

Universités de Bordeaux
Master 2 BioInformatique

MARIE BEURTON-AIMAR

Conception d'un Projet de Recherche

FRÈCHE Arnaud, HÉRICÉ Charlotte, MOLA Saraï,
PAYSAN-LAFOSSE Typhaine, SANSEN Joris

W E B
R E T
REG EFM TOOL



21 février 2013

Résumé

Le logiciel *regEfmtool* permet la modélisation de réseaux métaboliques et le calcul de modes élémentaires. Ce logiciel fonctionne sous environnement *UNIX* uniquement en ligne de commandes et nécessite plusieurs fichiers d'entrée au format texte. Ces derniers doivent être générés à la main et contiennent les informations sur ce réseau. Sa prise en main n'est donc pas aisée pour un utilisateur ne possédant aucune connaissance en programmation.

Dans le cadre de l'UE Conception d'un Projet de Recherche du Master 2 BioInformatique, notre but était de réaliser une interface graphique (en langages Web) afin de pallier au manque d'ergonomie de *regEfmtool*. Puisque jusqu'à aujourd'hui aucune méthode ne permet de générer les fichiers d'entrée de ce logiciel, ni de le lancer sans ligne de commande.

Cette interface nommée *WRET : WebRegEfmTool* permet de générer les fichiers nécessaires à l'utilisation de ce logiciel de modélisation sans avoir à l'installer sur son système. Elle permet également la création de réseaux au format *DAT* afin de lancer des simulations comparées sur le logiciel *METATOOL*.

Table des matières

Résumé	2
Introduction	5
1 Analyse	6
1.1 Contexte	6
1.2 État de l'existant	7
1.3 Analyse des besoins	10
2 Conception	12
2.1 Interface graphique	12
2.2 Création d'un nouveau réseau	13
2.3 Règles des gènes	14
2.4 Choix des options de lancement	16
2.5 Chargement d'un réseau pré-existant	17
2.6 Affichage des résultats	17
3 Réalisation	19
3.1 Ergonomie de l'interface	19
3.2 Création d'un nouveau réseau	22
3.3 Règles des gènes	26
3.4 Choix des options de lancement	28
3.5 Chargement d'un réseau pré-existant	29
3.6 Affichage des résultats	30
4 Difficultés et améliorations	35
Conclusion	36

Table des figures

2.1	Page d'accueil	12
2.2	Page de création d'un nouveau réseau	13
2.3	Suite de la page de création	14
2.4	Page d'aide à la création d'un nouveau réseau	14
2.5	Page de création des règles des gènes	15
2.6	Page d'aide à la création des règles des gènes	15
2.7	Partie de la page de choix des options de lancement	16
2.8	Partie de la page de chargement d'un réseau préexistant	17
2.9	Page du choix des fichiers à charger	18
2.10	Page de recherche sur le log	18
3.1	Légende de nos diagrammes d'organisation	19
3.2	Exemple d'organisation d'une catégorie	20
3.3	Diagramme d'organisation pour le changement de langue	21
3.4	Diagramme d'organisation pour la création d'un nouveau réseau	22
3.5	Bouton d'initialisation des fichiers	22
3.6	Bouton d'ajout d'une réaction	23
3.7	Bouton de modification	24
3.8	Zone de sélection des métabolites internes et externes	25
3.9	Bouton de création du fichier au format <i>DAT</i>	25
3.10	Diagramme d'organisation pour le changement de langue	26
3.11	Bouton du choix du nombre de réactions pour une règle	27
3.12	Menus déroulant si 2 réactions dans la règle	27
3.13	Menus déroulant si 3 réactions dans la règle	27
3.14	Diagramme d'organisation pour le choix des options après création	28
3.15	Sous catégorie d'affichage des résultats dans la catégorie d'enregistrement	28
3.16	Diagramme d'organisation pour le chargement des fichiers	30
3.17	Diagramme d'organisation pour l'affichage des résultats	30
3.18	Page d'affichage des résultats	31
3.19	Exemple de comparaison de deux résultats	32
3.20	Page de recherche sur le log	34
3.21	Exemple de recherche sur le log	34

Introduction

Afin de nous exercer au travail en milieu de recherche et de mettre en pratique les connaissances acquises au cours du Master BioInformatique, dans le cadre de l'UE : Conception d'un Projet de Recherche, nous devons réaliser un projet de programmation.

Les applications de la BioInformatique dans le domaine de la biochimie métabolique sont diverses et l'offre logiciel disponible en application métabolique est large.

RegEfmtool (Regulatory elementary flux mode tool) est un outil de calcul qui combine le calcul des modes élémentaires de flux dans les réseaux métaboliques avec des régulations transcriptionnelles. Plus précisément, *regEfmtool* ne calcule que les modes de flux élémentaires qui sont conformes à la réglementation transcriptionnelle. Comme celle-ci exclut la plupart des modes mathématiquement possibles de flux élémentaires pour des raisons biologiques, *regEfmtool* permet de réduire considérablement le coût de calcul de modes de flux élémentaires. Il s'agit donc là d'un outil performant et utile pour des biologistes. Cependant, il n'est pas très convivial pour un utilisateur n'ayant pas de notions en informatique. En effet, l'environnement *UNIX* et le fait de devoir utiliser ce logiciel en ligne de commandes pourrait être rédhibitoire pour certains, ce qui serait dommage.

Pour pallier à ce problème, nous nous proposons de créer une interface graphique la plus ergonomique possible. Nous avons choisi de l'implémenter en utilisant les langages Web (au détriment des langages orientés objet).

Nous allons détailler notre approche dans la construction de cette interface. Nous verrons tout d'abord le contexte de ce projet, puis comment nous concevons cette interface et enfin comment nous l'avons réalisée.

Chapitre 1

Analyse

1.1 Contexte

1.1.1 Sujet

Le métabolisme d'une cellule est un système complexe de transformations moléculaires et énergétiques qui se déroulent de manière ininterrompue dans la cellule et mettant en jeu un ensemble de réactions dites métaboliques. Ces réactions impliquent différents types de métabolites qui, suivant leurs positions dans la réaction, sont appelés substrats ou produits et catalysés par des enzymes.

La représentation visuelle des grandes fonctions métaboliques (glycolyse ou photosynthèse par exemple) sous la forme de réseaux pourrait être un outil précieux pour faciliter l'avancement des travaux des chercheurs. Cette modélisation de réactions permet de réaliser des requêtes complexes comme, par exemple, le calcul (et la prédiction) de tous les métabolites pouvant être générés à partir d'un ensemble de composés sources.

Quelques logiciels disponibles internationalement permettent de travailler et d'automatiser l'étude de ces réseaux, tout en proposant des fonctionnalités telles que le calcul des modes élémentaires de flux ou la recherche de *minimal cut sets*¹. Cependant, ils peuvent être dépendants de logiciels non libres comme MATLAB ou alors ne pas avoir d'interface utilisateur conviviale, ce qui est le cas avec *regEfmtree* (logiciel sur lequel nous nous appuyerons pour ce projet).

1.1.2 Objectif

Dans le cadre de cette U.E., il nous était demandé de créer un programme (package et documentation à fournir à la fin du projet) nous permettant d'approfondir ou d'apprendre un langage de programmation, et de réaliser une analyse critique du travail effectué. L'objectif de ce projet était donc de réaliser une interface graphique du programme *regEfmtree* qui ne pouvait être utilisé jusqu'alors que par des commandes à l'aide d'une console.

Deux possibilités s'offraient à nous pour réaliser cette interface : nous avons le choix entre les technologies Web et le langage Java. Nous avons choisi de nous appuyer sur les technologies Web. En effet, beaucoup de programmes utilisent une interface de type site Web leur permettant d'une part de créer une interface graphique conviviale, grâce aux feuilles de style, tout en permettant une certaine modularité.

1. Un *minimal cut sets* [1] (MCS) est un ensemble minimal (irréductible) de réactions dans le réseau dont l'activation va certainement conduire à une défaillance de certaines fonctions du réseau.

1.2 État de l'existant

1.2.1 Efmtree

Efmtree [2] calcule les modes élémentaires de flux de réseaux métaboliques. Il est implémenté en Java et a été intégré à MATLAB.

Il a été développé par Marco Terzer. La version courante est la 4.7.1 (Décembre 2009).

1.2.2 RegEfmtree

RegEfmtree [3] est un outil informatique qui combine le calcul des modes élémentaires de flux et la régulation transcriptionnelle du réseau métabolique. Il a été développé, entre autres, par Christian Jungreuthmayer. Il a été créé afin d'accélérer le calcul de jeux complets de modes élémentaires de flux d'un réseau métabolique.

RegEfmtree est une extension d'Efmtree qui prend en compte la régulation transcriptionnelle des réseaux pour le calcul des modes élémentaires de flux.

La prise en compte de la régulation des gènes réduit de façon importante le nombre de solutions et permet d'éliminer constamment les modes qui ne peuvent exister biologiquement pendant et après le processus de calcul. Elle permet aussi de réduire considérablement le coût du calcul.

L'installation et l'utilisation de *regEfmtree* a été exclusivement testée sous Linux. Elle pourrait cependant fonctionner sous d'autres systèmes d'exploitation puisqu'il s'agit d'un programme Java. Il n'existe pas d'interface graphique de cette application, elle s'exécute donc en lignes de commandes via le terminal. La version courante de *RegEfmtree* est la 2.0 (Août 2012).

1.2.3 METATOOL

METATOOL [4] est un programme écrit en C développé de 1998 à 2000 par Thomas Pfeiffer (Berlin) en coopération avec Juan Carlos Nuno (Madrid), Stefan Schuster (Berlin) et Ferdinand Moldenhauer (Berlin).

Il sert à étudier la structure des réseaux métaboliques à partir d'équations de réactions stoechiométriques et permet notamment de calculer les modes élémentaires.

Les premières versions de METATOOL (jusqu'à la 4.9) ont été développées en C. Aujourd'hui, nous trouvons aussi une version de METATOOL en C++ mais cette version n'est pas au point. Dans la version actuelle (5.1) l'exécutable est désormais un module de MATLAB 7 et GNU 3.0 Octave, il se présente sous la forme d'un ensemble de fichiers scripts de MATLAB.

