Développement d'un logiciel de visualisation de données métaboliques

Stage DESS CCI, Université F. Rabelais (Tours)

Agathe Madeleine

Parcours

- Formation initiale : maîtrise de biochimie
- Formation en informatique: CNAM et DESS CCI

Projet HELIX

Laboratoire de Biométrie et Biologie Evolutive
UMR CNRS 5558, Université C. Bernard, Lyon

Institut Suisse de Bioinformatique (ISB)

- Projet Helix (2001): Grenoble + Lyon
- > 40 personnes
  (chercheurs, ingénieurs, doctorants, étudiants)
- responsable : François Rechenmann

Thèmes de recherches :

- Informatique et génomique (bioinformatique)

Le projet Panoramix

Proteix

EC

EC

Chromosome A

Chromosome B

Genomix

Metabolix

gene

protéine

complexe protéique

EC enzyme

composés (ex: sucre...)

Réactions biochimiques

Développement d'un logiciel de visualisation de voies métaboliques

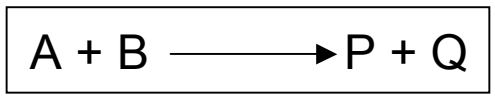
Cahier des charges :

- **Modélisation d'une voie métabolique**
- **Affichage**
- **Composant graphique Java**

Voie métabolique

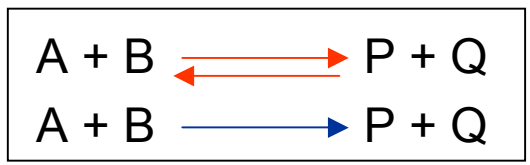
Ensemble de réactions chimiques

- **Réaction chimique :** transformation de substrats (A et B) en produits (P et Q)

