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

2016-2017

Conception d'un outil de comparaison des méthodes de Graph Matching

Tuteurs académiques
Mostafa DARWICHE
Donatello CONTE
Vincent T'KINDT
Romain RAVEAUX

Étudiant
Kévin MICHAUX (DI5)

2 avril 2017

Liste des intervenants

Nom	Email	Qualité
Kévin MICHAUX	kev.michaux@etu.univ-tours.fr	Étudiant DI5
Mostafa DARWICHE	mostafa.darwiche@etu.univ-tours.fr	Tuteur académique, Département Informatique
Donatello CONTE	donatello.conte@univ-tours.fr	Tuteur académique, Département Informatique
Vincent T'KINDT	vincent.tkindt@univ-tours.fr	Tuteur académique, Département Informatique
Romain RAVEAUX	romain.raveaux@univ-tours.fr	Tuteur académique, Département Informatique

Avertissement

Ce document a été rédigé par Kévin MICHAUX susnommé l'auteur.

L'Ecole Polytechnique de l'Université François Rabelais de Tours est représentée par Mostafa DARWICHE, Donatello CONTE, Vincent T'KINDT et Romain RAVEAUX 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 reconnait 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 reconnait 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

Kévin MICHAUX, *Conception d'un outil de comparaison des méthodes de Graph Matching*, Projet Recherche & Développement, Ecole Polytechnique de l'Université François Rabelais de Tours, Tours, France, 2016-2017.

```
@mastersthesis{
    author={MICHAUX, Kévin},
    title={Conception d'un outil de comparaison des méthodes de Graph Matching},
    type={Projet Recherche \& Développement},
    school={Ecole Polytechnique de l'Université François Rabelais de Tours},
    address={Tours, France},
    year={2016-2017}
}
```

Document mis en forme sous LaTeX
grâce à la classe de document `polytech.cls` V2.2.3 (2017/03/08)

Table des matières

Liste des intervenants a

Avertissement b

Pour citer ce document c

Table des matières i

Table des figures v

1 Introduction 1

1 Contexte 1
2 Objectifs 1
3 Hypothèses 1
4 Bases méthodologiques 2

2 Description générale 3

1 Environnement du projet 3
2 Caractéristiques des utilisateurs 3
3 Fonctionnalités et structure générale du système 3

3 État de l'art : graphes et graph matching 5

1 Introduction 5
2 Graphes [4] 5
2.1 Définition 5
2.2 Graphes orientés et non-orientés 5
2.3 Graphes attribués et non-attribués 5
3 Problème de Graph Matching [1] 6

3.1 Domaines d'application 6
3.2 Exact Graph Matching 6
3.2.1 Définitions [4] 6
3.2.2 Méthodes [1] 7
3.3 Inexact Graph Matching 7
3.3.1 Définitions [4] 7
3.3.2 Graph Edit Distance [1, 4] 8
3.3.3 Méthodes [1] 9
3.3.4 Programmation linéaire binaire 10
4 Évaluation et comparaison des méthodes [3] 13
5 Conclusion 13

4 Veille technologique : visualisation de données 15
1 Introduction 15
2 Types de graphiques 15
3 Outils de génération de graphiques 16
3.1 Google Charts [WWW4] 16
3.2 D3.js [WWW1] 16
3.3 MorrisJs [WWW6] 17
3.4 HighCharts [WWW5] 17
3.5 FlotR2 [WWW3] 17
3.6 Elycharts [WWW2] 17
4 Conclusion 17

5 Analyse et conception 18
1 Application bureau VS application web 18
1.1 Application bureau 18
1.2 Application web 18
2 Choix du langage de développement 19
2.1 Java EE 19
2.2 PHP 19
2.3 Conclusion 19
3 Structure de l'application 21
4 Gestion des données 22

6 Mise en œuvre 25
1 Outils et librairies utilisées 25
2 Implémentation 25
3 Qualité et évaluation des performances 26

7 Bilan et conclusion 27
1 Semestre recherche et conception ... 27
1.1 Planning prévisionnel ... 27
1.2 Planning effectif ... 28
1.3 Conclusion ... 28
2 Semestre développement ... 29
2.1 Planning prévisionnel ... 29
2.1.1 Sprint 1 ... 29
2.1.2 Sprint 2 ... 29
2.1.3 Sprint 3 ... 29
2.1.4 Sprint 4 ... 30
2.1.5 Sprint 5 ... 30
2.2 Planning effectif ... 30
2.3 Conclusion ... 31

Annexes 32

A Description des interfaces externes du logiciel 33
1 Interfaces matériel/logiciel ... 33
2 Interfaces homme/machine ... 33
3 Interfaces logiciel/logiciel ... 38

B Spécifications fonctionnelles 39
1 Ajout / modification d'un modèle ... 39
2 Ajout / modification d'une collection de graphes ... 39
3 Ajout d'un test ... 39
4 Lancement d'un test ... 40
5 Consultation des résultats ... 40

C Gestion de projet 41
1 Choix de la méthode ... 41
2 Outils utilisés ... 41

D Documentation utilisateur 42
1 Introduction ... 42
2 Menu principal ... 42
3 Gestion des modèles ... 42
3.1 Liste des modèles ... 42
3.2 Fiche modèle ... 43
4 Gestion des collections de graphes ... 43

4.1 Liste des collections de graphes 43
4.2 Fiche collection de graphes 44
5 Gestion des tests 44
5.1 Liste des tests 44
5.2 Fiche test 45
6 Gestion des exécutions 45
6.1 Lancement d'une exécution 45
6.2 Liste des exécution 46
6.3 Détails d'une exécution 46
7 Résultats d'une exécutions 46
7.1 Général 47
7.2 Temps CPU 47
7.3 Nœuds explorés 47
7.4 Solutions optimales 47
7.5 Instances traitées 48
7.6 Déviation 48
7.7 Appareillements 48

E Documentation technique **50**
1 Serveur et déploiement 50
2 Mise à jour de l'application 50
3 Dépendances 50
4 Architecture de l'application 51
4.1 Package DAO 51
4.2 Package Model 51
4.2.1 Classe Execution 52
4.3 Package Servlets 54
4.4 Package Tools 54
4.5 Package Tests 55
5 Base de données 55
5.1 Architecture 55

Comptes rendus hebdomadaires **56**

Webographie **62**

Bibliographie **63**

Table des figures

2 Description générale

1 Diagramme de cas d'utilisation 4

4 Veille technologique : visualisation de données

1 Exemple d'histogramme empilé pour la visualisation du temps CPU 16

5 Analyse et conception

1 Diagramme de classes simplifié de l'application 21
2 Diagramme d'objets 23
3 Arborescence de l'application 24

7 Bilan et conclusion

1 Diagramme de Gantt prévisionnel du semestre recherche et développement 27
2 Diagramme de Gantt du semestre recherche et développement 28
3 Diagramme de Gantt prévisionnel du semestre développement 29
4 Diagramme de Gantt du semestre développement 30

A Description des interfaces externes du logiciel

1 Menu de l'application 33
2 Interface d'édition d'un modèle 33
3 Interface de liste des modèles 34
4 Interface d'édition d'une collection de graphes 34
5 Interface de liste des collections de graphes 35
6 Interface de gestion des tests 35

7 Interface de liste des tests 36
8 Interface de liste des exécutions 36
9 Interface de consultation des résultats 37
10 Diagramme de composants 38

D Documentation utilisateur

1 Liste des modèles 43
2 Fiche modèle 43
3 Liste des collections de graphes 44
4 Fiche collection de graphes 44
5 Liste des tests 45
6 Fiche test 45
7 Fiche exécution 46
8 Liste des exécutions 46
9 Détails d'une exécution 46
10 Résultats onglet Général 47
11 Résultats onglet Temps CPU 47
12 Résultats onglet Nœuds explorés 47
13 Résultats onglet Solutions optimales 48
14 Résultats onglet Instances traitées 48
15 Résultats onglet Déviation 48
16 Résultats onglet Appareillements 49

E Documentation technique

1 Diagramme de classe du package DAO 51
2 Diagramme de classe simplifié du package Modèle 52
3 Diagramme de classe de la classe Execution 53
4 Diagramme de classe du package Servlet 54
5 Schéma de navigation web 54
6 Schéma de la base de données 55

1 Introduction

1 Contexte

Le problème de Graph Matching permet, entre autres, de reconnaitre des formes ou des objets. C'est un problème complexe pour lequel de nombreuses méthodes de résolution ont été proposées par les scientifiques depuis les années 70, notamment grâce à l'utilisation de modèles mathématiques. Cependant, ces méthodes ne sont pas optimales et des chercheurs travaillent encore aujourd'hui pour les améliorer. Il est donc nécessaire de pouvoir tester et comparer ces méthodes.
C'est pourquoi mon encadrant, monsieur Mostafa DARWICHE, et plus généralement les chercheurs de Polytech Tours souhaitent disposer d'un outil qui doit donc permettre d'automatiser l'appel de ces méthodes avec des données et des paramètres différents et de restituer les résultats et les indicateurs de comparaison afin de faciliter le travail des chercheurs.

2 Objectifs

L'objectif final de ce projet est donc de concevoir un outil sous la forme d'une application web qui permettra la gestion des différents modèles et des collections de graphes, la gestion et l'exécution des tests ainsi que la restitution des résultats sous forme de graphiques.
Pour mener à bien cet objectif, j'ai dut réaliser un état de l'art sur les graphes et les méthodes de comparaison pour bien comprendre le contexte du projet. Ensuite, j'ai produit un second état de l'art sur les outils de représentation graphique de données afin de terminer celui qui sera le plus adapté aux besoins.
Enfin, j'ai mis en place des méthodologies de gestion de projet afin de prévoir ce que je dois faire et mesurer mon avancement dans l'optique de proposer une application fonctionnelle correspondant aux besoins du client à la fin du projet.

3 Hypothèses

Représentation de l'appareillement de graphes

Représenter l'appareillement de deux graphes consiste à montrer à quel nœud / arc du graphe 2 et associé chaque nœud / arc du graphe 1. Cette tâche n'est pas facile à réaliser et il se

trouve qu'un chercheur du département informatique de Polytech Tours travaille sur une librairie permettant une représentation graphique de cet appareillement. Nous ne savons pas à l'heure actuelle s'il sera possible de l'utiliser, c'est pourquoi une solution alternative est prévue : représenter les appareillements par un tableau. Cela sera moins agréable à visualiser mais cette solution est plus simple à mettre en place et ne dépend pas d'une tierce personne.

Format du fichier de sortie de Cplex

Le fichier de sortie fourni par Cplex doit être interprété pour afficher les résultats des tests. Il se trouve qu'un modèle récemment créé produit un fichier de sortie dans un format différent. Étant donné le temps limité pour le développement de l'application, les fichiers de sortie seront considérés comme ayant tous le même format. Nous verrons par la suite, en fonction de l'avancement du projet, s'il est possible de modifier l'application pour qu'elle puisse lire une structure de fichier différente.

Concurrence des threads et ressources matérielles

Les performances des modèles dépendent en partie des ressources matérielles dont chaque thread dispose pour l'exécution d'un test (CPU, mémoire RAM...). Il est donc nécessaire de contrôler que l'ensemble des threads lancés ne demandent pas plus de ressources que celles disponibles sur le serveur. Dans un premier temps, les threads seront donc exécutés de manière séquentielle puis, si le temps restant le permet, nous mettrons en place des contrôles sur les ressources disponibles afin de déterminer le nombre de tests pouvant être exécutés simultanément.

4 Bases méthodologiques

Les fonctionnalités de cet outil ainsi que les données gérées par ce dernier seront modélisées grâce au langage UML. Ce projet sera mené en suivant une méthode agile permettant de fournir des livrables fréquemment et d'avoir un retour régulier du client. La gestion du projet sera gérée avec le logiciel MindView et le versionning des sources grâce à Git. Cette application web sera développée en utilisant le langage Java EE sous l'environnement de développement IntelliJ et sera hébergée sur un serveur Tomcat. La gestion des données sera réalisée grâce au framework Hibernate. Les graphiques seront générés en Javascript grâce à l'API Google Charts.

2 Description générale

1 Environnement du projet

Au début du projet, l'implémentation en langage C des modèles mathématiques est existante. Ils devront être modifiés afin de permettre la lecture du fichier généré par l'application web contenant les paramètres renseignés par l'utilisateur. Ils seront par la suite compilés sous forme d'exécutable pour être appelés par l'application. En revanche, l'application web est inexistante.

2 Caractéristiques des utilisateurs

L'application sera utilisée par les chercheurs du département informatique de Polytech Tours. Ils devront donc avoir des connaissances dans le domaine des graphes et surtout des méthodes de comparaison afin de pouvoir paramétrer les test et tirer des conclusions sur les résultats affichés. Aucune gestion des utilisateurs n'est requise.

3 Fonctionnalités et structure générale du système

Les utilisateurs auront la possibilité de gérer les modèles (implémentation des modèles mathématiques sous forme d'exécutable) et les collections de graphes sur lesquelles seront exécutés les tests. Ils pourront également créer et lancer des tests en choisissant des modèles et une ou plusieurs collection(s) de graphes et en y associant différents paramètres. Enfin, ils pourront consulter les résultats de ces tests et les performances associées.

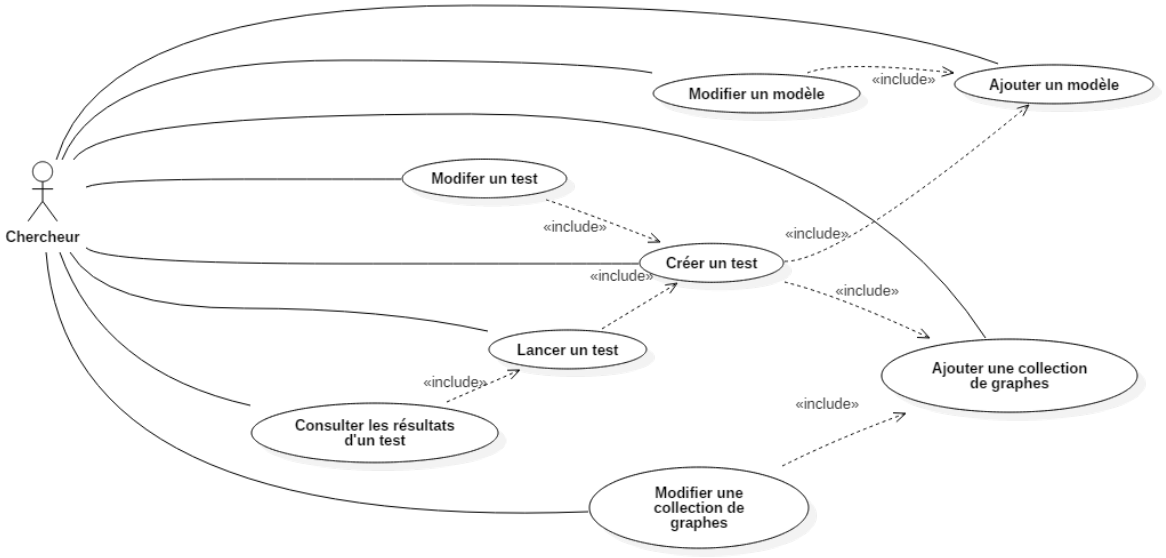


Figure 1 – *Diagramme de cas d'utilisation*

Ce diagramme de cas d'utilisation présente les différentes actions pouvant être réalisées par l'utilisateur. Par souci de lisibilité, la consultation des listes (modèles, collections, tests et résultats) de même que les suppressions ne sont pas représentées sur ce diagramme.

L'application sera chargée d'appeler les exécutables des modèles mathématiques qui, après avoir lu les paramètres et les graphes, feront eux-mêmes appel au logiciel d'optimisation Cplex. Ce dernier fournira les résultats sous la forme d'un fichier texte que l'application se chargera d'interpréter pour restituer les résultats.

3 État de l'art : graphes et graph matching

1 Introduction

L'objectif de cet état de l'art est dans un premier temps de pouvoir cerner le problème du Graph Matching sous ses différentes formes et d'évoquer les méthodes permettant de résoudre ce problème. Nous nous concentrerons particulièrement sur l'Inexact Graph Matching et sur la méthode de Graph Edit Distance et notamment sur les formulations de programmation linéaire. Par la suite, nous chercherons à déterminer la meilleure manière de visualiser les performances de ces différentes méthodes afin de les comparer et évaluer.

2 Graphes [4]

2.1 Définition

Un graphe est une structure de données composée de nœuds reliés entre eux par des arcs. Un graphe est noté $G = (V, E)$ où V est l'ensemble des nœuds et E tel que $E \subseteq VxV$ est l'ensemble des arcs. Un arc est notée $e = (a, b)$ où a et b sont les nœuds reliés par cet arc.

2.2 Graphes orientés et non-orientés

Un graphe non-orienté est un graphe dont les arcs n'ont pas de sens. Par conséquent, un arc $e1 = (a, b)$ est donc identique à l'arc $e2 = (b, a)$. A l'inverse, les arcs d'un graphe orienté ont un sens, c'est à dire que les arcs $e1 = (a, b)$ et $e2 = (b, a)$ sont donc différents.

2.3 Graphes attribués et non-attribués

Un graphe attribué possède des descripteurs associés aux nœuds et/ou aux arcs. Ces descripteurs peuvent-être de type numérique ou textuel. Un graphe attribué est noté $G = (V, E, \mu, \varsigma)$ où $\mu : V \longrightarrow L_V$ est une fonction qui associe un attribut (ou label) à un nœud et $\varsigma : E \longrightarrow L_E$ est une fonction qui associe un attribut (ou label) à un arc.

3 Problème de Graph Matching [1]

Le Graph Matching consiste à comparer des graphes représentant des objets dans le but de les classifier ou de reconnaitre des formes. Proposées depuis les années 1970, ces techniques ont réellement commencé à être utilisées par les scientifiques au début des années 2000 car les ordinateurs ne fournissaient pas jusque là une puissance de calcul suffisante pour faire fonctionner les algorithmes de Graph Matching de manière confortable.
Le problème de Graph Matching se décompose en deux types : le "Exact Graph Matching" et le "Inexact Graph Matching".

3.1 Domaines d'application

Le problème de Graph Matching peut être appliqué dans des domaines variés dont voici quelques exemples :

- Dans la biologie et le biomédical pour comparer des molécules et faire de la reconnaissance d'empreintes digitales.
- Pour l'analyse de document afin de reconnaitre des caractères ou des mots, d'analyser une écriture manuscrite.
- Dans l'analyse d'images ou de vidéos pour reconnaitre et détecter des objets, suivre un mouvement.

3.2 Exact Graph Matching

3.2.1 Définitions [4]

L'Exact Graph Matching consiste à déterminer s'il y a une correspondance exacte entre les deux graphes étudiés. On peut également chercher à savoir si un graphe est contenu dans l'autre.
On peut distinguer plusieurs formes d'Exact Graph Matching qui sont plus ou moins strictes :

Isomorphisme

Chaque nœud d'un graphe doit correspondre à un et un seul nœud de l'autre graphe et chaque arc reliant deux nœuds dans un graphe doit correspondre à un et un seul arc dans l'autre. Il existe donc une fonction bijective $f : V \longrightarrow V'$ de $G = (V, E, \mu, \varsigma)$ vers $G' = (V', E', \mu', \varsigma')$ respectant les conditions suivantes :

1. $\forall v \in V, \mu(v) = \mu'(f(v))$
2. $\forall v' \in V', \mu'(v') = \mu'(f^{-1}(v'))$
3. $\forall e \in E et \forall e' \in E', \varsigma(e(v_i, v_j)) = \varsigma'(e'(f(v_i), f(v_j)))$
4. $\forall e'(v'_i, v'_j) \in E', \exists e = (f^{-1}(v'_i), f^{-1}(v'_j)) \in E$ avec $\varsigma'(e') = \varsigma(e)$

Sous-graphe isomorphisme

Il doit y avoir un isomorphisme entre le premier graphe $G = (V, E, \mu, \varsigma)$ et un sous-graphe du deuxième graphe $G' = (V', E', \mu', \varsigma')$. Le deuxième graphe peut donc avoir des nœuds et des arcs qui n'existent pas dans le premier mais les nœuds communs doivent avoir les mêmes arcs communs. Il existe donc une fonction $f : V \longrightarrow V'$ qui à tout nœud $v \in V$ associe un nœud $v' \in V'$ respectant les conditions suivantes :

1. $\forall v \in V, \mu(v) = \mu'(f(v))$
2. $\forall e \in E$ et $\forall e' \in E', e(v_i, v_j) = e'(f(v_i), f(v_j)$

3. $\forall e'(v'_i, v'_j) \in E' \cap f(V)xf(V), \exists e = (f^{-1}(v'_i), f^{-1}(v'_j)) \in E avec \varsigma'(e') = \varsigma(e)$
4. $\mu(v) = \mu'(v')$
5. $\varsigma(e) = \varsigma'(e')$

Monomorphisme

Tous les nœuds et les arcs du premier graphe doivent avoir une correspondance dans le second mais un nœud peut être lié à plus d'arcs dans le deuxième graphe que dans le premier. Il existe donc une fonction injective $f : V \longrightarrow V'$ de $G = (V, E, \mu, \varsigma)$ vers $G' = (V', E', \mu', \varsigma')$ respectant les conditions suivantes :

1. $\forall v \in V, f(v) = v', f^{-1}(v') = v$
2. $\forall e = (v_i, v_j) \in E, \exists e' = (f(v_i), f(v_j) \in E'$
3. $\mu(v) = \mu'(v')$
4. $\varsigma(e) = \varsigma'(e')$

Sous-graphe commun maximal

On recherche ici le plus grand sous-graphe commun aux deux graphes.

3.2.2 Méthodes [1]

Il existe de nombreux algorithmes basés sur la méthode des arbres de recherche. Le principe est d'explorer toutes les possibilités de couplage entre les nœuds afin de voir si on peut trouver une correspondance entre les deux graphes. On peut également intégrer des euristiques afin de réduire le nombre de possibilités testées. On peut citer par exemple :

— *L'algorithme d'Ullmann* pour les problèmes d'isomorphisme, de sous-graphe isomorphisme et de monomorphisme.
— *L'algorithme Netgraph* pour les problèmes de monomorphisme.
— *L'algorithme VF* pour les problèmes d'isomorphisme et de sous-graphe isomorphisme.

On trouve également d'autres algorithmes comme :

— *L'algorithme de Nauty* basé sur la théorie des groupes capable de calculer un groupe d'automorphisme pour chaque graphe et d'en déduire une forme canonique. Il est ensuite capable de détecter un isomorphisme entre deux graphes et vérifiant l'égalité entre les matrices d'adjacences des formes canoniques.
— *L'algorithme de Messmer et Bunke* qui décompose récursivement les graphes en sous-graphes très simples pour déterminer les parties communes aux graphes de la collection. Ainsi ces parties communes ne seront pas comparées pour chaque graphe avec le graphe que l'on souhaite étudier. Il permet de résoudre des problèmes d'isomorphisme et de sous-graphe isomorphisme.