Les paramètres donnés en entrée pour le bon fonctionnement du logiciel METATOOL sont les suivants :

- La liste des réactions réversibles, ainsi que celle des réactions irréversibles, avec le nom des réactions,
- La liste des métabolites internes et externes impliqués dans les réactions,
- Les équations réactionnelles.

Le tout est rassemblé dans un fichier avec l'extension *.dat*

A la fin de son exécution, METATOOL a généré un fichier avec l'extension *.out* dans lequel se trouvent les résultats. Dans les versions de METATOOL écrites en C, le fichier de sortie contient l'ensemble des résultats sous forme de matrices, ainsi que des bilans qui permettent de décrire le réseau d'étude.

Les versions de METATOOL écrites en MATLAB produisent des résultats similaires en terme de calcul des matrices des modes élémentaires mais les résultats sont disposés différemment dans le fichier de sortie.

1.2.4 CellNetAnalyzer

CellNetAnalyzer [5] est un package de MATLAB (écrit en C) qui fournit un environnement compréhensible et convivial pour l'utilisateur et qui permet une analyse fonctionnelle et structurale de réseaux biochimiques. Il a été développé à l'institut Max Planck de Magdeburg par Steffen Klamt (depuis 2000) et Axel von Kamp (depuis 2007) notamment.

CellNetAnalyzer fournit une importante collection d'outils et d'algorithmes pour l'analyse structurale de réseaux.

C'est un programme gratuit pour une utilisation académique. Pour l'exécuter, il faut avoir installé MATLAB 7.0 ou une version ultérieure qui demande une licence. Il peut être utilisé sur Linux, Windows XP ou Mac.

Pour l'étude des modes élémentaires, CellNetAnalyzer fait appel à METATOOL via le logiciel MEX qui sert d'interface. MEX permet à MATLAB d'appeler tout logiciel C externe pour compléter les outils qu'il possède.

1.2.5 Yana

YANA [6] est un logiciel libre, écrit en JAVA, utilisant METATOOL pour l'étude des voies métaboliques. Il contient le logiciel METATOOL dans sa structure interne.

YANA [7] sert de façade et de sortie à METATOOL 6 tout en implémentant d'autres fonctions d'analyses du métabolisme (ex : quantification de l'activité enzymatique des réactions).

YANA s'occupe de l'entrée des données vers METATOOL puis il effectue une analyse syntaxique du fichier *.out* pour afficher les résultats sur une interface graphique et effectuer des analyses complémentaires sur ceux-ci.

1.2.6 Acom

ACoM [8] fonctionne à partir des fichiers de sortie de METATOOL. Il permet une classification automatique des modes élémentaires en fonction d'une taille minimale et d'un seuil de similarité. Il est surtout utilisé pour l'analyse des réseaux de grandes tailles où la manipulation des données à la main est laborieuse. ACoM est un programme C utilisable uniquement en mode console.

1.2.7 JACoMode

JACoMode est une interface Web permettant le lancement d'ACoM via le Web. En plus d'obtenir les résultats d'ACoM, JACoMode traite ces résultats pour obtenir d'autres résultats statistiques sur les modes élémentaires.

1.2.8 Langages

Notre projet nécessite l'utilisation d'une interface Web, dans ce cadre il existe :

- Mod Perl combiné avec Apache² et CGI³ mais la technologie utilisée est à l'heure actuelle dépassée
- Mod Python combiné avec Apache et CGI mais même remarque que précédemment
- CL-WHO avec Hunchentoot⁴ et ParenScript offre un bon environnement pour le développement Web

2. Apache HTTP Server

3. CGI : Common Gateway Interface.

4. Hunchentoot HTTP Server

- Java et ses applets⁵ apparaissent également comme un bon choix pour le développement Web
 - PHP⁶ couplé avec du JavaScript peut être un choix judicieux pour une application Web
- Parmi les différents choix précédemment cités, deux sortent du lot : Java et PHP avec leurs bibliothèques.

5. Applet Java : logiciel s'exécutant dans la fenêtre d'un navigateur Web (grâce à une machine virtuelle Java (JVM)), fournie aux utilisateurs sous la forme de bytecode Java.

6. PHP : Hypertext Preprocessor

1.3 Analyse des besoins

1.3.1 Besoins fonctionnels

Interface Homme Machine (IHM)

L'interface Web que nous avons réalisé permet de créer un nouveau réseau métabolique pour *regEfmTool* ou pour *METATOOL*.

Elle donne également le moyen à l'utilisateur de saisir les fonctions et options qui l'intéressent puis de lancer les calculs des modes élémentaires de flux du réseau d'intérêt.

Cette interface est disponible en trois langues (Français, Anglais, Allemand), via des icône sur le coté droit du menu accessible à tout moment sur chacune des pages .

Création d'un nouveau réseau

Si l'utilisateur clique sur le bouton de création d'un nouveau réseau métabolique, une première page Web s'affiche. Il peut alors initialiser les fichiers présents sur son disque dur et écrire les réactions qui composent son réseau. Il a aussi la possibilité de modifier une réaction (métabolites, coefficients stœchiométriques). Lorsqu'il ajoute ou modifie une réaction, les données sont récupérées afin de générer automatiquement les fichiers nécessaires au fonctionnement du logiciel. L'utilisateur peut aussi générer un fichier *DAT* s'il le désire.

Lorsqu'il choisi de passer à l'étape suivante, il peut créer les règles des gènes qui seront utilisées par *regEfmtool* pour calculer les modes élémentaires.

Réglage des paramètres de lancement

Il est possible pour un utilisateur confirmé ou habitué à l'interface d'avoir accès à un mode de réglage avancé des paramètres de lancement, s'il le désire. Ces derniers sont fixés à des valeurs par défaut pour les débutants.

Chargement d'un réseau préexistant

Si l'utilisateur clique sur le bouton de chargement d'un réseau depuis la page d'accueil, il doit charger une série de fichiers (depuis le disque dur de son ordinateur) nécessaires au bon fonctionnement du logiciel.

Lancement du programme

Lorsque l'utilisateur a créé son réseau manuellement, il doit ensuite choisir les paramètres de calcul de *regEfmtool*. Nous avons fait le choix d'utiliser des cases à cocher en fonction de ce qu'il choisi. Pour les utilisateurs non expérimentés, les choix de base seront pré-sélectionnés. Il suffira ensuite de cliquer sur le bouton *Lancement* pour avoir les résultats générés par le logiciel.

Résultats

Enfin, la visualisation des résultats apparaît de façon claire et conviviale à l'utilisateur au travers de l'interface Web. De plus, les résultats sont également présents dans des fichiers pour une analyse ultérieure.

Aide en ligne

Des pages d'aide sont aussi consultables pour la création d'un nouveau réseau. Sur chaque page un lien hypertexte fait apparaître une page d'aide indiquant à l'utilisateur le fonctionnement de la page où il se trouve.

Ces pages d'aide sont également disponibles en trois langues (Français, Anglais, Allemand).

1.3.2 Besoins non fonctionnels

Portabilité

L'utilisation de *regEfmtool* s'appuie sur d'autres logiciels, nécessitant par exemple la version 1.7 de Java. De ce fait, ils sont libres d'utilisation pour le secteur académique. L'interface est livrée avec tous les fichiers de configuration (pré-existants ou nouvellement créés) et indépendante du système d'exploitation.

Documentation

L'écriture du code est constituée de commentaires qui permettent la maintenance du code ainsi qu'une éventuelle amélioration de ce dernier par un tiers.

L'interface Web créée, quand à elle, est fournie avec une documentation sur son installation et son utilisation disponible directement sur la page d'accueil du site..

Chapitre 2

Conception

2.1 Interface graphique

Comme nous l'avons précisé précédemment, notre interface graphique se présente sous la forme d'un site Web.



FIGURE 2.1 – Page d'accueil

Lorsqu'un utilisateur arrive sur le site, une page d'accueil s'affiche (Figure 2.1). Elle contient des informations relatives à l'utilisation du programme (à quoi il sert, que peut-on y faire ...). A partir de là, deux choix s'offrent à lui :

- Soit il crée un nouveau réseau, en cliquant sur *Création* dans le menu situé en haut de page,
- Soit il charge un réseau qu'il a déjà créé en cliquant sur *Chargement*.

Chaque page du site se présente de la même façon : un menu en haut de la page et le logo du site Web dans un bandeau situé sur la gauche.

Le menu contient quatre onglets :

- *Accueil* qui permet de retourner sur la page d'accueil du site Web,
- *Création* qui permet de créer un nouveau réseau métabolique,
- *Chargement* qui permet de charger un réseau pré-existant,
- *Aide* qui permet de consulter une aide à l'utilisation du site.

On trouve aussi dans le menu, trois drapeaux qui permettent de changer la langue du site Web. Les trois langues proposées sont : l'anglais, le français et l'allemand. Par défaut le site est en français.

Nous avons également ajouté un lien vers une page contenant le *copyright* de notre site, et un autre menant à une page de précisions sur la configuration requise pour l'utilisation de

l'interface.

2.2 Création d'un nouveau réseau

Si l'utilisateur choisi de créer un nouveau réseau, il arrive sur une page qui va lui permettre d'ajouter de nouvelles réactions à celui-ci.

Une première partie permet d'initialiser les fichiers (Figure 2.2) dans le cas où l'utilisateur aurait déjà créé un réseau précédemment.

W R E T B
REG EFM TOOL

Accueil Création Chargement Aide

Création d'un nouveau réseau

Initialisez vos fichiers avant de créer un nouveau réseau

Initialiser fichiers

Ecrivez une nouvelle réaction

Aide

reaction : reag1 + reag2 => 2 prod1 + 4 prod2 .

Réversible :

☐ Oui ☐ Non

Attention !

Pas d'espaces dans les noms, utiliser un underscore '_' à la place
La syntaxe des réactions est la suivante :
Nom_reaction : 2 reactif1 + 5 reactif2 => 3 produit1 + 7 produit2
Attention a laisser un espace entre le coefficient stoechiométrique et le métabolite associé

Ajouter

Réactions déjà entrées :

reaction : reag1 + reag2 => 2 prod1 + 4 prod2 .
reaction2 : reag1 + reag2 => 2 prod1 + 4 prod2 .

Attention !