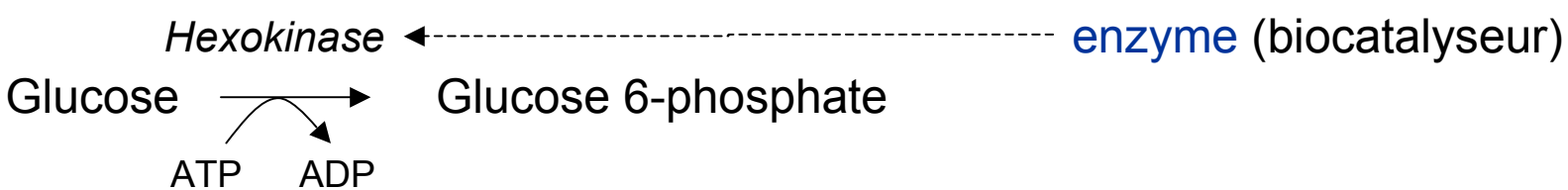


Exemple: Glucose + ATP ⟶ Glucose 6-phosphate + ADP

- **Réaction chimique :** réversible ou irréversible

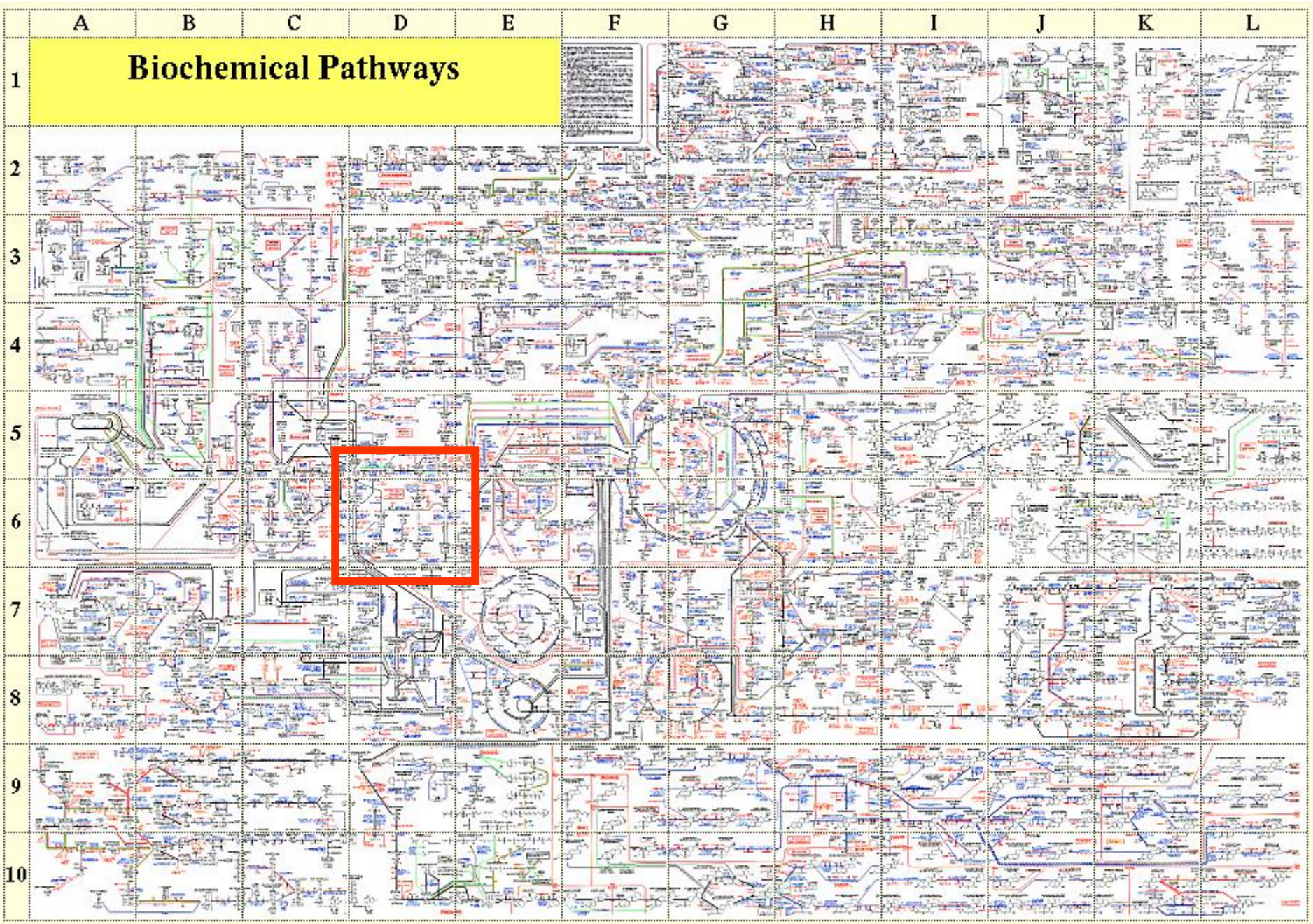


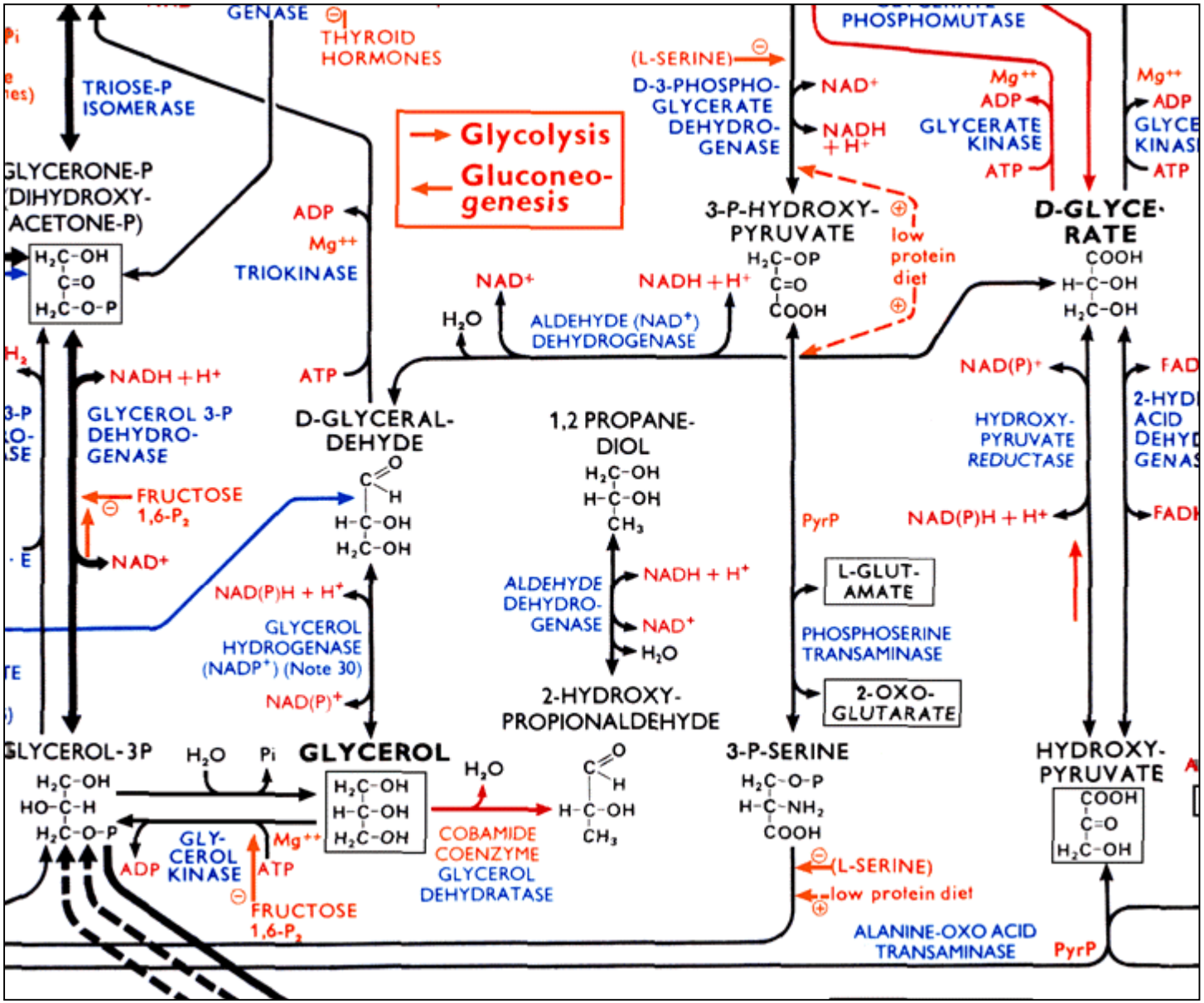
- **Réaction biochimique :** transformation chimique catalysée par une enzyme



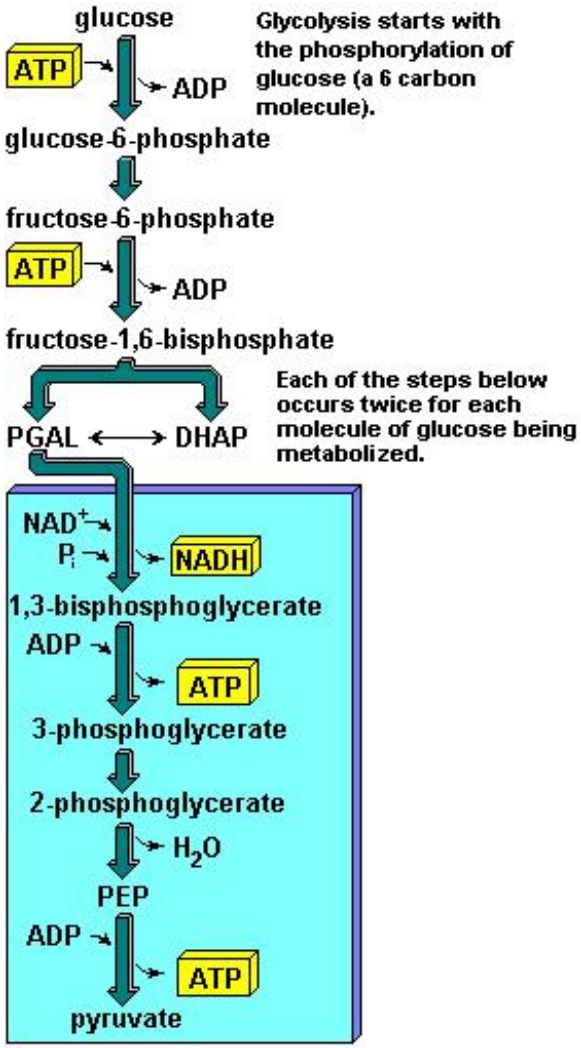
Composés primaires (principaux) : Glucose et le Glucose 6-phosphate

