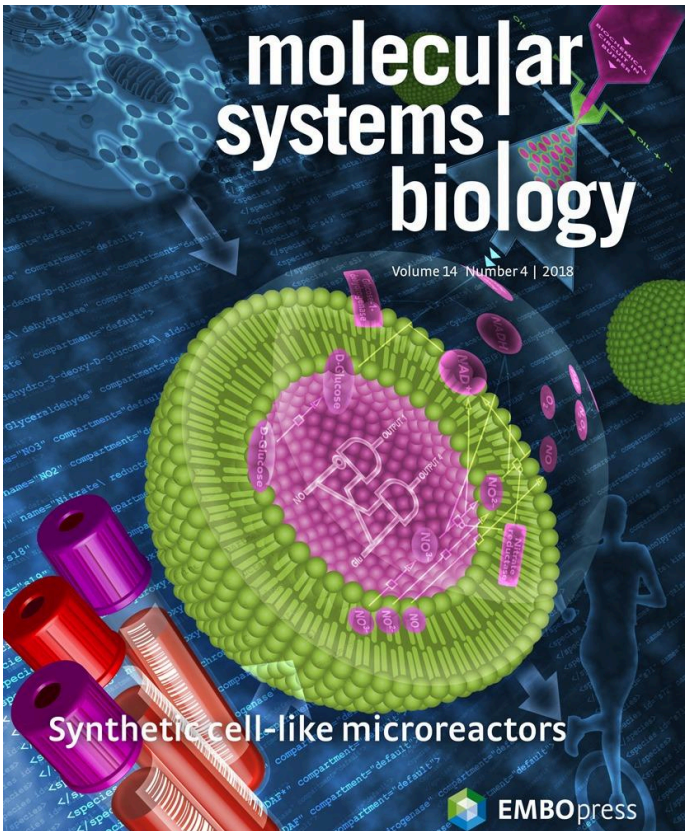
PSC INF/BIO 09

Compilation de fonctions mathématiques en réactions biochimiques

Rapport de PSC



Tuteur : FAGES François

BESSA Swann
DAGONNEAU Thomas
HAZERA Gabriel
MONTOYA Yann
LI Romain
TCHOMBA NGUEKO Rovanaud

PSC INF/BIO09 : Compilation de fonctions mathématiques en réaction biochimique

Tuteur : FAGES François

Coordinateur : Emmanuel Haucourt (INF)

Co-coordinateur: Yves Mechulam (BIO)

DAGONNEAU Thomas
MONTOYA Yann
TCHOMBA NGUEKO Rovanaud
BESSA Swann
HAZERA Gabriel
LI Romain

I. Table of Contents

Introduction :

La détection de maladie ou d'anomalies dans le corps humain ou dans l'environnement est un des enjeux majeurs du XXIème siècle. Cette détection peut passer par l'identification des molécules présentes ou absentes de certaines solutions. Pour ce faire, on peut produire des vésicules non vivantes dans lesquelles on introduit un ensemble de molécules. Ces molécules permettent alors d'attester de la présence ou de l'absence de certaines molécules : un ensemble de réaction conduit à la création d'un produit facilement identifiable (par exemple un indicateur coloré) attestant de la présence ou de l'absence des espèces recherchées.

La difficulté consiste à déterminer quelles molécules ajouter à la vésicule pour vérifier la propriété recherchée. Cette combinaison de molécules peut en réalité être déterminée informatiquement en développant des algorithmes. L'intérêt de cette recherche préalable est d'assurer la validité du résultat et de réduire les coûts de la recherche de la bonne vésicule. Le but de notre PSC, en collaboration avec l'équipe de François FAGES et de l'Inria, a été de développer de tels algorithmes permettant de déterminer à moindre coût les vésicules à créer pour effectuer des tests médicaux.

De manière plus concrète notre travail s'est concentré sur le passage d'un Chemical Reaction Network (CRN) abstrait à un CRN concret.

Un CRN abstrait est un ensemble de réactions fictives. Il permet de représenter un ensemble complexe de réactions réelles sous forme d'une équation booléenne. Ici une variable de l'équation représente la présence (vrai) ou l'absence (faux) d'une molécule, puisqu'on ne prend pas en compte sa concentration. Ces équations permettent de conditionner la présence d'une molécule à la présence d'autres molécules. Par exemple, on peut considérer le CRN abstrait " A + B → C ". Bien que la réaction réelle A + B → C soit probablement impossible, on veut dans ce cas que la molécule C ne soit présente que si les molécules A et B sont présentes. En revanche, si le CRN abstrait existe bien alors il existe

un ensemble de réactions qui produit C uniquement en présence de A et B, ainsi que d'autres espèces choisies au départ. La particularité d'un CRN abstrait est qu'il correspond avant tout à une équation logique. Par exemple "A ET B DONNENT C" et "A OU B DONNENT C" ne correspondent pas au même CRN. En effet dans le premier cas on attend un CRN tel que si A ou B est retiré alors il n'y a plus de C alors que dans l'autre on veut que si seulement l'un parmi A ou B est retiré il y ait du C.

Le CRN concret, contrairement au CRN abstrait, correspond à un ensemble de réactions réelles et exhaustives en vue des espèces présentes : toutes les réactions possibles en vue des espèces en présence doivent être représentées.

L'objet de notre PSC a donc été de déterminer un ensemble d'algorithmes permettant le passage d'un CRN abstrait à un CRN concret. À partir d'un système d'équations logiques entre des réactifs et un produit, on obtiendrait une liste d'espèces à mettre dans la vésicule pour qu'elle vérifie le système d'équations logiques une fois introduite en milieu. **Pour aboutir à ce but, la base de notre étude repose sur un algorithme du plus court chemin : étant donné un milieu réactionnel, des réactifs et un produit, l'algorithme renvoie un ensemble des mécanismes réactionnels les plus courts des réactifs au produit.**

Nous n'avons au début de nos recherches découvert aucun programme permettant de résoudre ce problème. Les logiciels se rapprochant le plus de notre objectif étaient des logiciels de chimie avec un aspect rétro synthèse tels que Sci-finder ou Reaxys. Cependant, ces logiciels ne permettent pas de spécifier les réactifs de départ mais servent à obtenir un moyen industriel de synthétiser la molécule voulue, ce qui n'est pas notre objectif.

Pour travailler sur ce projet nous avons réparti le travail comme suit : monsieur NGUEKO TCHOMBA Rovanaud a travaillé sur l'étude de la base de données BRENDA et le lien entre nos algorithmes et les données extraites, messieurs BESSA Swann, HAZERA Gabriel et LI Romain ont travaillé sur la première version de l'algorithme et enfin messieurs MONTOYA Yann et DAGONNEAU Thomas ont travaillé sur la formalisation des algorithmes et l'ensemble des preuves de correction et calculs de complexité.

L'ensemble de notre code est écrit dans le langage de programmation python et nous avons utilisé le logiciel Git pour travailler simultanément sur le projet.

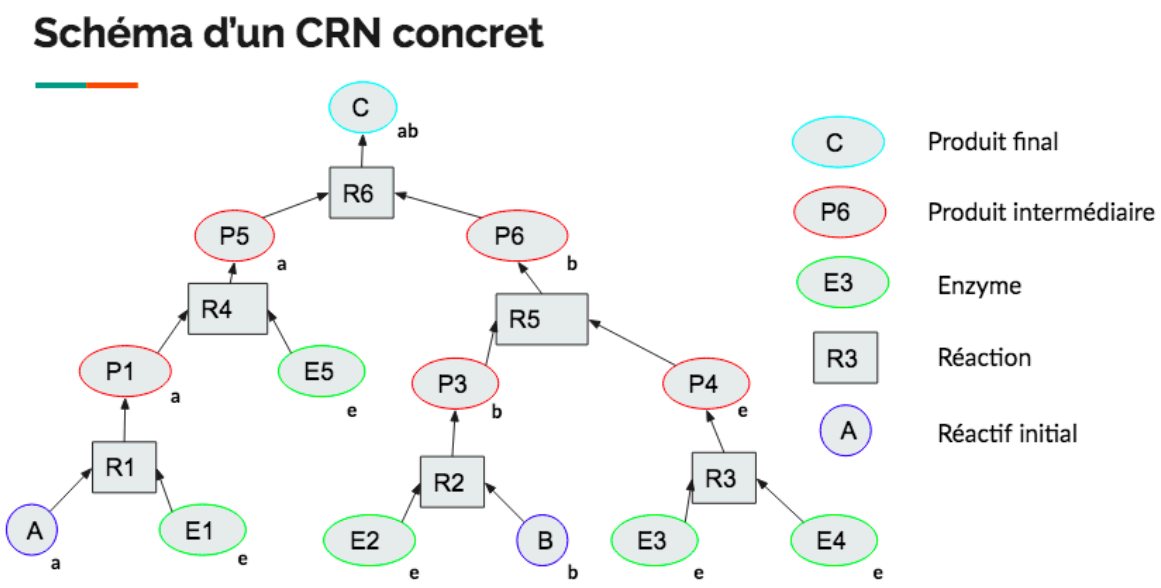
I. Un premier algorithme sur des ensembles restreints

1. Premier algorithme de résolution

A cause de l'absence de solutions existantes au problème, nous avons dû commencer par une phase de réflexion sur la méthode à adopter pour résoudre le problème. L'équipe de monsieur FAGES nous avait fourni des solutions validées de CRN concret correspondant à un CRN abstrait. Le premier objectif a donc été de développer un premier algorithme en le testant sur ce faible nombre de réactions pour retrouver leur résultat.

Nous avons donc commencé par une phase de réflexion séparée où chaque membre du groupe a réfléchi aux solutions envisageables. Nous avons ensuite mis en commun l'ensemble de nos idées pour décider quelle serait celle que nous allions conserver.