Vous ne pouvez ni supprimer ni ajouter de réaction dans cette zone de modification
Il ne doit pas y avoir d'espace après le dernier métabolite

Modifier

FIGURE 2.2 – Page de création d'un nouveau réseau

Une seconde partie permet de créer une nouvelle réaction. Celle-ci doit être de la forme $reaction : reag1 + reag2 \Rightarrow 2prod1 + 4prod2$.

L'utilisateur doit également préciser si la réaction est réversible ou non. Ensuite il clique sur le bouton *Ajouter*, la réaction est alors ajoutée au réseau et apparaît dans la zone de texte située en-dessous. Dans cette zone de texte, l'utilisateur a la possibilité de modifier les réactions déjà créées (mais ne peut pas en ajouter ou en supprimer car les règles de réversibilités ne seraient alors plus respectées pour l'écriture d'un fichier au format *DAT*). Il valide ensuite les modifications apportées en cliquant sur le bouton *Modifier*.

Enfin, l'utilisateur peut aussi générer un fichier au format *DAT* (Figure 2.3) qui pourra être utilisé dans METATOOL par exemple.

Si l'utilisateur ne sait pas comment faire, il lui suffit de cliquer sur *Aide* et une nouvelle fenêtre de navigation (Figure 2.4) apparaîtra afin de le guider. Cette page est automatiquement traduite dans la langue de la page courante.

Quand l'utilisateur a fini de créer toutes les réactions qui composent son réseau, il clique sur le bouton *Étape suivante* et arrive alors sur la page de création des règles des gènes.

FIGURE 2.3 – Suite de la page de création

FIGURE 2.4 – Page d'aide à la création d'un nouveau réseau

2.3 Règles des gènes

Ces règles sont utilisées pour éliminer les modes élémentaires non possibles dans un réseau métabolique réel.

Une réaction R peut être 1-active ($R=1$), 0-active ($R=0$) ou encore full-active ($R=f$).

La création des règles est régie selon certains principes :

- La réaction de sortie ne peut jamais être une réaction d'entrée.
On ne peut pas écrire : $R3 = (! (! 0R3) | 1R1)$
- Une réaction peut être utilisée plusieurs fois comme réaction d'entrée.
Ex : $R11 = ((! (! 0R3) | 1R5)) \& 1R3$
- Le préfixe d'une réaction n'est valable que pour une opération.
Ainsi dans la règle $R11 = ((! (! 0R3) | 1R5)) \& 1R3$, la réaction $R3$ est 0-active dans l'opération OR et 1-active dans l'opération AND.
- Chaque réaction doit être entourée d'exactly une paire de parenthèses.
 $R3 = (! 0R1)$ est correcte, mais $R3 = ((! 0R1)$ et $R3 = ! 0R1$ sont incorrectes
- Il n'existe pas d'opération permettant de représenter la relation $R2 = R1$, elle sera représentée de la façon suivante : $R2 = (fR1 | fR1)$.

Tout d'abord, l'utilisateur choisit le nombre de réactions qui composent la règle qu'il veut créer. Une fois ce choix effectué, il clique sur le bouton *Ok*. S'affichent alors de deux ou trois menus déroulant par réaction. Quand il y en a trois (Figure 2.5), le premier permet de choisir l'opérateur, le second la réaction et le dernier sa valeur (0, 1 ou f). Quand il y en a deux, ils correspondent à la réaction et à sa valeur. L'utilisateur doit entrer un nombre de réactions au moins égal à deux.

FIGURE 2.5 – Page de création des règles des gènes

L'utilisateur ne peut sélectionner le nom d'une réaction que lorsqu'il a choisi la précédente. Ceci permet de ne proposer que les réactions non sélectionnées précédemment pour le choix de la réaction de sortie (principe utilisé dans la documentation de *regEfmtool*).

Enfin, quand l'utilisateur a sélectionné toutes les réactions ainsi que leur valeur, il clique sur le bouton *Ajouter*. La règle est alors écrite dans le fichier *generules.grfile* si l'utilisateur a bien rempli tous les champs. Celle-ci apparaît alors dans la zone de texte située en-dessous. Dans cette zone, l'utilisateur a la possibilité de modifier, ajouter ou supprimer une règle déjà écrite, mais également d'en écrire de nouvelles manuellement. Lorsqu'il a fini ses modifications, il clique sur le bouton *Modifier*, le fichier *generules.grfile* est alors modifié en conséquence.

FIGURE 2.6 – Page d'aide à la création des règles des gènes

Si l'utilisateur ne sait pas comment faire, il lui suffit de cliquer sur *Aide* comme précédemment et une nouvelle fenêtre de navigation (Figure 2.6) apparaîtra afin de la guider. Cette page est également automatiquement traduite dans la langue de la page courante.

Une fois les règles entrées, l'utilisateur peut passer à l'étape suivante en cliquant sur le bouton correspondant. Il arrive alors sur la page de sélection des options nécessaires au lancement de la commande *regEfmtool*.

2.4 Choix des options de lancement

Comme nous l'avons dit précédemment, *regEfmtree* se lance grâce à une ligne de commande. Cette page d'options (Figure 2.7) permet donc à l'utilisateur de sélectionner les paramètres de calcul qu'il souhaite, par la biais de boutons à cocher. Certaines options sont pré-cochées, se sont les options de base nécessaires au bon déroulement des calculs du logiciel. Cela permet à un utilisateur non expérimenté de lancer ses calculs avec le réseau qu'il vient de créer. Un utilisateur plus averti pourra modifier ces options à sa guise afin d'avoir un niveau de complexité d'exécution des calculs supérieur.

FIGURE 2.7 – Partie de la page de choix des options de lancement

Les options sont regroupées par catégories (en fonction de ce qui était précisé dans le fichier *metabolic-efm.xml* de *regEfmtree*) :

Enregistrement Cette catégorie contient trois sous catégories précisant le type d'affichage des résultats, le niveau d'information sur le déroulement des calculs et le format d'enregistrement. Par défaut, l'affichage se fera en ligne sur le site et le type de message d'information sur le déroulement des calculs sera le plus complet possible.

Types de fichiers stoechiométriques Cette catégorie possède deux sous catégories précisant le type de passage en entrée et l'analyseur de flux. Cette dernière permet de définir les types de fichiers d'entrée que *regEfmtree* utilisera. Ces fichiers étant créés lors des étapes précédentes, leurs noms sont déjà pré-remplis dans les cases correspondantes.

Compression et paramètres de sortie Ces deux catégories contiennent chacune une seule sous catégorie, servant respectivement à définir le niveau de compression (et également l'utilisation de la récursivité dans les calculs) et le type de fichier désiré en paramètre de sortie. L'option "text-doubles" est cochée par défaut, avec le nom du fichier qui contiendra les modes élémentaires calculés. Nous avons appelé ce fichier *results.txt*.

Paramètres d'Efmtree Cette dernière catégorie contient sept sous catégories : le "rowordering" (ou la ligne de commandes à utiliser), la méthode d'adjacence, le nombre maximal de "threads" (prédéfini à 2 suivant les exemples de *regEfmtree*), l'arithmétique des nombres, le

type de normalisation pour la sortie (prédéfinie à "aucune"), la précision fractionnaire et l'activation ou non d'un auto-test après chaque itération.

Une fois que l'utilisateur a coché toutes les options qu'il souhaite, il lui suffit de cliquer sur le bouton *Lancement* afin de générer la commande et lancer *regEfmtreeol*. Ensuite, une page d'affichage des résultats s'affiche.

2.5 Chargement d'un réseau pré-existant

2.5.1 Choix des paramètres

Cette page, accessible depuis le menu général, permet à un utilisateur de charger un réseau préexistant depuis son disque dur (Figure 2.8). Il pourra également générer la ligne de commandes nécessaire au lancement de *regEfmtreeol*. Cette page est basée sur le modèle de la page du choix des options après création, à un détail près : dans la catégorie du type des fichiers stoechiométriques, il n'y a plus la sous-catégorie des fichiers d'analyseur de flux. Le reste fonctionne de la même façon que précédemment.

FIGURE 2.8 – Partie de la page de chargement d'un réseau préexistant

L'utilisateur n'aura plus qu'à cliquer sur le bouton *Étape suivante* pour générer le début de la ligne de commande. Tous les paramètres sauf ceux concernant les fichiers d'entrée seront alors récupérés. L'utilisateur est renvoyé sur une nouvelle page qui permet de charger ses fichiers depuis son disque dur et de lancer le logiciel.

2.5.2 Chargement des fichiers

Cette page permet à l'utilisateur de sélectionner les fichiers à charger depuis son disque dur (Figure 2.9). Il y a un ordre précis du type des fichiers à charger (indiqué à l'utilisateur).

Une fois les fichiers sélectionnés, il suffit de cliquer sur *Submit* pour effectuer le chargement.

2.6 Affichage des résultats

La page des résultats permet deux choses : l'exécution de *regEfmtreeol* et l'affichage des modes élémentaires, obtenus par le parse du fichier généré lors de l'exécution du logiciel.



FIGURE 2.9 – Page du choix des fichiers à charger

Comme nous l'avons dit précédemment, la commande d'exécution de *regEfmtool* générée en partie depuis la page de choix des options est récupérée sur la page des résultats pour permettre son exécution.

L'exécution génère deux types de résultats : le fichier contenant les modes élémentaires ainsi que le \log^1 s'il est demandé.

Une fois l'exécution terminée, les résultats précédents seront affichés.

Les modes élémentaires quant à eux sont affichés sous la forme d'un tableau contenant :

- En abscisses, les modes élémentaires obtenus
- En ordonnées, toutes les réactions pouvant être impliquées

Cette page contient aussi trois boutons :

- l'un permet de finir les calculs et de revenir à la page d'accueil,
- le deuxième permet de revenir à la page d'accueil, mais pour analyser un nouveau réseau et ainsi comparer les deux résultats générés,
- le troisième permet d'afficher l'outil de recherche sur le log et de sélectionner la catégorie que l'utilisateur veut afficher ou la totalité.

La page d'extraction des résultats offre à l'utilisateur la possibilité de visualiser l'ensemble des résultats générés par *regEfmtool* ou de pouvoir en extraire une partie à l'aide d'un mot clé sélectionné dans un menu déroulant (Figure 2.10).