Composés secondaires : ATP et ADP

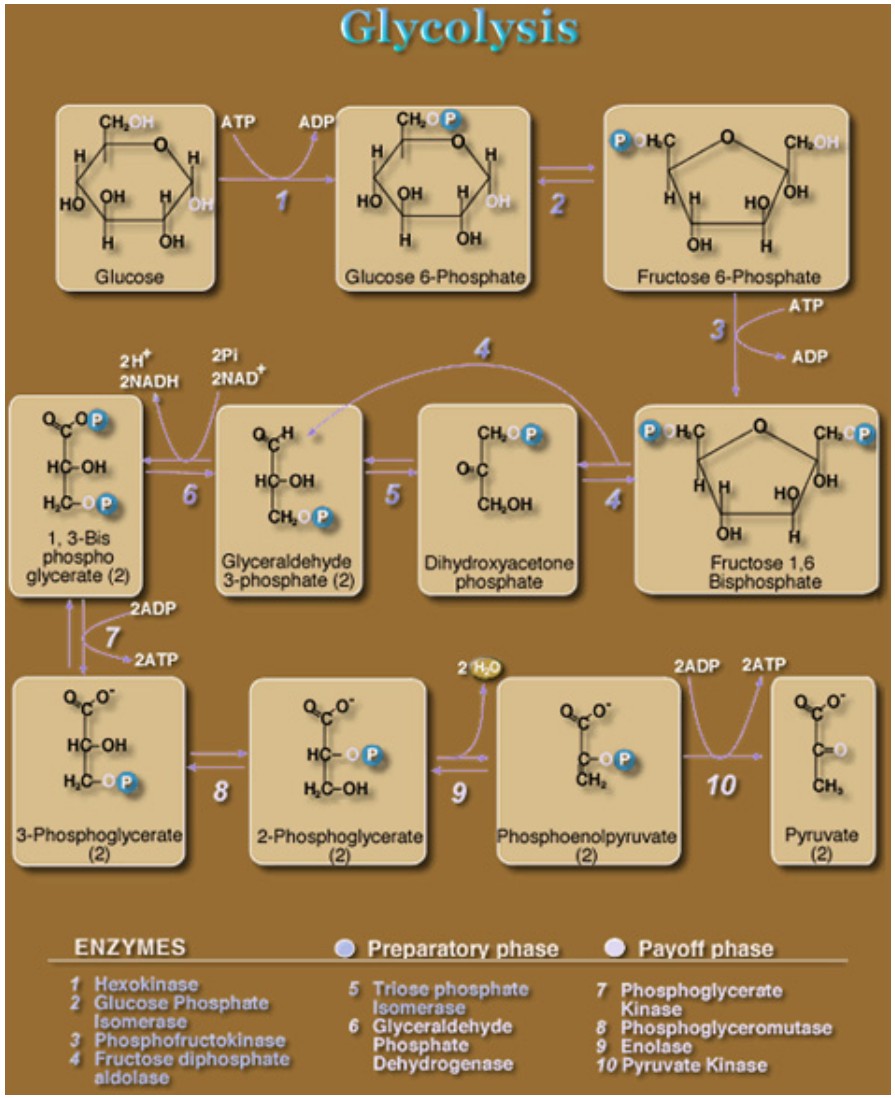

A
B
C
D
E
F
G
H
I
J
K
L
1
2
3
4
5
6
7
8
9
10
Biochemical Pathways


Glycolysis
Gluconeo-genesis
THYROID HORMONES
TRIOSE-P ISOMERASE
GLYCERONE-P (DIHYDROXY-ACETONE-P)
TRIOKINASE
ADP
Mg++
ATP
(L-SERINE)
D-3-PHOSPHO-GLYCERATE DEHYDRO-GENASE
NAD+
NADH + H+
PHOSPHOMUTASE
GLYCERATE KINASE
3-P-HYDROXY-PYRUVATE
low protein diet
D-GLYCE-RATE
ALDEHYDE (NAD+) DEHYDROGENASE
H2O
GLYCEROL 3-P DEHYDRO-GENASE
FRUCTOSE 1,6-P2
D-GLYCERAL-DEHYDE
1,2 PROPANE-DIOL
ALDEHYDE DEHYDRO-GENASE
2-HYDROXY-PROPIONALDEHYDE
NAD(P)+
NAD(P)H + H+
GLYCEROL HYDROGENASE (NADP+) (Note 30)
HYDROXY-PYRUVATE REDUCTASE
PyrP
L-GLUT-AMATE
PHOSPHOSERINE TRANSAMINASE
2-OXO-GLUTARATE
GLYCEROL-3P
Pi
GLYCEROL
GLY-CEROL KINASE
COBAMIDE COENZYME GLYCEROL DEHYDRATASE
3-P-SERINE
HYDROXY-PYRUVATE
ALANINE-OXO ACID TRANSAMINASE