L'inconvénient de ces algorithmes est qu'ils nécessitent la plupart du temps énormément de temps de calcul, il ne peuvent donc être appliqués que sur des graphes de petite taille.

3.3 Inexact Graph Matching

3.3.1 Définitions [4]

Dans certains cas d'utilisation, les images que l'on souhaite étudier peuvent contenir du bruit ce qui se traduit par des graphes légèrement modifiés. Dans d'autres cas, on peut chercher à reconnaitre des objets de même nature mais comportant des différences (par exemple deux voitures de marque différente, un mot écrit par deux personnes, etc...). Pour cela, il est plus

adapté de chercher une ressemblance plutôt qu'une correspondance exacte. Il devient donc nécessaire d'évaluer cette différence entre ces deux graphes. De plus, les algorithmes d'Exact Graph Matching pouvant nécessiter un temps de calcul exponentiel, il est plus pertinent d'utiliser des algorithmes capables de fournir une solution approchée en un temps raisonnable. On peut distinguer plusieurs formes d'inexact Graph Matching :

Sous-graphe isomorphisme tolérant à la substitution

Il s'agit d'un isomorphisme entre un graphe G et un sous-graphe de G' au niveau de la structure du graphe (ses nœuds et ses arcs) mais dont les valeurs associées peuvent être différentes (par exemple, des valeurs numériques associées aux arcs). Il existe donc une fonction injective $f : V \longrightarrow V'$ de $G = (V, E, \mu, \varsigma)$ vers $G' = (V', E', \mu', \varsigma')$ respectant les conditions suivantes :

1. $\forall v \in V, f(v) = v', f^{-1}(v') = v$
2. $\forall e = (v_i, v_j) \in E, \exists e' = (f(v_i), f(v_j) \in E'$
3. $\mu(v) \approx \mu'(v')$
4. $\varsigma(e) \approx \varsigma'(e')$

Inexacte sous-graphe isomorphisme

Il est possible d'ajouter des nœuds et des arcs fictifs dans le second graphe afin d'obtenir un isomorphisme avec le premier. Il existe donc une fonction injective $f : V \longrightarrow V'$ de $G = (V, E, \mu, \varsigma)$ vers $G' = (V', E', \mu', \varsigma')$ respectant les conditions suivantes :

1. Soit $\bigwedge'_v$ un ensemble de nœuds fictifs
2. Soit $\bigwedge'_e$ un ensemble d'arcs fictives
3. $\forall v \in V, f(v) = v' \in V' \cup \bigwedge'_v$
4. $\forall e = (v_i, v_j) \in E, \exists e' = (f(v_i), f(v_j) \in E' \cup \bigwedge'_e$
5. $\mu(v) \approx \mu'(v')$
6. $\varsigma(e) \approx \varsigma'(e')$

Inexacte graphe isomorphisme

On peut ajouter des nœuds et des arcs fictifs dans les deux graphes afin d'obtenir un isomorphisme structurel entre les deux tout en ayant éventuellement des valeurs différentes pour chaque nœuds ou arcs associés. Il existe donc une fonction bijective $f : V \longrightarrow V'$ de $G = (V, E, \mu, \varsigma)$ vers $G' = (V', E', \mu', \varsigma')$ respectant les conditions suivantes :

1. Soient $\bigwedge_v$ et $\bigwedge'_v$ deux ensembles de nœuds fictifs
2. Soient $\bigwedge_e$ et $\bigwedge'_e$ deux ensembles d'arcs fictives
3. $\forall v \in V \cup \bigwedge_v, f(v) = v' \in V' \cup \bigwedge'_v, f^{-1}(v') = v$
4. $\forall e = (v_i, v_j) \in E \cup \bigwedge_e, \exists e' = (f(v_i), f(v_j) \in E' \cup \bigwedge'_e$
5. $\forall e' = (v'_i, v'_j) \in E' \cup \bigwedge'_e, \exists e = (f^{-1}(v'_i), f^{-1}(v'_j) \in E \cup \bigwedge_e$
6. $\mu(v) \approx \mu'(v')$
7. $\varsigma(e) \approx \varsigma'(e')$

3.3.2 Graph Edit Distance [1, 4]

Nous nous intéressons particulièrement à cette méthode car c'est l'une des plus étudiées et utilisées pour la reconnaissance de formes, elle permet de répondre aux problèmes d'isomorphisme et de sous-graphes isomorphisme. De plus, il a été démontré que le coût de transformation peut être interprété comme une distance à condition de respecter certaines

propriétés mathématiques. Cette méthode nous permet donc de mesurer la dissimilarité entre deux graphes. Le principe de cette méthode est d'effectuer une suite de transformation sur le premier graphe pour obtenir le second qui est appelée le chemin d'édition. Un coût est associé à chaque transformation (ajout, suppression ou substitution de nœuds et d'arcs). La somme de ces coûts est appelée le coût d'édition du graphe et peut être utilisée pour mesurer la distance entre les deux graphes. L'objectif est donc de trouver la solution optimale minimisant ce coût afin d'obtenir un graphe le plus proche possible du premier. L'inconvénient de cette méthode est que le temps d'exécution dépend du nombre de chemins d'édition possibles et donc de la taille du graphe. Elle n'est donc applicable que sur des graphes de petite taille.
La distance d'édition entre les graphes $G_1 = (V_1, E_1, \mu_1, \varsigma_1)$ et $G_2 = (V_2, E_2, \mu_2, \varsigma_2)$ est définie de la manière suivante :

$$d_{plain}(G_1, G_2) = \min_{e_1,\ldots,e_k \in EP} \sum_{i=1}^{k} c(e_i)$$

Où $EP = e_1, \ldots, e_k$ est le chemin d'édition correspondant aux k transformations effectuées réalisées pour transformer G_1 en G_2 et c la fonction de coût telle que $c(e_i)$ donne le coût associé à la transformation e_i

3.3.3 Méthodes [1]

Il existe des méthodes optimales qui sont capables de trouver la meilleure solution. Cependant, ces méthodes nécessitent un temps de calcul important, voir supérieur à celui des algorithmes d'Exact Graph Matching. Heureusement, il existe aussi des méthodes non optimales qui sont capables de trouver des solutions proches de la solution optimale en un temps souvent polynomial. Voici une liste non exhaustive des différents types de méthodes existantes.
On retrouve là encore de nombreuses méthodes basées sur des arbres de recherche comme par exemple :

- *L'algorithme de Tsai et Fu* est applicable aux graphes attribués. Il existe plusieurs versions de cette méthode optimale permettant de résoudre respectivement des problèmes d'isomorphisme (en prenant en compte le coût des substitutions), de sous-graphe isomorphisme (avec les coûts d'insertion et de suppression en plus) et de monomorphisme. Eshera et Fu on également proposé une version non-optimale de cette méthode qui décompose les graphes en un ensemble de sous-graphes et recherche le plus grand ensemble commun entre les deux graphes.
- *L'Algorithme de Gharaman* permet de trouver une solution optimale en parcourant l'arbre de recherche grâce à une heuristique inspirée de l'algorithme d'Exact Graph Matching Netgraph. Gharaman a également proposé une méthode à base d'euristique permettant de trouver un homomorphisme entre deux hypergraphes.
- *Les algorithmes basés sur A** explorent les différents chemins d'édition possibles grâce à un arbre de recherche. Des euristiques permettent d'estimer le coût de chaque branche correspondant à un chemin d'édition et ainsi d'explorer uniquement les solutions susceptibles de fournir la meilleure solution et ainsi réduire le temps d'exécution qui reste cependant dépendant de la taille du graphe. On distingue ceux proposés par Dumay, Berretti et Gregory et Kittler.

Il existe également des méthodes mathématiques non optimales basées sur l'optimisation continue parmi lesquelles on distingue :

- *La relaxation de labels*, proposée par Fischler et Elschager, consiste à associer à chaque nœud un vecteur contenant les probabilités de correspondance avec chaque nœud d'un autre graphe qui sont calculées en plusieurs itérations grâce à une euristique en fonction des attributs et des connexions des nœuds. De nombreux algorithmes utilisent cette méthode comme ceux de Kittler et Hancock, Christmas, Wilson et Hancock et Myers.

— *L'algorithme EM (Expectation-Maximisation)* (espérence-maximisation) consiste à maximiser itérativement les paramètres de vraisemblance. Il est utilisé par l'algorithme de Luo et Hancock où les nœuds du modèle sont considérés comme les variables aléatoires cachées.
— *Weighted Graph Matching (WGM)* Les arcs sont pondérés par des nombres réels, l'objectif étant de trouver une correspondance entre le poids de deux arcs associés. L'association est faite grâce à une matrice de 0 et de 1 dont la somme de chaque ligne et chaque colonne ne doit pas être supérieure à 1 afin de s'assurer qu'un nœud soit associé à un seul nœud du second graphe. Cette méthode est applicable uniquement aux graphes non attribués (sauf le poids associé aux arcs). De nombreuses méthodes de résolution ont été conçues comme, par exemple, par Almohamad et Duffua qui ont proposé une formulation linéaire de ce problème qui peut être résolue avec l'algorithme du Simplex.

Les méthodes spectrales sont basées sur l'étude des valeurs propres et des vecteurs propres de la matrice d'adjacence d'un graphe car leurs valeurs ne sont pas modifiées par la permutation des nœuds. Ainsi, les matrices d'adjacence de deux graphes isomorphes auront les mêmes valeurs propres et vecteurs propres. Cependant, l'inverse n'est pas vrai, deux graphes dont les matrices d'adjacence ont les mêmes valeurs propres et vecteurs propres ne sont pas forcement isomorphes. C'est pourquoi ces méthodes ne permettent pas de répondre au problème du Graph Matching mais permettent de calculer la matrice de permutation optimale entre deux graphes isomorphes comme l'a démontré Umeyama.
Cette méthode a été combinée avec des techniques d'optimisation continue par Xu et King afin d'améliorer les performances des algorithmes. Elle a également été utilisée pour définir des groupes de nœuds qui peuvent être appareillés dans la solution optimale par Carcassoni et Hancock ainsi que par Kosinov et Caelli.
Certaines techniques d'Exact Graph Matching ont été adaptées pour répondre au problème d'Inexact Graph Matching, notamment l'algorithme de décomposition de Messmer et Bunke permettant de trouver un sous-graphe isomorphisme.
Enfin, il existe des techniques basées sur l'utilisation des réseaux de neurones.

3.3.4 Programmation linéaire binaire

L'idée est de transformer le problème de Graph Edit Distance en problème d'optimisation. Pour cela, le problème est formulé de manière mathématique afin d'obtenir une fonction objectif à minimiser représentant le coût du chemin d'édition et des contraintes. Ce modèle mathématique peut ensuite être résolu grâce à un logiciel d'optimisation, comme Cplex par exemple.

Modèle F1 [3]

Définition des données :
Soient deux graphes orientés définis par $G_1 = (V_1, E_1, \mu_1, \varsigma_1)$ et $G_2 = (V_2, E_2, \mu_2, \varsigma_2)$.

Définition des variables correspondantes aux transformations effectuées :
1. $\forall (i,k) \in V_1 \times V_2$, $x_{i,k} = 1$ si i est substitué avec k, 0 sinon.
2. $\forall (ij,kl) \in E_1 \times E_2$, $y_{ij,kl} = 1$ si ij est substitué avec kl, 0 sinon.
3. $\forall i \in V_1$, $u_i = 1$ si i est supprimé de G_1, 0 sinon.
4. $\forall ij \in E_1$, $e_{ij} = 1$ si ij est supprimé de G_1, 0 sinon.
5. $\forall k \in V_2$, $v_k = 1$ si k est inséré dans G_1, 0 sinon.
6. $\forall kl \in E_2$, $f_{kl} = 1$ si kl est inséré dans G_1, 0 sinon.

Définition des variables correspondantes au coût des transformations :
1. $\forall (i,k) \in V_1 \times V_2$, $c(i \to k)$ est le coût de substitution du nœud i avec k.
2. $\forall (ij,kl) \in E_1 \times E_2$, $c(ij \to kl)$ est le coût de substitution de l'arc ij avec kl.
3. $\forall i \in V_1$, $c(i \to e)$ est le coût de suppression du nœud i de G_1.

4. $\forall ij \in E_1$, $c(ij \to e)$ est le coût de suppression de l'arc ij de G_1.
5. $\forall k \in V_2$, $c(e \to k)$ est le coût d'insertion du nœud k dans G_1.
6. $\forall kl \in E_2$, $c(e \to kl)$ est le coût d'insertion de l'arc kl dans G_1.

Le chemin d'édition entre les graphes G_1 et G_2 est défini par l'ensemble (x, y, u, v, e, f) où :

1. $x = (x_{i,k})_{(i,k)\in V_1 \times V_2}$,
2. $y = (y_{ij,kl})_{(ij,kl)\in E_1 \times E_2}$,
3. $u = (u_i)_{i\in V_1}$,
4. $v = (v_k)_{k\in V_2}$
5. $e = (e_{ij})_{ij\in E_1}$,
6. $f = (f_{kl})_{kl\in E_2}$

Fonction objectif :

$$\begin{aligned}&\min_{(x,y,u,v,e,f)}\Big(\sum_{i\in V_1}\sum_{k\in V_2} c(i \to k)\cdot x_{i,k}\\ &+\sum_{ij\in E_1}\sum_{kl\in E_2} c(ij \to kl)\cdot y_{ij,kl}\\ &+\sum_{i\in V_1} c(i \to \epsilon)\cdot u_i\\ &+\sum_{k\in V_2} c(\epsilon \to k)\cdot v_k\\ &+\sum_{ij\in E_1} c(ij \to \epsilon)\cdot e_{ij}\\ &+\sum_{kl\in E_2} c(\epsilon \to kl)\cdot f_{kl}\Big)\end{aligned}$$

Le but de la fonction objectif est de minimiser le coût total des transformations effectuées de G_1 en G_2.

Contraintes :

1. $u_i + \sum_{k\in V_2} x_{i,k} = 1 \ \forall i \in V_1$
2. $v_k + \sum_{i\in V_1} x_{i,k} = 1 \ \forall k \in V_2$
3. $e_{ij} + \sum_{kl\in E_2} y_{ij,kl} = 1 \ \forall ij \in E_1$
4. $f_{kl} + \sum_{ij\in E_1} y_{ij,kl} = 1 \ \forall kl \in E_2$
5. $y_{ij,kl} \le x_{i,k} \ \forall (ij,kl) \in E_1 \times E_2$
6. $y_{ij,kl} \le x_{j,l} \ \forall (ij,kl) \in E_1 \times E_2$

Les contraintes 1 et 2 assurent que chaque nœud restant dans G_1 corresponde à un et un seul nœud de G_2 et inversement.
Les contraintes 3 et 4 assurent que chaque arc restant dans G_1 corresponde à un et un seul arc de G_2 et inversement.
Soient un arc ij allant du nœud i vers le nœud j appartenant à G_1 et un arc kl allant du nœud k vers le nœud l appartenant à G_2. Les contraintes 5 et 6 assurent que si l'arc ij est associé à l'arc kl, les nœuds i et j sont respectivement associés aux nœuds k et l et inversement.

Modèle F2 [3]

Le modèle F2 est en fait une autre formulation du modèle F1 qui vise à réduire le nombre de variables et de contraintes afin d'améliorer le temps de calcul.
Dans le but de réduire le nombre de variables, on peut considérer les insertions et les suppressions comme des substitutions ce qui nous permet de supprimer les variables u,v, e et f. Les contraintes 1 à 4 du modèle F1 sont donc remplacées par les inégalités suivantes équivalentes :

1. $\sum_{k\in V_2} x_{i,k} \le 1 \ \forall i \in V_1$
2. $\sum_{i\in V_1} x_{i,k} \le 1 \ \forall k \in V_2$
3. $\sum_{kl\in E_2} y_{ij,kl} \le 1 \ \forall ij \in E_1$
4. $\sum_{ij\in E_1} y_{ij,kl} \le 1 \ \forall kl \in E_2$

La suppression de ces variables implique donc également de modifier la fonction objectif. On défini une matrice $C = \sum_{i \in V_1} c(i \to \epsilon) + \sum_{k \in V_2} c(\epsilon \to k) + \sum_{ij \in E_1} c(ij \to \epsilon) + \sum_{kl \in E_2} c(\epsilon \to kl)$ qui représente les coûts d'insertion et de suppression des nœuds et des arcs. On obtient ainsi la fonction objectif suivante :

$$\min_{(x,y)}(\sum_{i \in V_1} \sum_{k \in V_2} (c(i \to k) - c(i \to \epsilon) - c(\epsilon \to k)) \cdot x_{i,k} + \sum_{ij \in E_1} \sum_{kl \in E_2} (c(ij \to kl) - c(ij \to \epsilon) - c(\epsilon \to kl)) \cdot y_{ij,kl} + C$$

Il est également possible de formuler les contraintes 3 à 6 du modèle F1 différemment afin d'en réduire le nombre tout en obtenant le même résultat de la manière suivante :

1. $\sum_{kl \in E_2} y_{ij,kl} \leq x_{i,k} \ \forall k in V_2, \forall ij \in E_1$
2. $\sum_{kl \in E_2} y_{ij,kl} \leq x_{j,l} \ \forall l in V_2, \forall ij \in E_1$

Suite à ces modifications ont obtient donc le modèle suivant :

Variables :

1. $x = (x_{i,k})_{(i,k) \in V_1 \times V_2}$,
2. $y = (y_{ij,kl})_{(ij,kl) \in E_1 \times E_2}$,

Contraintes :

1. $\sum_{k \in V_2} x_{i,k} \leq 1 \ \forall i \in V_1$
2. $\sum_{i \in V_1} x_{i,k} \leq 1 \ \forall k \in V_2$
3. $\sum_{kl \in E_2} y_{ij,kl} \leq x_{i,k} \ \forall k in V_2, \forall ij \in E_1$
4. $\sum_{kl \in E_2} y_{ij,kl} \leq x_{j,l} \ \forall l in V_2, \forall ij \in E_1$

Fonction objectif :

$$\min_{(x,y)}(\sum_{i \in V_1} \sum_{k \in V_2} (c(i \to k) - c(i \to \epsilon) - c(\epsilon \to k)) \cdot x_{i,k} + \sum_{ij \in E_1} \sum_{kl \in E_2} (c(ij \to kl) - c(ij \to \epsilon) - c(\epsilon \to kl)) \cdot y_{ij,kl} + C$$

Grâce à cette nouvelle formulation, on passe de :

— $|V_1| + |V_2| + |E_1| + |E_2| + |V_1| \cdot |V_2| + |E_1| \cdot |E_2|$ à $|V_1| \cdot |V_2| + |E_1| \cdot |E_2|$ variables
— $|V_1| + |V_2| + |E_1| + |E_2| + 2 \cdot |E_1| \cdot |E_2|$ à $|V_1| + |V_2| + 2|V_2| \cdot |E_1|$ contraintes

Modèle 3 [2]

Cette méthode est applicable aux graphes non orientés avec des attributs associés aux nœuds et des arcs sans attributs. Elle consiste à partir d'un graphe complet $G_\Omega = (\Omega, \Omega \times \Omega, l_\Omega)$ où Ω est l'ensemble des nœuds, $\Omega \times \Omega$ l'ensemble des arcs et l_Ω la fonction associant un nœud à un attribut de l'alphabet $\sum$. G_Ω est appelé grille d'édition. Les nœuds sont numérotés de 1 à N et les arcs de N+1 à 2N. On place G_1 dans la grille d'édition puis on va effectuer des transformations (insertion, suppression, substitution) sur G_1 jusqu'à obtenir G_2. Naturellement, il existe plusieurs séquences de transformations possibles avec donc des coûts d'édition différents. L'objectif est de trouver la séquence ayant le coût le plus faible afin de déterminer la distance entre G_1 et G_2.

Données :

Soient deux graphes non orientés $G_1 = (V_1, E_1, l_1)$ et $G_2 = (V_2, E_2, l_2)$. Soient c la fonction de coût définie par $c : (\sum \bigcup \phi)^2 \bigcup 0,1^2 \longrightarrow \Re\varepsilon_+$ qui associe un coût positif à chaque transformation et C la matrice des coûts. Le coût d'une transition d'un graphe vers un autre correspond à la somme des coûts de toutes les permutations effectuées pour passer d'un état à l'autre. $C(\eta_{k-1}, \eta_k) = \sum_{i=1}^{I} c(\eta_{k-1}^i, \eta_k^i)$ où I correspond au nombre d'éléments (nœuds + arcs) de la grille

d'édition.

Variables :

- $\eta \in (\sum \bigcup \phi)_N \times 0,1^{\frac{1}{2}(N^2-N)}$ est le vecteur contenant les attributs associés aux nœuds et aux arcs.
- π est le vecteur contenant les permutations de G_1 vers un isomorphisme de ce dernier. Par exemple, si $\pi^4 = 5$ cela signifie que le nœud qui était à la position 4 dans G_Ω se trouve maintenant à la position 5.
- Π est l'ensemble contenant touts les vecteurs de permutation possibles de G_1 vers un isomorphisme de ce dernier.
- $P \in 0,1^{NxN}$ est la matrice de permutation associée à π telle que $P^{ij} = \delta(\pi_i, j)$
  Où δ est le delta de Kronecker qui est une fonction définie telle que : $\delta_{ij} = \begin{cases} 1, & \text{si } i = j \\ 0, & \text{si } i \neq j \end{cases}$
- $A_k \in 0,1^{NxN}$ est la matrice d'adjacence correspondant au vecteur η_k où chaque ligne correspond à un label et qui associe chaque nœud à un label
- la matrice $T \in 0,1^{NxN} = A_0 - A_1$
- la matrice $S \in 0,1^{NxN} = A_1 - A_0$

Contraintes :

1. $(A_0P - PA_1 + S - T)^{ij} = 0 \forall i, j$
2. $\sum_i P^{ik} = \sum_j P^{kj} = 1 \forall k$

Fonction objectif :

$$\min_{P,S,T \in 0,1^{NxN}} \sum_{i=1}^{N} \sum_{j=1}^{N} c(l(A_0^i), l(A_1^j))P^{ij} + \frac{1}{2}c(0,1)(S+T)^{ij}$$

La fonction objectif a pour but de déterminer l'isomorphisme de G_1 parmi Π ayant le coût d'édition le plus petit possible qui est déterminé par la somme des coûts des transformations effectuées.

4 Évaluation et comparaison des méthodes [3]

Étant donné le nombre important de méthodes existantes, il est important de pouvoir les comparer afin de déterminer laquelle sera le plus adapté à un problème donné. Pour cela, il convient de les tester sur des bases de données différentes (par exemple GREC, MUTA, PROT, ILPSO) et avec des graphes de taille différente. Bien entendu, il faut veiller à ne pas comparer les méthodes exactes avec les méthodes inexactes. Enfin, les test doivent être réalisées dans les mêmes conditions : c'est à dire des ordinateurs ayant la même puissance de calcul avec un temps maximum d'exécution identique.
Ensuite, les méthodes sont comparées suivant deux critères importants :

- La pertinence des solutions
- Le temps de calcul

Il faut donc trouver un compromis entre ces deux critères qui dépendra du problème que l'on a à résoudre.