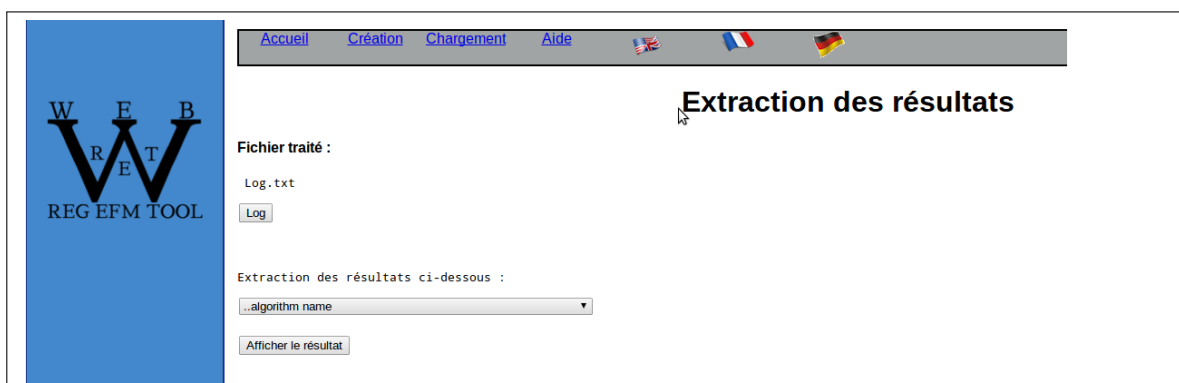


FIGURE 2.10 – Page de recherche sur le log

1. Un fichier log est un fichier texte classique, reprenant de façon chronologique l'ensemble des événements qui ont affecté un système informatique et l'ensemble des actions qui ont résulté de ces événements.

Chapitre 3

Réalisation

Nous allons intégrer des digrammes d'organisation des pages Web le long de notre explication afin d'avoir un support visuel. La légende est représentée par la Figure 3.1.

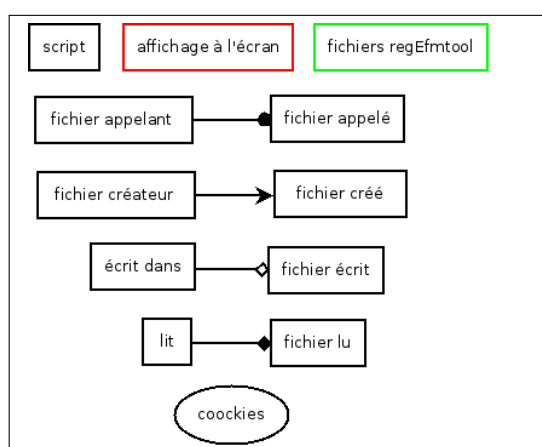


FIGURE 3.1 – Légende de nos diagrammes d'organisation

3.1 Ergonomie de l'interface

3.1.1 Mise en page

Nous avons utilisé le CSS afin de régler la mise en page de notre interface. Nous avons intégré une bande verticale sur la gauche du site, contenant un logo de notre création. Nous avons dû adapter la zone d'affichage en fonction de cette bande, mais également faire attention à ce que l'affichage soit ajusté à la taille de l'écran de l'utilisateur. Ce dernier critère était très important, notamment pour l'affichage du menu des onglets en haut des pages, mais aussi pour celui des pages de choix d'options de lancement.

Exemple de la page du choix des options Cette page est composée d'un emboîtement de plusieurs sections, grâce à la balise HTML `<div>`, qui est une sorte de conteneur. Comme nous l'avons précisé précédemment, nous avons séparé les options en plusieurs catégories et sous-catégories, distinguées par une combinaison de dégradé de bleus.

Nous avons pris l'exemple des paramètres d'enregistrement des fichiers avec trois sous-catégories, contenu dans une *division*.

- L'intitulé de la catégorie générale (*Enregistrement*) est en bleu foncé (le même que celui de la bande verticale) et représente une sorte de *sous-division*.

Enregistrement		
Affichage des résultats	Type de messages d'erreurs / Niveau d'informations sur le déroulement des calculs	Format/méthode de l'enregistrement
<input checked="" type="radio"/> En ligne sur WRET <input type="radio"/> Dans un fichier : Nom du fichier : <input type="text"/>	<input type="radio"/> Warning <input type="radio"/> Config <input type="radio"/> Info <input type="radio"/> Fine <input type="radio"/> Finer <input checked="" type="radio"/> Finest	<input type="radio"/> Par défaut <input checked="" type="radio"/> Complet

FIGURE 3.2 – Exemple d'organisation d'une catégorie

- Les titres des trois sous catégories (*Affichage des résultats*, *Type de messages d'erreurs* et *Méthode d'enregistrement*) sont en bleu clair et représentent également une *sous division*. Cette dernière est aussi divisée en trois sortes de *sous-sous-division* (en orange).
- Les contenus correspondants aux trois sous catégories sont également divisés en trois *sous-sous-division* (en vert).

Il a fallu ajuster la taille de ces sous catégories de manière relative à la taille de l'écran. Voici un exemple du code CSS correspondant :

```

1  div#part {
2      text-align: center;
3      width: 100%;
4      height: 40px;
5      background-color: #4388CC;
6      margin-top: 30px;
7      margin-bottom: 0px;
8      padding: 0px;}
9  div#subPart {
10     margin-bottom: 0px;
11     margin-top: 0px;
12     width: 100%;
13     height: 80px;
14     background-color: #9ECAE1;
15     padding: 0px;}
16  div#subPart3-2 {
17     float: left;
18     width: 33.3%;
19     height: 60px;
20     background-color: #9ECAE1;
21     text-align: center;}
22  div#buttons {
23     width: 100%;
24     height: 175px;
25     background-color: #DEEBF7;
26     clear: both;}
27  div#buttons3-2 {
28     float: left;
29     width: 33.3%;
30     height: 140px;
31     background-color: #DEEBF7;
32     text-align: justify;
33     margin-right: -10%;
34     margin-left: 10%;}

```

La `div#part` est utilisée pour le titre de la catégorie. Sa hauteur est fixe mais sa largeur s'adapte à celle de l'écran. La `div#subPart` sert de conteneur à `div#subPart3-1`, `div#subPart3-2` et `div#subPart3-3` qui sont en vert (ici, seul le code relatif à `div#subPart3-2` est montré pour des raisons d'économie de place car les 3 sont très semblables). De la même manière, `div#buttons` contient `div#buttons3-1` `div#buttons3-2` `div#buttons3-3` qui sont en orange.

3.1.2 Langues

Dans un soucis de convivialité, notre interface est développée de manière à pouvoir être traduite aisément et propose trois langues possibles dans sa version actuelle (français par défaut, anglais et allemand). Pour changer la langue, l'utilisateur n'a qu'à cliquer sur le drapeau correspondant dans le menu en haut de page. Cela fera appel au script *choosen_languages.php* (Figure 3.3), qui lui-même fait appel à *de_lang.php* pour l'allemand, *en_lang.php* pour l'anglais et *fr_lang.php* pour le français, en fonction du drapeau sélectionné.

Le fichier *choosen_languages.php* vérifie soit dans la barre d'adresse soit dans les variables de session php la langue sélectionnée par l'utilisateur.

```

1 session_start();
2 if (isset($_SESSION['lang']) && !isset($_GET['lang'])) {
3     $lang=$_SESSION['lang'];
4 }
5 else if (isset($_GET['lang'])) {
6     $lang=$_GET['lang'];
7     $_SESSION['lang']=$lang;
8 }

```

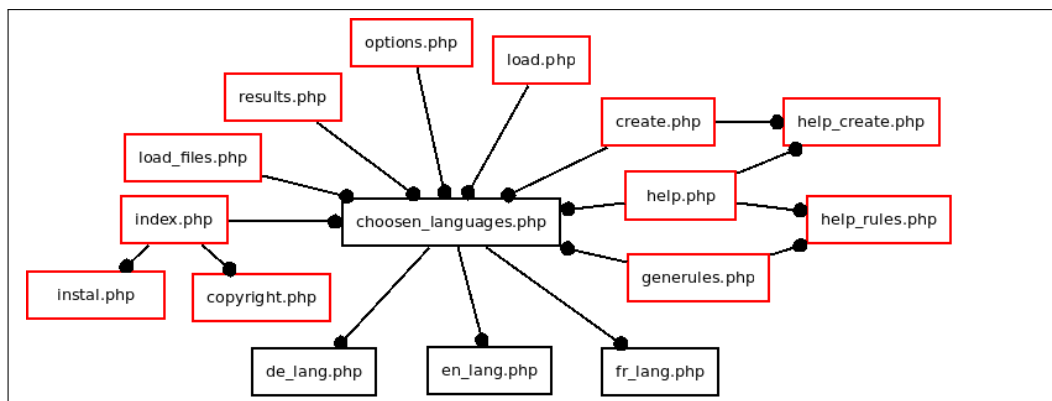


FIGURE 3.3 – Diagramme d'organisation pour le changement de langue

Pour ce faire, nous n'avons pas mis de texte dans le code des pages affichant à l'écran. Nous l'avons remplacé par des variables, qui afficheront le texte qu'elles contiennent dans la langue désirée. Par exemple, le titre de la page d'accueil "Bienvenue sur WebRegEfmTool" est appelé par `<?php echo TXT_HOMEPAGE_TITLE; ?>` et est contenu dans le fichier *fr_lang.php* de la manière suivante : `define('TXT_HOMEPAGE_TITLE', 'Bienvenue sur WebRegEfmTool');`. De ce fait, un nouveau fichier du type *XX_lang.php* pourrait être créé afin d'intégrer une nouvelle langue. Bien sûr, il faudrait ajouter les appels à ce fichier dans toutes les pages affichant à l'écran par la suite.