Pour diriger l'étude du problème, on peut regarder la représentation suivante des réactions possibles entre un jeu restreint de molécules.



- Il y a 2 types de nœuds : les molécules (forme ovale/circulaire) et les réactions (forme rectangulaire).
- Une arête est dirigée d'une molécule vers une réaction quand cette molécule est un réactif de la réaction. Une arête est dirigée d'une réaction vers une molécule quand cette molécule est un produit de la réaction.

Notre but était le suivant : trouver un algorithme qui permet de calculer tout les mécanismes réactionnels des réactifs au produit.

Notre idée principale était la suivante : réaliser une recherche de type Breadth-First-Search (BFS, adapté à notre type particulier de graphe et à nos autres besoins) pour trouver des chemins entre les réactifs et le produit.

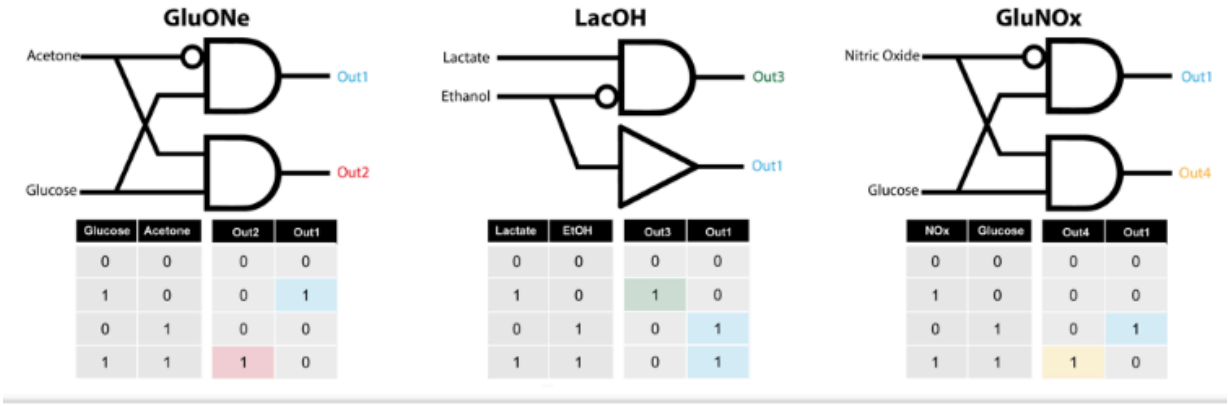
La question était alors la suivante : Faut-il partir du produit ou des réactifs ?

- Partir des produits nous a paru trop complexe. En effet, la BFS ne pouvant pas considérer les réactifs, de nombreux chemins réactionnels inutiles sont utilisés.
- Partir des réactifs nous est alors apparu comme la solution. En effet, en utilisant les réactifs initiaux et les enzymes disponibles au début dans le milieu réactionnel, on

ne progresse dans l'arbre qu'avec les réactions pour lesquelles tous les réactifs sont présents, et on explore alors beaucoup moins de chemins.

- Ce modèle a ensuite été optimisé pour s'adapter à des bases de données plus grandes (voir IV.2.)

Pour passer d'un CRN abstrait (équation logique) à un CRN concret (mécanisme réactionnel), il faut trouver un chemin réactionnel partant des réactifs et espèces initiales et allant au produit voulu, tout en vérifiant une équation logique entre la présence des réactifs et la présence du produit. Par exemple : " A + B → C ". Le produit C n'est obtenu que si A et B sont présents ensembles dans le milieu.



Glucose	Acetone	Out2	Out1
0	0	0	0
1	0	0	1
0	1	0	0
1	1	1	0

Lactate	EtOH	Out3	Out1
0	0	0	0
1	0	1	0
0	1	0	1
1	1	0	1

NOx	Glucose	Out4	Out1
0	0	0	0
1	0	0	0
0	1	0	1
1	1	1	0

Équations logiques recherchées dans les vésicules

L'initialisation consiste à récupérer les listes de molécules et réactions possibles à partir d'un fichier texte. Cependant, l'importation des molécules sous un nom littéral nécessite un tableau d'indexage où l'on associe un numéro à chaque molécule. On peut alors convertir la liste des réactions avec les noms des molécules en liste des réactions avec les indices des molécules, plus efficace pour la recherche de chemin possible. Par exemple, au lieu d'avoir ("acétone" + "eau" → "géminal-diol"), on représentera la réaction sous la forme ("6" + "2" → "15") (acétone indexé à 6, eau à 2 et géminal-diol à 15).***

De même, on indexe les réactions par numéro.

Dans notre algorithme, la liste des molécules est incrémentée lors de la lecture de chaque réaction. On part d'une liste avec les réactifs initiaux, et à chaque étape on ajoute à cette liste les produits des réactions entre les molécules présentes dans la liste. Pour faciliter cela, on crée une liste de listes pour accéder rapidement à toutes les réactions dans lesquelles une molécule donnée réagit. L'élément d'indice i est la liste des numéros de réaction où la molécule d'indice i réagit. ***Ainsi, à chaque étape, on considère comme nouvelles réactions potentielles les réactions dans lesquelles les nouveaux produits réagissent.

On effectue une recherche d'un CRN concret à partir d'un CRN abstrait sous la forme "Réactif_1 + Réactif_2 → Produit_1" interprétant le + comme un 'et logique'. ***

Au départ de l'algorithme, la liste contient "Réactif_1", "Réactif_2", et d'autres molécules prédéfinies déjà présents dans le milieu réactionnel, éventuellement catalyseurs. On considère (en première approximation) la présence des molécules selon un schéma simplificateur booléen : les molécules sont soit présentes soit absentes, et une fois qu'elles sont produites, elles deviennent présentes jusqu'à la fin de l'exécution de l'algorithme.

Afin de rechercher un mécanisme réactionnel, on applique la BFS définie en 1. en sauvegardant une trace du mécanisme réactionnel. Une fois arrivé au produit, on retrace le mécanisme réactionnel grâce à la sauvegarde.

Seulement, rien n'assure que le mécanisme réactionnel nous permet de vérifier l'équation logique. Dans le cas "A+B → C", C pourrait être obtenu seulement avec A, seulement avec B ou même sans aucun des 2 avec les molécules du milieu réactionnel. On pourrait également avoir C à partir de A et B, sauf qu'un autre mécanisme réactionnel impliquant seulement A et produisant C pourrait avoir lieu sans qu'il soit détecté. D'où la nécessité d'un système d'étiquettes, pour distinguer les différents mécanismes réactionnels.

2. Un système d'étiquettes pour distinguer les mécanismes réactionnels en fonction des réactifs intervenant

La difficulté était maintenant de vérifier l'équation logique. Pour ce faire, nous avons développé un système d'étiquettes vérifiant une simple relation. Si par exemple on étudie le CRN abstrait "A et B donnent C", il y a quatre types d'étiquettes : a, b, e, ab.

- Une espèce étiquetée **e** est une espèce qui n'a été produite par un mécanisme n'impliquant ni A ni B : elle a été produite à partir des molécules déjà dans le milieu réactionnel.
- Une espèce étiquetée **a** (resp. **b**) est une espèce qui a été produite par un mécanisme réactionnel impliquant A (resp. B) et des molécules étiquetées e.
- Une espèce étiquetée **ab** est une espèce qui a été produite par un mécanisme réactionnel impliquant A et B, et des molécules étiquetées e.

L'algorithme renvoie ainsi tous les chemins possibles en précisant l'étiquette obtenue pour le produit, ce qui permet plus tard de trouver les milieux réactionnels pour une fonction logique "ou".

On crée une relation binaire * pour calculer les étiquettes***. e agit comme un élément neutre, et a*b donne ab. Lors d'une réaction de la forme X + Y = Z si les étiquettes de X et Y sont x et y alors l'étiquette z de Z est :

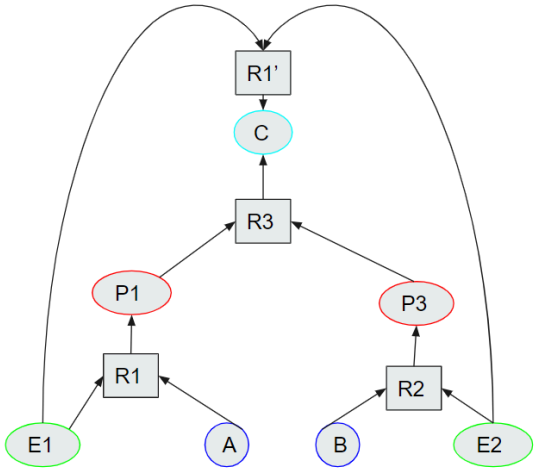
Étiquette de X	**Étiquette de Y**	**Étiquette de Z**
a (resp. b)	b (resp. a)	ab
a (resp. b)	e	a (resp. b)
a (resp b,ab,e)	a (resp b,ab,e)	a (resp b,ab,e)
ab	a (resp. b)	ab
ab (resp. e)	e (resp. ab)	ab

Il faut cependant traiter une difficulté : quelle étiquette porte concrètement une molécule formée par des réactions différentes ? En effet, comme l'algorithme traite plusieurs

chemins réactionnels à la fois, une molécule peut avoir des étiquettes différentes suivant le chemin.

Pour un chemin donné dans lequel le produit est étiqueté 'ab', cela signifie que A et B sont présents au début du chemin. Ainsi, lorsque l'on remonte le mécanisme, il est nécessaire que l'on conserve des molécules étiquetées 'a' et 'b'.