5 Conclusion

Le problème de Graph Matching peut être appliqué dans de nombreux domaines, c'est pourquoi il a beaucoup de scientifiques qui ont et qui continuent à travailler dessus. Cette diversité

explique qu'il y ait de nombreuses formes de définition existantes (exactes ou inexactes) qui sont plus ou moins strictes, chacune étant plus ou moins adaptée pour une application spécifique. D'autre part, la forte complexité de ce problème incite les chercheur à trouver des solutions de plus en plus performantes au regard de deux critères qui sont la pertinence de résultat et la rapidité de calcul. Cela explique le nombre important d'algorithmes qui ont été conçus.
A l'heure actuelle, il n'existe pas de méthode permettant de trouver une solution optimale dans un temps de calcul raisonnable. Il faut donc faire un compromis entre la précision et la vitesse qui dépendra du cas d'utilisation. En effet, si on souhaite par exemple reconnaitre des voitures dans une vidéo il faudra une méthode qui soit rapide alors que si on souhaite comparer des molécules en chimie on préférera une méthode plus lente mais plus précise.

4 Veille technologique : visualisation de données

1 Introduction

Afin de pouvoir évaluer et comparer les différents modèles mathématiques de comparaison de graphes, il est nécessaire d'étudier différents critères comme le temps CPU, le nombre de solutions optimales trouvées, le nombre de nœuds parcours pour trouver une solution par exemple. Comparer toutes ces valeurs numériques en fonction des modèles, des paramètres et des données peut s'avérer fastidieux. C'est pour cela que nous chercheront à représenter ces indicateurs numériques de façon graphique afin d'en faciliter la compréhension. Cette section a pour objectif de faire un tour d'horizon des différentes solutions permettant de générer des graphiques en vue de les intégrer dans une interface web afin de choisir celle qui sera le plus adaptée pour répondre aux besoins du projet.

2 Types de graphiques

Avant de choisir comment générer les graphiques il est nécessaire de déterminer quel(s) graphique(s) nous allons utiliser. En effet, en fonction des données à visualiser et des informations que l'on souhaite en tirer, certaines formes visuelles seront plus adaptées que d'autres. Le point commun entre toutes les données que l'on souhaite analyser est que l'on devra toujours comparer les résultats en fonction des modèles.
En prenant en exemple les données sur le temps CPU, on souhaite comparer les temps d'exécution CPU minimum, moyens et maximum en fonction des différents modèles. Un histogramme empilé semble adapté pour visualiser ces données. En effet, un modèle peut être représenté par une barre sur laquelle on peut aisément visualiser les valeurs minimales, moyennes et maximales.

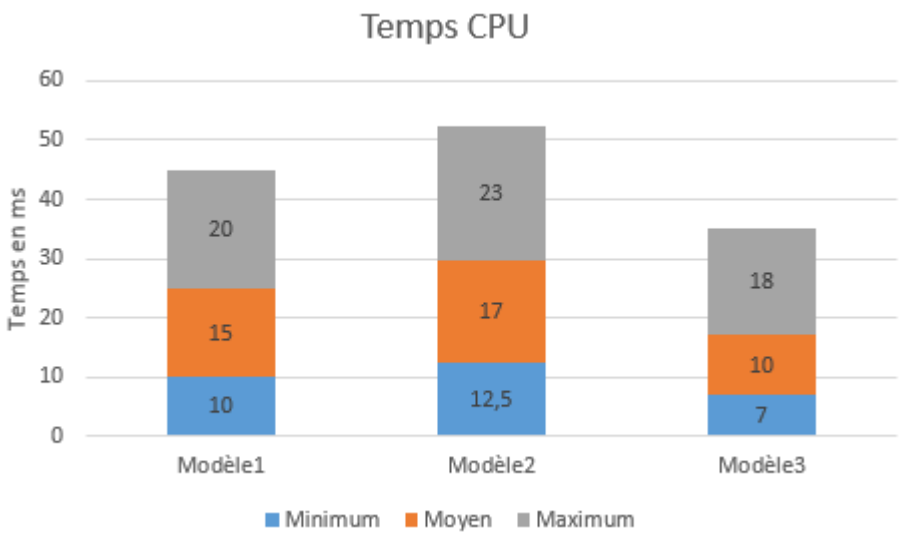


Figure 1 – *Exemple d'histogramme empilé pour la visualisation du temps CPU*

Cette représentation est également adaptée pour les autres données à analyser. Le nombre de solutions optimales trouvées sera aussi représenté par un histogramme mais qui ne sera pas empilé car il n'y aura qu'une valeur à représenter par modèle.

3 Outils de génération de graphiques

Pour déterminer l'outil le plus adapté au besoin du projet, il est nécessaire de comparer les solutions suivant plusieurs critères :

- le format des données d'entrée
- les types de graphiques proposés
- la richesse de la documentation
- la facilité d'utilisation
- le niveau de personnalisation
- les options proposées (zoom, etc...)

J'ai donc étudié plusieurs solutions parmi les plus populaires sur le web en me limitant à celles dont l'utilisation est gratuite.

3.1 Google Charts [WWW4]

L'API Google Charts est facilement utilisable en JavaScript, il suffit de lui transmettre les données que l'on souhaite représenter ainsi que le type de graphique voulu et les éventuels paramètres de personnalisation. Les données sont passées au moyen d'un DataTable, qui est une classe proposant de nombreuses méthodes pour manipuler ces données. La technologie d'affichage des graphiques assure une compatibilité avec tous les navigateurs. Cette API est très bien documentée ce qui rend son apprentissage et son utilisation facile. De plus, elle propose de nombreux types de graphiques assez simples et clairs (colonnes, courbes, histogrammes, diagrammes, tableaux, etc...) et la possibilité d'afficher le tableau des données en plus du graphique. Google Charts permet également à l'utilisateur d'interagir avec le graphique en affichant des infobulles au survol des éléments, de trier et filtrer les données par exemple.

3.2 D3.js [WWW1]

D3.js est une librairie JavaScript proposant de nombreux types de graphiques très évolués. Elle offre la possibilité de coder des objets manuellement en utilisant le langage HTML mais aussi

d'en construire automatiquement en fonction des données en utilisant JavaScript. Les données peuvent être lues à partir de sources diverses comme des fichiers CSV ou fournies sous forme de vecteurs. Il existe de nombreux tutoriels sous formes diverses permettant l'apprentissage de cette librairie. D3.js propose des interactions avec l'utilisateur mais elles semblent assez limitées. Il est également possible de personnaliser les graphiques, notamment au niveau des couleurs et des échelles.

3.3 MorrisJs [WWW6]

MorrisJs est une librairie JavaScript permettant de générer des graphiques sous forme de courbes, histogrammes et donuts incluant des infobulles au survol de la souris. Ils sont très peu personnalisables mais sont, en contrepartie, faciles à concevoir. Étant donné la simplicité d'utilisation et le peu de choix possibles au niveau des types de graphiques, la documentation est très limitée.

3.4 HighCharts [WWW5]

HighCharts propose de nombreuses formes de graphiques allant des plus simples aux plus complètes compatibles avec tous les navigateurs. Cette API s'utilise avec JavaScript et les données sont transmises sous la forme de vecteurs. Les graphiques sont personnalisables grâce à l'utilisation de thèmes permettant de modifier les couleurs. L'utilisateur à la possibilité de masquer ou afficher certaines données et d'afficher des infobulles au survol de la souris. Enfin, la documentation est très complète et propose de nombreux exemples.

3.5 FlotR2 [WWW3]

L'API FlotR2 propose les types de graphiques les plus courants avec quelques options de personnalisation. Elle ne permet pas d'interaction avec l'utilisateur comme les infobulles et les valeurs ne sont pas toujours affichages sur les graphiques. La documentation est présente mais manque d'explications.

3.6 Elycharts [WWW2]

ElyCharts permet de créer de graphiques assez complets sous les principales formes. Ils sont facilement personnalisables au niveau des couleurs, des tailles, des infobulles au survol de la souris. La documentation est simple mais suffisante à la compréhension.

4 Conclusion

Étant donné que l'histogramme empilé semble être le type de représentation le plus adapté, nous allons restreindre notre choix aux bibliothèques capables de générer ce graphique, à savoir : Google Charts, HighCharts, D3.js et FlotR2.
Parmi celles-ci, FlotR2 offre moins de possibilités de personnalisation et D3.js semble plus compliquée à prendre en main. Le choix se fera donc entre HighCharts et Google Charts.
Enfin Google Charts offre un avantage considérable sur HighCharts, elle permet de gérer les tableaux associés aux graphiques et de créer des tableaux de bord qui permettent de manipuler dynamiquement les données associées. Nous utiliseront donc Google Charts pour visualiser les résultats des tests.

5 Analyse et conception

1 Application bureau VS application web

L'objectif de cette section est d'étudier les possibilités techniques offertes par chaque solution afin de déterminer celle qui sera le plus adaptée pour répondre aux besoins du client. Pour faire ce choix, il est donc nécessaire de lister les avantages et inconvénients de chaque solution.

1.1 Application bureau

Avantages

Une application bureau peut fonctionner sans connexion internet et ne nécessite donc pas de configurer un serveur.

Inconvénients

Les applications bureau sont dépendantes du système d'exploitation et de ses configurations ce qui peut impliquer des problèmes de droits d'accès aux fichiers. De plus, l'application devrait être déployée sur chaque poste client, la maintenance serai compliquée car il faudrait mettre à jour tous les postes. Enfin, les tests exécutés sur des postes différents ne pourraient pas comparés dans le cas où les machines auraient des configurations matérielles différentes.

1.2 Application web

Avantages

Une applications web utilise les ressources du serveur, cela garanti les même performances pour tous les tests lancés depuis n'importe quel poste client. De plus, les résultats des tests seraient centralisés sur le serveur et donc accessibles de n'importe où. La maintenance d'une application web est simplifiée car il suffit de mettre l'application à jour sur le serveur. Enfin, le logiciel devant utiliser Cplex, il n'y aura pas de problèmes de configuration, version, ou installations différentes suivant les postes car Cplex sera directement installé sur le serveur.

Inconvénients

Une application web nécessite de disposer d'un serveur web configuré et d'une connexion internet pour y accéder. De plus, il est nécessaire de prendre en compte l'existence des différents navigateurs et de leur spécificités.

Conclusion

Les deux solutions proposées proposent chacune leurs avantages et inconvénients. Cependant, étant donné les arguments précédemment énoncés, une application web semble la plus adaptée pour répondre aux besoins des utilisateurs.

2 Choix du langage de développement

L'objectif de cette section est de déterminer le langage de développement le plus adapté à la réalisation de cette solution. Étant donné le temps limité dont je dispose pour mener à bien ce projet, j'ai décidé de me restreindre aux technologies que je maitrise (Java ou PHP) car j'estime ne pas avoir suffisamment de temps pour me former à une nouvelle technologie. J'ai donc effectué des recherches sur les spécificités de ces deux langages pour choisir celui qui sera le plus adapté.

2.1 Java EE

Le langage Java est un langage structuré orienté objet permettant la structuration selon le modèle MVC. Il offre la possibilité de générer des interfaces graphiques grâce à JavaFX et de gérer la persistance des données avec Hbernate. De plus, ma connaissance de JEE facilitera le développement de l'application. Enfin, Java offre des fonctionnalités de gestions des thread très complètes.

2.2 PHP

Le langage PHP est le plus couramment utilisé dans le monde du web. De plus, la mise en place d'un serveur web pour une application PHP est plus simple et généralement plus courante chez les hébergeurs que pour une application développée en Java. Cependant, la conception de fonctionnalités avancées nécessite l'utilisation de frameworks que je ne maitrise pas.

2.3 Conclusion

Étant donnée mes compétences personnelles et les fonctionnalités proposées par ces deux langages, l'utilisation de Java EE me permettra plus facilement de fournir une application fonctionnelle à la fin de ce projet.

3 Structure de l'application

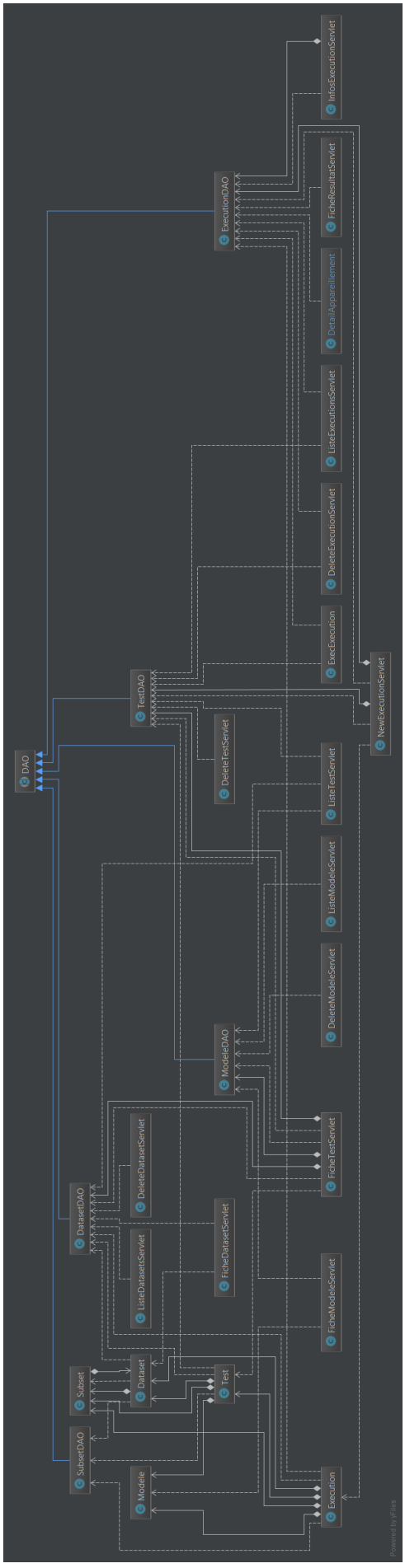


Figure 1 – *Diagramme de classes simplifié de l'application*

Ce diagramme permet d'avoir une vue globale des interactions entre les classes des différents packages (Modele, Servlet, DAO). Les classes du package tools ne sont pas visibles sur le diagramme. Dans ce package, on trouve la classe DatabaseConnexion qui utilisée par les classes du package DAO pour la connexion à la base de données, la classe Executor qui permet d'exécuter les threads de la classe Execution et enfin la classe Fichier qui contient de nombreuse méthodes de manipulation de fichiers.
Les classes de l'application sont expliquées plus en détail dans la partie Architecture de l'application de la documentation technique.

4 Gestion des données

Les données seront gérées et sauvegardées grâce au framework Hibernate qui permet de gérer la persistance des données en réalisant un mapping avec les classes. (voir Schéma de la base de données).
Parmi les données à gérer on retrouve les modèles, les collections de graphes, les tests et les résultats. Afin de pouvoir conserver l'intégrité relationnelle entre les objets, aucune donnée ne sera supprimée du système, elle seront masquées pour l'affichage mais toujours présentes dans la base de données.

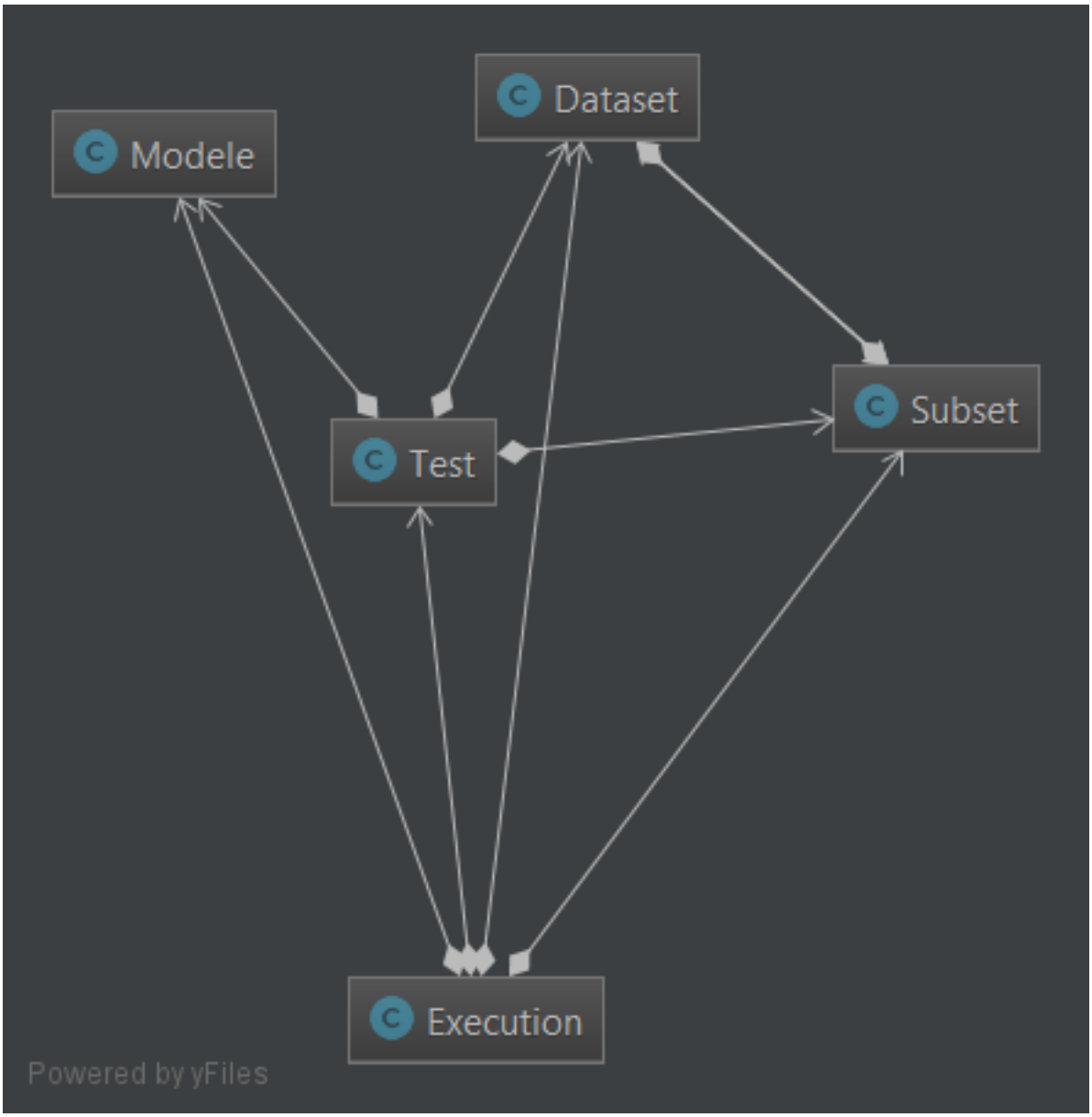


Figure 2 – *Diagramme d'objets*

Ce diagramme montre la relation entre les différents objets. On voit qu'un test contient au minimum deux modèles et au minimum un dataset (ou subset) et qu'un modèle ou dataset (ou subset) peut être lié à aucun ou plusieurs tests. On note également qu'un test peut générer plusieurs exécutions mais qu'une exécution correspond à un seul test.

En plus des objets créés, un certain nombre de fichiers devront être stockés dans le répertoire :

- Les exécutables des modèles
- Les dossiers contenant les graphes des collection et les fichier décrivant les sous-collections
- Les fichiers texte générés par Cplex lors de l'exécution des tests

Ces fichiers seront organisés selon l'arborescence suivante :

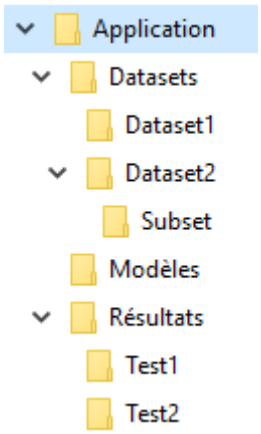


Figure 3 – *Arborescence de l'application*

6 Mise en œuvre

1 Outils et librairies utilisées

Afin d'obtenir un affichage de qualité, plusieurs librairies ont été utilisées.

DataTables

Les tables permettant d'afficher les liste des modèles, des datasets, des tests et des exécutions sont gérées grâce à JQuery (https://jquery.com/) et le plug-in DataTables (https://datatables.net/). Cela permet d'une part de donner aux tables une apparence agréable grâce aux CSS proposés dans ce plug-in. De plus, il offre notamment la possibilité de proposer des tables dynamiques grâce à des possibilités de filtres et de tri mais également d'effectuer des sélections dans les tables grâce à des cases à cocher.

Google Charts

De plus, l'affichage des résultats est réalisé grâce à l'API Google Charts (https://developers.google.com/chart/). Elle permet de générer des tableaux et des graphiques facilement avec de nombreuses options de personnalisation. De plus, un des avantages principaux est que les tableaux et les graphiques peuvent êtres générés à partir de la même source de données.

Jstl

Enfin, la librairie jstl (https://jstl.java.net/) a été utilisée pour permettre la lecture et l'affichage des objets Java passés en paramètres aux pages jsp de manière simple et efficace.

2 Implémentation

D'un point de vue implémentation, ce projet met en œuvre de nombreuses technologies :

- Les annotations Hibernate pour faire le lien entre les objets java et la base de données.
- Le JSP pour générer des pages web directement.
- Les tags Jstl pour faciliter l'interaction entre le code Java et le JSP.
- L'utilisation de servlets pour la navigation entre les pages web et la gestion des paramètres.
- Le langage Java pour réaliser les traitements.
- Le langage JavaScript pour gérer l'interaction avec l'utilisateur au sein des pages web.

Une des particularités de cette application est qu'elle va utiliser des exécutables indépendants. Le risque est que si un exécutable rencontre une erreur et ne se termine pas correctement, le

thread ayant appelé ce dernier va se retrouver bloqué. Il faut donc s'assurer que toutes les exceptions possibles soient gérées et permettent de terminer l'exécutable.
De plus, ces exécutables font appel à Cplex qui devra donc être installé et configuré correctement sur le serveur.

3 Qualité et évaluation des performances

Documentation et nommage

Un code source de qualité doit être facilement maintenable. Pour cela, les variables sont nommées de manière explicite. De plus, chaque méthode a été commentée de manière à fournir une Javadoc complète. Des commentaires supplémentaires ont été insérés pour expliquer les passages les plus complexes. En complément, une Documentation technique détaillée a été rédigée de manière a expliquer la structure de l'application (packages, classes, base de données), les dépendances, l'environnement d'exécution et la démarche pour effectuer une mise à jour.