3.2 Création d'un nouveau réseau

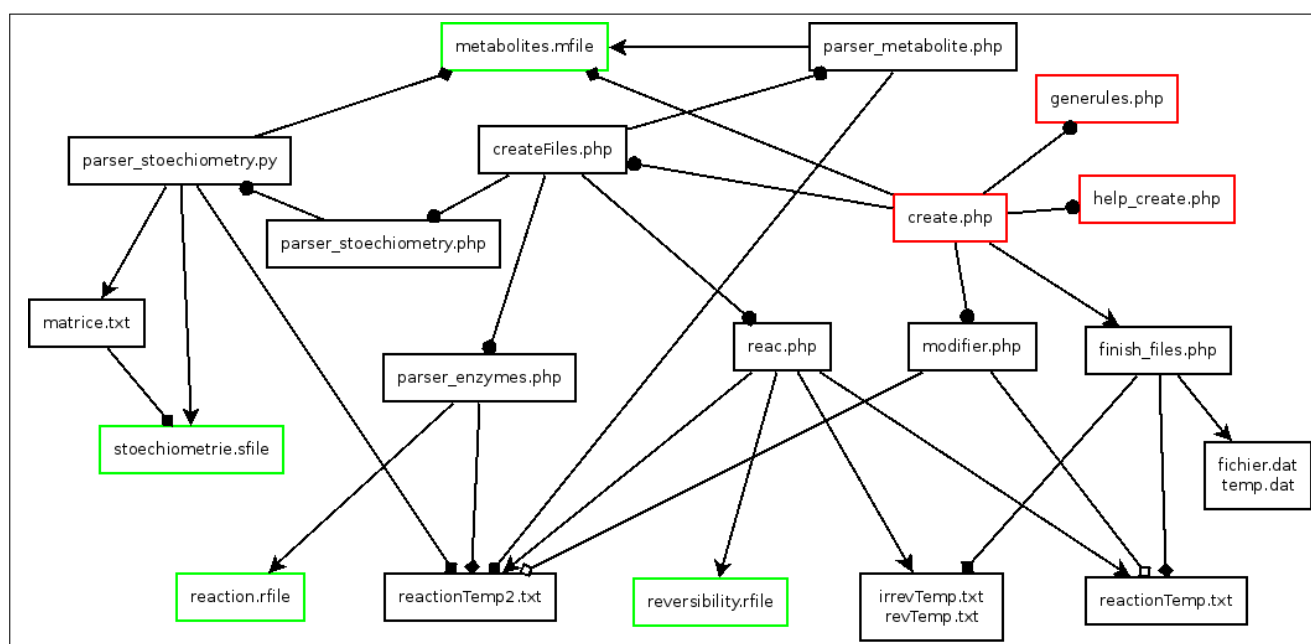


FIGURE 3.4 – Diagramme d'organisation pour la création d'un nouveau réseau

3.2.1 Initialisation des fichiers

La création d'un nouveau réseau dans WRET se fait via la page *create.php*. A partir de cette page (Figure 3.4), l'utilisateur doit dans un premier temps appuyer sur le bouton *Initialiser les fichiers* (Figure 3.5) s'il désire initialiser ses fichiers.

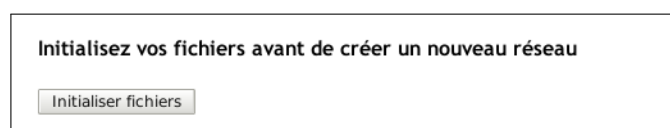


FIGURE 3.5 – Bouton d'initialisation des fichiers

Ce bouton appelle le fichier *initfiles.php* qui va créer les 11 fichiers nécessaires à la mise en place d'un nouveau réseau.

Dans ces fichiers, nous trouvons des fichiers qui seront utilisés dans le lancement de *regEfntool* (rfile, mfile, rvfile, sfile) et également des fichiers temporaires (*irrevTemp*, *revTemp*, *reactionTemp*, *reactionTemp2*, *matrice*, *matrice2*) ainsi que la base du fichier au format *DAT*.

Tous ces fichiers sont donc créés et donnent les droits d'édition, de lecture et d'exécution à tous les utilisateurs pour chacun d'entre eux, afin de pouvoir être modifiés sur le serveur.

3.2.2 Réactions et réversibilité

Sous la touche *Initialiser les fichiers* de la page *create.php*, une fenêtre de texte permet de rentrer les réactions du réseau métabolique une à une ainsi que la réversibilité de la réaction (Figure 3.6).

Ecrivez une nouvelle réaction

[Aide](#)

reaction : reag1 + reag2 => 2 prod1 + 4 prod2 .

Réversible :

☐ Oui ☐ Non

FIGURE 3.6 – Bouton d’ajout d’une réaction

Si l’utilisateur oublie de cocher la réversibilité de la réaction et clique sur *Ajouter*, un message d’erreur s’affichera et empêchera le passage à l’étape suivante. Les réactions doivent être enregistrées par l’utilisateur en respectant la syntaxe des fichier au format *DAT*. Ces informations vont alors être envoyées au fichier *createFiles.php*, via le bouton *Ajouter*.

Ce fichier redirige vers différentes pages, dans l’ordre : *reac.php*, *parser_enzyme.php*, *parser_reversibility.php*, *parser_metabolite.php*, *parser_stoechiometry.php*. La première page *reac.php* va permettre d’écrire dans un fichier temporaire (*reactionTemp.txt*) les réactions, et également, de sauvegarder la réversibilité de la réaction (0 pour non réversible, et 1 pour réversible).

L’ordre est ainsi conservé entre les réactions et leur réversibilité.

3.2.3 Nom de réactions : enzymes

Une fois les données de réactions et de réversibilité enregistrées, la page *parser_enzymes.php* est appelée. Elle va parser le fichier temporaire des réactions (*reactionTemp.txt*) et va extraire le premier élément de la réaction situé avant le " : ", qui se trouve être le nom de la réaction (nom de l’enzyme généralement).

Ces noms sont enregistrés dans un fichier (*reactions.rfile*) en respectant les espaces et la syntaxe nécessaires à son utilisation au sein de *regEfmtree*.

3.2.4 Métabolites

Après enregistrement des enzymes, le fichier *parser_metabolites.php* est appelé. Ce script va parser le fichier temporaire contenant les réactions (*reactionTemp.txt*) et va enregistrer chacun des métabolites dans un fichier (*metabolites.mfile*). Au cours de ce passage, seuls les éléments situés après le nom de l’enzyme seront pris en compte. Les noms présents plusieurs fois dans le fichier de réactions sont enregistrés une seule fois dans le fichier *metabolites.mfile*.

3.2.5 Stœchiométrie

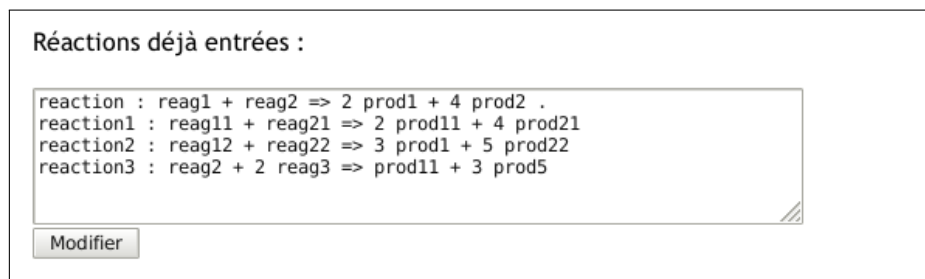
Enfin après génération des fichiers : *rvfile*, *mfile*, *sfile*, *rfile*, le script *parser_stoechiometry* est appelé. Il va lancer le script *parser_stoechiometry.py*. Ce dernier permet de générer la matrice de stœchiométrie nécessaire à *regEfmtree*. Pour ce faire, il prend les fichiers *reactionTemp.txt* et *metabolites.mfile* en entrée. Il génère la matrice ligne par ligne (une ligne correspondant à une réaction).

Pour chaque ligne du fichier *reactionTemp.txt*, une liste est créée et, pour chaque métabolite de

cette réaction, sa stœchiométrie est enregistrée en respectant son ordre dans le fichier contenant les réactifs. Ce script fourni alors en sortie le fichier *stoechiometry.sfile*.

3.2.6 Modification du réseau

Une fois les fichiers générés, l'utilisateur est redirigé vers la page *create.php*. Sous la touche *Ajouter* se trouve une zone de texte (Figure 3.6) où l'utilisateur peut modifier un réseau déjà rentré.



```

Réactions déjà entrées :

reaction : reag1 + reag2 => 2 prod1 + 4 prod2 .
reaction1 : reag11 + reag21 => 2 prod11 + 4 prod21
reaction2 : reag12 + reag22 => 3 prod1 + 5 prod22
reaction3 : reag2 + 2 reag3 => prod11 + 3 prod5

[Modifier]
```

FIGURE 3.7 – Bouton de modification

En effet, cette zone appelle le fichier *reactionTemp.txt* et permet la modification de son contenu (pour corriger une erreur de frappe notamment). Cette zone de texte va appeler la page *modifier.php* lors de l'utilisation du bouton *Modifier*.

Cette page efface le précédent fichier *reactionTemp.txt* et insère le nouveau contenu que l'utilisateur a modifié.

Les fichiers *metabolites.mfile*, *stoechiometry.sfile*, *reactions.rfile*, *reversibility.rvfile*, sont également générés à nouveau, à chaque modification du fichier *reactionTemp.txt*.

3.2.7 Création du fichier *DAT*

En dessous de la zone de texte modifiable, une touche nommée *Choisir les métabolites internes et externes* (Figure 3.8) permet à l'utilisateur de définir dans son réseau les métabolites internes et externes. Ce bouton va effectuer un passage du fichier *metabolites.mfile* et afficher dans une colonne à gauche tous les métabolites. Une colonne à droite vide peut être remplie ou vidée via les boutons *Ajouter*, *Supprimer* situés entre les colonnes. Ces touches d'ajout et de suppression font appelle à des fonctions *JavaScript*, directement intégrées à la page *create.php*.

Sous la zone de sélection des métabolites internes et externes, un bouton *DAT* permet de générer un fichier au format *DAT* du réseau métabolique précédemment créé.

Ce bouton (Figure 3.9) appelle la page *finish_files.php* qui va récupérer les choix de l'utilisateur en ce qui concerne les métabolites internes et externes, et concaténer trois fichiers :

- *irrevTemp.txt*
- *revTemp.txt*
- *reactiontemp2.txt*

Le premier fichier contient l'ensemble des enzymes catalysant les réactions irréversibles. Le second contient l'ensemble des enzymes catalysant les réactions réversibles, et enfin le dernier fichier contient l'ensemble des réactions du réseau.