Un exemple de voie métabolique : la glycolyse

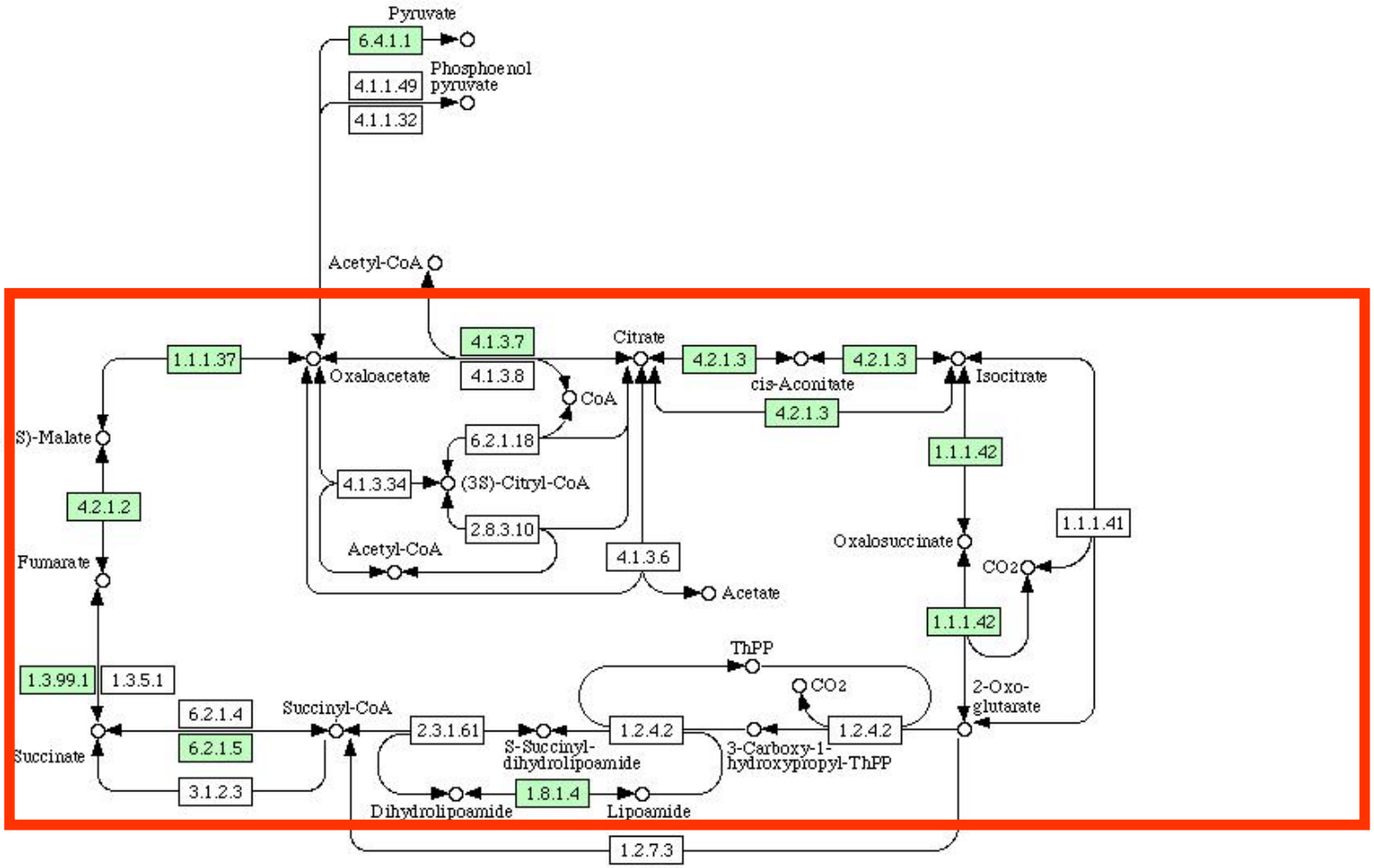


(Members.tripod.com/beckysroom)

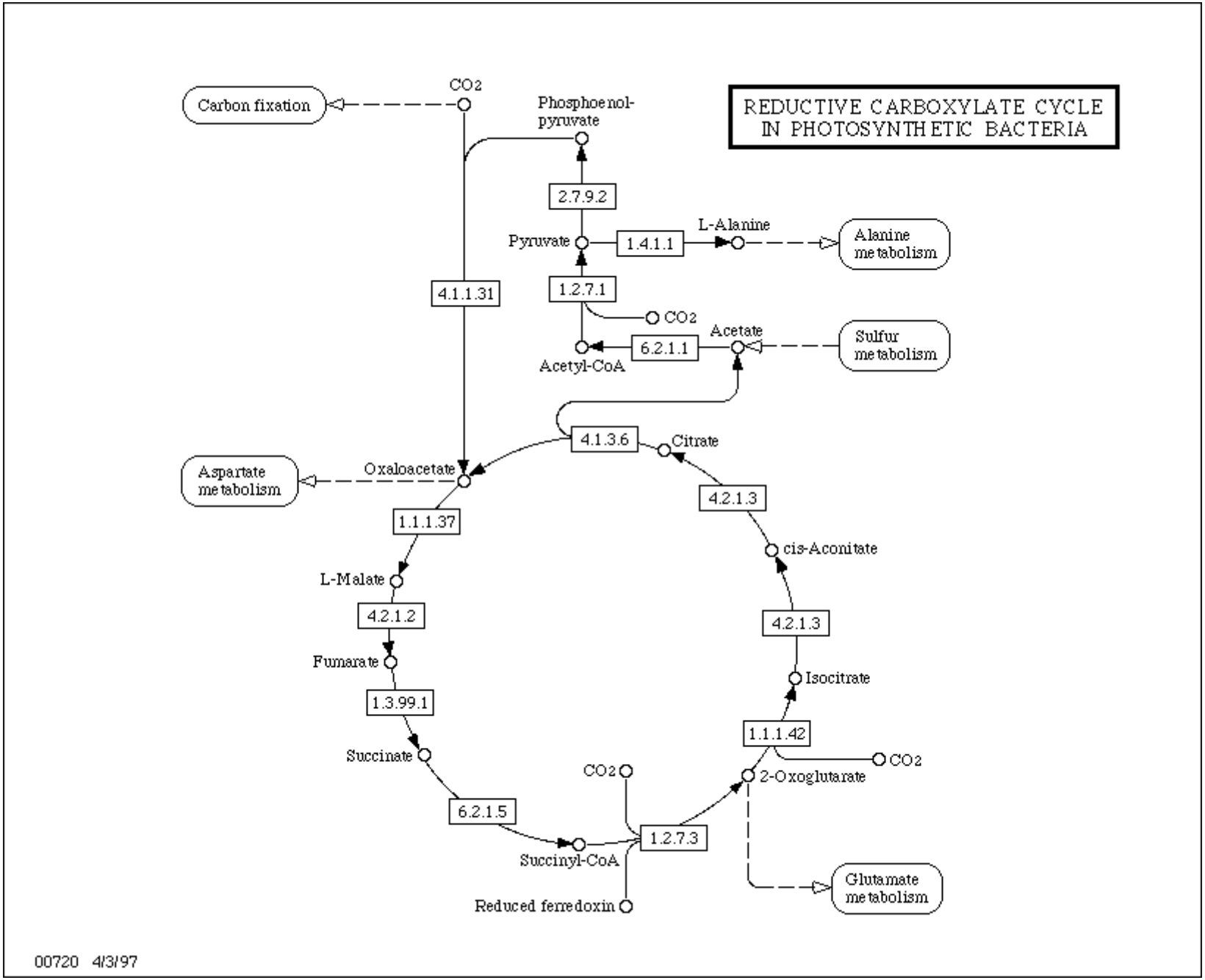


(www.techedlab.com)

Voie métabolique avec cycles



Extrait de la base KEGG: Kyoto Encyclopedia of Genes and Genomes


REDUCTIVE CARBOXYLATE CYCLE
IN PHOSPHOSYNTHETIC BACTERIA
Carbon fixation
CO2
Phosphoenol-
pyruvate
2.7.9.2
Pyruvate
1.4.1.1
L-Alanine
Alanine
metabolism
4.1.1.31
1.2.7.1
CO2
Acetate
6.2.1.1
Sulfur
metabolism
Acetyl-CoA
4.1.3.6
Citrate
Aspartate
metabolism
Oxaloacetate
4.2.1.3
1.1.1.37
cis-Aconitate
L-Malate
4.2.1.2
4.2.1.3
Fumarate
Isocitrate
1.3.99.1
1.1.1.42
Succinate
CO2
CO2
2-Oxoglutarate
6.2.1.5
1.2.7.3
Succinyl-CoA
Glutamate
metabolism
Reduced ferredoxin
00720 4/3/97