SonarQube

Afin d'améliorer le code produit, j'ai utilisé le plugin SonarQube (`https://www.sonarqube.org/`) qui permet de détecter de nombreux défauts comme les portions de code mort, le code dupliqué, les exceptions non gérées, etc...

Tests

Enfin, le code a été testée de plusieurs manières. Des test unitaire ont été rédigés (pas pour toutes les classes par manque de temps) afin de vérifier le bon fonctionnement des classes. Ensuite, des tests fonctionnels ont été réalisés sur l'environnement de production afin de vérifier que les fonctionnalités développées fonctionnent comme prévu.

7 Bilan et conclusion

1 Semestre recherche et conception

Au cours de ce premier semestre, j'ai été amené à réaliser les tâches suivantes :

- Réalisation d'un état de l'art sur les méthodes de graph matching et des solutions web de représentation graphiques
- Analyse du besoin du client et choix d'une solution adaptée
- Rédaction des spécifications de l'application et conception des maquettes d'IHM

1.1 Planning prévisionnel

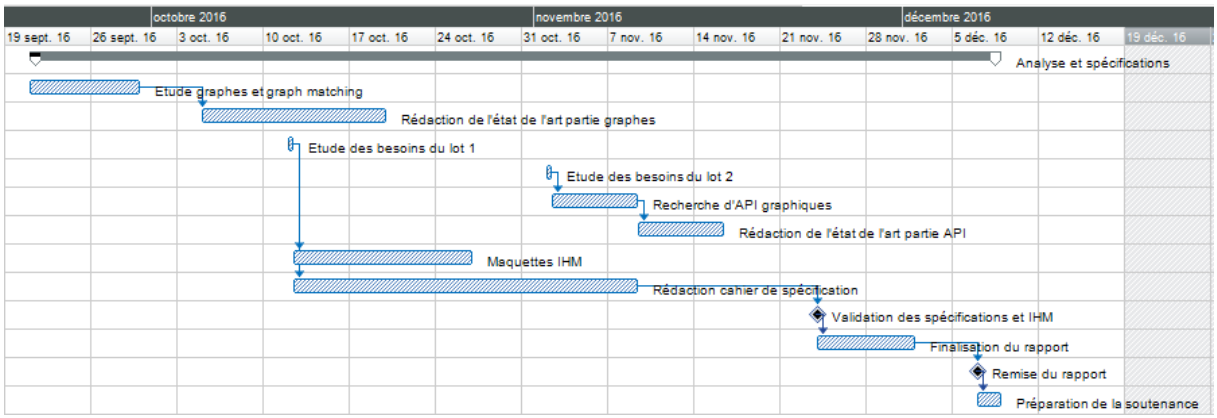


Figure 1 – *Diagramme de Gantt prévisionnel du semestre recherche et développement*

Ce diagramme de Gantt montre l'organisation que j'avais prévue pour ce premier semestre. J'ai commencé par étudier des articles sur le graph matching. Bien que je n'aurais pas à mettre en œuvre ces méthodes car elles sont déjà implémentées, cette première étape était néanmoins indispensable pour comprendre le contexte du projet. Ensuite, j'ai commencé à rédiger l'état de l'art et à concevoir les maquettes des IHM et rédiger le cahier des spécifications pour la première partie en parallèle. Ces tâches en parallèle m'ont permis, d'une part de varier les activités et donc de rester productif et motivé et d'autre part de ne pas rester bloqué sur des questions jusqu'à la prochaine réunion avec mon encadrant car j'avais toujours une autre activité sur laquelle je pouvais avancer. Une fois ces tâches bien avancées, j'ai étudié les besoins de la partie 2 et effectué un état de l'art sur les solutions d'affichage graphique de données.

1.2 Planning effectif

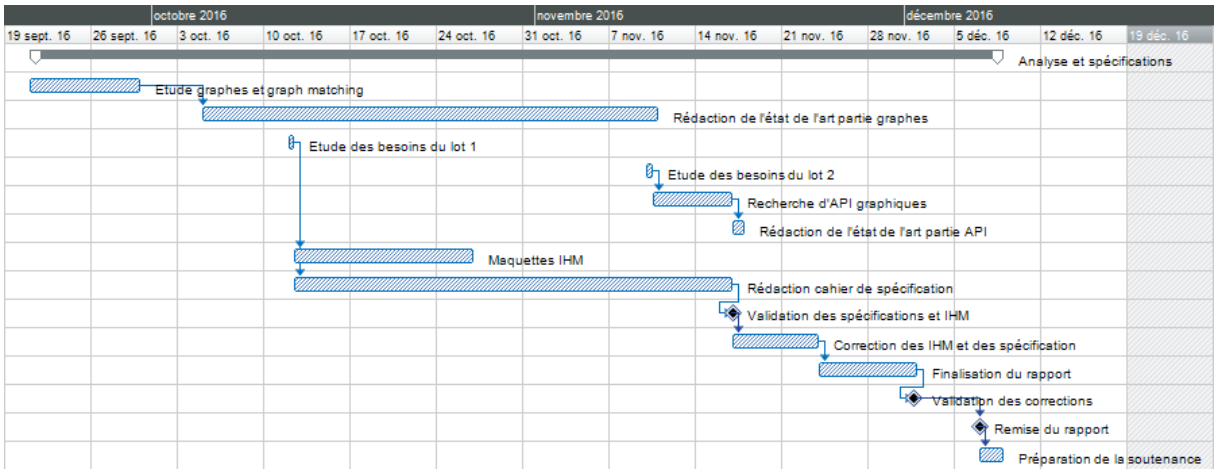


Figure 2 – *Diagramme de Gantt du semestre recherche et développement*

Ce second diagramme présente l'organisation réelle des tâches, il met en évidence certaines différences avec ce qui était prévu :

- La rédaction de l'état de l'art sur le graphe matching a été beaucoup plus longue car je n'avais pas pris en compte la partie sur les graphes ni les définitions sous forme mathématiques. Cela à donc retardé l'étude des besoins de la partie 2.
- La rédaction du cahier des spécifications m'a également pris un peu plus de temps car j'ai dut prendre un temps de réflexion pour certaines parties.
- En revanche, la rédaction de l'état de l'art sur la visualisation de données a été plus rapide que ce que je pensais. Cela m'a permis de compenser le retard pris sur le début de cette tâche.
- Enfin, j'ai dut rajouter une phase de correction des IHM et des spécifications et un jalon de validation de corrections que je n'avais pas explicitement prévue dans le planning prévisionnel.

J'avais initialement prévu un écart entre la fin des tâches de rédaction et le jalon de validation des IHM et spécifications afin de laisser le temps à mon encadrant de les étudier et de compenser un éventuel retard pris sur les tâches précédentes. Sur les deux semaines prévue, une a été utilisée pour rattraper mon retard. J'ai donc pu soumettre mes productions à mon encadrant une semaine plus tôt, cela ma donc permis de rajouter une tâche de correction des IHM et des spécifications que je n'avais pas prévue et de finaliser mon rapport dans les délais prévus.

1.3 Conclusion

Malgré quelques retards sur les tâches de rédaction qui sont dus en partie à ma non-maîtrise de Latex, j'ai pu mener à terme toutes les tâches prévues. Cependant, j'ai manqué de temps pour réaliser l'analyse de l'application, notamment sur la seconde partie de l'application qui concerne la gestion des threads chargés de l'exécution des tests. Je devrai donc consacrer un peu de temps au début du sprint 4 pour approfondir cette analyse.
A la fin de ce premier semestre, j'ai donc bien compris le problème du graph matching et ses applications ce qui m'a permis de bien cerner le contexte et les objectifs de ce projet. De plus, j'ai une idée assez précise de l'application que je vais devoir développer même si certains aspects techniques devront être approfondis.

2 Semestre développement

2.1 Planning prévisionnel

Ce diagramme de Gantt présente le planning prévu pour le développement de l'application. Il a été établi avant le début de la phase de développement.

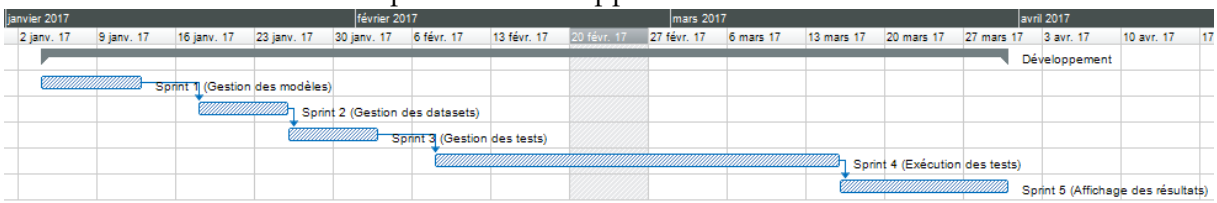


Figure 3 – *Diagramme de Gantt prévisionnel du semestre développement*

2.1.1 Sprint 1

Ce sprint correspond à la réalisation du menu et des interfaces et des classes nécessaires à la gestion des modèles. La documentation utilisateur concernant ces interfaces sera rédigée en parallèle. A l'issue de ce sprint, le client recevra donc les deux interfaces liées aux modèles et la documentation associée.
Bien que les fonctionnalités de ce sprint ne soient pas plus longues à développer que celles du sprint 2, sa durée sera un peu plus longue car il inclut le temps nécessaire à la mise en place du projet et à l'acquisition ou à la remise à niveau de certaines compétences nécessaires au développement.

Liste des tâches :

- Création du projet et mise en place d'Hibernate et de TomCat
- Création du menu principal
- Développement des classes Modele et ModeleDAO
- Conception des IHM Modèle et ListeModèle
- Conception des tests unitaires associés
- Rédaction de la documentation

2.1.2 Sprint 2

Ce sprint correspond à la réalisation des interfaces et des classes nécessaires à la gestion des collections de graphes. La documentation utilisateur concernant ces interfaces sera rédigée en parallèle. A l'issue de ce sprint, le client recevra donc les deux interfaces liées aux collections de graphes et la documentation associée.

Liste des tâches :

- Développement des classes Dataset et DatasetDAO
- Conception des IHM Dataset et ListeDatasets
- Conception des tests unitaires associés
- Rédaction de la documentation

2.1.3 Sprint 3

Ce sprint correspond à la réalisation des interfaces et des classes nécessaires à la gestion des tests. A l'issue de ce sprint, le client recevra donc les deux interfaces liées aux tests et la

documentation associée.

Liste des tâches :

- Développement des classes Test et TestDAO
- Conception des IHM Test et ListeTests
- Conception des tests unitaires associés
- Rédaction de la documentation

2.1.4 Sprint 4

Dans ce sprint, la gestion des threads qui vont exécuter les tests et les méthodes permettant de mesurer les performances et l'interprétation des résultats fournis par Cplex seront développées.

Liste des tâches :

- Développement des classes ExecutionManager, ExecutionTest et Résultat
- Conception des tests unitaires associés
- Rédaction de la documentation

2.1.5 Sprint 5

Ce sprint correspond à la réalisation des interfaces et les classes permettant de consulter les résultats des tests. A l'issue de ce sprint, le client recevra une application fonctionnelle contenant toutes les fonctionnalités principales ainsi que la documentation utilisateur complète.

Liste des tâches :

- Conception des IHM Résultat et ListeRésultats
- Conception des tests unitaires associés
- Rédaction de la documentation

2.2 Planning effectif

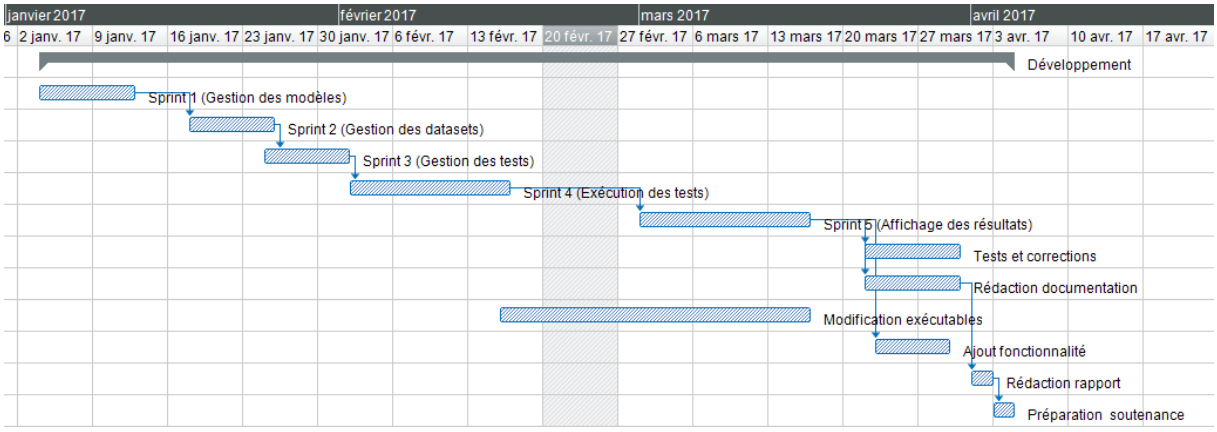


Figure 4 – *Diagramme de Gantt du semestre développement*

On remarque que les trois premiers sprints ont été réalisés dans le temps prévu, avec même un jour d'avance. Contrairement à ce qui été prévu, la documentation n'a pas été rédigée en parallèle mais à la fin de la phase de développement. Ce choix a été fait combler le retard du à la prise en main des technologies mises en œuvre. On note également que le sprint 3 a été commencé avant la fin du sprint 2 car j'avais besoin de précisions de la part du client pour le terminer. J'ai donc pris la décision de mettre en suspend le sprint 2 pour ne pas partir sur de fausses piste et de commencer le sprint 3 pour ne pas perdre de temps.

J'ai donc pu commencer le sprint 4 avec un jour d'avance. Il a d'ailleurs été plus court que prévu puisqu'il a été développé en 5 jours au lieu des 9 prévus. Cela est du au fait que la difficulté technique était moins importance que ce que j'avais prévu. De plus, la modification des exécutables était initialement prévue dans ce sprint mais fait finalement l'objet d'une tâche séparée car ces modifications ont été diluées dans le temps. Ce choix a été fait car j'ai eut besoin de temps et de l'aide de mon encadrant pour comprendre ces programmes. J'ai donc dans un premier temps effectué les modifications minimales sur un exécutable afin de pouvoir continuer le développement de l'application. Par la suite, les autre exécutables ont été modifiés.
Le sprint 5 a donc pu commencer avec 4 jours d'avance mais a duré un jour de plus que prévu car l'exploitation des résultats a été plus complexe que prévue. Grâce à l'avance prise au sprint précédent, ce sprint a été terminé 4 jours en avance.
L'avance prise sur le planning prévu m'a permis de rédiger la documentation qui devait être faite au fur et à mesure. Cela m'a également dégagé du temps pour réaliser des tests et effectuer des corrections. J'ai même pu ajouter une nouvelle fonctionnalité qui avait été mise de coté pour être sur de tenir les délais de développement.

2.3 Conclusion

Globalement, le développement de l'application a été réalisé dans le temps imparti. L'organisation réelle a été légèrement modifiée car, d'une part, les difficultés techniques n'étaient pas forcement là où je les attendait, et d'autre part, car j'avais parfois besoin de précisions de la part du client.
Afin de tenir les délais et en accord avec le client, certaines fonctionnalités ont été simplifiées. Par exemple, les fichiers compressés sont uniquement gérés au format zip, le format rar n'est pas géré comme il était prévu initialement.
La livraison des lots n'apparait pas sur le planning car les livraisons ne se faisaient pas a date fixe. L'application était mise à jour sur le serveur régulièrement, cela me permettait d'effectuer des tests sur l'environnement de production et permettait au client de voir l'avancement du projet au jour le jour.

Annexes

A Description des interfaces externes du logiciel

1 Interfaces matériel/logiciel

Dans un premier temps, l'application sera hébergée sur le poste client qui devra donc disposer d'un serveur web TomCat, de Cplex et d'un navigateur. Par la suite, l'application pourra être hébergée sur un serveur de l'université où Cplex sera installé. Dans ce cas, le poste client devra disposer d'une connexion internet et d'un navigateur.

2 Interfaces homme/machine

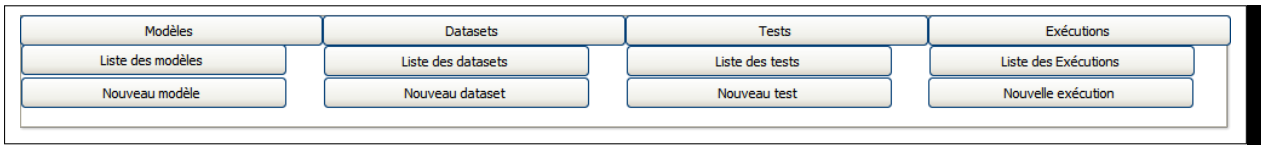


Figure 1 – *Menu de l'application*

Ce menu sera intégré à toutes les interfaces de l'application. Il permet de naviguer entre les différentes pages. Il est constitué de quatre menus déroulants regroupant chacun l'accès à la liste et à la page de création d'un objet correspondant (modèle, collection, test, exécution).

Figure 2 – *Interface d'édition d'un modèle*

L'utilisateur pourra saisir et modifier les informations relatives à un modèle. Lors d'une création, l'exécutable associé au modèle sera copié dans le répertoire de l'application.

Figure 3 – *Interface de liste des modèles*

L'utilisateur pourra visualiser la liste des modèles existants qui permettra d'accéder à la modification du modèle sélectionné. Cette liste pourra être filtrée par nom et description.

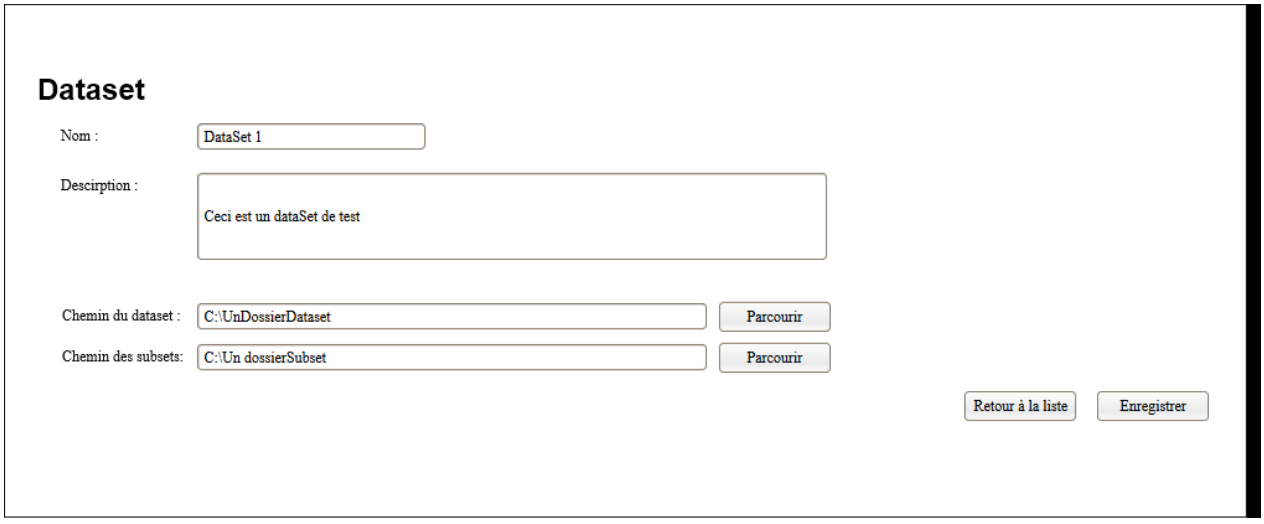


Figure 4 – *Interface d'édition d'une collection de graphes*

L'utilisateur pourra saisir et modifier les informations et paramètres relatifs à une collection de graphes. Lors d'une création, la collection de graphes sera copiée dans le répertoire de l'application. Avant la copie, cette collection sera compressée si elle ne l'est pas déjà afin de réduire les temps de transfert.

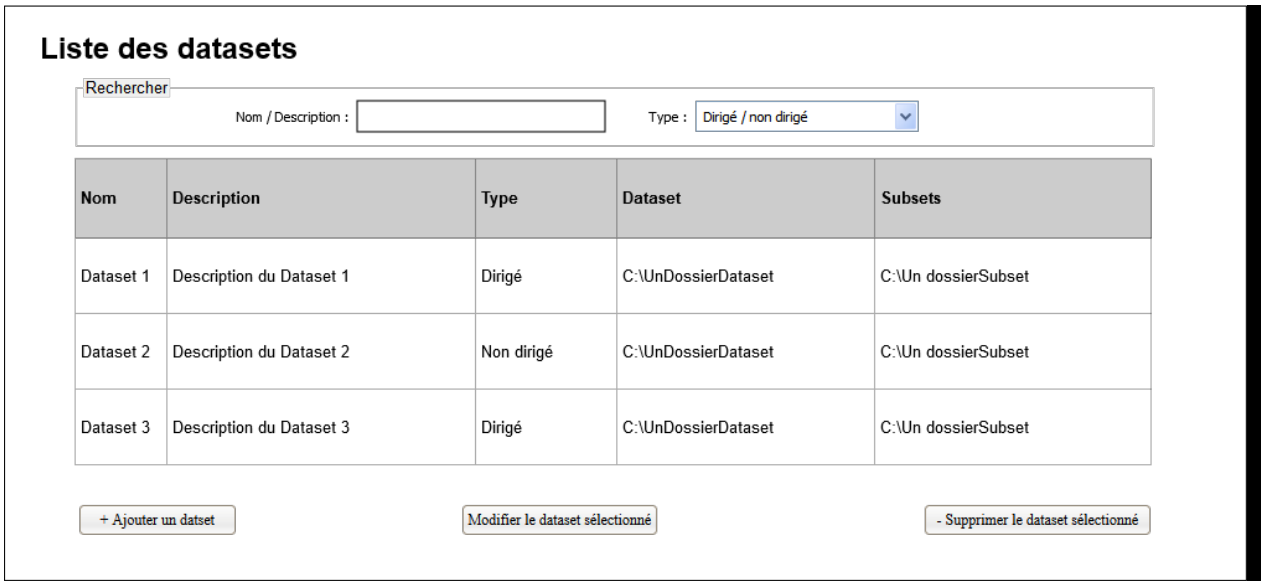


Figure 5 – *Interface de liste des collections de graphes*

L'utilisateur pourra visualiser la liste des collections existantes qui permettra d'accéder à la modification de la collection sélectionnée. Cette liste pourra être filtrée par nom, description et type de graphes.