Chacun de ces fichiers contient les balises *-ENZIRREV*, *-ENZREV*, *-CAT*, selon son contenu et respecte la syntaxe propre au format *DAT*. Les noms des métabolites sont insérés dans le

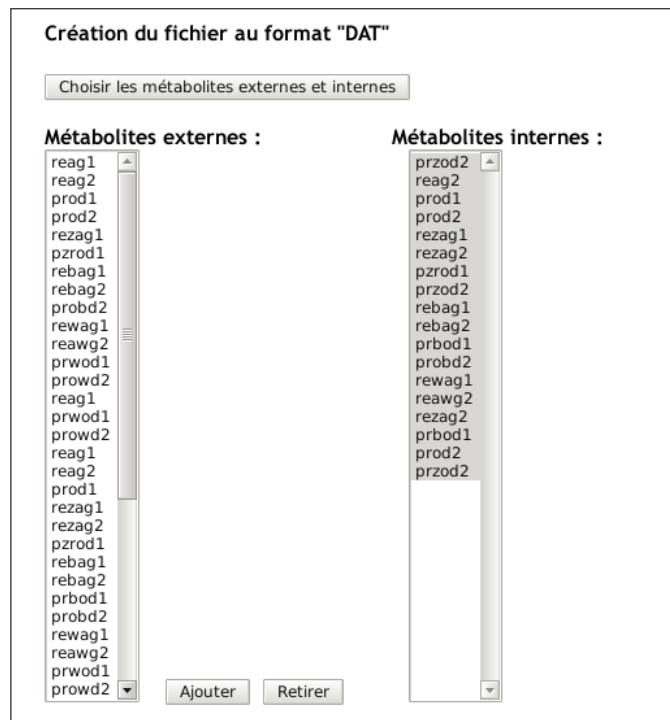


FIGURE 3.8 – Zone de sélection des métabolites internes et externes

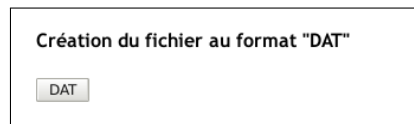


FIGURE 3.9 – Bouton de création du fichier au format *DAT*

fichier au format *DAT* après une balise *-METINT* ou *-METEXT*, selon leur rôle dans le réseau défini par l'utilisateur.

Il en résulte en sortie du script *finish_files.php* un fichier au format *DAT*, de la forme :

```

1 -ENZREV
2 R9 R12 R13 R14 R15 T1 T2 T5 T7 T12
3
4 -ENZIRREV
5 R6i R7i R8i R10i R11i T6
6
7 -METINT
8 OAA ACoA Cit Akg SucCoA Succ Fum Mal Isocit Pi Glu Asp Pyr
9
10 -METEXT
11 Pyr_ext Glu_ext NAD NADH2 FAD FADH2 CoA ADP ATP H2O CO2 Mal_ext Cit_ext
    AKG_ext Pi_ext Asp_ext
12
13 -CAT
14
15 R6i : Pyr + CO2 + ATP = OAA + Pi + ADP .
16 R7i : Pyr + NAD + CoA = ACoA + NADH2 + CO2 .
17 R8i : OAA + ACoA + H2O = Cit + CoA .
18 R9 : Cit = Isocit .
19 R10i : Isocit + NAD = Akg + NADH2 + CO2 .
20 R11i : Akg + NAD + CoA = SucCoA + NADH2 + CO2 .
21 R12 : SucCoA + Pi + ADP = Succ + CoA + ATP .
22 R13 : Succ + FAD = Fum + FADH2 .
23 R14 : Fum + H2O = Mal .
24 R15 : Mal + NAD = OAA + NADH2 .
25 T1 : Cit + Mal_ext = Mal + Cit_ext .
26 T2 : AKG_ext + Mal = Mal_ext + Akg .
27 T5 : Pi_ext = Pi .
28 T6 : Pyr_ext = Pyr .
29 T7 : Mal + Pi_ext = Pi + Mal_ext .
30 T12 : Asp + Glu_ext = Asp_ext + Glu .

```

3.3 Règles des gènes

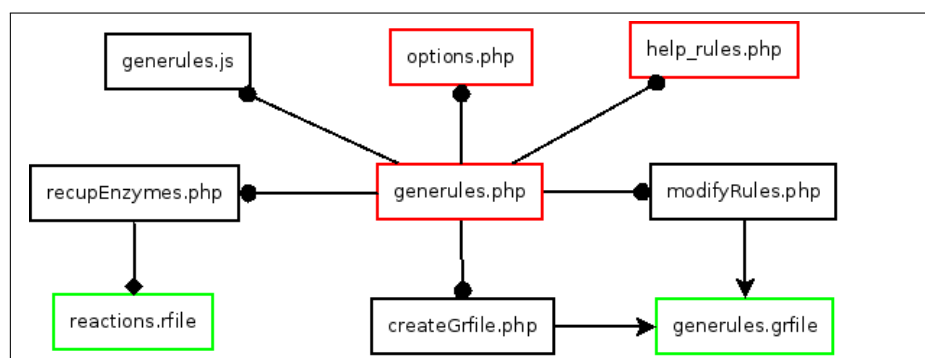


FIGURE 3.10 – Diagramme d’organisation pour le changement de langue

La page permettant la saisie des règles générales est obtenue à l’aide du fichier *generules.php* (Figure 3.10).

L’affichage à l’ouverture de la page n’est composé que d’une zone de texte et d’un bouton *Ok* (Figure 3.11) qui est de type *submit*.

Au clic, ce bouton fait appel à la fonction `add_reaction()` qui crée deux menus déroulants

Ecrivez une nouvelle règle :

Choisissez le nombre de réaction :

FIGURE 3.11 – Bouton du choix du nombre de réactions pour une règle

(pour la première et la dernière, Figure 3.12) ou trois menus déroulants (pour les autres, Figure 3.13) par réaction jusqu'à atteindre le nombre de réactions entrées par l'utilisateur.

Choisissez le nombre de réaction :

2

IF Réaction: Choisir une réaction Valeur: Choisir une valeur

THEN Réaction: Choisir une réaction Valeur: Choisir une valeur

FIGURE 3.12 – Menus déroulant si 2 réactions dans la règle

Choisissez le nombre de réaction :

3

IF Réaction: Choisir une réaction Valeur: Choisir une valeur

Opérateur: Choisir un opérateur Réaction: Choisir une réaction Valeur: Choisir une valeur

THEN Réaction: Choisir une réaction Valeur: Choisir une valeur

FIGURE 3.13 – Menus déroulant si 3 réactions dans la règle

Elle vérifie également que le nombre de réactions entrées par l'utilisateur est au moins égal à 2, sinon elle affiche un message sous la forme d'une alerte à l'utilisateur.

Les réactions sont récupérées à partir de *reactions.rfile*.

L'utilisateur peut sélectionner plusieurs fois la même réaction dans sa règle, sauf pour le choix de la dernière (ligne THEN). Cette particularité est gérée par la fonction `choice(form, val)`. Celle-ci permet de remplir les menus déroulants quand une réaction a été choisie et pour la dernière seules les réactions non sélectionnées précédemment apparaissent.

Lorsque l'utilisateur a choisi toutes ses réactions, leur opérateur et leur valeur, il clique sur le bouton *Ajouter*. Celui-ci fait appel à la fonction `validateForm()` qui vérifie que tous les champs ont bien été sélectionnés. Si la fonction retourne VRAI, il fait appel au fichier *createGrfile.php* qui écrit la règle dans le fichier *generules.grfile*.

Le fichier *createGrfile.php* écrit dans le fichier *grfile* selon certaines règles. En effet, l'écriture de la règle dépend des valeurs associées aux réactions et des opérateurs choisis.

Si la valeur de la réaction qui suit le "THEN" est 0, alors il y aura le symbole "!" au début de la règle (juste après le "="), sauf si la règle ne contient que deux réactions.

Si les autres réactions ont pour valeur :

- 0, alors on écrira (!0reac)
- 1, alors on écrira (!1reac)
- f, alors on écrira (!freac)

En revanche, si la valeur de la réaction après "THEN" est 1, les autres réactions s'écriront :

- 0reac si sa valeur est 0

- lreac si sa valeur est 1
- freac si sa valeur est f

De plus, si l'opérateur "AND" est sélectionné il sera écrit sous la forme & dans le fichier, l'opérateur "OR" sera lui écrit |. Quand la règle est composée d'au moins trois réactions, des parenthèses sont ajoutées après chaque réaction sauf la première et la dernière. Des parenthèses entourent l'ensemble des réactions situées après le "=".

Pour mieux comprendre l'écriture du fichier *generules.grfile*, voici un exemple :

Choix de l'utilisateur sur la page Web :

```
1 IF reaction: R1 valeur: 1
2 Operateur: AND reaction: R2 valeur: 0
3 Operateur: OR reaction: R3 valeur: 0
4 THEN reaction: R4 valeur: 0
```

Règle écrite dans le fichier :

```
1 R4 = (!((1R1 & (!0R2)) | (!0R3)))
```

3.4 Choix des options de lancement

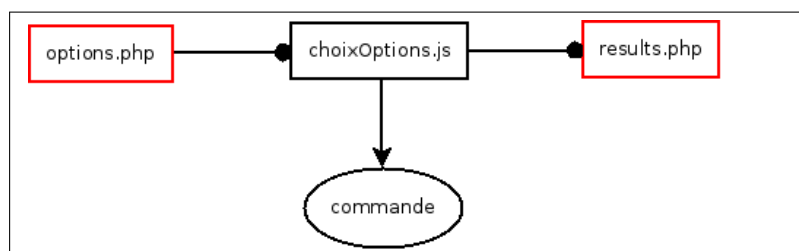


FIGURE 3.14 – Diagramme d'organisation pour le choix des options après création

La page permettant la sélection des options de lancement est obtenue à l'aide du fichier *options.php* (Figure 3.14). Comme nous l'avons dit précédemment, nous avons séparé les options en une série de catégories et sous catégories, le tout contenu dans un formulaire afin de gérer l'interaction avec l'utilisateur. Ce dernier peut sélectionner les options qu'il désire avec une combinaison de *radioboutons* et de zones de texte. Les *radioboutons* d'une même sous-section ont le même attribut *name* afin de ne pouvoir en sélectionner qu'un seul.