Par exemple si la réaction finale aboutissant au produit est "P1+P2 → C", les réactifs initiaux étant "A" et "B", et C étant étiquetée "ab" : alors si P1 possède l'étiquette a et P2 l'étiquette b, on va chercher récursivement pour P1 un mécanisme dans lequel il est produit par A, et pour P2 un mécanisme dans lequel il est produit par B. Il y a d'autres configurations potentielles (par exemple P1 étiqueté ab et P2 étiqueté e) : pour choisir la bonne on sauvegarde les étiquettes que P1 et P2 ont obtenues lors de la BFS et on choisit la réelle configuration en fonction.



3. Les difficultés pour traiter la négation et les systèmes d'équations logiques

Une des équations logiques que nous avons été amenés à devoir traiter a été la négation : "Non B donne C". Ce CRN abstrait correspond au cas où l'absence de B est nécessaire et suffisante pour obtenir C. Cela impose que la présence de B crée une molécule consommant C. Ce cas est différent des précédent (le et et le ou). ***
En effet la consommation de C peut avoir lieu sous plusieurs formes :

- une réaction directe avec B : $C + B \rightarrow X$ (1)
- consommation suite à une réaction en cascade $B \rightarrow B1$; $B1 \rightarrow B2$; $B2 \rightarrow B3$; $B3 + C \rightarrow X$ (2)
- consommation suite au déplacement d'un équilibre chimique $B \rightarrow A$; $A + C \leftrightarrow D + E$(3)

Ces trois possibilités nous ont contraint à un traitement différencié dans l'algorithme.
Dans le cas de la réaction directe (1) C est consommé par B, aussi la vérification est rapide et efficace. Dans le cas des réactions en cascade (2) c'est B3 qui consomme C. L'enjeu est alors de trouver les molécules potentielles B3 pouvant consommer C, puis pour chaque B3 un chemin réactionnel entre B et B3. Pour le cas du déplacement chimique (3) la molécule qui consomme C est A. Notons dans les trois cas P la molécule qui consomme C.
Pour déterminer le lien entre la molécule P et B nous utilisons l'algorithme du plus court chemin que nous avons développé. Le problème est que cet algorithme a un temps d'exécution assez long et le tester sur chaque molécule réagissant avec C est problématique. Pour remédier à ce problème, nous avons réalisé une variante de l'algorithme du plus court chemin, dans lequel on traite toutes les molécules en même temps, ce qui nous a permis un grand gain de temps de calcul.

Dans un second temps l'objectif a été de traiter le cas où plusieurs équations logiques servaient à représenter un système logique. Le passage de l'équation au système pose une nouvelle difficulté car le simple fait de résoudre chacune des lignes de manière indépendante et de faire la réunion des solutions ne permet pas d'obtenir le bon résultat. En effet si par exemple on a un "A et B donnent C" peut être que le milieu réactionnel solution d'une autre équation permet d'obtenir du C sans A ou B. Pour pallier ce problème nous avons envisagé deux solutions.

La première consisterait à réaliser un algorithme de recherche de chemin qui recherche les chemins de toutes les lignes simultanément en vérifiant des contraintes afin de ne pas avoir d'interférences entre chacun de ces chemins. N'ayant pas réussi à bien définir ces contraintes, nous nous sommes intéressés à la deuxième option.
La deuxième solution naïve, qui correspond à celle que nous avons conservé est la suivante : nous cherchons l'ensemble des milieux réactionnels permettant de vérifier chaque ligne du système une par une (dans une certaine limite du nombre de milieux réactionnels), puis

nous testons toutes les combinaisons de ces milieux réactionnels pour vérifier lesquels fonctionnent. ***

II. Utilisation de l'algorithme et premiers résultats

1. Travail sur des bases de données restreintes

Comme évoqué, pour mettre en place l'algorithme de recherche de chemin, nous avons utilisé une base simplifiée des réactions possibles. Le fichier ne contient presque que les réactions qui ont lieu dans les vésicules trouvées par les chercheurs. Cette première base permet de tester si un chemin possible existant est bien trouvé.

Chaque réaction est sous forme de texte avec un ou deux réactifs donnant un ou deux produits. La première implémentation de l'algorithme ne prend donc en compte que des réactions à un ou deux réactifs avec un ou deux produits.

Une deuxième hypothèse, lors de la recherche de CRN est la logique booléenne sur la présence : une molécule produite ne disparaît jamais.

Nous avons récupéré différents milieux réactionnels utilisés dans des tests pharmacologiques de l'industrie par des documents de l'ancien chercheur à l'Inria Patrick Amar. L'un d'eux permet d'obtenir le test « Acetone + Glucose → Resorufin » et « Non(Acetone) + Glucose → NADH ». Les molécules initialement présentes dans la vésicule doivent être : Alcohol dehydrogenase, Alcohol oxydase, Peroxydase, Glucose-1-dehydrogenase ***

Pour l'équation logique « Acetone + Glucose → Resorufin », deux mécanismes sont trouvés et un seul correspond à l'étiquette *ab* d'un ET logique. Les étapes réactionnelles du mécanisme sont les suivantes :

```
mécanisme 2:
G_1DH + NAD -> CCia1 --- glucoseext -> glucose --- acetoneext -> acetone
CCia1 + glucose -> CCib1
CCib1 -> CCfa1 + NADH
ADH + NADH -> CCia2
CCia2 + acetone -> CCib2
CCib2 -> CCfa2 + NAD
CCfa2 -> ADH + isopropanol
AO + isopropanol -> CCf3
CCf3 -> CCio3 + H_2O_2
HRP + H_2O_2 -> CCia5
CCia5 + resazurin -> CCib5
CCib5 -> HRP + resorufin
```

Résultat n°2 de l'algorithme de recherche de mécanismes, avec glucose et acétone en réactifs, et résorufine en produit

Cela correspond bien au mécanisme fourni par l'INRIA, ce qui fournit une validation avant de passer à la base Brenda. À noter que nous avons obtenu des résultats concordants pour l'ensemble des 3 vésicules à notre disposition, en validant chacune des équations logiques.

2. Améliorations de l'algorithme

Nous avions pris la décision d'implémenter notre algorithme en partant des réactifs pour arriver au produit et non le sens inverse. Cependant l'explosion du nombre de molécules présentes avec Brenda nous a obligés à adapter la stratégie de recherche. Pour cela, nous avons choisi à chaque exécution de l'algorithme de restreindre le nombre de molécules considérées aux "voisinages" des produits et des réactifs.

Définition : le voisinage d'une molécule A de profondeur n correspond à toutes les réactions faisables et aux molécules alors produites à partir de A en moins de n étapes réactionnelles.

Ainsi, si l'on veut déterminer des mécanismes réactionnelles en 2n (ou 2n-1) étapes réactionnelles, on considérera les voisinages de profondeur n des espèces initiales et du produit à obtenir pour mener notre étude.

Ci-dessous se trouve colorié en vert le voisinage de profondeur 2 des réactifs A et B avec les molécules initialement dans le milieu réactionnel (notées "E…") sur la 1ère image. En le couplant au voisinage de profondeur 1 du produit C (2ème image), on obtient un ensemble (3ème image) nous permettant d'accéder à tous les mécanismes réactionnels des produits aux réactifs (dont le mécanisme représenté en I.1). On notera que l'utilisation de cet ensemble nous évite ici d'explorer la branche à droite, ce qui représenterait un coût de calcul supplémentaire. En pratique, les branches parallèles inutiles sont bien plus importantes et représentent beaucoup plus de molécules, d'où la pertinence de notre optimisation.

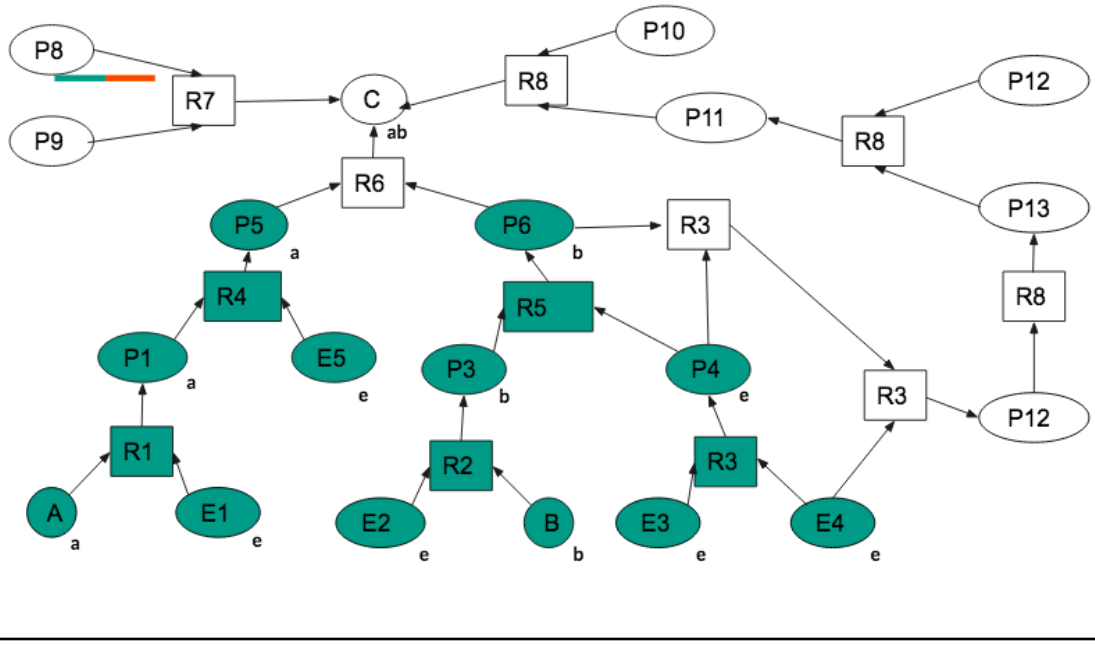


Image 1 : espèces obtenues à partir des molécules initiales (A,B,E1,E2,E3,E4) en 2 étapes