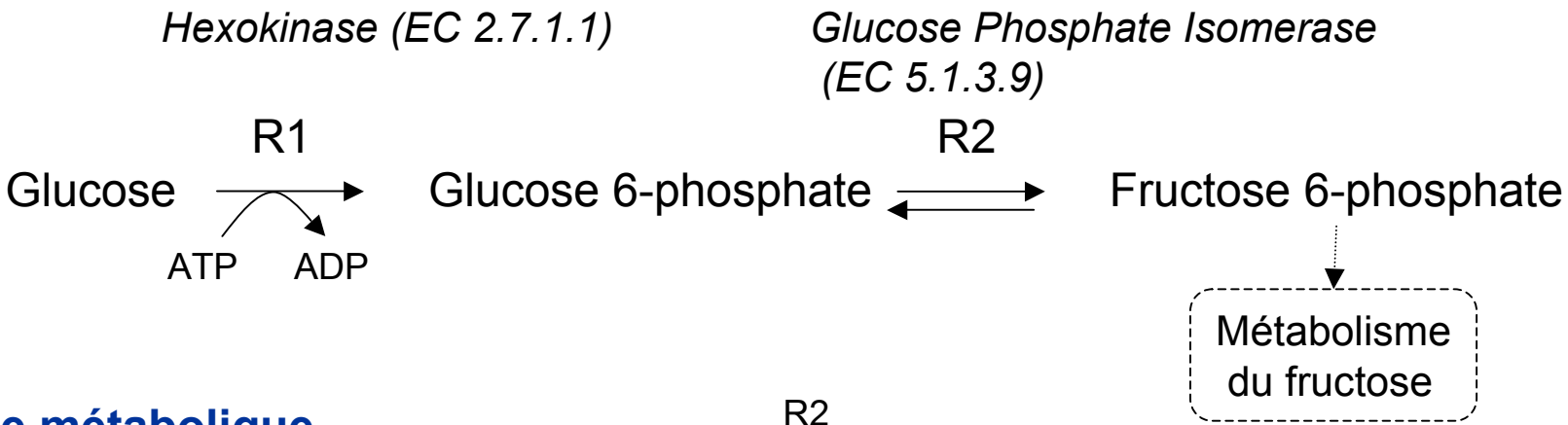
Représentation graphique d'un *pathway*

Etat de l'art:

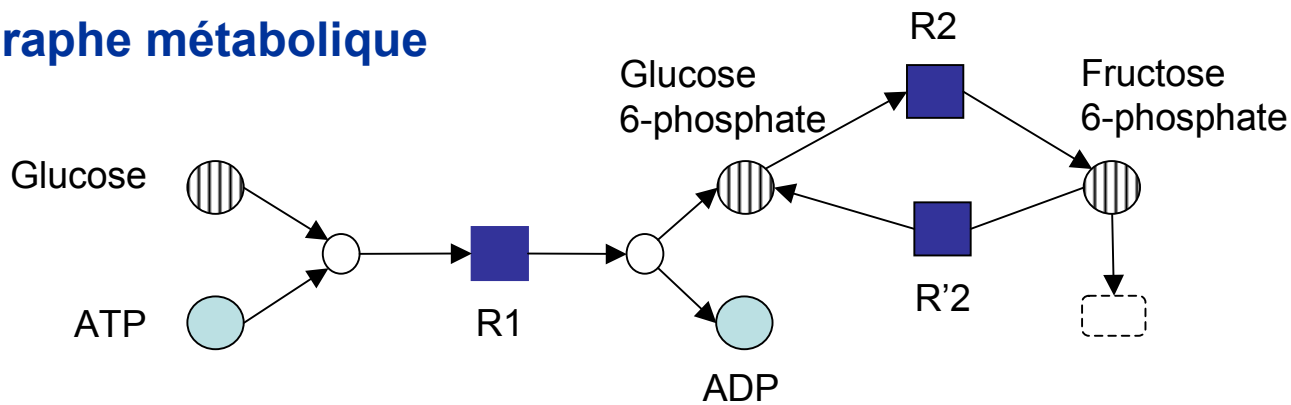
Logiciels	Description
BioPath biopath.fmi.uni-passau.de/	Logiciel développé par l'université de Passau. Implémentation en C^{++} et Tcl/Tk.
BioCyc ecocyc.org:1555/server.html	Site internet développé par SRI International. Propose un ensemble d'utilitaires de visualisation de pathways.
PathVis http://www.eml.villa-bosch.de/english/Research/sdbv/1	Logiciel développé par European Media Laboratory (EML). PathVis est un prototype développé en Java.
PathDB www.ncgr.org/pathdb	Logiciel développé par le NCGR (National Center for Genome Research). Une version prototype est disponible pour les laboratoires académiques.
PathwayBrowser www.yworks.de.en	Développé en Java. Commercialisé par la société yWorks.

Abstraction d'une voie métabolique à un graphe

Voie métabolique



Graphe métabolique



« nœud composé primaire »

« nœud composé secondaire »

« nœud réaction »

« nœud d'addition »

« nœud de liaison »

Choix d'implémentation du graphe

- Librairies Java : OpenJgraph, GEF, JGraph

- Choix: JGraph
  - ☺ Gratuit
  - ☺ Documentation : +++

  - ☹ *aucune méthode de placement automatique*

Développement d'un logiciel de visualisation de voies métaboliques

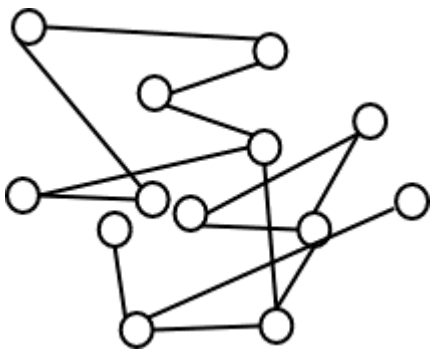
Cahier des charges :

- **Modélisation d'une voie métabolique**
- **Affichage**
- **Composant graphique Java**

Spécifications : affichage du graphe

- **Placement automatique**
- Pas de chevauchement des noeuds
- Distances voisines entre les noeuds
- Motif cyclique représenté par un cercle