Prenons comme exemple la première sous catégorie (Figure 3.15) :

FIGURE 3.15 – Sous catégorie d'affichage des résultats dans la catégorie d'enregistrement

Voici le code HTML/PHP qui correspond :

```
1 <input type="radio" name="choix1" value="log console"
   checked="checked"> <?php echo TXT_OPTIONS_SAVING_3; ?>
2 <input type="radio" name="choix1" value="log file"> <?php echo
   TXT_OPTIONS_SAVING_4; ?>
3 <?php echo TXT_OPTIONS_SAVING_5; ?> <input type="text"
   name="log_nomFichier" size="10" id="texte1">
```

Le premier *radiobouton* est pré-coché avec l'attribut *checked="checked"* afin de guider l'utilisateur. Si cette option ne lui convient pas, il lui suffit de cocher le deuxième *radiobouton* et de remplir la zone de texte associée.

Ce sont les attributs *value* de ces boutons qui sont récupérés avec un script *JavaScript*. Ce dernier, nommé *choixOptions.js*, permet de vérifier quels *radioboutons* sont cochés et récupère ce qu'il a dans l'attribut *value* de ceux-ci quand c'est le cas. Il est appelé lors du clic sur le bouton *Lancement* en bas de la page. De plus, c'est dans ce script qu'est générée la commande qui permettra le lancement de *regEfmtool*. Elle est contenue dans une variable nommée *commande*, qui se remplit grâce aux paramètres sélectionnés.

Exemple du code *JavaScript* qui permet la récupération des *values* précédentes :

```
1 var commande = "java -Xmx1G -jar ../regEfmtool.jar";
2
3 // SAVE
4 // Display
5 if (formulaire.choix1[0].checked) {
6     valeur1 = " -" + formulaire.choix1[0].value;
7     commande = commande + valeur1;
8 }
9 else if (formulaire.choix1[1].checked) {
10    var nom1 = document.getElementById('texte1').value;
11    valeur1 = " -" + formulaire.choix1[1].value + " " + nom1;
12    commande = commande + valeur1;
13 }
```

La variable *commande* est une chaîne de caractères qui contient le début de la commande Java. Les paramètres sont ensuite ajoutés au fur et à mesure de la vérification de la sélection des boutons.

Cette commande est sauvegardée dans un *cookie* qui sera récupéré à la page d'affichage des résultats pour le lancement de *regEfmtool*.

3.5 Chargement d'un réseau pré-existant

Ce sont les pages *load.php* et *load_files.php* qui permettent le chargement des fichiers contenant un réseau pré-existant (Figure 3.16) depuis le disque dur de l'utilisateur.

Tout d'abord, la page *load.php* permet à ce dernier de choisir les paramètres de la commande de lancement de *regEfmtool*, de la même façon que pour la page *options.php*. Cependant, la

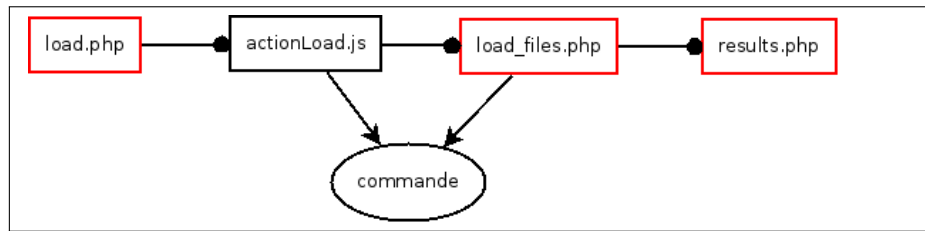


FIGURE 3.16 – Diagramme d'organisation pour le chargement des fichiers

deuxième grande catégorie diffère car le choix des fichiers à charger se fera lors d'une étape ultérieure. Ici, l'utilisateur coche les options qu'il désire et clique sur le bouton *Étape suivante*. Ce bouton fait appel au script *JavaScript* *actionLoad.js*, qui fonctionne sur le même principe que *choixOptions.js*.

Ensuite, l'utilisateur est dirigé vers la page *load_files.php*, qui permet à l'utilisateur d'aller "chercher" les fichiers aux formats *sfile*, *mfile*, *rvfile*, *grfile* et *rfile*. Nous copions et renommons ces fichiers dans le dossier courant afin de pouvoir les utiliser dans *regEfmttool*. Ce processus est faisable grâce à un script PHP, dont voici un exemple pour le fichier au format *sfile* :

```

1 move_uploaded_file($_FILES["sfile"]["tmp_name"], $_FILES["sfile"]["name"]);
2 shell_exec('mv ' . $_FILES["sfile"]["name"] . ' sfile');
```

Ce script est contenu dans l'attribut *value* du bouton de chargement du fichier en question. La ligne de commande est donc complétée et le lancement du logiciel peut commencer.

3.6 Affichage des résultats

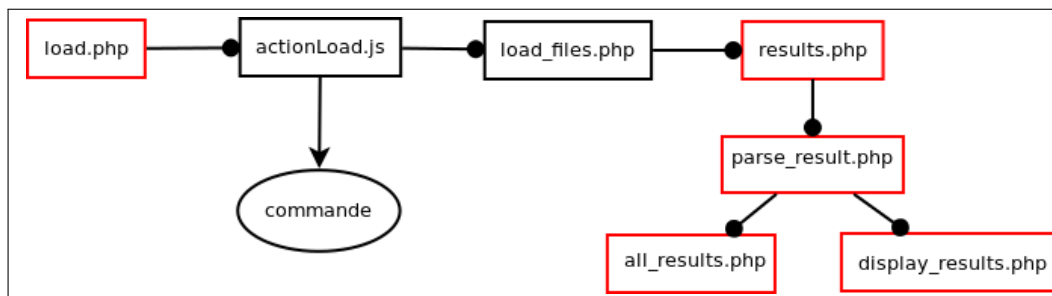


FIGURE 3.17 – Diagramme d'organisation pour l'affichage des résultats

Le fichier *results.php* permet l'affichage des résultats de la commande générée précédemment. Celle-ci est exécutée grâce à la fonction `shell_exec()`. Les fichiers de résultat et de log sont analysés pour permettre leur affichage (Figure 3.17).

L'affichage (Figure 3.18) se fait en plusieurs étapes : tout d'abord, nous vérifions grâce à la variable de session PHP `isCompared` si l'utilisateur veut afficher un ou plusieurs résultats.

Prenons le cas d'un utilisateur qui calcule une première fois les modes élémentaires d'un réseau : la variable `isCompared` indique à l'interface qu'il n'y a qu'un résultat à afficher, la



FIGURE 3.18 – Page d’affichage des résultats

`<div id='new'>` est créée. Nous affichons tout d’abord une image *waiting.gif* grâce à la fonction `waiting()`, qui se charge aussi d’exécuter la commande précédemment conservée dans les cookies HTML .

```

1 function waiting(){
2     echo ('');
3     $com = $_COOKIE['commande'];
4     if ($_SESSION['isCompared']==1) {
5         $com = $com . ' > log.txt';
6         shell_exec($com);
7     }
8     else {
9         $com = $com . ' > log1.txt';
10        shell_exec($com);
11    }
12 }

```

Cette fonction vérifie donc si l’utilisateur voulait comparer son résultat et sauvegarde le log dans le fichier *log1.txt* ou *log.txt* suivant le cas.

```

1 <div id="new" name="new" title="new results">
2     <?php
3         if (count($modes)<2)
4             echo '<script>document.getElementById("new").innerHTML =
5                 "';
6         }
7         $res2=parse_res();
8         echo '<script>document.getElementById("new").innerHTML =
9             "<p> . TXT_DISPLAY_RESULTS_NEW . </p>';
10        show_results($res2);
11        $_SESSION['compare']=$res2;
12    ?>

```

Ensuite, nous effectuons un passage du résultat avec la fonction `parse_res()` qui ouvre le fichier de résultats généré et le sauvegarde dans un tableau pour faciliter son affichage.

Finalement, la fonction `show_res(resultat)` permet d'afficher le résultat en ajoutant au tableau les noms des réactions, ainsi qu'une colonne indiquant le mode élémentaire. Dans le cas où l'utilisateur souhaite comparer deux résultats, une `<div id='original'>` HTML est créée (Figure 3.19). Celle-ci contient le premier résultat calculé et nous continuons avec le nouveau résultat à afficher.

```
1 if ($_SESSION["isCompared"]==1){
2     $res=$_SESSION["compare"];
3     echo "<div id='original' name='original' title='original results' >";
4     echo '<p>' . TXT_DISPLAY_RESULTS_ORIGINAL . '</p>';
5     show_results($res);
6     echo "</div>";
7 }
```

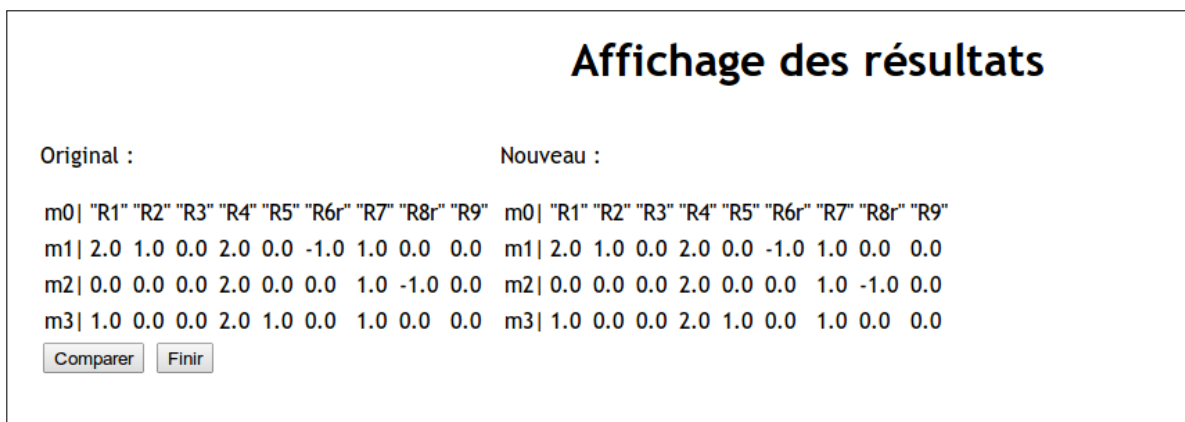


FIGURE 3.19 – Exemple de comparaison de deux résultats

Une fois toutes ces étapes finies, le résultat est sauvegardé dans la variable `compare` pour le cas où l'utilisateur voudrait comparer son résultat avec un autre (avec par exemple des choix différents d'options, de nouvelles règles).