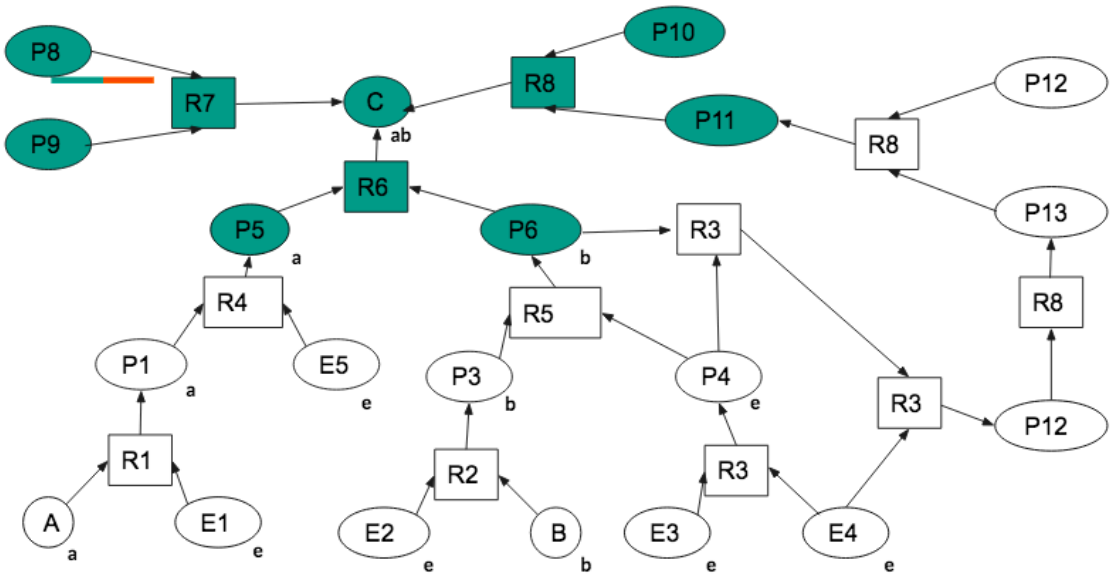


Image 2 : Voisinage de profondeur 1 du produit C

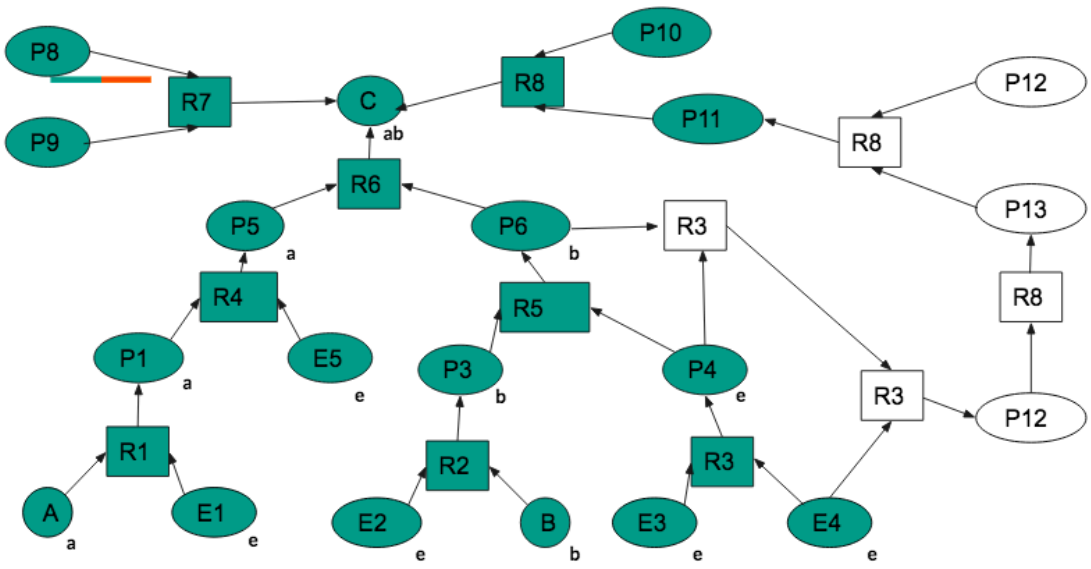


Image 3 : Combinaison des 2 voisinages précédents calculés

Une autre optimisation a porté sur l'itération à chaque étape sur les réactions potentielles. Initialement, on itérait sur toutes les molécules présentes, et pour chacune de ces molécules on itérait sur la liste des réactions possibles avec cette molécule. L'amélioration a consisté à itérer que sur les molécules nouvellement produites, car c'est elles qui vont permettre de réaliser de nouvelles réactions.

Le passage de listes Python à des vecteurs numpy nous a permis de gagner en espace de stockage. La mise à profit de sets Python a permis des gains de temps considérables. **Nous avons remplacé des listes par des sets afin d'avoir une procédure find en temps constant.**

3. La base Brenda

Pour ce projet, l'objectif est d'appliquer nos algorithmes directement sur la Base BRENDA. Il s'agit d'une base de données en ligne qui fournit des informations sur les enzymes, protéines et réactions chimiques. Elle est organisée en différentes sections, notamment une section de recherche d'enzymes par nom, numéro EC ou séquence, une section de recherche de protéines par nom, identifiant Uniport ou séquence, et une section de recherche de réactions enzymatiques par nom ou numéro EC. Elle contient également des informations sur les propriétés des réactions telles que la cinétique, la spécificité, la température optimale et le pH optimal.

Les informations disponibles sur la base de données ne sont pas utilisables telles quelles par le programme que nous avons implémenté. Une extraction des données est donc nécessaire pour la suite du projet pour les tests notamment. Le traitement en question présente quelques difficultés notamment au niveau de la cohérence des données. En particulier, il est possible qu'une molécule ait plusieurs noms différents dans la base de données Brenda. Il serait donc intéressant de vérifier les différents noms d'une molécule dans la base de données pour s'assurer d'avoir des tests consistants. De même certaines

équations sont incohérentes et ne doivent pas être considérées (par exemple certaines réactions n'ont pas de produit).

L'ensemble des réactions de la base sont fournies par différents organismes. Il arrive parfois que différents organismes y ajoutent des réactions similaires, ce qui a permis de pouvoir choisir les réactions les plus cohérentes parmi celles proposées. Nous avons cependant limité le traitement des données de la base au minimum acceptable pour que notre algorithme puisse disposer de tests consistant, parce que l'objectif est de pouvoir tester qu'il est opérationnel pour de grandes bases de données.

À cause des problèmes de formatage, la lecture des données sous format texte présente quelques difficultés. Les données peuvent être structurées de manière irrégulière ou contenir des caractères spéciaux qui rendent difficile leur traitement par des programmes informatiques. Il faut utiliser des outils de nettoyage de données pour standardiser le format des données et éliminer les caractères indésirables. Ce travail a été l'un des objets du stage de recherche de 3A de Julien Bienvenu, qui a travaillé sur le nettoyage de la base Brenda.

Nous avons pris connaissance et apprécié son travail qui nous a donné un aperçu de la base Brenda. Il s'est ensuite avéré nécessaire de refaire de nous même la lecture le prétraitement des données à partir d'un fichier .csv fourni par Julien. Pour cause, les programmes écrits par ce dernier pour traiter ces données ne correspondaient pas exactement au travail que nous nous apprêtions à effectuer. Nous avons alors opté pour la bibliothèque pandas de python pour l'exploitation des données.

Nous avons cependant limité le traitement des données de la base au minimum acceptable pour que notre algorithme puisse disposer de tests consistant, parce que l'objectif est de pouvoir tester s'assurer qu'il est opérationnel pour de grandes bases de données, indépendamment de la source. Après exploration, visualisation et transformation des données, nous les avons stockées dans un fichier pour que cette étape n'aie plus à être refaite. Il suffit alors de disposer d'un fichier rassemblant l'ensemble des réactions possibles et le programme se chargera du reste.

Plus précisément, les informations dont on dispose à la base, et seulement celles dont on a besoin sont un catalogue de réactions chimiques renseignant à chaque fois les réactifs et les produits pour chaque réaction. On peut alors examiner chaque réaction pour en déduire un lexique contenant l'ensemble des molécules. Pour renseigner nos algorithmes nous avons aussi besoin pour chaque molécule d'une liste d'amis, que nous définissons comme l'ensemble de molécules avec laquelle la molécule interagit. Cela permettra d'évaluer les réactions possibles en fonction des molécules présentes dans le milieu réactionnel.

En plus des données nécessaires aux algorithmes, nous avons pendant cette phase d'analyse de cette encyclopédie partielle de Brenda nous en avons extrait quelques faits pertinents pour en tirer des statistiques utiles notamment pour des questions de d'intéraction des molécules entre elles.