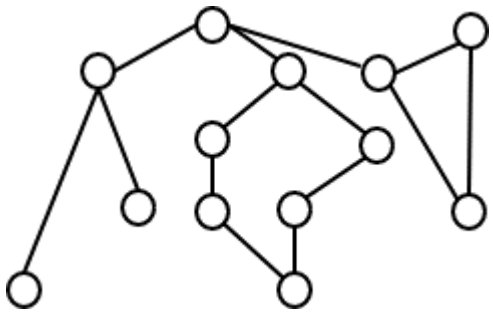
Algorithmes de placement

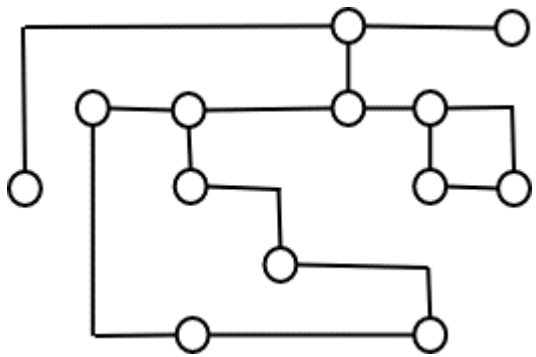


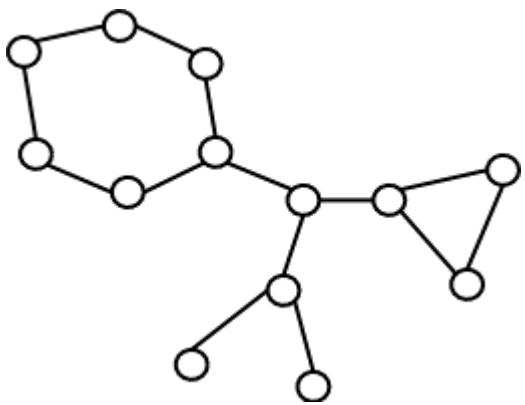
hiérarchique

orthogonal

par forces







Algorithme de placement par forces

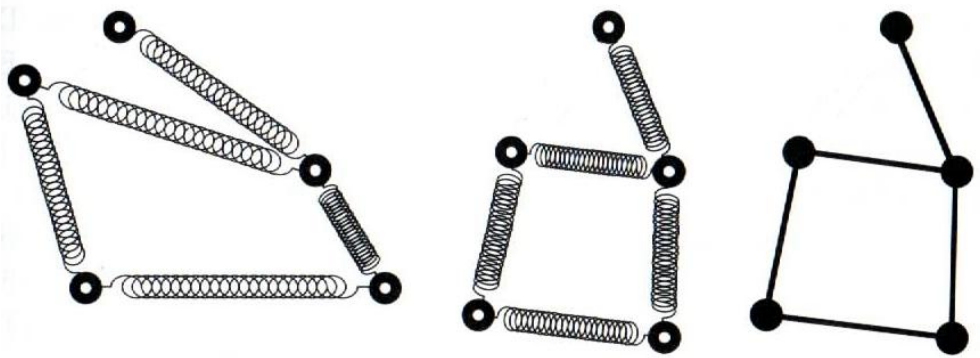
- Nœuds du graphe: particules
- Application de forces aux particules
- Recherche de la configuration de plus faible énergie

Méthode des ressorts et des charges électriques

- Nœuds : particules (charge, masse)
- Arêtes : ressorts amortis

Intérêts

- Distances voisines
- Pas de chevauchement



État initial | Equilibre | Représentation graphique

Modèle physique de la méthode des ressorts et des charges électriques

Fonctionnement général

- 2 systèmes physiques
  - ❑Système de masses-ressort
  - ❑Système d'interaction électromagnétique

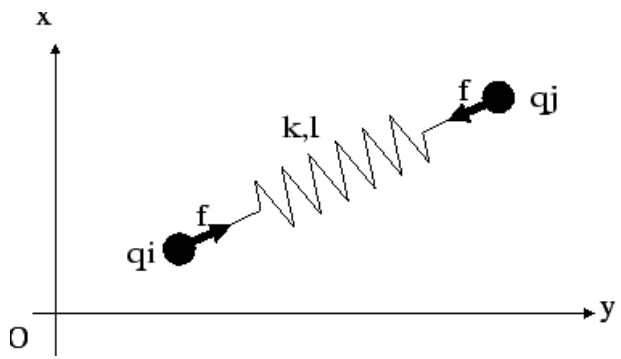
- Calcul des forces s'appliquant aux particules

- Calcul des nouvelles positions des particules

Modèle physique de la méthode des ressorts et des charges électriques

Le système masses-ressort

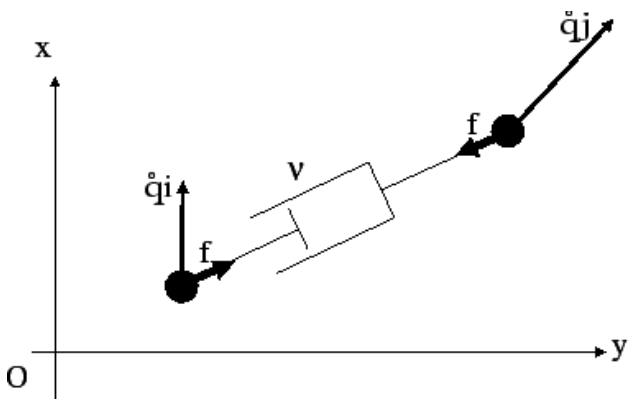
- Le ressort a un comportement élastique.



$$f_{j\to i}^{(k)} = k\left(\frac{\|q_i - q_j\| - l}{\|q_i - q_j\|}\right)\frac{q_i - q_j}{\|q_i - q_j\|}$$

k: constante de raideur
l: longueur au repos

- L'amortisseur réduit le mouvement relatif des particules.