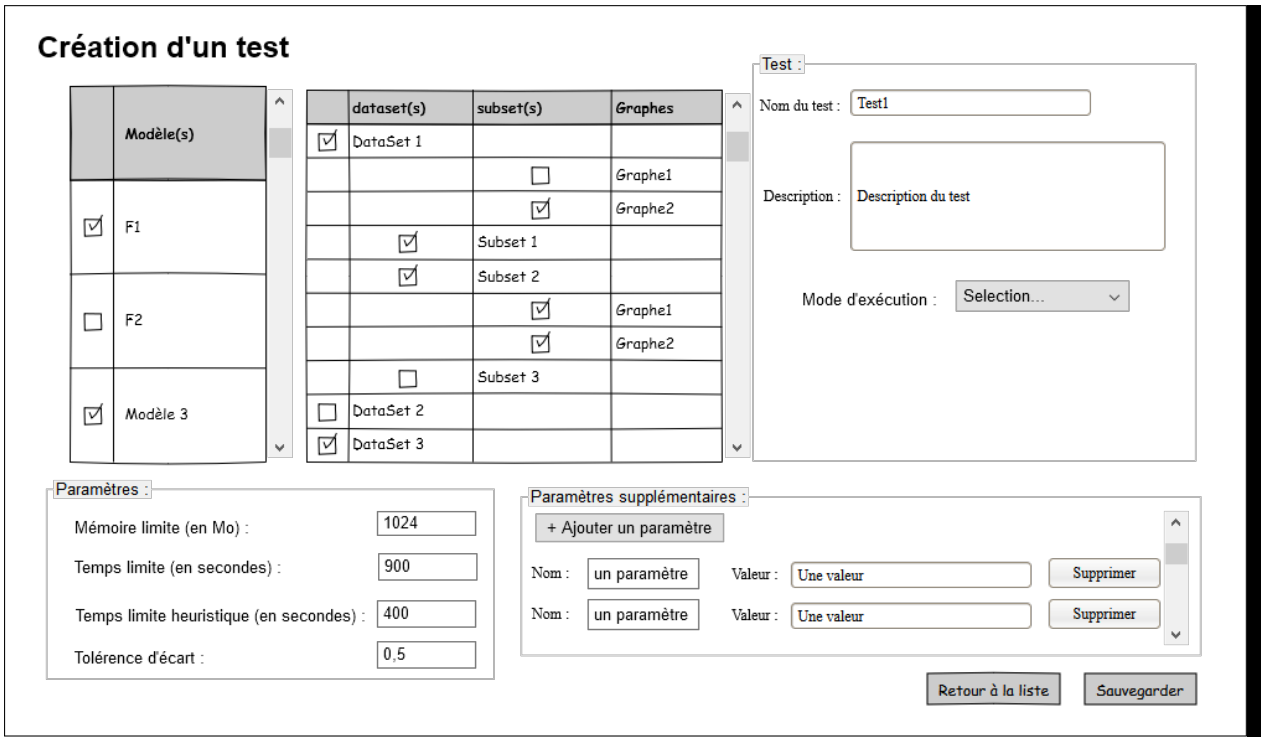


Figure 6 – *Interface de gestion des tests*

L'utilisateur pourra sélectionner les modèles et la ou les collection(s) de graphes parmi la liste de modèles / collections de graphes existants et enregistrer le test après avoir renseigné différents paramètres s'il le souhaite. Il existe des paramètres prédéfinis mais l'utilisateur aura également la possibilité d'en définir de nouveaux.

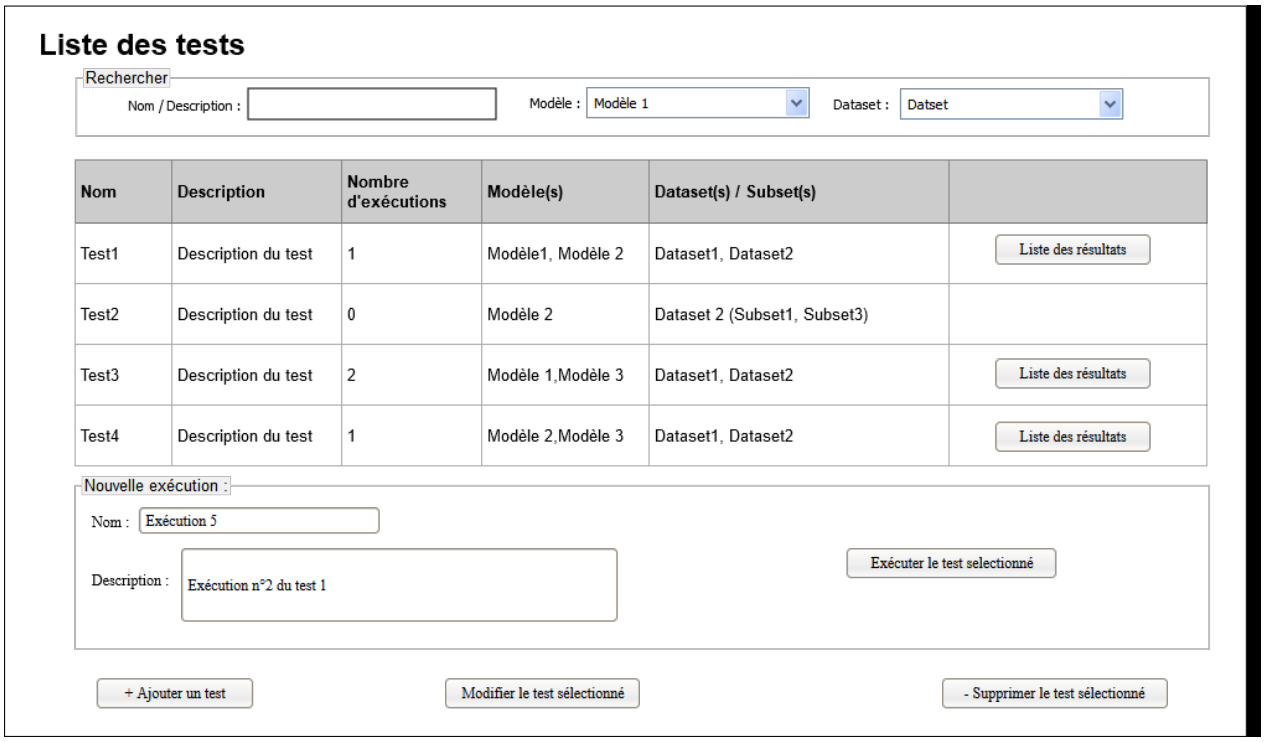


Figure 7 – *Interface de liste des tests*

L'utilisateur pourra consulter la liste des tests effectués qui permettra de lancer une nouvelle exécution ou d'accéder à la liste des exécutions du test sélectionné. Cette liste pourra être filtrée par nom et description.

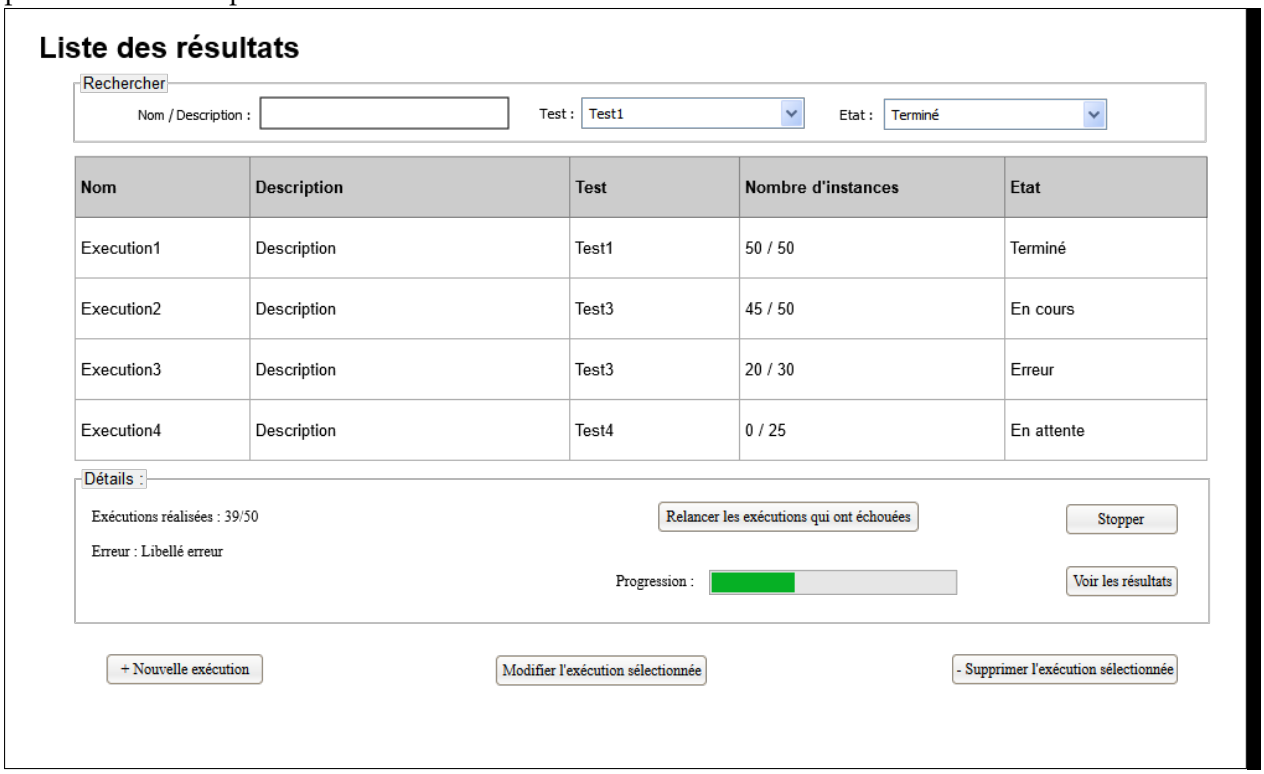


Figure 8 – *Interface de liste des exécutions*

L'utilisateur pourra consulter la liste des tests exécutés qui permettra d'accéder à la page des résultats correspondante. Cette interface donnera également des informations sur l'état du test sélectionné comme son statut, les éventuelles erreurs relatives à son exécution, le nombre

d'instances réalisées.

Figure 9 – *Interface de consultation des résultats*

L'utilisateur pourra consulter les résultats de l'exécution sélectionnée sous forme de graphiques. Elle comporte plusieurs onglets correspondant aux différents types d'information :

- Général : rappel les modèles, les graphes et les paramètres associés au test, nombre de solutions optimales trouvées.
- Temps CPU : temps CPU minimum, moyen et maximum.
- Noeuds explorés : informations sur le nombre de noeuds explorés par Cplex.
- Instances traitées : information sur le nombres d'instances traitées.
- Pourcentage de pré-processing : taux de variables pré-calculées.
- Écart bornes : écart entre les bornes minimales et maximales utilisées par Cplex.
- les appareillements entre les noeuds et entre les arcs.

3 Interfaces logiciel/logiciel

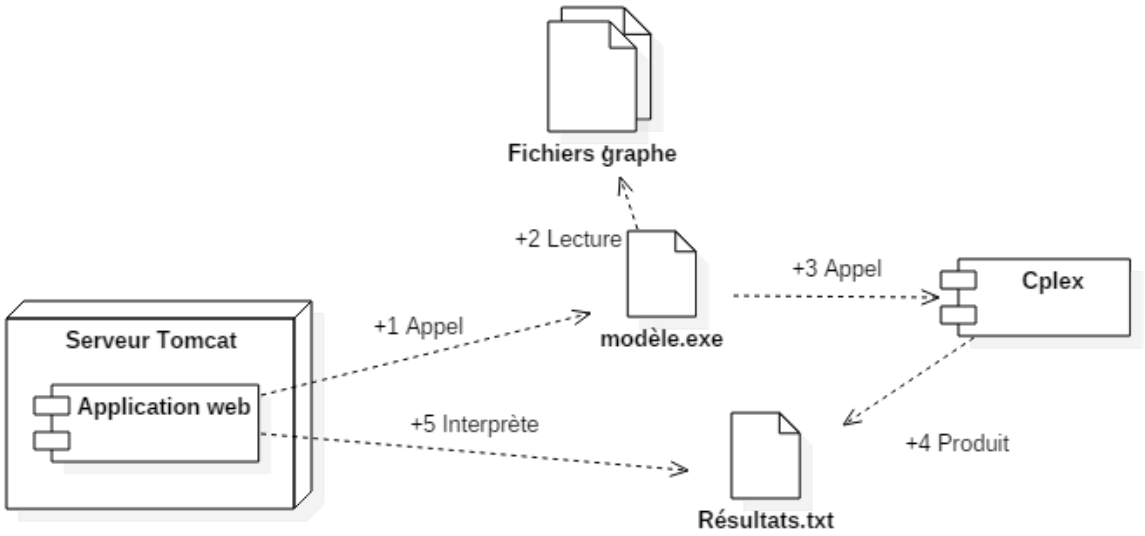


Figure 10 – *Diagramme de composants*

L'application devra être en mesure de faire appel à des exécutables situés sur le serveur (modèles) pour chaque paire de graphes parmis la liste associée au test. Ces exécutables feront eux-même appel au logiciel Cplex (qui sera installé sur le même poste que l'application). Ensuite l'application exploitera les résultats retournés sous forme de fichiers texte par Cplex.

B Spécifications fonctionnelles

1 Ajout / modification d'un modèle

Pré-condition : Disposer de l'exécutable du modèle
Entrées : nom, description et chemin de l'exécutable du modèle
Sortie : Un objet Modèle

L'ajout d'un modèle consiste à mettre en mémoire le nom, la description, le chemin de l'exécutable. Lors de la création d'un modèle, l'exécutable sera copié dans le répertoire de l'application. Le modèle créé sera ensuite accessible depuis la liste des modèles.

2 Ajout / modification d'une collection de graphes

Pré-condition : Disposer de la collection de graphes Entrées : nom, description et chemin de l'exécutable du répertoire contenant la collection
Sortie : Un objet Dataset

L'ajout d'une collection de graphes consiste à mettre en mémoire le nom, la description, le chemin du répertoire contenant la collection, éventuellement le chemin du répertoire contenant les fichiers qui définissent les sous-ensembles de graphes. La collection créée sera ensuite compressée et copié dans le répertoire de l'application puis sera accessible depuis la liste des collections.

3 Ajout d'un test

Pré-condition : Avoir au moins 2 objets Modèle et 1 objet Dataset existants
Entrées : nom, description, liste d'objet Modèle, liste d'objet Dataset, liste de paramètres
Sortie : Un objet Test, un fichier texte contenant les paramètres

Pour lancer un test, l'utilisateur devra sélectionner un modèle et une ou plusieurs collection(s) ou sous-ensemble(s) de graphes et renseigner un nom, une description permettant d'identifier le test et éventuellement des paramètres associés qui seront stockés dans un fichier texte. Bien qu'il existe des paramètres prédéfinis, l'utilisateur à la possibilité d'ajouter des paramètres supplémentaires caractérisés par un nom et une valeur.

4 Lancement d'un test

Pré-condition : Avoir 1 objet Test existant
Entrées : nom, description, objet Test
Sortie : Un objet Execution

Un thread fera appel à l'exécutable associé au modèle en lui passant le fichier contenant les paramètres s'il existe, pour chaque paire de graphes. L'exécutable fera lui même appel à Cplex qui fournira un résultat sous forme de fichier texte qui sera stocké dans un répertoire du système. Enfin, le système analysera ce fichier et mesurera les performances (comme le temps d'exécution CPU par exemple) pour restituer les résultats par la suite sous forme graphique.

5 Consultation des résultats

Pré-condition : Avoir 1 objet Execution existant dans un état Terminé
Entrées : objet Execution, le fichier texte des résultats produit par Cplex
Sortie : Affichages divers

L'utilisateur pourra consulter la liste des tests qui ont été exécutés. Cette liste l'informera de l'état du test (en cours, terminé, échec). Lorsqu'un test sera terminé, l'utilisateur pourra depuis cette liste accéder à la visualisation des résultats.

C Gestion de projet

1 Choix de la méthode

Ce projet sera conçu suivant une méthode agile. En plus de la forte disponibilité du client, ce choix est motivé par les principales raisons suivantes :

Définition des besoins

Le client ayant exprimé des besoins généraux, ils devront être précisés par la suite. C'est pour cela que des livraisons régulières de fonctionnalités permettront au client de voir si les productions correspondent ou non à ses attentes et donc d'avoir une idée plus précise de ce qu'il souhaite vraiment. Dans l'autre sens, cela évite de développer une application basée sur des besoins pas assez précis qui ne satisfera pas le client et de devoir tout reprendre après la livraison finale.

Fonctionnalités séparables en lots

Ce projet est composé de deux grandes parties :

1. Interfaces de paramétrage (qui peuvent être livrées séparément car elles sont indépendantes)
2. Lancement des tests, interprétation et restitution des résultats

De plus, il peut être facilement séparé en lots selon la décomposition suivante :

1. Gestion des modèles (formulaire d'édition + liste)
2. Gestion des collections de graphes (formulaire d'édition + liste)
3. Gestion des tests (formulaire d'édition + liste)
4. Exécution des tests et gestion des threads
5. liste des exécutions + Restitution des résultats

Chaque lot correspondra à un sprint et sera livrée à l'issue de ce dernier.

2 Outils utilisés

— Le logiciel MindView sera utilisé pour la planification des tâches et la création du diagramme de Gantt. Il permettra de mesurer l'avancement du projet.
— L'application sera développée avec l'IDE IntelliJ.
— La gestion des sources sera faite grâce à Git.
— Les diagrammes UML ont été créés avec StarULM.
— Les maquettes d'IHM ont été réalisées avec Pencil.

D Documentation utilisateur

1 Introduction

L'objectif de cette application web est de permettre aux utilisateurs de tester des modèles mathématiques de comparaison de graphes. Pour cela, l'utilisateur peut gérer des modèles mathématiques et des collections de graphes, exécuter des tests et consulter les résultats. L'application est actuellement installée sur une machine virtuelle de l'école et est accessible à l'adresse suivante : `10.108.99.40\GraphMatching`.

2 Menu principal

Le menu principal divisé en 4 partie correspondant chacune à un type d'objets : Modèles, Datasets, Tests, Exécutions. Chaque partie du menu permet d'accéder à la liste des objets existants ou à un formulaire pour ajouter un nouvel objet.

3 Gestion des modèles

3.1 Liste des modèles

La liste des modèles présente les modèles existants dans l'application. On peut voir leur nom, leur description et le chemin de l'exécutable. Il est possible de filtrer la liste en fonction du contenu du nom et de la description. Un double clic sur un modèle dans le tableau permet d'accéder à la fiche du modèle. Un bouton permet également d'ouvrir une fiche vierge pour ajouter un nouveau modèle.

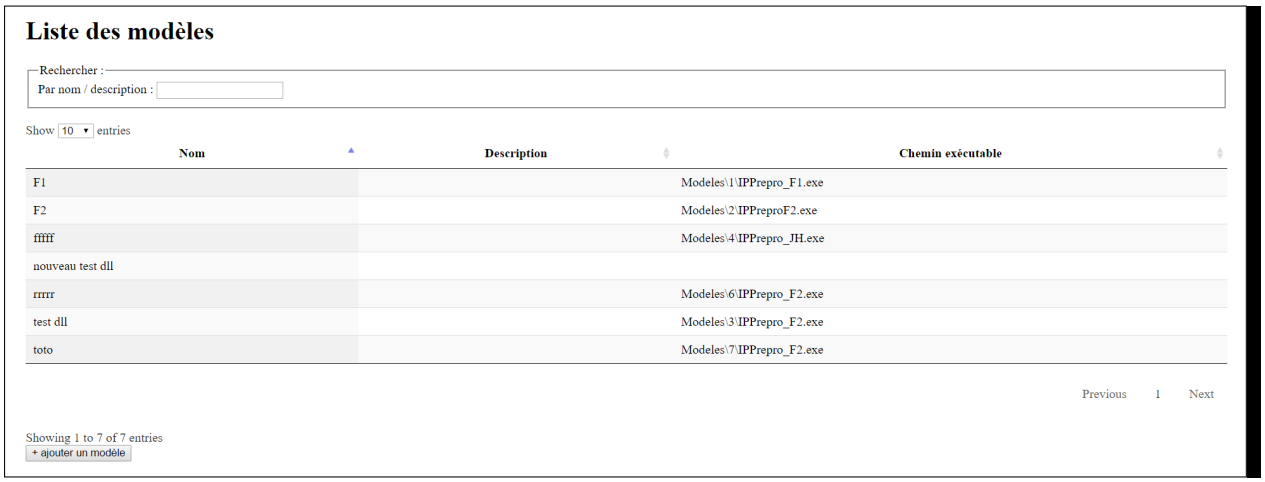


Figure 1 – *Liste des modèles*

3.2 Fiche modèle

Pour ajouter un modèle, l'utilisateur doit obligatoirement saisir un nom et sélectionner un fichier correspondant à l'exécutable du modèle au format .exe. Il peut également renseigner une description. L'utilisateur peut à tout modifier un modèle à tout moment. Depuis cette page, l'utilisateur peut enregistrer les modifications de la fiche, retourner à la liste des modèles, ouvrir une fiche vierge et masquer le modèle en cours (dans ce cas, le modèle ne sera plus visible par l'utilisateur mais toujours présent dans les données de l'application).

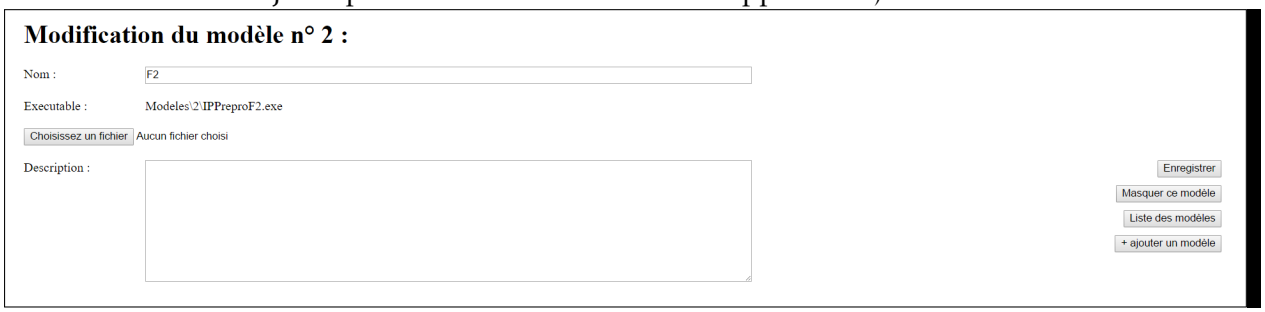


Figure 2 – *Fiche modèle*

4 Gestion des collections de graphes

4.1 Liste des collections de graphes

La liste des datasets présente les collections de graphes existantes dans l'application. On peut voir leur nom, leur description, s'il s'agit de graphes dirigés ou non, le nombre de graphes de la collection, le chemin du dossier contenant les graphes et le chemin du dossier contenant les sous-collections de graphes. Il est possible de filtrer la liste en fonction du contenu du nom et de la description et en fonction du type de graphe (dirigé ou non). Un double clic sur un dataset dans le tableau permet d'accéder à la fiche de la collection de graphes. Un bouton permet également d'ouvrir une fiche vierge pour ajouter une nouvelle collection.