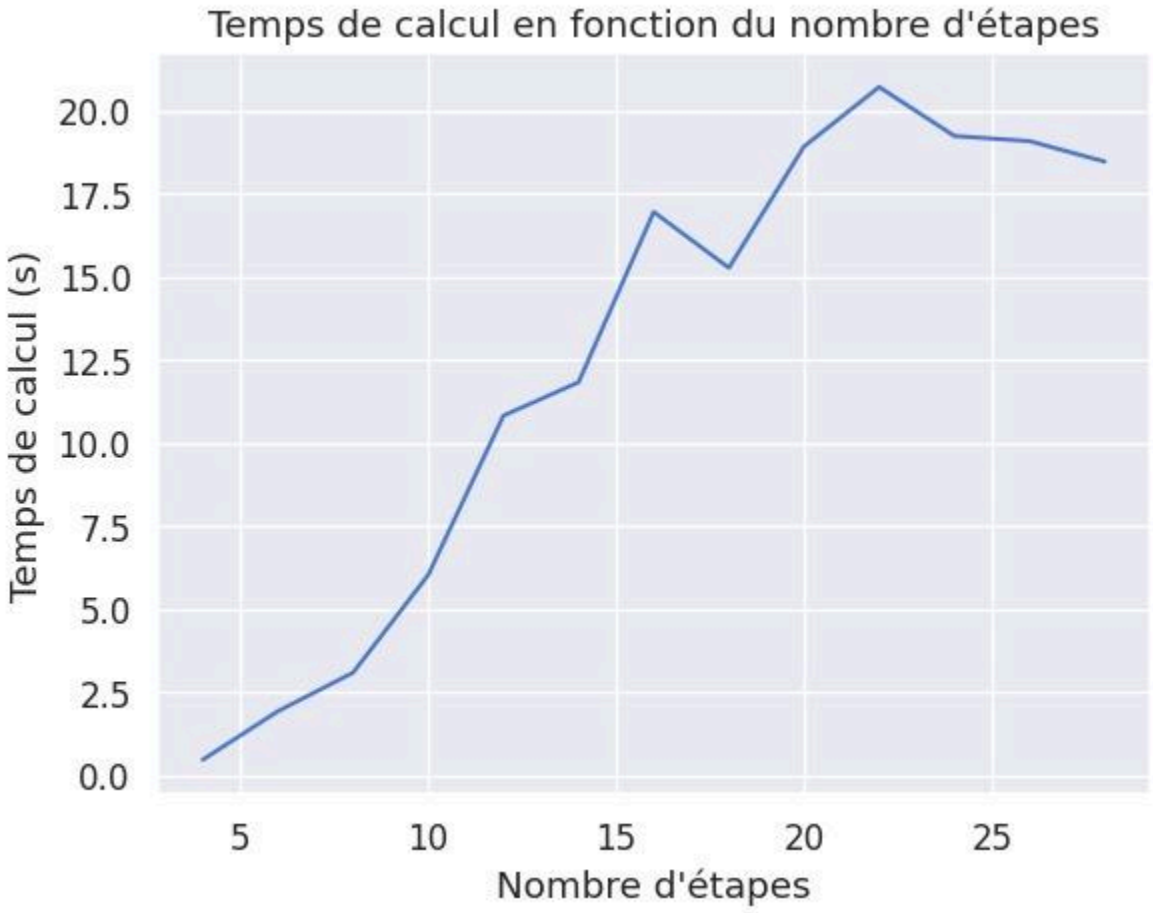
Différentes questions sont posées par la base Brenda, à commencer par sa taille lors des phases de tests. Une fois les optimisations mises en place, nous avons pu passer à la base de données finale : la base Brenda. Pour cela, nous avons utilisé plusieurs configurations de référence sur lesquelles nous avons évalué plusieurs grandeurs :

- Le nombre de mécanismes obtenus
- Le nombre total de molécules explorées
- Le temps d'exécution de l'algorithme

Ces grandeurs ont été évaluées en fonction du nombre maximal d'itérations de l'algorithme.

Pour donner un ordre d'idée par rapport aux performances de notre algorithme, la base Brenda contient 115 000 molécules et 180 000 réactions. Elles sont toutes prises en compte dans notre étude.

Ci-dessous, nous présentons les résultats de notre étude pour les réactifs "phosphoenolpyruvate" et "L-aspartate", le produit "oxaloacetate", avec dans le milieu réactionnel "H2O" et "NADH".

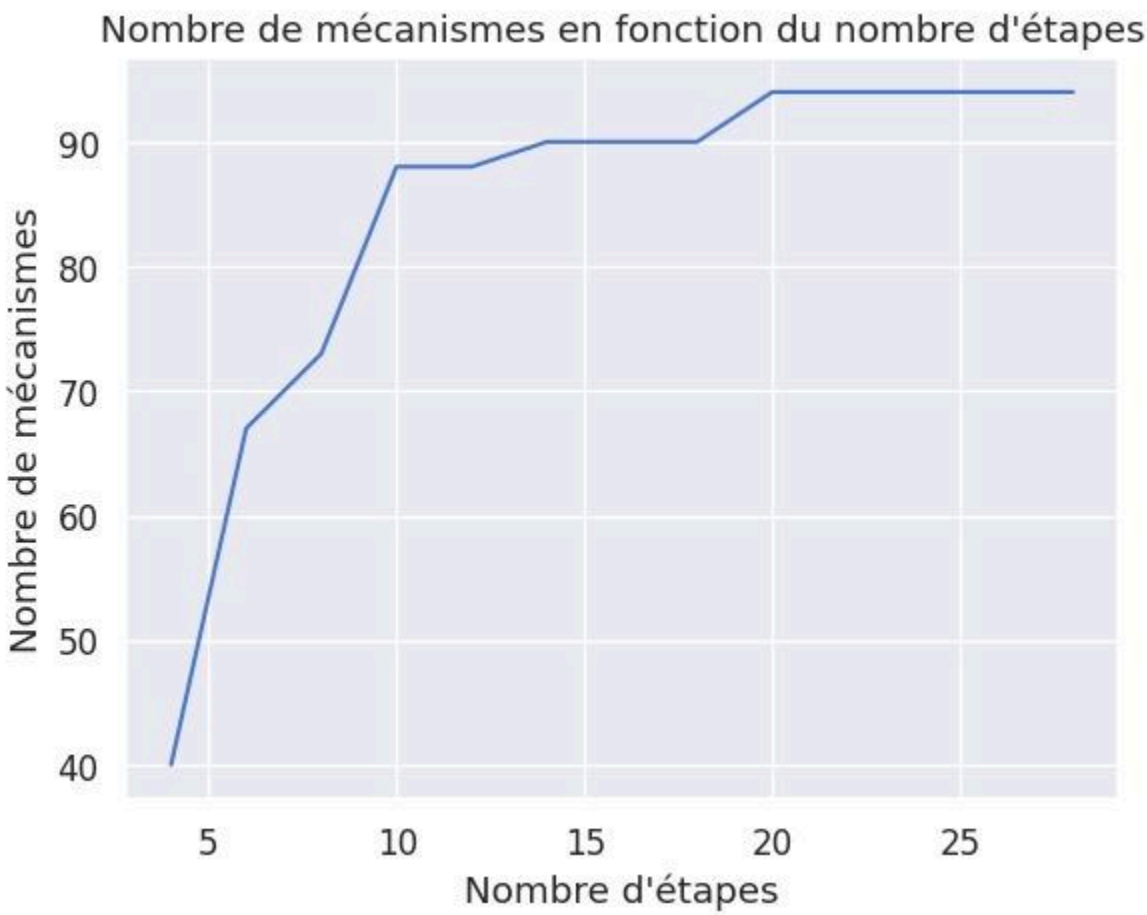


Durée d'exécution de l'algorithme pour chaque numéro d'étapes maximale entre 4 et 28

On constate une croissance exponentielle au début de l'algorithme, lors de l'explosion du nombre de molécules à partir des réactifs initiaux dans les premières étapes. Puis vient une phase de stabilisation correspondant au moment où toutes les molécules possibles ont été explorées : l'ensemble du CRN concret a été exploré et le temps de calcul devient alors relativement constant (variabilité de l'ordinateur).

Remarque : Le minimum local à 8 étapes est dû à la variabilité du temps de calcul de l'ordinateur, il devrait en théorie être au-dessus du temps à 7 étapes.

Les analyses précédentes sont confirmées par le schéma suivant :



Nombre de mécanismes déterminés par l'algorithme pour chaque numéro d'étapes maximal entre 2 et 9

5. Un algorithme de simulation

Afin de vérifier si les milieux réactionnels que nous obtenons par nos algorithmes fonctionnent réellement (ou plutôt en réalité discriminer les milieux qui sont sûrs de ne pas fonctionner), nous avons mis en place une simulation pour étudier les différentes réactions

chimiques en tenant compte du facteur de vitesse et de la concentration des différentes espèces présentes dans le milieu. L'objectif de cette simulation était de valider les résultats obtenus par l'algorithme binaire théorique avant de passer aux différents tests en laboratoire.

Nous avons d'abord réalisé une simulation en fonction du temps, mais il est apparu difficile de définir une échelle de temps adaptée en raison des variations importantes des constantes de vitesse d'une réaction à l'autre. Nous avons donc opté pour une approche basée sur les quantités de matière présentes dans le système, où chaque boucle de simulation fait avancer d'un pas fixe le nombre de molécules qui réagissent.

Pour mettre en œuvre cette simulation, nous avons utilisé des listes pour coder les réactions et les molécules par des numéros. Nous avons estimé qu'il n'était pas nécessaire de recourir à un dictionnaire ou à une autre forme de base de données, car il fallait de toute façon parcourir tous les éléments à chaque boucle.