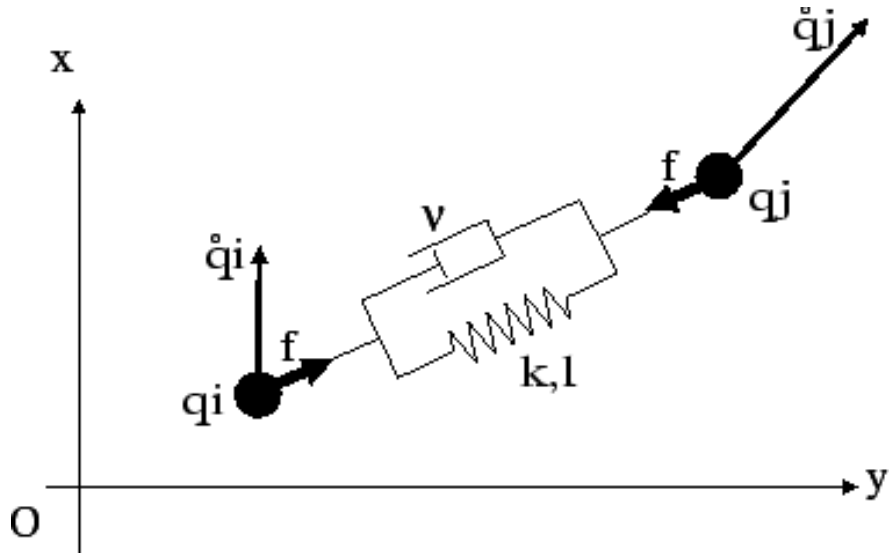
$$f_{j\to i}^{(\nu)} = \nu\left(\frac{\dot{q}_i - \dot{q}_j}{\|q_i - q_j\|}\frac{q_i - q_j}{\|q_i - q_j\|}\right)\frac{q_i - q_j}{\|q_i - q_j\|}$$

v: constante de viscosité

Modèle physique de la méthode des ressorts et des charges électriques

Le système masses-ressort

• Le ressort amorti combine les effets d'un ressort et d'un amortisseur.

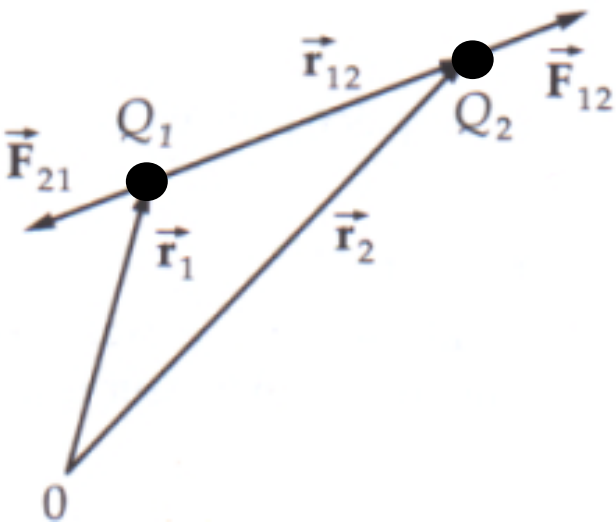


$$f_{j\to i} = f^{(k)}_{j\to i} + f^{(\nu)}_{j\to i}$$

Modèle physique de la méthode des ressorts et des charges électriques

Le système d'interaction électromagnétique

L'interaction électromagnétique est la force responsable de l'attraction et de la répulsion de particules chargées.



$$\vec{\mathbf{F}} = \frac{1}{4\pi\varepsilon_0}\frac{Q_1 Q_2}{r^2}\frac{\vec{\mathbf{r}}_{12}}{r_{12}}$$

Modèle physique de la méthode des ressorts et des charges électriques

Calcul des positions des particules

1 – Calcul de l'accélération

$$\ddot{q} = f / m$$

$\ddot{q}$: accélération de la particule
f : force s'appliquant à la particule
m : la masse de la particule

2 – Calcul de la vitesse

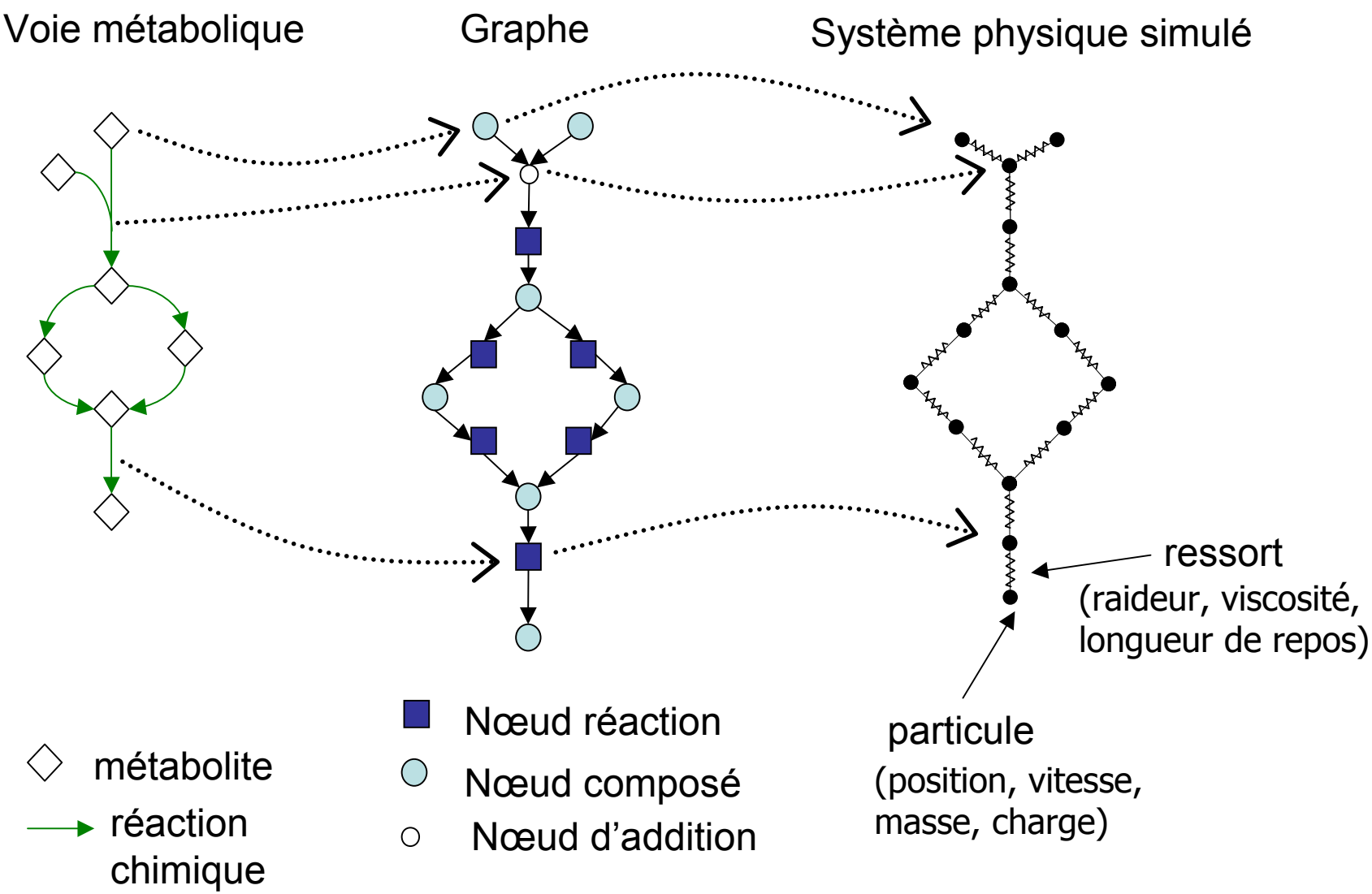
$$\dot{q}(t+h) = \dot{q}(t) + \ddot{q}(t) * h$$