L'utilisateur a donc le choix de terminer ses calculs ou de comparer le résultat obtenu. Pour cela, deux boutons *Comparer* et *Finir* lui permettent de retourner à l'écran d'accueil. Cependant, *Comparer* fait appel à la fonction `compare()` qui conserve en mémoire le fait que l'utilisateur veut comparer son résultat. Ceci est possible grâce à une variable de session PHP `isCompared`. Le bouton *Finir* quant à lui réinitialise cette même variable avant d'effectuer la redirection.

Sur cette même page, l'utilisateur a aussi la possibilité d'afficher les résultats de la commande qu'il vient de lancer. Pour cela, le fichier `parse_results.php` récupère le fichier `log.txt` dans lequel les résultats générés par `regEfmtool` ont été redirigés.

Cette page offre deux options :

- visualiser l'ensemble des résultats ,
- choisir une catégorie de résultats à visualiser.

Cette page parcourt le fichier log et sélectionne les mots-clés à l'aide de la fonction `item()`, qui récupère toute les phrases suivies de ":".


```

1 function item(){
2
3     $fichier = 'log.txt';
4     global $item;
5     define ('FICHIER', $fichier);
6     $numligne = 0;
7
8     $fichier = fopen( FICHIER, 'r')or die('Ouverture en lecture de "' .
9         FICHIER . '" impossible !');
10
11     while (!feof($fichier)){
12         $numligne++;
13         $lignes = fgets($fichier, 1024);
14         $posi = strpos($lignes, '|');
15         $ligne = trim(substr($lignes,$posi+1,strlen($lignes)));
16         $pos = strpos($ligne, ':');
17         if ( $pos == true && $numligne >= 32){
18             $line = trim(substr($ligne,0, $pos));
19             $item[] = $line;
20         }
21
22         $count = count($item);
23
24         for ($numero = 0; $numero < $count; $numero++){
25             echo '<OPTION VALUE="' . $item[$numero] . '"> ' . $item[$numero]
26                 . ' </OPTION>';
27         }
28     }
29     return $item;
30 }

```

Les mots-clés sont stockés dans un menu déroulant, qui est obtenu par affichage de chaque valeur du tableau dans la balises HTML '`<OPTION VALUE="" . $item[numero] . '"> ' . $item[numero] . ' </OPTION>`'. Ainsi l'utilisateur n'a plus qu'à sélectionner la catégorie qu'il souhaite visualiser à partir du mot-clé.

Ainsi la visualisation peut se faire sur l'ensemble du fichier *log.txt* (Figure 3.20) et une redirection sur la page *all_results.php* sera effectuée. La visualisation peut également se faire sur une seule catégorie et la redirection sera alors sur *display_results.php*. De plus, la page d'affichage des résultats offre la possibilité à l'utilisateur de revenir en arrière à l'aide de deux boutons, dont un présent au-dessus de l'affichage des résultats et l'autre, en-dessous.

Nous avons tenté de faire en sorte que les résultats extraits soient affichés de façon claire et lisible pour l'utilisateur grâce au fichier *display_results.php*. Ce dernier récupère le mot-clé sélectionné dans le menu déroulant de la page précédente et le recherche dans l'ensemble du fichier. Si ce dernier comporte un 'R', seule la ligne comportant le mot clé est retournée. Dans le cas contraire, un des mot-clé suivant délimitera la catégorie sélectionnée. Cette recherche offre deux possibilités grâce à la fonction `display_results()` du fichier *display_results.php* :

- dans le premier cas, si le mot-clé est suivi d'autres mots-clés contenant eux-même un 'R', le mot-clé délimitant sera le premier qui ne possédera pas de 'R',
- dans le second cas, le mot-clé suivant dans la liste, servira de délimitant.

Nous avons également ajouté un bouton permettant de retourner sur la page d'extraction des résultats (Figure 3.21).

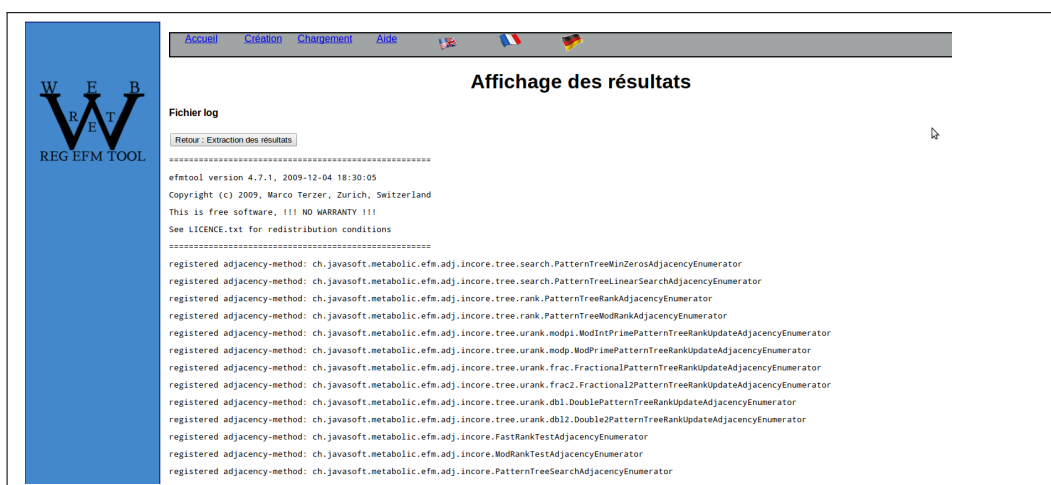


FIGURE 3.20 – Page de recherche sur le log

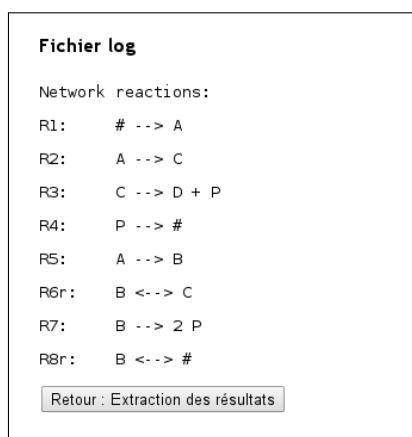


FIGURE 3.21 – Exemple de recherche sur le log

Chapitre 4

Difficultés et améliorations

Nous allons maintenant voir les difficultés que nous avons rencontré au cours de ce projet, ainsi que les améliorations possibles.

Suivant le navigateur utilisé, la récupération des fichiers par la fonction `move_uploaded_file()` ne fourni pas le même résultat, en particulier pour le dossier récupérant les fichiers.

De plus, il semblerait que les variables permettant de récupérer les fichiers `$FILES` ne fonctionnent pas sur toutes les configurations. Jusqu'à présent, nous n'avons pas compris quel facteur influençait l'utilisation de ces variables et n'avons pas pu exécuter *regEfmtree* depuis l'interface sur toutes les machines.

Les variables de sessions PHP étant propres à ce langage, nous avons parfois dû utiliser les cookies, en particulier lorsque certaines valeurs devaient être sauvegardées depuis un script *JavaScript*. Cela provoque l'utilisation de ces deux types de variables, ce qui n'est pas très uniforme.

L'affichage des résultats n'a été fait que superficiellement, d'où la présence de *mode élémentaire 0* devant les noms de réactions, ou encore un simple affichage en tableau des résultats brut, indiquant la présence de réactions avec une valeur à 0 (c'est à dire non représentées dans les modes élémentaires).

Pour finir, l'utilisation de *regEfmtree* depuis l'interface que nous avons créé dépend de la configuration du serveur Web installé sur la machine. Dans notre cas les tests étaient fait sous *apache2* sur nos machines personnelles (la configuration du serveur sur les machines du *CREMI* étant trop restrictive, il nous était impossible d'exécuter le logiciel).

Conclusion

Le but de ce projet été de réaliser une interface graphique pour le logiciel de calculs de modes élémentaires *regEfmtree*. L'interface réalisée se présente sous la forme d'un site Web. Elle permet d'utiliser *regEfmtree* sans avoir à passer par un terminal et permet la création de tous les fichiers d'entrée nécessaires au fonctionnement de ce logiciel.

Elle offre aussi la possibilité de créer des réseaux au format *DAT*, rendant possible l'édition de ces derniers pour le logiciel METATOOL afin de permettre une comparaison des résultats de calcul de modes élémentaires.

Ce projet nous a permis de nous familiariser avec les langages propres aux communications et protocoles Web. Nous avons également pu mesurer l'importance d'une bonne gestion de l'emploi du temps au sein de l'équipe et de la communication dans la répartition des tâches.

Tout en restant dans un contexte d'enseignement, ce projet nous a donné un aperçu du travail en vie active, et plus particulièrement de la gestion de plusieurs projets simultanément.

Bibliographie

- [1] <http://bioinformatics.oxfordjournals.org/content/20/2/226>.
- [2] <http://www.csb.ethz.ch/tools/efmtool>.
- [3] Christian Jungreuthmayer, David E. Ruckerbauer, and Jürgen Zanghellini. Utilizing gene regulatory information to speed up the calculation of elementary flux modes. 2012.
- [4] Axel von Kamp and Stefan Schuster. Metatool 5.0 : fast and flexible elementary modes analysis. *Bioinformatics*, August 2006.
- [5] <http://www.mpi-magdeburg.mpg.de/projects/cna/cna.html>.
- [6] Roland Schwarz, Patrick Musch, Axel von Kamp, Bernd Engels, Heiner Schirmer, Stefan Schuster, and Thomas Dandekar. Yana – a software tool for analyzing flux modes, gene-expression and enzyme activities. *BMC Bioinformatics*, June 2006.
- [7] <http://yana.bioapps.biozentrum.uni-wuerzburg.de/>.
- [8] S. Pérès, F. Vallée, M. Beurton-Aimar, and J.P. Mazat. Acom : A classification method for elementaryfluxmodes based on motif finding. *Biosystems*, March 2011.