L'algorithme de simulation se déroule comme suit : ***

1. Le logiciel définit d'abord le pas, c'est-à-dire la quantité de matière à faire avancer à chaque itération de la boucle. On va prendre le minimum des quantités initiales et le diviser par une précision choisie.
2. Ensuite, à l'aide des formules cinétiques classiques en chimie, on calcule les différentes vitesses de réaction.
3. On impose la réaction la plus rapide avec une vitesse de 1 et on détermine les autres vitesses relatives en effectuant une règle de trois.
4. On fait avancer la réaction la plus rapide d'un avancement correspondant au pas et les autres réactions d'un avancement proportionnel à leur facteur de vitesse.
5. On ajoute les produits formés, on retire les réactifs consommés et on lance la boucle suivante.

```python
def simulation(nombre_etape, nbequationminimale):
    compteur = 0  # Compteur pour le nombre de pas
    # la quantité de matière que je veux créer a chaque étape dans le produit, (peut faire disparaitre +, problème de création de matière)
    PRESENCE = copy.deepcopy(MOLECULESS)
    while compteur < nombre_etape:
        # Liste des réactions à considérer
        Inter = Num_Reaction_Actif(PRESENCE)
        # calcul des différentes vitesse à considérer
        Vitesse_Inter = Calc_Vitesse(PRESENCE, Inter)
        if sum(Vitesse_Inter) < 0.0000001:
            return "session terminer en avance", PRESENCE[29]
        vitesse_max = max(Vitesse_Inter);
        if len(Inter) > nbequationminimale:
            Vitesse_Inter, Inter = Calc_Reaction_retenues(Vitesse_Inter, Inter, vitesse_max, nbequationminimale) #on récupère la liste des réactions ainsi que les vitesse q
        Reactifs_Necessaires = Calc_Reactifs_Necessaires(Inter)
        Vitesse_Reactifs_Necessaires, Indice_Composante_Des_Vitesses= Calc_Vitesse_Reactifs_Necessaires(Reactifs_Necessaires, Inter, Vitesse_Inter)
        nmax_par_molecule = [PRESENCE[i][1] for i in Reactifs_Necessaires]
        pas, Rapport_Vitesse_Molecules = Calc_pas(nmax_par_molecule, Vitesse_Reactifs_Necessaires)
        reactifs_a_enlever, produit_a_ajouter = Simulation_Reaction_Etape(pas, Rapport_Vitesse_Molecules, Reactifs_Necessaires, Indice_Composante_Des_Vitesses, Vitesse_Inte
        Produit = Cre_Produit(Inter, produit_a_ajouter)  # Je crée les produits

        for k in range(len(Produit)): ##ca sert a acutaliser la liste en ajoutant les produits, j'aurai pu le mettre dans une fonction auxilliaire
            i = Produit[k][0]
            if Produit[k][1] > 0:
                PRESENCE[i][0] = True
                # j'ajoute les produits à la liste PRESENCE
                PRESENCE[i][1] = PRESENCE[i][1] + Produit[k][1]

        PRESENCE = Cre_PresenceBis(PRESENCE, reactifs_a_enlever, Reactifs_Necessaires) # Je soustrais aux réactifs ce qui a été consommé
        compteur = compteur + 1

    # Retourne la composition du milieu Ici le 29 c'est le resorufin, cohérent, on voit que si on ajoute ou non NAD on obtien soit 1 soit un chiffre enorme.
    return (PRESENCE[29], time.time() - start)


simulation(10348,10)
```

Les résultats obtenus sont plutôt concluants. On remarque que la quantité avec ou sans NAD sont proportionnellement cohérent avec les résultats attendus :

Résultat théorique		
	En absence de NAD	En presence de NAD
Présence de Resorufin	True	False
Présence de Gluconolacrone	True	False

Résultat Simulation(90 000, 10), 8s execution		
	En absence de NAD	En présence de NAD
Présence de Resorufin	[True,9335]	[True, 9E-8]
Présence de Gluconolacrone	[True, 21.25]	[True, 9E-15]

Résultat Simulation(900 000, 10), 60s execution		
	En absence de NAD	En présence de NAD
Présence de Resorufin	[True,20393]	[True, 74.65]
Présence de Gluconolacrone	[True, 2263]	[True, 68.135]

Toutefois, l'algorithme rencontre plusieurs problèmes :

1. Les constantes utilisées sont celles de Michaelis et non les constantes de réaction classiques. Cependant, ce problème est mineur car la logique du programme reste la même ; il suffit de modifier la manière dont on calcule la vitesse.

2. Le problème de complexité : l'algorithme a une complexité en O(m*n), avec m étant le nombre de molécules actuellement présentes et n étant le nombre de réactions possibles avec toutes ces molécules. Cela peut poser des problèmes de temps d'exécution lorsqu'on utilise la base BRENDA. De plus ici, on voit bien qu'on n'atteint pas l'équilibre même avec 900000 cycles qui nous prend déjà plus d'une minute pour être exécuter. Ainsi avec cette efficacité, il suffit que la molécule arrive un peu tardivement dans la chaîne des réactions et nous ne la verrons jamais.
3. Le problème de surconsommation : l'algorithme peut consommer plus de réactifs qu'il n'y en a actuellement disponibles, car on ne vérifie jamais si les réactifs sont suffisants. Cela devient un problème grave, car on peut former des produits qui ne devraient pas se former, ou qui devraient se former en quantités infimes. Par exemple, on peut imaginer une situation où des produits de base impossibles à former sont créés, ou des quantités négligeables de certains produits sont générées :

Pas = 100		
Vitesse de Réaction	1	0,2
Quantité des réactifs présents	1000	10
Quantité consomée lors de l'étape	100	20

On voit que dans cette situation, la deuxième réaction ne devrait pas faire réagir 20 unités de réactifs. Cette situation peut être raccroché au problème numéro 2 du fait qu'on n'atteignait jamais l'équilibre car la surconsommation permet la création en continue et infini de produits. Ainsi, il est difficile de définir un seuil où on considère qu'il n'y en a pas dans la vraie vie car avec suffisamment de temps on pourra en créer autant que l'on désire dans les deux cas et ce n'est que la comparaison des 2 résultats en fonction de la présence ou l'absence de NAD qui nous permet de conclure.

Afin de résoudre les problèmes rencontrés, nous avons exploré plusieurs solutions. Pour la complexité en O(n*m), étant donné que cette complexité est liée au fait de considérer toutes les réactions, une solution serait de ne prendre en compte que les réactions ayant une vitesse suffisante. En effet, les lois de vitesse classiques en chimie indiquent un lien

entre la vitesse de réaction et la quantité de réactifs, rendant cette hypothèse plausible. Cependant, il reste à déterminer la méthode de sélection des réactions.

Le problème de surconsommation semble, a priori, simple à résoudre. Une possibilité serait d'ajuster le pas en fonction de la quantité de molécules les moins abondantes.

Cependant, il faudrait alors s'assurer que le programme n'atteint pas une situation d'immobilisme, similaire au problème d'Achille et la tortue. Malgré cette solution, le problème ne serait pas entièrement résolu, car un même réactif peut être présent dans plusieurs réactions, ce qui peut entraîner une surconsommation globale.

Après plusieurs essais en explorant différentes options, nous avons trouvé la solution la plus adaptée à notre problème. Nous avons conclu que la manière la plus efficace de résoudre la complexité était de limiter le nombre maximal de réactions, tout en garantissant un nombre minimal de réactions. Cette approche permet d'éviter que certaines réactions cruciales pour les mécanismes ultérieurs soient ignorées.

En limitant le nombre de réactions, nous nous assurons également de ne pas rencontrer le problème d'Achille et la tortue, car lorsque la quantité de matière devient faible, sa constante de réaction diminue également. Pour résoudre le problème de surconsommation, nous avons introduit une vitesse supplémentaire correspondant à la vitesse de réaction de la molécule elle-même en additionnant les vitesses des réactions auxquelles elle participe. Nous raisonnons désormais sur la "vitesse des molécules" plutôt que sur les vitesses individuelles des réactions.

L'algorithme fonctionne comme suit :

1. On calcule la vitesse des molécules en utilisant les vitesses de réaction, puis on détermine le pas dynamique pour chaque étape en prenant le minimum de la division de la quantité par la vitesse normalisée. Ainsi, on évite la surconsommation.
2. Grâce à un tableau intermédiaire indiquant le poids de chaque réaction sur la vitesse des molécules, on attribue un avancement pour chaque réaction à partir de l'avancement des molécules.
3. On ajoute ensuite les produits et les réactifs, et on relance la boucle de simulation.

En appliquant cette méthode, nous avons réussi à atténuer les problèmes de complexité et de surconsommation tout en préservant l'exactitude de la simulation des réactions chimiques.

```python
def simulation(nombre_etape, nbequationminimale):
    compteur = 0  # Compteur pour le nombre de pas
    # la quantité de matière que je veux créer a chaque étape dans le produit, (peut faire disparaitre +, problème de création de matière)
    PRESENCE = copy.deepcopy(MOLECULESS)
    while compteur < nombre_etape:
        # Liste des réactions à considérer
        Inter = Num_Reaction_Actif(PRESENCE)
        # calcul des différentes vitesse à considérer
        Vitesse_Inter = Calc_Vitesse(PRESENCE, Inter)
        if sum(Vitesse_Inter) < 0.0000001:
            return "session terminer en avance", PRESENCE[29]
        vitesse_max = max(Vitesse_Inter);
        if len(Inter) > nbequationminimale:
            Vitesse_Inter, Inter = Calc_Reaction_retenues(Vitesse_Inter, Inter, vitesse_max, nbequationminimale) #on récupère la liste des réactions ainsi que les vitesse q
        Reactifs_Necessaires = Calc_Reactifs_Necessaires(Inter)
        Vitesse_Reactifs_Necessaires, Indice_Composante_Des_Vitesses= Calc_Vitesse_Reactifs_Necessaires(Reactifs_Necessaires, Inter, Vitesse_Inter)
        nmax_par_molecule = [PRESENCE[i][1] for i in Reactifs_Necessaires]
        pas, Rapport_Vitesse_Molecules = Calc_pas(nmax_par_molecule, Vitesse_Reactifs_Necessaires)
        reactifs_a_enlever, produit_a_ajouter = Simulation_Reaction_Etape(pas, Rapport_Vitesse_Molecules, Reactifs_Necessaires, Indice_Composante_Des_Vitesses, Vitesse_Inte
        Produit = Cre_Produit(Inter, produit_a_ajouter)  # Je crée les produits

        for k in range(len(Produit)): ##ca sert a acutaliser la liste en ajoutant les produits, j'aurai pu le mettre dans une fonction auxilliaire
            i = Produit[k][0]
            if Produit[k][1] > 0:
                PRESENCE[i][0] = True
                # j'ajoute les produits à la liste PRESENCE
                PRESENCE[i][1] = PRESENCE[i][1] + Produit[k][1]

        PRESENCE = Cre_PresenceBis(PRESENCE, reactifs_a_enlever, Reactifs_Necessaires) # Je soustrais aux réactifs ce qui a été consommé
        compteur = compteur + 1

    # Retourne la composition du milieu Ici le 29 c'est le resorufin, cohérent, on voit que si on ajoute ou non NAD on obtien soit 1 soit un chiffre enorme.
    return (PRESENCE[29], time.time() - start)


simulation(10348,10)
```

Les résultats que nous avons obtenus sont comparables à ceux obtenus précédemment, à l'exception d'une petite quantité en présence de NAD, qui est maintenant remplacée par False en raison de l'interdiction de surconsommation. De plus, nous avons atteint un résultat final après seulement quelques milliers d'étapes et en moins de 5 secondes, grâce à la résolution des problèmes de complexité et de surconsommation.