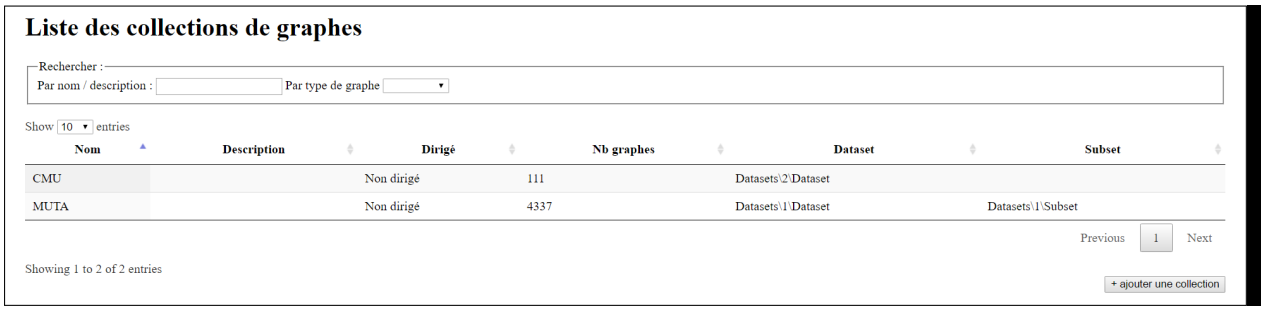


Figure 3 – *Liste des collections de graphes*

4.2 Fiche collection de graphes

Pour ajouter une collection de graphes, l'utilisateur doit obligatoirement renseigner un nom et sélectionner le dossier contenant les fichiers graphe préalablement compressé au format .zip. Les fichier graphes doivent être au format .dat ou .gxl. Les fichiers.gxl seront convertis en .dat par l'application. L'utilisateur peut également renseigner une description et sélectionner un dossier au format .zip contenant les fichiers de sous-collection de graphes. L'utilisateur peut à tout modifier une collection à tout moment. Depuis cette page, l'utilisateur peut enregistrer les modifications de la fiche, retourner à la liste des collections, ouvrir une fiche vierge et masquer la collection en cours (dans ce cas, elle ne sera plus visible par l'utilisateur mais toujours présent dans les données de l'application).

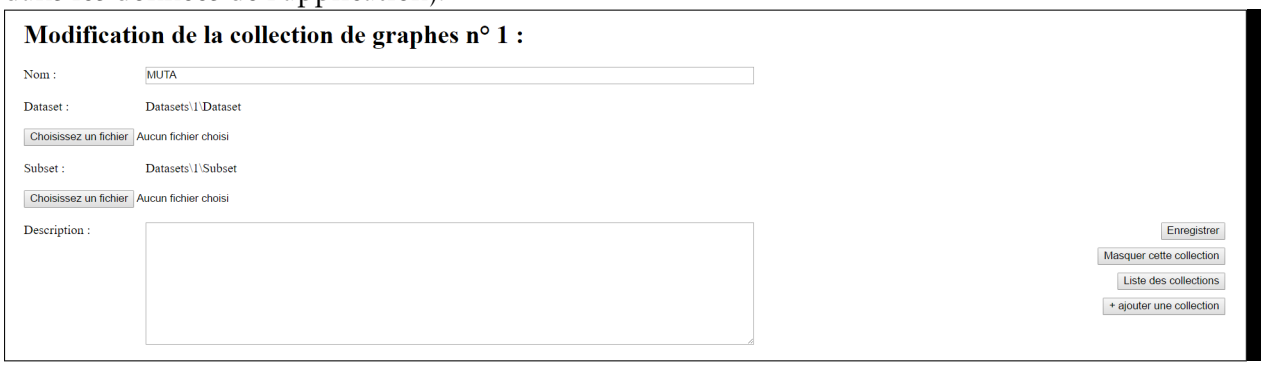


Figure 4 – *Fiche collection de graphes*

5 Gestion des tests

Un test correspond à l'association entre au moins 2 modèles et une ou plusieurs collections de graphes. Des paramètres peuvent être associés à un test.

5.1 Liste des tests

La liste des tests présente les tests existants dans l'application. On peut voir leur nom, leur description, le mode et le nombre d'exécutions du test ainsi que les modèles et les collections de graphes associés. Il est possible de filtrer la liste en fonction du contenu du nom et de la description et également par modèle ou collection de graphes. Un clic simple sur une ligne de la liste d'affiche une zone en bas de la liste permettant d'exécuter le test sur lequel on a cliqué. Un double clic permet d'ouvrir la fiche du test concerné. Un bouton permet également d'ouvrir une fiche vierge pour ajouter un nouveau test.

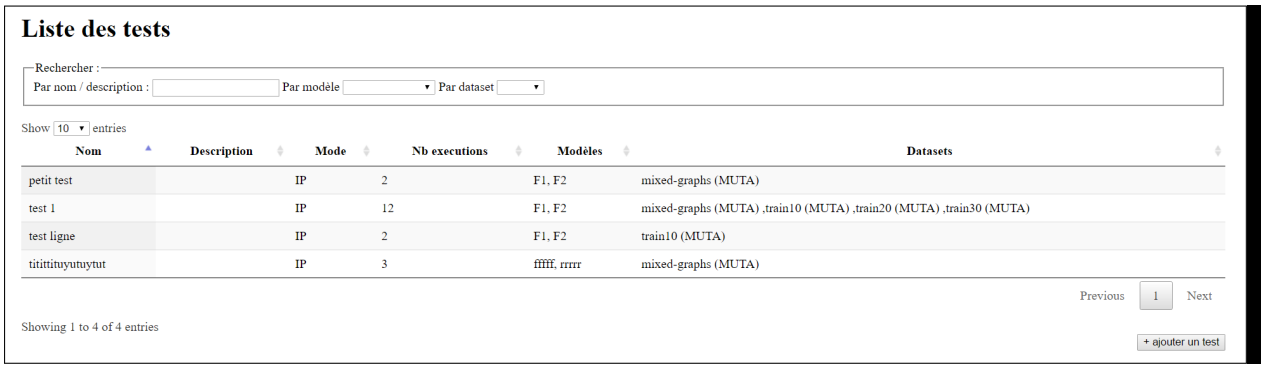


Figure 5 – *Liste des tests*

5.2 Fiche test

Pour ajouter un test, l'utilisateur doit sélectionner au minimum 2 modèles et au moins une collection (ou sous-collection) de graphes. Il doit également saisir un nom du test et sélectionner le mode d'exécution. L'utilisateur peut également associer des paramètres au test. Il y a certains paramètres prévus par défaut (mémoire limite, temps limite, etc...). D'autres peuvent être ajoutés par l'utilisateur en renseignant le nom et la valeur du paramètre. Enfin, l'utilisateur peut également renseigner une description. Un test peut être modifié à tout moment. Depuis cette page, l'utilisateur peut enregistrer les modifications de la fiche, retourner à la liste des test, ouvrir une fiche vierge et masquer le test en cours (dans ce cas, le test ne sera plus visible par l'utilisateur mais toujours présent dans les données de l'application).

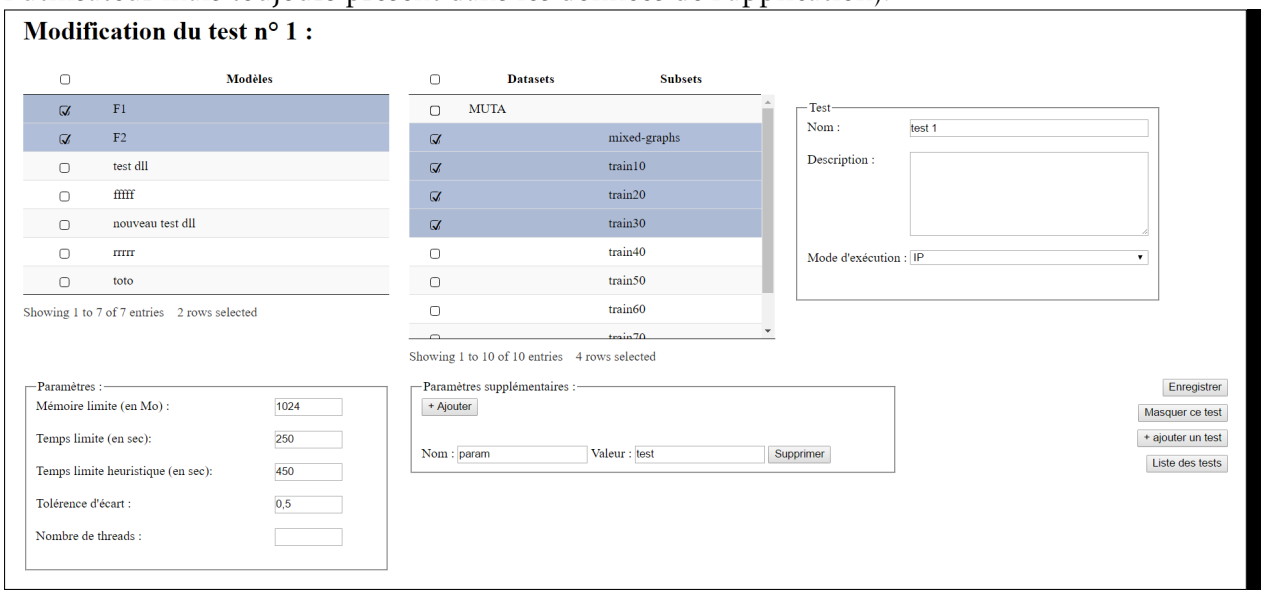


Figure 6 – *Fiche test*

6 Gestion des exécutions

6.1 Lancement d'une exécution

Une exécution peut être lancée avec un clic simple sur une ligne de la liste des tests. L'utilisateur devra saisir un nom et éventuellement une description avant de lancer l'exécution. Au démarrage de l'exécution, les paramètres du test associé sont dupliqués. Ainsi, la modification ultérieure du test n'impacte pas les exécutions lancées avant.

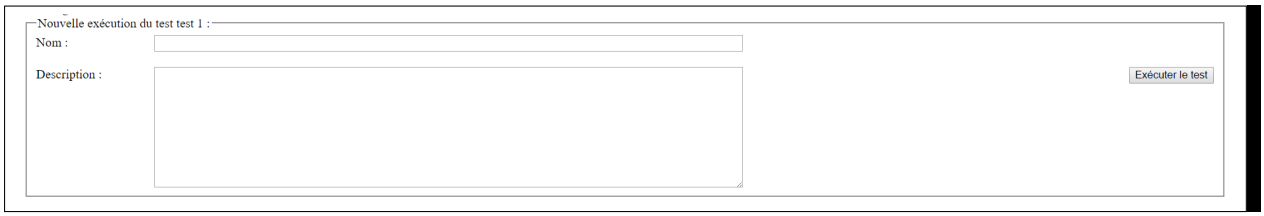


Figure 7 – *Fiche exécution*

6.2 Liste des exécution

La liste des exécutions présente le nom, la description, le nom du test associé, le nombre d'instances exécutes sur le nombre d'instances totales et l'état de l'exécution (en attente, en cours, terminé).

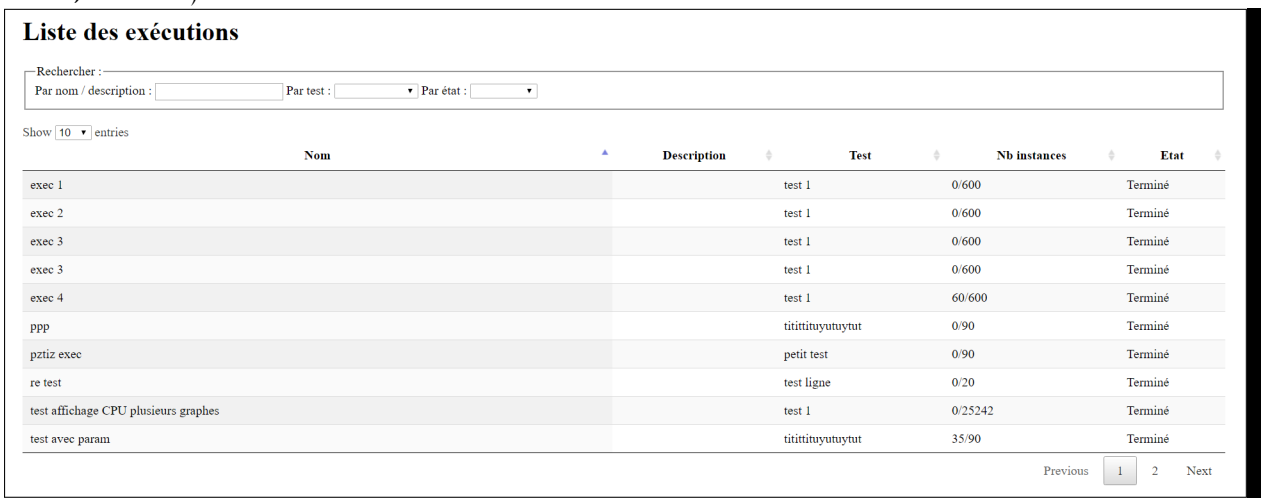


Figure 8 – *Liste des exécutions*

6.3 Détails d'une exécution

Un clic simple sur une ligne permet d'afficher une zone en dessous de la liste contenant des détails sur l'exécution courante. On peut ainsi voir l'avancement de l'exécution, le nombre d'erreurs. L'utilisateur peut accéder au résultats de l'exécution, relancer les instances qui ont échouées et masquer l'exécution.

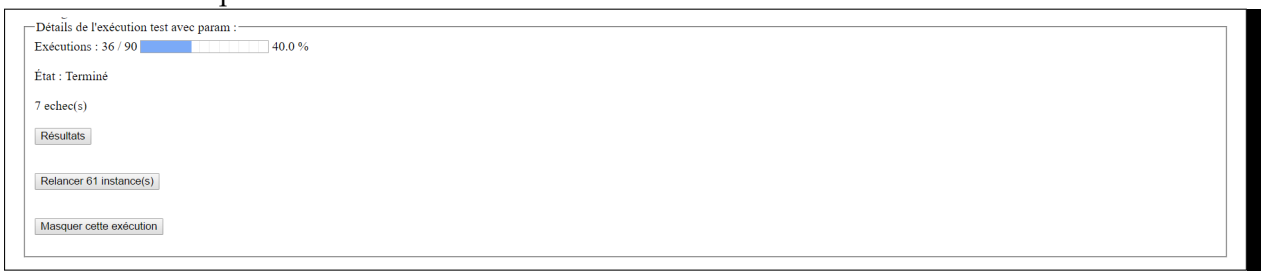


Figure 9 – *Détails d'une exécution*

7 Résultats d'une exécutions

Les résultats d'une exécution sont présentés sous forme de différents onglets : Général, temps CPU, Nœuds explorés, Solutions optimales, Instances traitées, Déviation, Appareillements. L'objectif est de représenter ces résultats sous forme graphique afin de permettre une comparaison facile et rapide.

7.1 Général

Cet onglet rappel les modèles, les collections de graphes, le mode d'exécution, le nombre d'instances et les paramètres correspondants à l'exécution.

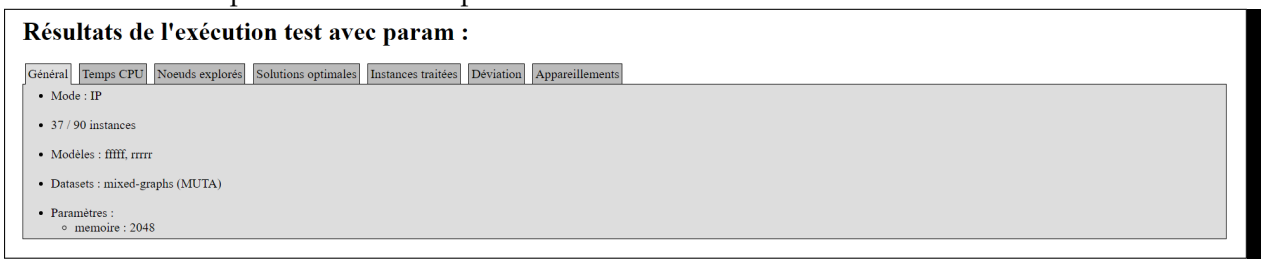


Figure 10 – *Résultats onglet Général*

7.2 Temps CPU

L'onglet Temps CPU présente un tableau et un graphique pour chaque collection de graphes (ou sous-collection) de l'exécution. Ils permettent de visualiser les valeurs minimums, moyennes et maximums des temps CPU de l'exécution pour chaque modèle.

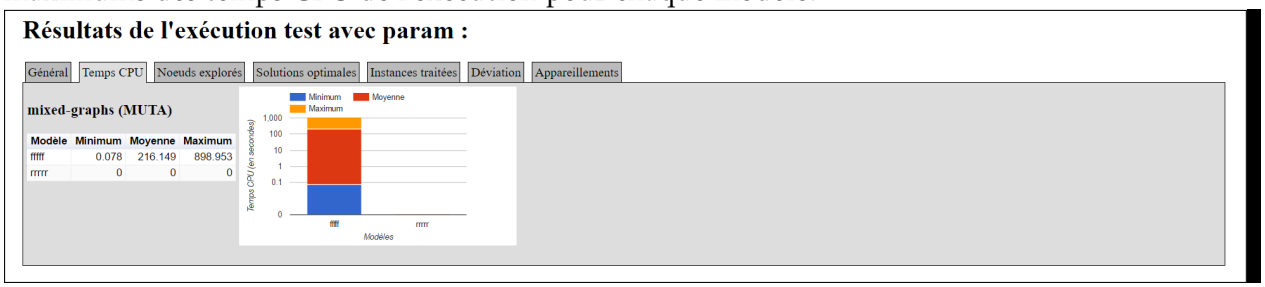


Figure 11 – *Résultats onglet Temps CPU*

7.3 Nœuds explorés

L'onglet Noeuds explorés présente un tableau et un graphique pour chaque collection de graphes (ou sous-collection) de l'exécution. Ils permettent de visualiser les valeurs minimums, moyennes et maximums du nombre de nœuds explorés par chaque modèle.

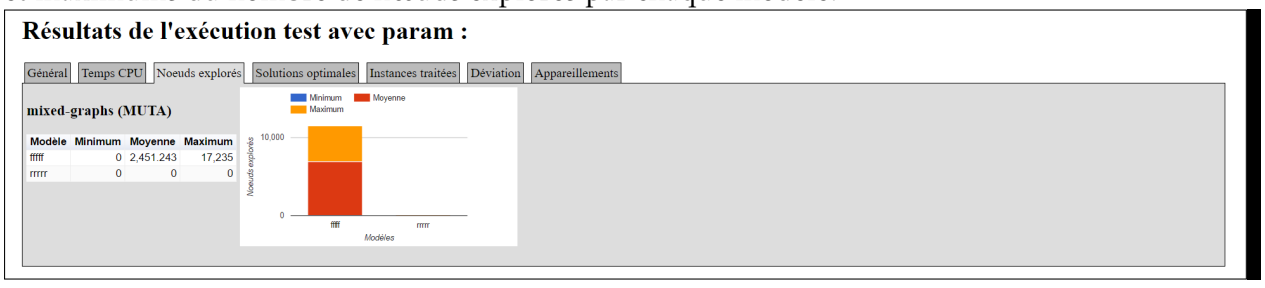


Figure 12 – *Résultats onglet Nœuds explorés*

7.4 Solutions optimales

L'onglet Solutions optimales présente un tableau et un graphique pour chaque collection de graphes (ou sous-collection) de l'exécution. Ils permettent de visualiser le nombre de solutions optimales trouvées par chaque modèle.

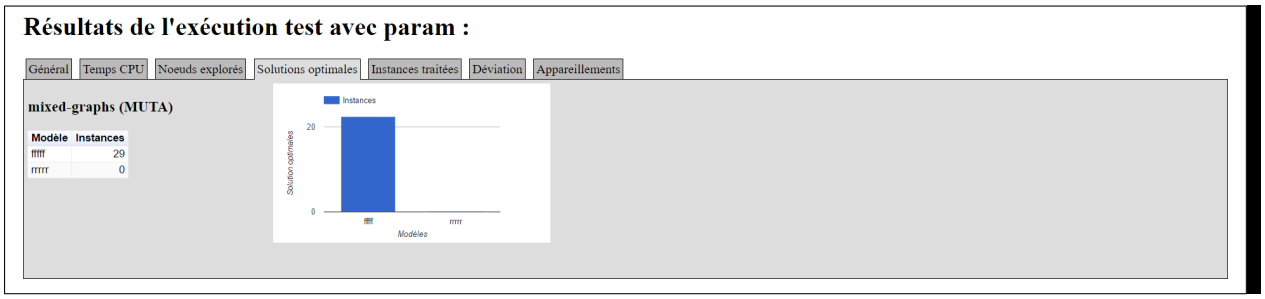


Figure 13 – *Résultats onglet Solutions optimales*

7.5 Instances traitées

L'onglet Instances traitées présente un tableau et un graphique pour chaque collection de graphes (ou sous-collection) de l'exécution. Ils permettent de visualiser le nombre d'instances traitées par chaque modèle.

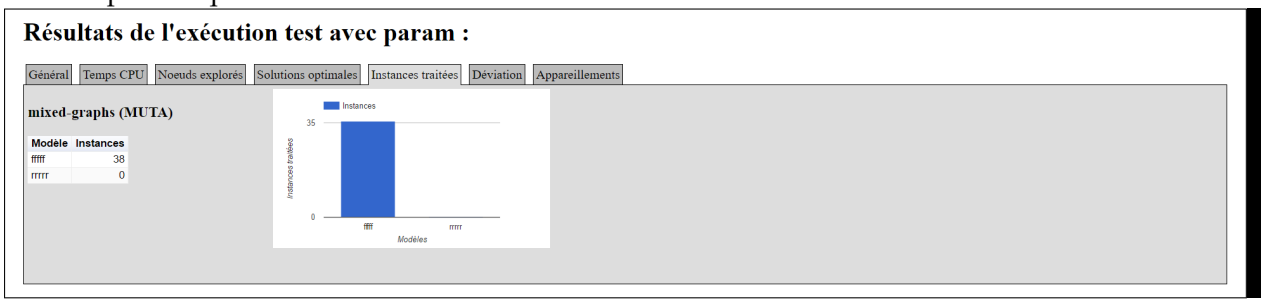


Figure 14 – *Résultats onglet Instances traitées*

7.6 Déviation

L'onglet Déviation présente un tableau et un graphique pour chaque collection de graphes (ou sous-collection) de l'exécution. Ils permettent de visualiser les valeurs minimums, moyennes et maximums de la déviation de chaque modèle. La déviation correspond à l'écart entre la plus petite valeur de la fonction objectif parmi tous les modèles et les valeurs de la fonctions objectif de chaque modèle.