III. Théorisation du problème

1. Passage à un problème mathématique

Le problème de détermination d'un CRN concret à partir d'un CRN abstrait nécessitant un ensemble d'algorithmes plus ou moins complexes et dont les résultats sont difficilement vérifiables, il est vite apparu comme une nécessité de définir le problème sous un aspect mathématique pour pouvoir effectuer des preuves sur la validité de nos résultats. Lors de

la première moitié du PSC nous avons tenté de prouver la correction et de déterminer la complexité des algorithmes que nous avions développés. Cependant les preuves étaient compliquées notamment à cause de la définition du système d'étiquettes. Nous avons donc décidé d'aborder le problème sous une autre approche.

L'idée a été de partir des espèces initiales et de travailler uniquement sur ces espèces pour trouver le CRN. De fait, la construction d'un CRN à partir des espèces initiales est facile et le travail sur les espèces initiales permettait de résoudre le problème d'étiquettes.

Notre algorithme est composé de deux parties majeures similaires à l'algorithme original : la descente et la remontée. La descente a le but suivant : en partant de l'ensemble des espèces initiales disponibles pour le système, on détermine l'ensemble des combinaisons de ces espèces initiales permettant d'obtenir le produit souhaité. Une des différences majeures de notre algorithme réside dans le fait que nous fixons le nombre n d'étapes autorisées pour la descente de sorte que l'algorithme finit nécessairement. Le résultat de la descente est donc le suivant : l'ensemble des combinaisons d'espèces initiales permettant d'obtenir l'espèce recherchée en moins de n étapes. De là la remontée a pour but de tester les différents CRN obtenus avec ces conditions initiales jusqu'à obtenir le CRN recherché. Une étape de tri des combinaisons d'espèces initiales peut ensuite être réalisée en fonction de la caractéristique recherchée sur le CRN. Le résultat de la remontée est le CRN recherché.

La complexité d'un tel algorithme serait très grande: construire un CRN nécessite de parcourir plusieurs fois l'ensemble des molécules présentes ce qui, testé potentiellement sur toutes les combinaisons possibles, peut rapidement faire exploser le nombre d'opérations. De fait, nous avons déterminé des conditions à vérifier avant de générer des CRN telles que la présence ou non des espèces étudiées, le fait qu'un sous-ensemble déjà testé ne renvoyant pas le bon résultat soit présent… Ces vérifications permettent de réduire le temps de calcul en générant moins de CRN. De plus, nous avons modifié la manière de stocker les informations sur les réactions en créant un « dictionnaire » (R dans le pseudo-code) qui contient dans la case (i,j) l'ensemble des produits créés par la réaction entre i et j. Bien sûr ce choix impose que les réactions considérées n'aient que 2 réactifs mais quitte à augmenter la dimension de la matrice on peut se ramener sans difficulté à des réactions à n réactifs. Ce choix, quoique gourmand en stockage, permet d'optimiser la recherche de la réaction entre i et j.

Nous avons de plus dû penser à des algorithmes prenant en compte les concentrations. En effet l'algorithme de négation nécessite cette prise en compte. Nous n'avons pas développé d'algorithmes permettant ce calcul des concentrations car le logiciel BioCham développé par les équipes de monsieur FAGES le permet déjà. Une API devrait être créée par ces mêmes équipes permettant de faire le lien entre notre algorithme et BioCham pour le calcul des concentrations.

2. Un nouvel algorithme

Cette partie a pour but de présenter l'ensemble des algorithmes théoriques que nous avons développés pour répondre à la problématique de détermination d'un CRN.
Algorithme de détermination du CRN à partir d'espèces initiales (nécessaire au fonctionnement global) :

Entrées :
ini = [espèces initiales étudiées]
Initialisation :
Pres = [ensemble des molécules présentes dans le CRN] = I
Cree = [ensemble des molécules créées à la dernière étape] = I
CRN = [ensemble des réactions effectuées] = [ ]
Exécution :

```
Tant que Cree [ ] :
        newEsp = [ ]
        Pour mol1 dans Cree :
                Pour mol2 dans Pres :
                        Pour mol3 dans R[i][j] :
                                Si mol3 Pres :
                                        newEsp = newEspU{mol3}
                                CRN = CRNU{(mol1,mol2,mol3)}
        Cree = newEsp
        Pres = PresUnewEsp
Sortie :  CRN
```

Algorithme de descente :

Entrées :

I=[espèces initiales]

n=nombre d'étapes limites

C=molécule étudiée

Initialisation :

k=nombre d'étape déjà effectuées=0

Pres=[espèces présentes à l'étape k]=I

Cree=[espèces crées à l'étape k]=I

listeIni=[…,[combinaisons d'espèces initiales permettant d'obtenir du i en moins de k étapes],…]=[[],…,[]]

Exécution :

```
Tant que k<n :
        newEsp=[ ]
        Pour mol1 dans Cree :
                Pour mol2 dans Pres :
                        Pour mol3 dans Reac[mol1][mol2] :
                                newEsp = newEsp+mol3
                                listeIni[mol3] = listeIni[mol3]+(listeIni[mol1]+listeIni[mol2])
        Cree=newEsp
        Pres=Pres+newEsp
        k=k+1
Sortie :  listeIni[C]
```

Dans le cas ou on ne prend pas en compte les concentrations :

Algorithme du ET : résolution pour "A ET B DONNE C"

Entrées :

listeIni = [liste des ensembles d'espèces initiales permettant d'obtenir C]

A

B

C

Initialisation :

tri(listeIni) // classe la liste selon un ordre choisi

Exécution :

Pour ini dans listeIni :

    Si A ou B n'appartient pas à ini :

        None

    Sinon :

        sansA = CRN(ini\{A})

        sansB = CRN(ini\{B})

        Si csansA ou csansB :

            None

        Sinon :

            Sortie : CRN(ini)

Algorithme du OU : résolution pour "A OU B DONNE C"

Entrées :

listeIni = [liste des ensembles d'espèces initiales permettant d'obtenir C]

A

B

C

Initialisation :

tri(listeIni) // classe la liste selon un ordre choisi

Exécution :

Pour ini1 dans listeIni :

    Pour ini2 dans listeIni :

        ini = ini1+ini2

        Si A ou B n'appartient pas à ini:

            None

        Sinon :

            sansRien = CRN(ini\{A,B})

            Si csansRien :

                None

Sinon :

Sortie : CRN(ini)

Dans le cas où on prend en compte en compte les concentrations :

Algorithme du ET : résolution pour “A ET B DONNE C”

Entrées :

listeIni = [liste des ensembles d’espèces initiales permettant d’obtenir C]

A

B

C

Initialisation :

tri(listeIni) // classe la liste selon un ordre choisi

Exécution :

Pour ini dans listeIni :

Si A ou B n’appartient pas à ini :

None

Sinon :

sansA = CRN(ini\{A})

sansB = CRN(ini\{B})

Si [C]>seuil dans sansA ou sansB :

None

Sinon :

Resultat = CRN(ini)

Si [C]>seuil dans CRN(ini) :

Sortie : Resultat

Sinon :

None

Algorithme du OU : résolution pour “A OU B DONNE C”

Entrées :

listeIni = [liste des ensembles d’espèces initiales permettant d’obtenir C]

A

B

C

Initialisation :

tri(listeIni) // classe la liste selon un ordre choisi

Exécution :

```
Pour ini1 dans listeIni :
    Pour ini2 dans listeIni :
        ini = ini1+ini2
        Si A ou B n'appartient pas à ini:
            None
        Sinon :
            sansRien = CRN(ini\{A,B})
            Si [C]>seuil dans sansRien:
                None
            Sinon :
            Resultat = CRN(ini)
            Si [C]>seuil dans CRN(ini) :
                Sortie :  Resultat
            Sinon :
                None
```

Remarque : On remarque que la différence majeure avec le cas où on ne prend pas en compte les concentrations est que ici un test est effectué sur le résultat pour vérifier que C est présent en quantité suffisante (par rapport à un seuil défini de manière globale).

Algorithme du NON : résolution pour "A ET NON B DONNE C"

Entrées :

listeIni = [liste des ensembles d'espèces initiales permettant d'obtenir C]

A

B

C

Initialisation :

tri(listeIni) // classe la liste selon un ordre choisi

Exécution :

Pour ini dans listeIni :

    Si A n'appartient pas à ini ou B appartient à ini :

        None

    Sinon :

        sansA = CRN(ini\{A})

        avecB = CRN(iniU{B})

        Si [C]>seuil dans sansA ou dans avecB :

            None

        Sinon :

            Resultat = CRN(ini)

            Si [C]>seuil dans CRN(ini) :

                Sortie : Resultat

            Sinon :

                None

La combinaison de l'ensemble de ces algorithmes donne l'algorithme globale de réponse suivant :

Algorithme global pour la relation : "A R B DONNE C"

Entrées :

A

B

C

R

I = [ensemble d'espèces initiales disponibles]

n = nombre maximum d'étapes autorisées

Descente :

listeIni = descente(I,n,C)

Remontée :

Si R=ET :

    RemontéeET(listeIni,A,B,C)

Si R=OU :

RemontéeOU(listeIni,A,B,C)

Si R=NON :

RemontéeNON(listeIni,A,B,C)

3. Correction de l'algorithme

Cette partie est dédiée à l'ensemble des preuves de correction des différents algorithmes.

Preuve de correction de l'algorithme de création du CRN :

L'objectif de cet algorithme est de renvoyer le CRN engendré par les espèces initiales fournies c'est-à-dire l'ensemble des réactions possibles à partir des réactifs présents.

Premièrement l'algorithme termine car on a l'invariant de boucle suivant: le nombre de molécules ajoutées au CRN à chaque étape est supérieur à 1 et le nombre de molécules disponibles diminue de 1. De là le nombre de molécules disponibles étant fini l'algorithme termine nécessairement.

De plus, on remarque que l'ensemble des actions possibles est épuisé pour chaque espèce introduite. De fait l'ensemble de réactions renvoyées correspond bien au CRN associé à l'ensemble des espèces initiales.

Preuve de correction de l'algorithme de la descente :

Cet algorithme doit nous renvoyer la liste des ensembles d'espèces initiales permettant d'obtenir du C en moins de n réactions.

On remarque déjà que si une combinaison est renvoyée dans la liste des résultats alors elle permet la création de C par construction. De plus, le fait de tester l'ensemble des nouvelles réactions possibles à chaque étape assure que si une combinaison d'espèces initiales permet d'obtenir C en moins de n étapes alors elle est identifiée et renvoyée avec le résultat.

Preuve de correction de l'algorithme du ET : résolution pour "A ET B DONNE C" sans concentration :

L'objectif de l'algorithme dans le cas du ET est de renvoyer un CRN correspondant à l'équation logique "A ET B DONNE C". Pour ce faire nous partons d'un ensemble de combinaisons d'espèces initiales possibles (obtenu pendant la descente), et nous construisons deux CRN qui sont des CRN "tests".

Le premier CRN est le CRN formé à partir des espèces initiales sans A et le deuxième sans B. Si C appartient à l'un de ces deux CRN alors le CRN de l'ensemble des espèces initiales ne vérifie pas "A ET B DONNE C". Sinon le CRN généré par les espèces initiales vérifie la bonne relation logique car :

si A est retiré alors pas de C

si B est retiré alors pas de C

C est bien formé car les ensembles d'espèces initiales fournis sont ceux obtenus lors de la descente qui forment du C en moins de n étapes

Cet algorithme renvoie donc bien le résultat demandé.

<u>Preuve de correction de l'algorithme du OU : résolution pour "A OU B DONNE C" sans concentration :</u>

L'objectif de l'algorithme dans le cas du OU est de renvoyer un CRN correspondant à l'équation logique "A OU B DONNE C". Pour ce faire nous partons de deux ensembles de combinaisons d'espèces initiales possibles (obtenu pendant la descente) l'un contenant au moins A et l'autre au moins B, et nous construisons un CRN qui est un CRN "test" à partir de deux ensembles d'espèces initiales.

Le CRN test est le CRN formé à partir de la fusion de deux ensembles d'espèces initiales auquels on retire A et B. Si C appartient au CRN alors le CRN de l'ensemble fusionné des espèces initiales ne vérifie pas "A OU B DONNE C". Sinon le CRN généré par la fusion des deux ensembles d'espèces initiales vérifie la bonne relation logique car :

si A et B sont retirés alors pas de C

si A ou B est présent alors C

Cet algorithme renvoie donc bien le résultat demandé.

<u>Preuve de correction de l'algorithme du ET et du OU : résolution pour "A ET B DONNE C" et résolution pour "A OU B DONNE C" avec concentration :</u>

L'algorithme est sensiblement le même que dans le cas sans les vérifications de concentration. La seule différence consiste dans la définition de "C appartient au CRN" qui

dans le premier cas est un booléen présent ou absent alors qu'ici on vérifie que la concentration est bien supérieure à un seuil. Ainsi ces deux algorithmes fonctionnent pour les mêmes raisons que dans le cas sans les concentrations.

Preuve de correction de l'algorithme du NON : résolution pour "A ET NON B DONNE C"

Le cas du NON est intrinsèquement lié aux concentrations, c'est pour cela que nous n'en avons pas développé avant. Ici la méthode pour trouver le CRN associé est la suivante : pour chaque ensemble d'espèces initiales on cherche à savoir si cet ensemble donne le CRN recherché ou non. Dès que c'est le cas, on s'arrête et on renvoie le CRN.

Par conséquent, pour vérifier que l'on a trouvé le bon CRN on procède comme suit : premièrement il faut que A appartienne aux espèces initiales et que B n'y appartiennent pas. Ensuite si on retire A ou si on ajoute B le CRN ne doit plus contenir de C. De là on effectue les deux tests proposés. Si ces deux tests sont passés, il reste un dernier test à effectuer : vérifier que le CRN final C est présent en quantités suffisantes. D'où le dernier test effectué sur le CRN renvoyé.

Finalement, si un CRN est renvoyé, il vérifie bien toutes les conditions.

Preuve de correction de l'algorithme global :

On remarque que cet algorithme est basé à 100% sur les fonctions auxiliaires décrites précédemment. Ainsi avec les résultats énoncés précédemment on sait que si un résultat est renvoyé il s'agira d'un CRN associé à la relation logique dans lequel C est obtenu en moins de n étapes.

Conclusion

In fine, le problème de recherche de chemin réactionnel s'est avéré bien plus compliqué qu'il ne nous a semblé à première vue, tant par la grandeur de la base Brenda qui force à la confection d'algorithmes très efficaces que par les difficultés à représenter une réaction biologique de manière informatique. Nous avons dû effectuer une approximation sur la seconde difficulté en supposant un comportement booléen (présence/absence) des molécules en jeu. Cela nous a permis dans un premier temps de développer un algorithme très efficace afin de traiter tous les CRN abstraits ainsi que tous les systèmes de CRNs abstraits.

Cependant, afin de peser la validité de l'approximation faite, il fallait réaliser une simulation du milieu réactionnel en traitant les différentes molécules sous forme de concentrations dans le milieu réactionnel et non plus comme simplement une présence/absence. Pour ce faire nous avons programmé une simulation basique afin d'observer le comportement du milieu réactionnel d'une manière moins approximative une fois que nous avons obtenu des idées d'ensembles de molécules à mettre dans le milieu réactionnel.

L'étape suivante, que nous aurions pu essayer d'effectuer avec un peu plus de temps, aurait été les tests directement dans un tube à essai pour vérifier la faisabilité des tests pharmacologiques trouvés.

Après une réunion avec notre co-coordinateur de biologie Yves Mechulam, nous avons déterminé que nous pouvions réaliser cet essai en vérifiant quelques contraintes. Tout d'abord il fallait limiter l'utilisation de l'algorithme à une base de données de molécules disponibles en laboratoire pas trop cher, ce qui semble aussi être une contrainte pour l'utilisation finale sous forme de tests pharmacologiques.

De plus, pour tester la présence jointe de deux molécules qu'il nous fallait déterminer, il fallait choisir un produit final facilement observable. Pour ce faire, une molécule ayant une émission lumineuse stimulée (dans le visible ou non) comme un fluorophore conjugué était un bon candidat. Ces tests seraient donc la continuation logique de ce PSC.

Un autre prolongement consisterait à sélectionner les mécanismes chimiquement cohérents que notre algorithme renvoie. Notre exécution en exemple de l'algorithme fournit près de 100 mécanismes potentiels pour un grand nombre d'étapes, mais la plupart n'auront en réalité pas lieu en raison de la cinétique et en fonction des conditions réactionnelles. Notre outil fournit une bonne première approximation des mécanismes chimiquement cohérents, de façon efficace.

Parallèlement, nous nous sommes rendu compte que le premier algorithme que nous avions programmé était structurellement plus difficile à démontrer: bien qu'étant utilisable dans les faits il n'était pas pratique mathématiquement. Nous avons donc décidé de le restructurer afin de fournir une étude théorique plus compréhensible.

Bibliographie

Cover image : Molecular Systems Biology, 14(4), 2018
[1] Patrick Amar. Bio-calculateur booléen à base de réseaux métaboliques intégré dans une vésicule artificielle : temporisation par oscillateur enzymatique. Séminaire de la Société Francophone de Biologie Théorique, Jun 2018, Saint Flour, France. ⟨hal-01817607⟩
[2] François Fages. Machinerie cellulaire et programmation biochimique: vers une informatique de la cellule. Forum des lauréats en informatique et mathématiques appliquées 2014, 2014, Collège de France, Paris, France. hal-01103343 (14 Jan 2015)

[3] François Fages, Franck Molina. La cellule, un calculateur analogique chimique. Approches symboliques de la modelisation et de l'analyse des systèmes biologiques., A paraître. hal 02518419 (25 Mar 2020)
[4] https://www.reaxys.com
[5] https://www.cas.org/solutions/cas-scifinder-discovery-platform/cas-scifinder
[6] Martin Davy, Etude de mécanismes biochimiques pouvant implémenter des portes logiques. Rapport de stage Sys2Diag CNRS 2018/2019