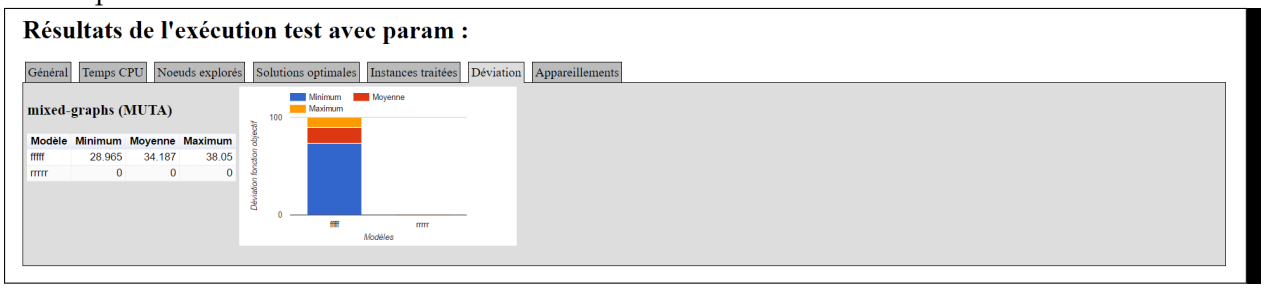


Figure 15 – *Résultats onglet Déviation*

7.7 Appareillements

L'onglet Appareillements, permet d'afficher un tableau représentant l'association faite entre les nœuds et les arcs d'une paire de graphes par un modèle. L'utilisateur devra donc sélectionner le modèle et les deux graphes dont il souhaite connaitre les appareillements.

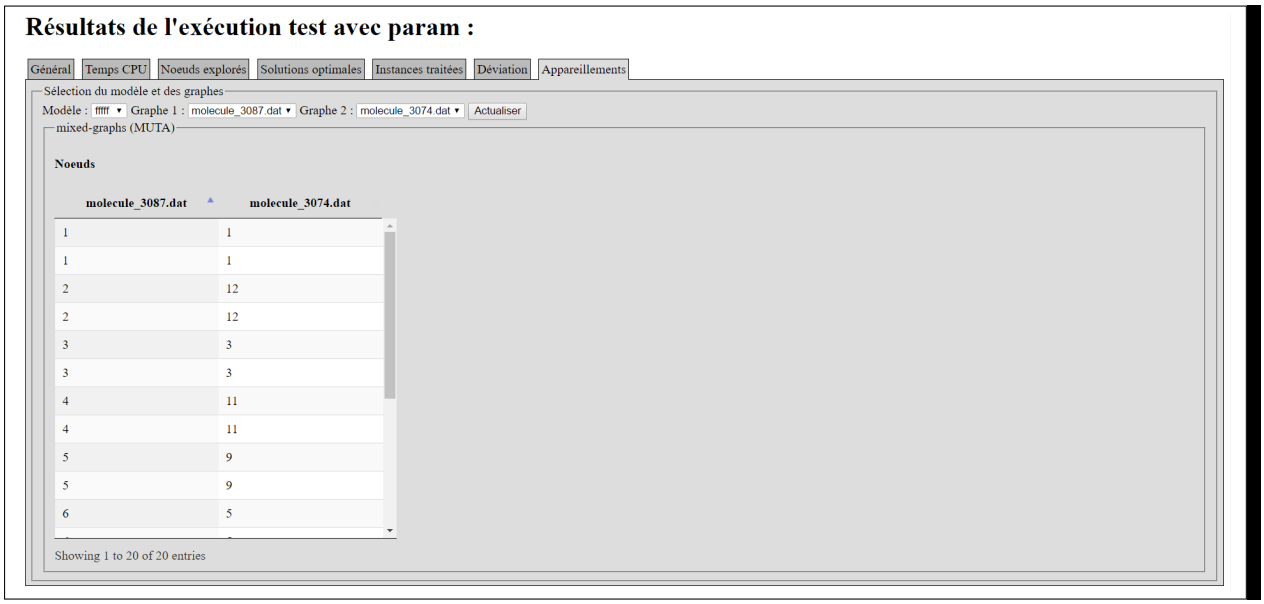


Figure 16 – *Résultats onglet Appareillements*

E Documentation technique

1 Serveur et déploiement

L'application est actuellement hébergée sur une machine virtuelle de l'école avec un serveur TomCat. Elle est accessible à l'adresse suivante : `10.108.99.40\GraphMatching`. Le serveur est installé dans un dossier prod situé sur le bureau de la machine virtuelle.

2 Mise à jour de l'application

Pour mettre à jour l'application, il faut dans un premier temps stopper le serveur TomCat. Ensuite, copier et remplacer l'ancien le fichier GraphMatching.war généré à partir du code source dans le répertoire webapps du serveur. Dans ce même répertoire, s'il existe déjà un dossier GraphMatching, le supprimer. Enfin, redémarrer le serveur.

3 Dépendances

La majorité des librairies nécessaires au fonctionnement de l'application (annotation Hibernate, servlets, junit...) sont incluses dans le répertoire web\WEB-INF\lib du projet.

ParseXML.exe

L'exécutable ParseXML.exe est indispensable pour que l'application puisse convertir les fichiers graphes en format .gxl en .dat. Il doit être placé dans le répertoire d'exécution de l'application, donc dans le dossier bin du serveur TomCat.

cplex1260.dll

La DLL cplex1260.dll est nécessaire à l'exécution des modèles mathématiques. Afin que l'application puisse la copier dans chaque dossier contenant un exécutable d'un modèle, elle devra être placé dans le répertoire bin du serveur.

Cplex

Les modèles mathématiques font appel à Cplex pour trouver des solutions. Cplex doit donc être installé et configuré (variable d'environnement) sur le serveur.

4 Architecture de l'application

Voici une decription synthétique de l'architecture de l'application. Pour plus de précisions, consulter la Javadoc.

4.1 Package DAO

Ce package contient les classes faisant le lien avec l'application et la base de données. La classe DAO défini les méthodes permettant d'ajouter, de mettre à jour et de supprimer un objet dans la base de données. Il y a ensuite 5 classes DatasetDAO, ModeleDAO, SubsetDAO, TestDAO et ExecutionDAO qui héritent de la classe DAO permettant d'interroger la base de données sur des objets spécifiques associés.

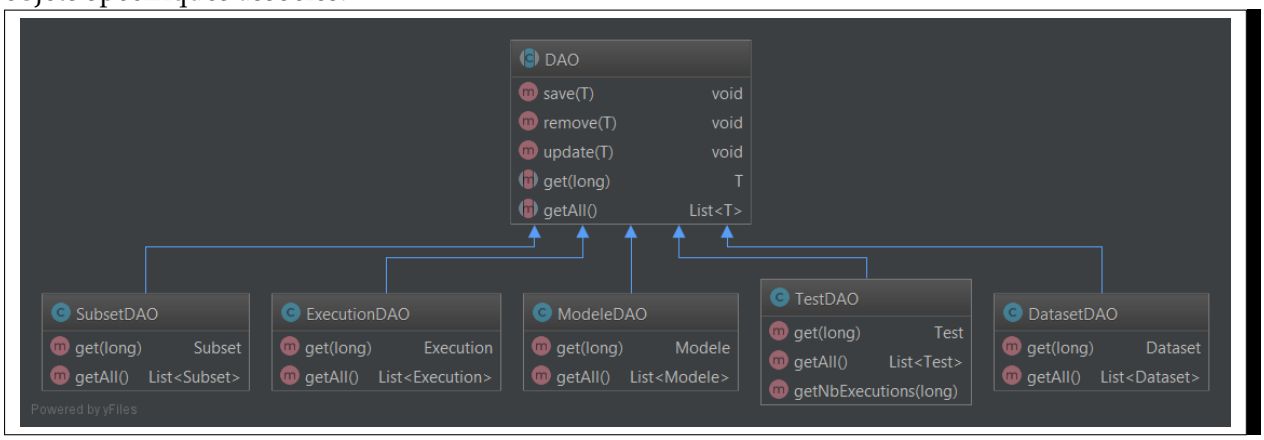


Figure 1 – *Diagramme de classe du package DAO*

4.2 Package Model

Ce package contient les classes qui correspondent aux entités gérés par l'application. Les classes sont liées aux données grâce au mapping réalisé avec les annotations Hibernate. Par soucis de lisibilité, les méthodes sont masquées sur le diagramme ci-ci-dessous. Les classes Modele, Dataset, Subset et Test contiennent des méthodes classiques permettant la gestion des propriétés des classes (voir la Javadoc pour plus de détails). La classe Execution est plus complexe que les autres car elle gère l'exécution des tests et l'interprétation des résultats, elle fait donc l'objet d'une section plus détaillée ci-dessous.

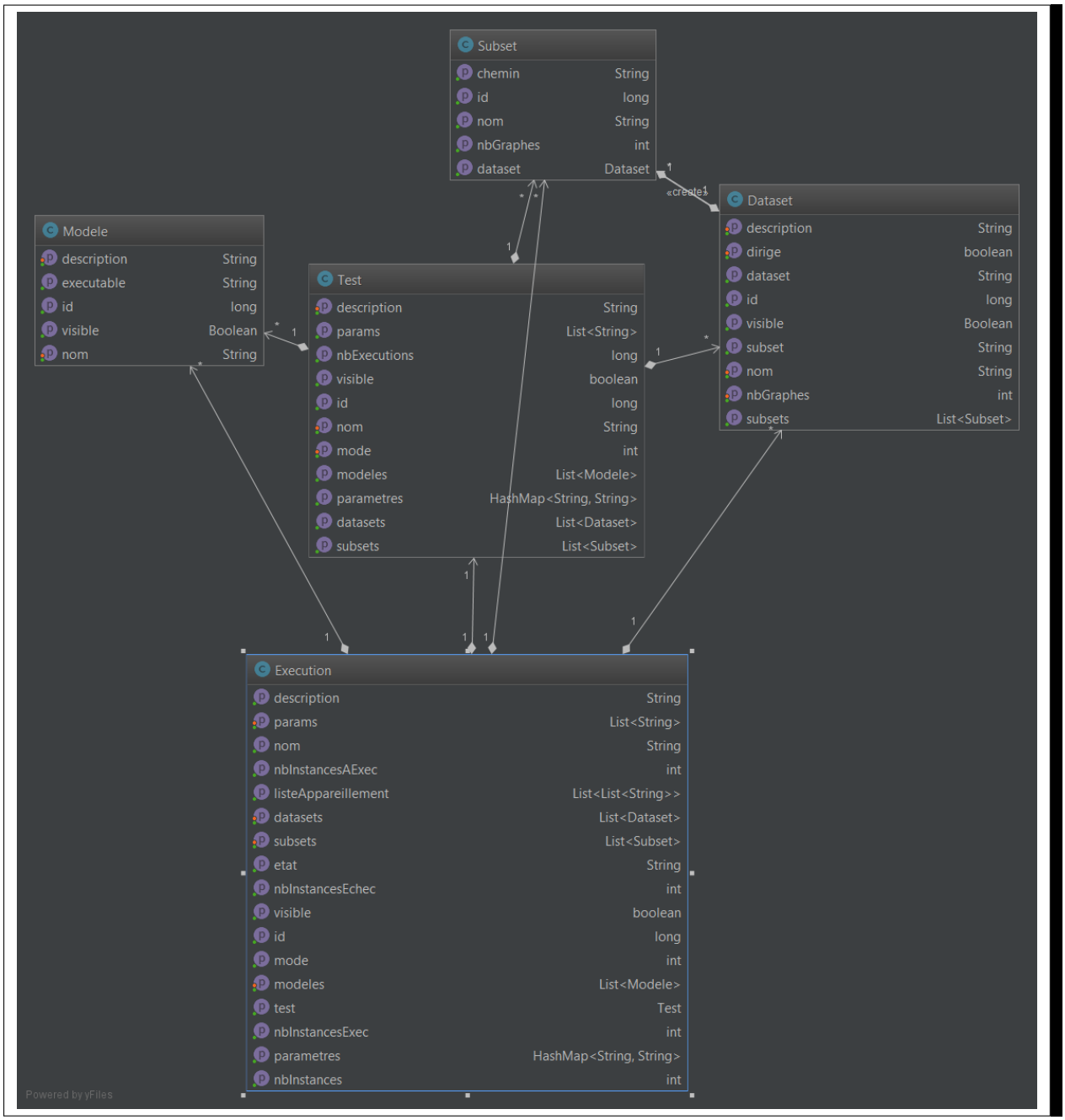


Figure 2 – *Diagramme de classe simplifié du package Modèle*

4.2.1 Classe Execution

On note que toutes les caractéristiques d'un test (modèles, graphes, paramètres) sont copiés dans l'objet Execution afin de garantir son intégrité même après la modification de l'objet Test. La classe Execution implémente l'interface Runnable car la méthode run() permettant l'exécution d'un test doit être exécutée dans un thread afin de ne pas figer l'application. Cette méthode Run() fait appel à lancerExecution() qui est chargée de lancer l'exécutable du modèle en lui passant les bon paramètres. On retrouve également la méthode aExecuter() qui détermine pour chaque modèle et paire de graphe si une exécution doit être lancée. Concernant les résultats, la classe dispose de plusieurs méthode permettant le parcours des fichiers résultats qui formatent les données de manière à être affichées : getResultats(), calculDeviation(), getCount(), getListeAppareillement(), getDetailsAppareillements(). Chaque méthode correspond à un format de données spécifique ou à un calcul a effectuer en fonction de l'information demandée.

Afin de gérer les structures de données stockant les résultats (tableau HTML, HashMap de HashMap, liste), des méthodes ont été créée : construitTableau(), initArrayList(), initHashMap().

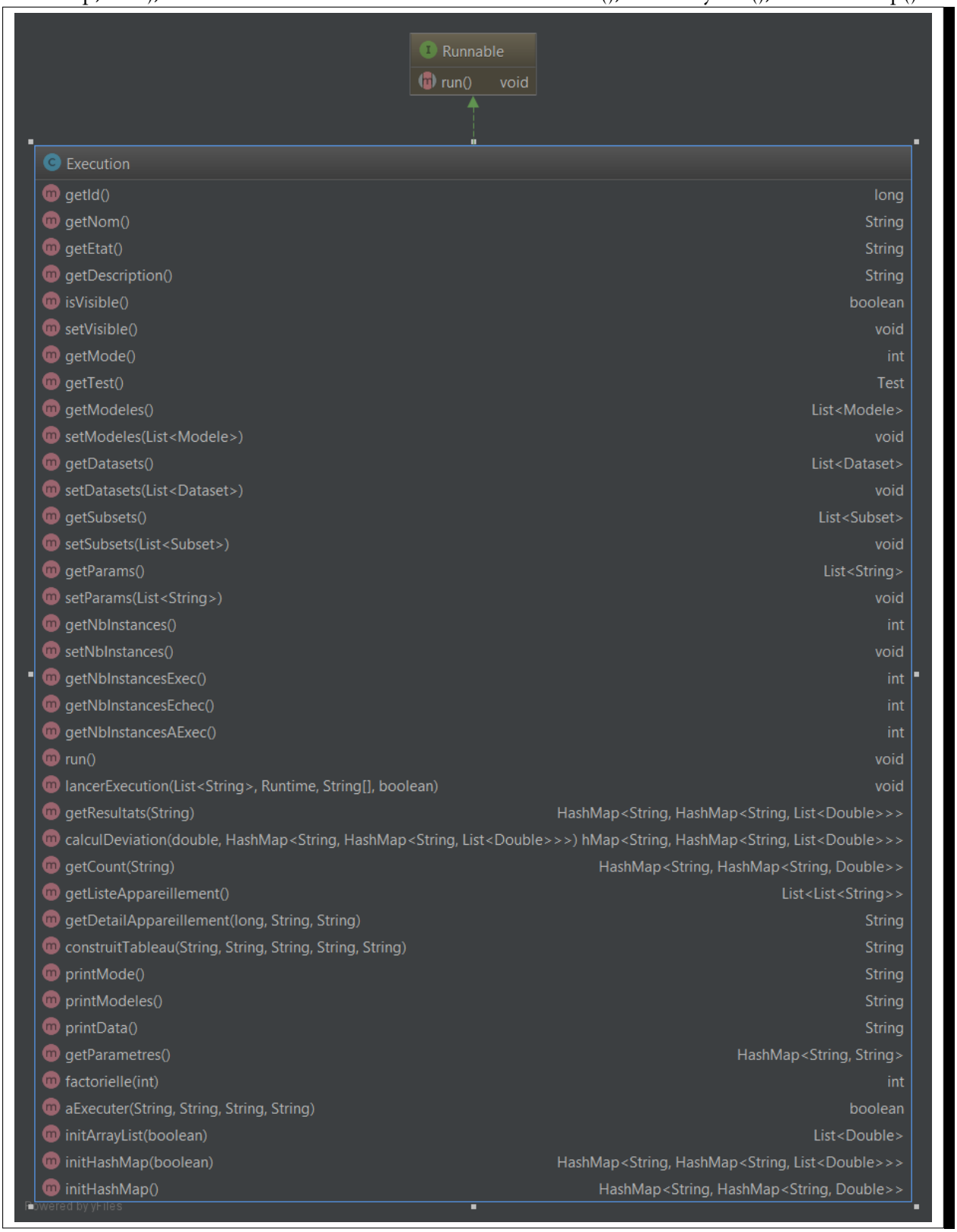


Figure 3 – *Diagramme de classe de la classe Execution*

4.3 Package Servlets

Ce package contient les classes servlet faisant le lien entre la partie objets et la partie graphique réalisée en jsp. Elles sont réparties en sous-packages, chacun correspondant à un objet du modèle.

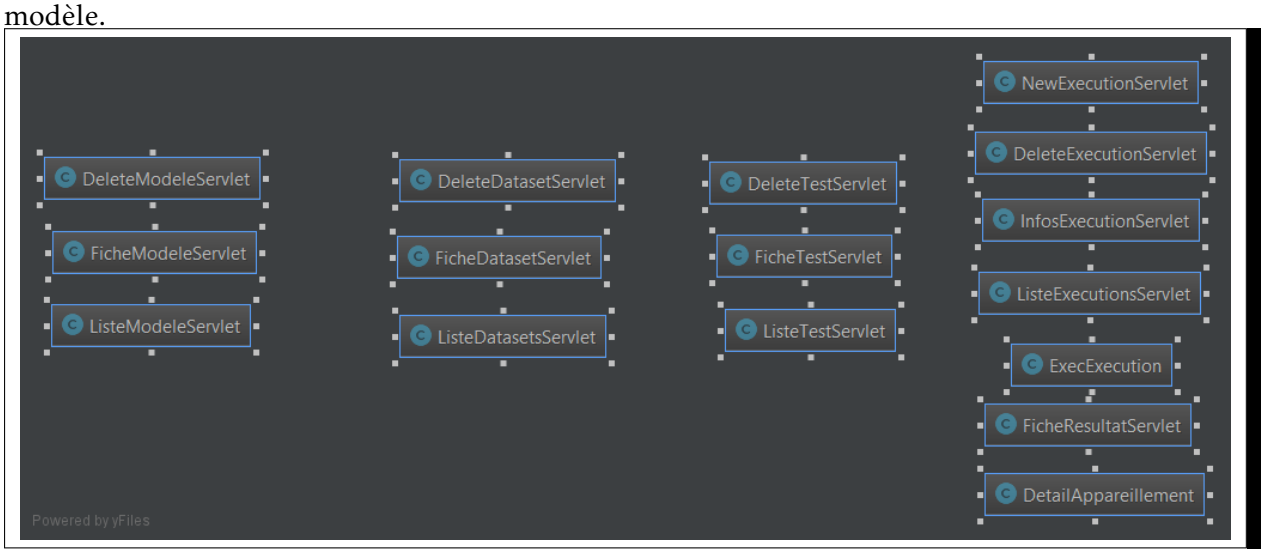


Figure 4 – *Diagramme de classe du package Servlet*

Le schéma suivant montre les interactions entre les différentes servlet et les pages .jsp :

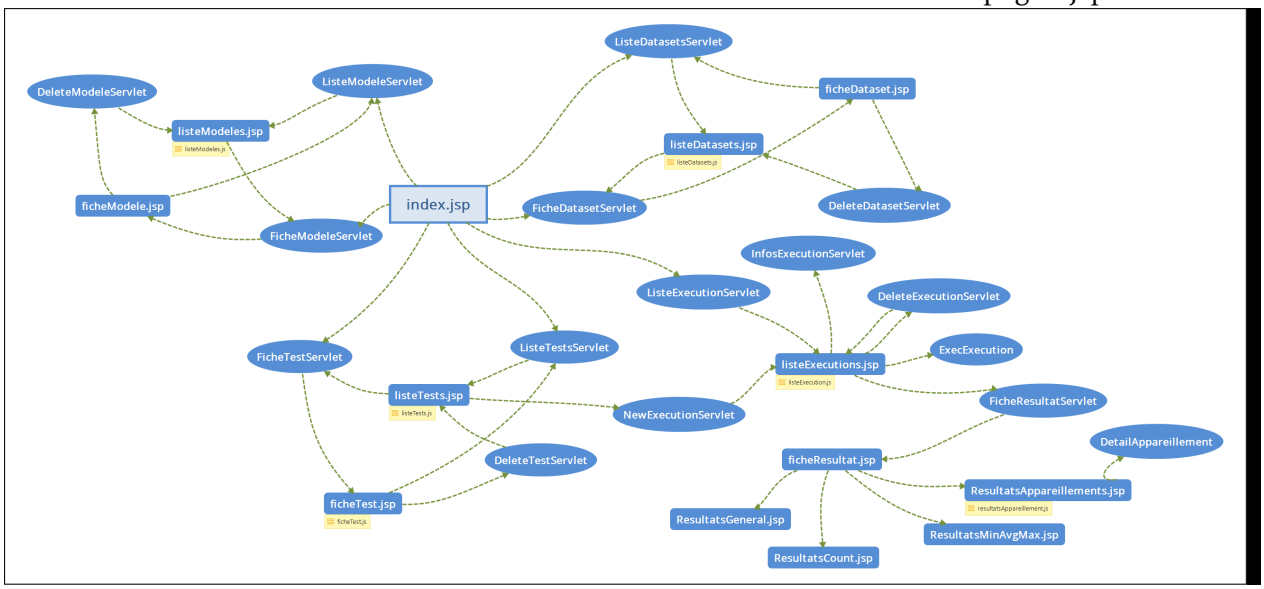


Figure 5 – *Schéma de navigation web*

4.4 Package Tools

Ce package contient des classses annexes. La classe DatabaseConnexion se charge de la lecture du fichier de configuration de la base de donnée pour ouvrir des sessions nécessaire aux classe du package DAO pour l'accès aux données. La classe Executor est chargée de fournir un objet ExecutorService permettant l'exécution des threads de la classe Execution. Enfin, la classe Fichier contient de nombreuses méthodes de gestion de fichiers (copie, lecture, suppression, convertion, décompression...).

4.5 Package Tests

Ce package contient des tests unitaires pour les classe Modele, Dataset et Test.

5 Base de données

Les données sont stockées dans une base de données H2. Le fichier physique se trouve dans . Les configurations de la base de données utilisées par l'application sont stockés dans le fichier hibernate.cfg.xml.

5.1 Architecture

Voici la structure de la base de données générée par Hibernate.

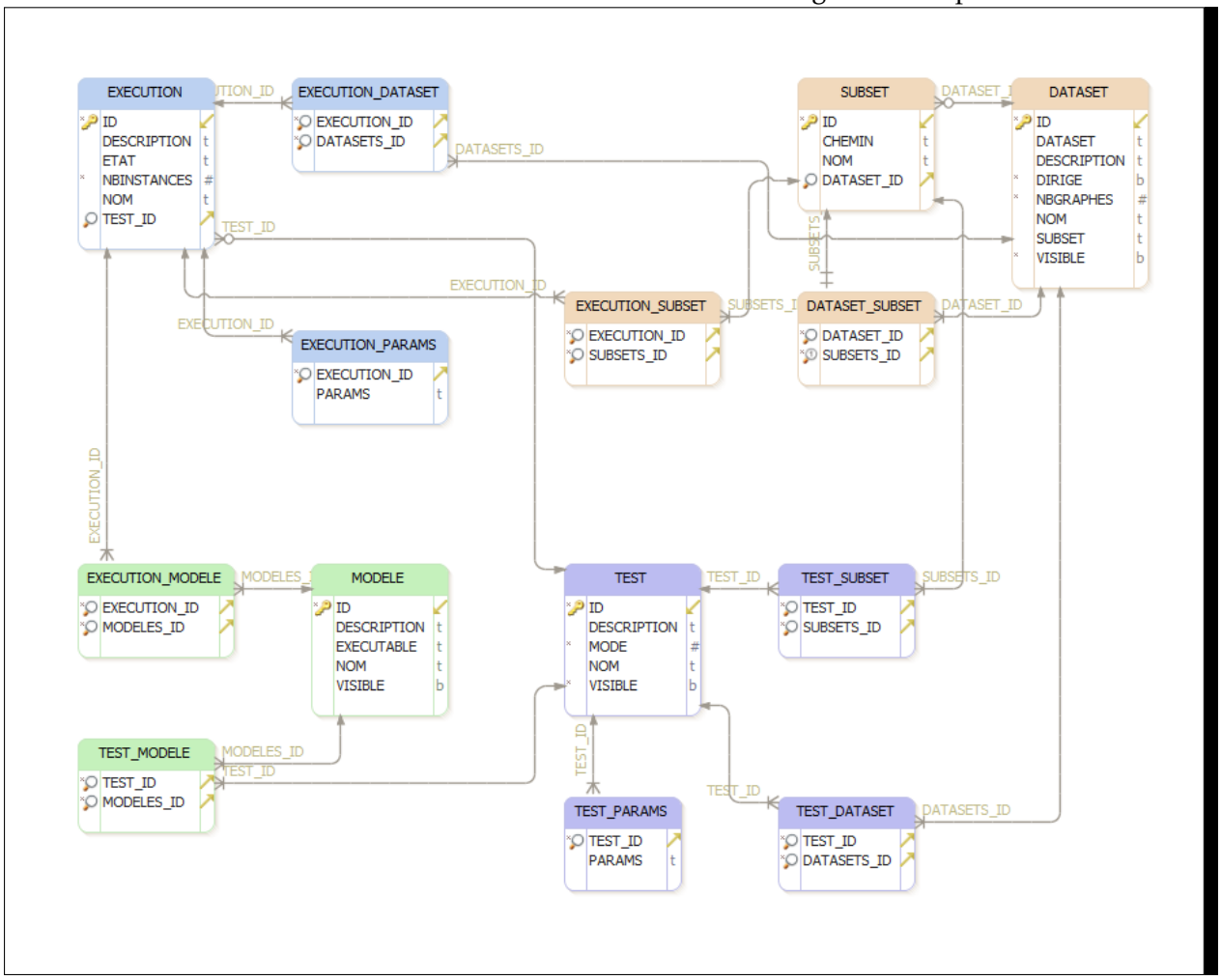


Figure 6 – *Schéma de la base de données*

Les classes permettant de faire le lien entre l'application et la base de données se trouvent dans le package DAO.

Comptes rendus hebdomadaires

Compte rendu n°1 du 22/09/2016

Bonsoir,

Les deux documents que vous m'avez donné hier m'ont permis de mieux cerner le problème du Graph Matching, les différents moyens de le résoudre et les domaines d'application possibles.
Je suis actuellement en train d'étudier les modèles mathématiques que vous m'avez transmis ce matin. Pour le moment ma compréhension est assez bonne malgré quelques détails qui sont un peu flous mais je pense que ça sera plus clair lorsque j'aurai parcouru la totalité du document.
Je viendrais vous poser des questions si nécessaire lorsque j'aurai terminé.

Cordialement,

Kévin MICHAUX.

Compte rendu n°2 du 30/09/2016

Bonjour,

Cette semaine j'ai terminé d'étudier les modèles mathématiques et j'ai commencé a réfléchir au plan de mon état de l'art.
La semaine prochaine je prendrais connaissance de l'implémentation du modèle que vous allez me fournir et commencerai la rédaction de l'état de l'art sur le principe du graphe matching, le GED et les différentes techniques existantes.

Cordialement,

Kévin MICHAUX

Compte rendu n°3 du 06/10/2016

Bonjour,

Cette semaine j'ai avancé sur la rédaction de l'état de l'art et j'ai également parcouru le code que vous m'avez fourni.
J'ai été malade toute la semaine donc je n'ai pas avancé autant que je voulais, c'est pour ça que je n'ai pas pris contact avec nous pour une réunion cette semaine.
Pouvons-nous faire le point la semaine prochaine (mercredi matin si possible)? J'aimerai vous poser des questions sur le contenu de l'état de l'art (quels sont les points à ajouter ou à développer) et éventuellement parler de l'outil que je vais devoir développer pour pouvoir commencer à rédiger les spécifications et réaliser des recherches sur les technologies adaptées.
Je joins donc à ce mail une ébauche de l'état de l'art si vous souhaitez en prendre connaissance avant notre entrevue.

Cordialement,

Kévin MICHAUX.

Compte rendu n°4 du 14/10/2016

Bonjour,
Cette semaine j'ai complété l'état de l'art suite à vos remarques, commencé les maquettes d'IHM, réfléchi à la manière de concevoir l'application et à la méthode de gestion de projet.
J'ai besoin de précisions supplémentaires pour terminer les maquettes d'IHM. Pouvons-nous nous voir mercredi matin svp ?
Bon weekend,
Kévin MICHAUX.

Compte rendu n°5 du 25/10/2016

Bonjour,
J'ai oublié de vous faire un compte rendu la semaine dernière. J'ai terminé les maquettes d'IHM et avancé sur le cahier des spécifications.
Mercredi je vais travailler sur le cahier des spécifications, compléter l'état de l'art avec les formules mathématiques et créer un diagramme de Gant prévisionnel.
Cordialement,
Kévin MICHAUX.

Compte rendu n°6 du 07/11/2016

Bonjour,
La semaine dernière j'ai presque terminé l'état de l'art, il manque juste les modèles mathématiques et la bibliographie. J'ai également rédigé les spécification concernant le paramétrage et le lancement des tests.
Pouvons-nous nous voir jeudi pour faire le point sur mon travail et discuter des spécifications concernant la restitution des résultats des tests ?
Cordialement,
Kévin MICHAUX.

Compte rendu n°7 du 14/11/2016

Bonjour,
La semaine dernière j'ai terminé l'état de l'art sur les graphes mercredi. Jeudi, j'ai modifié les maquettes d'IHM et le cahier des spécifications en fonction des points évoqués jeudi matin.
J'ai également commencé à rechercher des API pour générer des graphiques et à les comparer en elles.
Pouvons-nous nous voir mercredi ? J'ai des questions à vous poser et nous pourrons en profiter pour valider la nouvelle version des IHM.
Cordialement,
Kévin MICHAUX.

Compte rendu n°8 du 17/11/2016

Bonsoir,
Comme prévu voici une première version de mon rapport. A noter que le diagramme n'est pas bon (je vais le terminer mercredi) et qu'il manque la bibliographie que je n'arrive pas à compiler. De plus, je n'ai pas pris le temps de le relire.
La semaine prochaine je vais donc terminer le diagramme de classes et réfléchir à la manière de stocker les données.
Il serai bien de faire une petite réunion pour que vous puissiez me faire part de vos éventuelles remarques (notamment s'il y a certains points que je n'ai pas assez développé et si les spécifications vous conviennent) et que l'on voit ce qu'il me reste à faire d'ici les soutenances de décembre.
Cordialement,
Kévin MICHAUX.

Compte rendu n°9 du 28/11/2016

Bonjour,
La semaine dernière j'ai modifié les IHM en fonction de vos remarques et mon rapport selon le plan fourni par Mr Ragot. J'ai également modifié le diagramme de classes mais il faut encore que je travaille dessus au niveau des classes qui gèrent l'exécution des tests. J'ai aussi ajouté certains points sur la gestion des données (qui n'est pas encore complète) et un paragraphe pour faire le lien entre les graphes et la visualisation de données.
Pouvons-nous nous voir jeudi pour valider les IHM ?
Cordialement,
Kévin MICHAUX.

Compte rendu n°10 du 01/12/2016

Bonsoir,
Suite à notre entrevue de cette après-midi, je vous joins mon rapport.
J'ai donc eut le temps de :

- Modifier les IHM
- De lister les tâches dans les sprints
- D'ajouter un diagramme d'objets pour représenter le lien entre les données
- De refaire le diagramme de structure du système
- D'ajouter des hypothèses pour évoquer la visualisation des appareillements de graphes et le problème du format du fichier de sortie de Cplex

Il me restera donc Mercredi à modifier le diagramme de classes et rédiger la conclusion du rapport (si je n'ai rien oublié) et de faire une relecture du rapport pour corriger les fautes.
De plus, j'ai pris la liberté de modifier le titre du projet qui était "Conception et implémentation des algorithmes de Graph Matching dans un framework interactif" en "Conception d'un outil de comparaison des méthodes de Graph Matching" car les algorithmes sont déjà conçus et implémentés. Dites moi si vous trouver un titre qui convient mieux.
Pour information, la date limite de dépôt du rapport sur Célène est fixée au vendredi 8 à 8h mais je vais essayer de vous transmettre la version définitive mercredi (cela va dépendre du temps de réflexion nécessaire pour terminer le diagramme de classes).
Merci de me dire si vous avez un moment de libre la semaine prochaine pour faire un dernier point avant la soutenance.
Cordialement,
kévin MICHAUX.

Compte rendu n°11 du 06/01/2017

Bonjour,
Je vous présente mes meilleurs vœux pour cette année.
Voici ce que j'ai fais cette semaine : - mise en place du projet : configuration du serveur Tomcat et des librairies - Création de la classe modèle et des tests unitaires associés - Mise en place de la base de données et des classes DAO - Affichage de la liste des modèles présents dans la base de données sur une page web
Il me reste donc à faire : - l'IHM pour l'insertion et la modification des modèles - compléter l'IHM de la liste des modèles avec la recherche et l'ouverture de la page de modification au clic dans la table - la création du menu
J'ai perdu beaucoup de temps (presque une journée) pour la mise en place et la configuration, maintenant que tout est en place le développement devrait aller plus vite, je pense donc être en mesure de terminer la gestion des modèles d'ici la semaine prochaine.
Je vous transmettrai prochainement un lien vers le Git de mon projet.
Il faudrait trouver une solution pour que vous puissiez valider mes productions car si vous récupérez juste mon projet, vous n'aurez pas d'environnement configuré pour le lancer (serveur et librairies) je pense donc à mettre en place une VM qui serait configurée comme il faut dans

la quelle vous n'aurez plus qu'a charger le projet. L'inconvénient étant que la mise en place de cette VM va me retarder dans le développement. Que pensez-vous de cette solution ?
Cordialement,
Kévin MICHAUX.

Compte rendu n°12 du 13/01/2017

Bonjour,
La partie gestion de modèles est fonctionnelle. Il reste juste quelques petits détails à régler : encodage des caractères spéciaux contrôles de saisie affichage des erreurs dans le navigateur Je n'ai pas eut le temps de mettre en place le serveur sur la VM ni de rédiger la documentation, je vais faire tout ça mercredi prochain.
En attendant, vous pouvez passer me voir mercredi pour que je vous montre ce que j'ai fais si vous voulez.
Je suis légèrement en retard sur le planning prévu, j'espère avancer vite sur la gestion des datasets car la majorité des fonctionnalités sont identiques à celles des modèles donc cela devrait aller vite.
Bon weekend,
Kévin MICHAUX.

Compte rendu n°13 du 24/01/2017

Bonjour,
Désolé j'ai oublié de vous envoyer un mail la semaine dernière.
J'ai bien avancé sur la partie gestion des datasets mais j'ai quelques questions à vous poser. Du coup pour ne pas perdre de temps j'ai commencé à travailler sur l'interface des tests.
Pouvons-nous nous voir demain matin svp ?
Cordialement,
Kévin.

Compte rendu n°14 du 26/01/2017

Bonsoir,
Depuis hier j'ai fais quelques corrections sur la gestion des fichiers et j'ai fais la convertion des fichiers .gxl en .dat, j'ai rajouté les contrôles de saisie et j'ai avancé la gestion des tests.
J'ai eut du mal à récupérer les identifiants des modèles et des datasets sélectionnés dans les tableaux du coup je n'ai pas eut le temps de mettre le serveur en place sur la VM.
La décompression des archives rar ne fonctionne pas avec la méthode que j'utilise pour les ZIP. Dois-je m'occuper des rar maintenant ou est-ce qu'on fonctionne avec les ZIP pour l'instant ?
La gestion des tests sera terminée la semaine prochaine et le serveur mis en place s'il n'y a pas trop de problèmes de configuration réseau. Je pourrai donc m'attaquer à l'exécution des test d'ici 2 semaines normalement.
Bonne soirée,
Kévin.

Compte rendu n°15 du 02/02/2017

Bonsoir,
Cette semaine j'ai terminé les IHM associées aux tests et j'ai fais la gestion des objets correspondant à l'exécution d'un test. Je n'ai plus qu'a générer le fichier contenant les paramètres je pourrai commencer la gestion des threads qui exécuteront les tests.
J'ai également mis en place le serveur sur la VM. Il fonctionne, l'application est accessible depuis un autre poste. Par contre j'ai mis ma version de développement de l'application pour tester le serveur donc il se peut qu'il reste des bugs. Je déposerait une version de production la semaine prochaine.
Pouvous-nous nous voir la semaine prochaine svp ? J'ai quelques questions à vous poser.
Bonne fin de semaine,

Kévin.

Compte rendu n°16 du 16/02/2017

Bonsoir,
Cette semaine j'ai modifié un modèle pour qu'il puisse lire les paramètres. L'application peut donc maintenant lancer des tests. J'ai commencé le parsing du fichier de sortie mais je n'ai pas encore terminé.
Pouvons-nous nous voir la semaine de la rentrée svp ? J'ai des questions à vous poser.
Bonnes vacances,
Kévin.

Compte rendu n°17 du 06/03/2017

Bonjour,
La semaine dernière j'ai réalisé l'affichage des tableaux et des graphiques pour les temps CPU et les nœuds explorés.
Je pense que les affichages restants ne devraient pas me prendre trop de temps.
La semaine prochaine je vais également lancer des tests sur la VM pour vérifier les affichages sur un cas réel. Je vais en profiter pour modifier les modèles présents sur la VM . pouvez vous me dire les quels je dois modifier en priorité ou les mettre dans un dossier spécifique sur la VM svp ?
Bonne journée,
Kévin.

Compte rendu n°18 du 09/03/2017

Bonsoir,
J'ai eut le temps de modifier un modèle sur les 3. Concernant l'application il ne manque plus que l'appareillent et l'onglet général (rappel des paramètres, etc...). Je pense avoir le temps de finir tout ça mercredi prochain, voir jeudi au pire des cas.
Bonne soirée,
Kévin.

Compte rendu n°19 du 21/03/2017

Bonjour,
Désolé pour le mail tardif. La semaine dernière j'ai terminé le développement de l'application et la modification des 3 modèles. J'ai commencé à faire des tests sur la VM ce qui m'a permis de trouver quelques bugs mineurs que j'ai corrigés. J'ai également commenté une bonne partie de mon code pour générer la Javadoc.
Cette semaine je vais donc continuer les tests et la rédaction de la documentation et du rapport en parallèle.
Bonne journée,
Kévin.

Compte rendu n°20 du 22/03/2017

Bonjour,
Oui je suis en train de corriger quelques bugs et j'ai rajouté une fonctionnalité pour relancer les instances qui ont échouées. Pour l'instant je n'ai pas de questions particulières mais vous pouvez venir demain pour voir le résultat si vous voulez !
Bonne journée,
Kévin.

Compte rendu n°21 du 24/03/2017

Bonjour, aucun problème je ne suis pas bloqué. J'ai presque terminé la correction des bugs, j'ai juste un problème de retour à la ligne dans les fichiers que je n'ai pas eut le temps de corriger hier.
Bonne journée,

Kévin MICHAUX

Webographie

[WWW1] *Documentation D3.js*. URL : https://github.com/d3/d3/wiki.

[WWW2] *Documentation Elycharts*. URL : http://elycharts.com/docs.

[WWW3] *Documentation FlotR2*. URL : http://www.humblesoftware.com/flotr2/index.

[WWW4] *Documentation Google Charts*. URL : https://developers.google.com/chart/interactive/docs/.

[WWW5] *Documentation HighCharts*. URL : http://www.highcharts.com/docs.

[WWW6] *Documentation Morris.js*. URL : http://morrisjs.github.io/morris.js/index.html.

Bibliographie

[1] D. Conte, P. Foggia, C. Sansone et M.Vento. « Thirty years of graph matching in pattern recognition ». In : *International Journal of pattern recognition* 18.3 (2004), p. 265–298.

[2] Derek Justice et Alfred Fellow. « A binary linear programming formulation of the graph edit distance ». In : *IEEE transactions on pattern analysis and machine intelligence* 28.8 (juin 2006), p. 1200–1214.

[3] Julien Lerouge, Zeina Abu-Aisheh, Romain Raveaux, Pierre Héroux et Sébastien Adam. « Graph edit distance : a new binary linear programming formulation ». In : *IEEE transactions on pattern analysis and machine intelligence* (). Non publié.

[4] Romain Raveaux et Zeina Abu-Aisheh. *International master of research in computer science : Computer aided decision support. Graph for pattern recognition*. Groupe RFAI de l'université de Tours, oct. 2013.

Projet Recherche & Développement

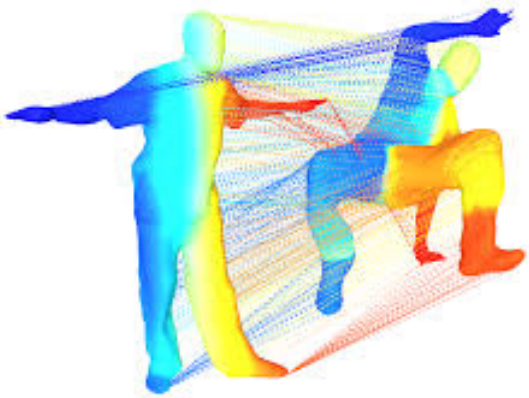
Conception d'un outil de comparaison des méthodes de Graph Matching

Kévin MICHAUX

Encadrement : Mostafa DARWICHE, Donatello CONTE, Vincent T'KINDT et Romain RAVEAUX

Contexte

La comparaison de graphes est utilisée dans de nombreux domaines comme la biologie ou l'analyse d'images et de documents. Cependant, aucune méthode de comparaison n'est parfaite. Il est donc nécessaire de pouvoir mesurer les performances de ces méthodes afin de pouvoir les améliorer.



Objectif

L'objectif de ce projet est de proposer une solution permettant de tester les implémentations des modèles mathématiques de comparaison de graphes simplement et de visualiser les résultats de manière graphique afin de pouvoir en tirer des conclusions.

Résultat attendu

Une application web permettant :

- la gestion des modèles, des collections de graphes et des tests
- L'exécution des tests de manière automatisée
- L'analyse des résultats restitués sous forme graphique

Interface de visualisation des résultats

Ecole Polytechnique de l'Université François Rabelais de Tours
Département Informatique
64 avenue Jean Portalis, 37200 Tours, France
polytech.univ-tours.fr

POLYTECH TOURS
Informatique

Projet Recherche & Développement

Conception d'un outil de comparaison des méthodes de Graph Matching

Kévin MICHAUX

Encadrement : Mostafa DARWICHE, Donatello CONTE, Vincent T'KINDT et Romain RAVEAUX

Contexte

La comparaison de graphes est utilisée dans de nombreux domaines comme la biologie ou l'analyse d'images et de documents. Cependant, aucune méthode de comparaison n'est parfaite. Il est donc nécessaire de pouvoir mesurer les performances de ces méthodes afin de pouvoir les améliorer.

Objectif

L'objectif de ce projet est de proposer une solution permettant de tester les implémentations des modèles mathématiques de comparaison de graphes simplement et de visualiser les résultats de manière graphique afin de pouvoir en tirer des conclusions.

Résultat attendu

Une application web permettant :

- la gestion des modèles, des collections de graphes et des tests
- L'exécution des tests de manière automatisée
- L'analyse des résultats restitués sous forme graphique

Interface de visualisation des résultats

Ecole Polytechnique de l'Université François Rabelais de Tours
Département Informatique
64 avenue Jean Portalis, 37200 Tours, France
polytech.univ-tours.fr

POLYTECH TOURS
Informatique

Conception d'un outil de comparaison des méthodes de Graph Matching

Résumé
Ce rapport présente un état de l'art sur les graphes et les méthodes de comparaison ainsi qu'une étude sur les outils de visualisation de données sous forme graphique dans une application web. Il contient également le cahier des spécifications de l'application permettant de comparer les différentes méthodes de graph matching qui doit être développée ainsi qu'une partie sur les méthodes de gestion de projet mises en œuvre. Il a ensuite été enrichi avec des éléments concernant le développement de l'application
Mots-clés
comparaison de graphes, programmation linéaire, évaluation des méthodes, application web

Abstract
This report present a state of the art on graphs and comparison methods and a study on the tools of data visualization in graphical form on a web application. It also contains the specification booklet of the application allowing to compare the different methods of graph matching that must be developped and a part on project management methods implemented. It was then enriched with elements concerning the development of the application
Keywords
graph matching, linear programming, evaluation of methods, web application

Tuteurs académiques
Mostafa DARWICHE
Donatello CONTE
Vincent T'KINDT
Romain RAVEAUX

Étudiant
Kévin MICHAUX (DI5)