$\dot{q}$: vitesse de la particule
h : longueur du pas de temps

3 – Calcul des positions

$$q(t+h) = q(t) + \dot{q}(t) * h$$

Modélisation

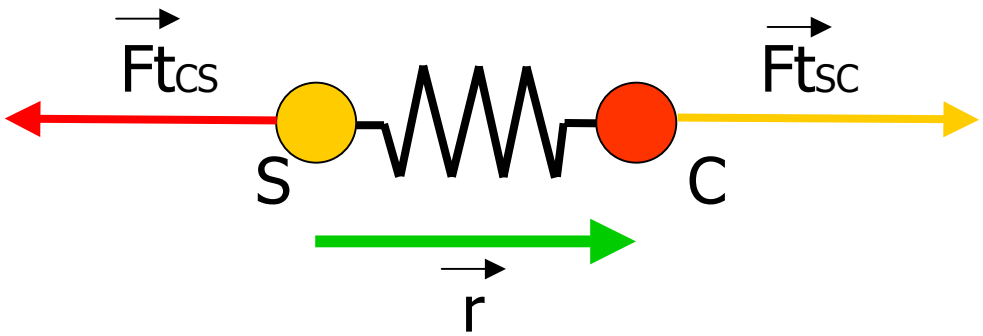


Description générale de l'algorithme

```
Procédure LAYOUT
Paramètres : Graphe : g ;
Variables : SystèmePhysique : s,
            Vecteur(taille 2) : force, ft, fi,
            TableauDeVecteurs(taille 2) : V ;
// créerSystèmePhysique crée le système physique correspondant
// au graphe donné en paramètre
s ← créerSystèmePhysique(g) ;
// creation et initialisation du tableau contenant les vitesses
V ← nouveau TableauDeVecteurs(taille 2)(nombreDeNoeuds(s)) ;
// boucle principale

tant que nonEquilibre(s) faire
    pour chaque n_i ∈ noeuds(s) faire
        // initialisation des forces
        force ← (0,0) ;
        fi ← (0,0) ;
        ft ← (0,0) ;

        pour chaque n_j ∈ noeuds(s) tel que n_j relié à n_i par un
        ressort faire
            ft ← ft + calculForceTension(n_i,n_j) ;

        pour chaque n_j ∈ noeuds(s) tel que n_j ≠ n_i faire
            fi ← fi + calculForceRépulsion(n_i,n_j) ;

        // calcul de la force résultante
        force ← ft + fi ;
        // mise à jour de la vitesse du noeud n_i
        V[i] ← V[i] + calculVitesseNoeud(force,n_i) ;

    pour chaque n_i ∈ noeuds(s) faire
        miseAJourCoordonnées(n_i,V[i]) ;

AppliquerCoordonnéesDuSytèmePhysique(g,s) ;
fin
```

Calcul de la force de tension

- Vecteur distance entre un noeud source et un noeud cible



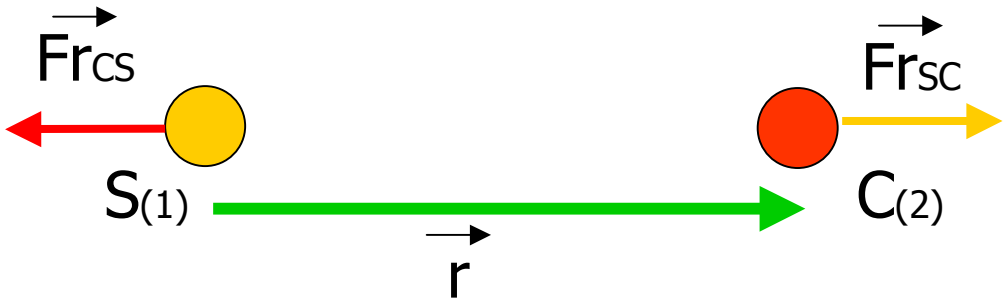
- Calcul des vecteurs de force de tension
(somme de la force de tension et de la force d'amortissement)

Description générale de l'algorithme

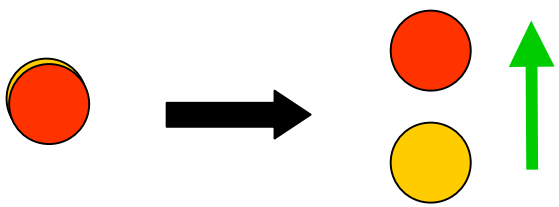
```
Procédure Layout
Paramètres : Graphe : g ;
Variables : SystèmePhysique : s,
            Vecteur(taille 2) : force, ft, fi,
            TableauDeVecteurs(taille 2) : V ;
début
    // créerSystèmePhysique crée le système physique correspondant
    // au graphe donné en paramètre
    s ← créerSystèmePhysique(g) ;
    // creation et initialisation du tableau contenant les vitesses
    V ← nouveau TableauDeVecteurs(taille 2)(nombreDeNoeuds(s)) ;
    // boucle principale

    tant que nonEquilibre(s) faire
        pour chaque n_i ∈ noeuds(s) faire
            // initialisation des forces
            force ← (0,0) ;
            fi ← (0,0) ;
            ft ← (0,0) ;

            pour chaque n_j ∈ noeuds(s) tel que n_j relié à n_i par un
            ressort faire
                ft ← ft + calculForceTension(n_i,n_j) ;

            pour chaque n_j ∈ noeuds(s) tel que n_j ≠ n_i faire
                fi ← fi + calculForceRépulsion(n_i,n_j) ;

            // calcul de la force résultante
            force ← ft + fi ;
            // mise à jour de la vitesse du noeud n_i
            V[i] ← V[i] + calculVitesseNoeud(force,n_i) ;

        pour chaque n_i ∈ noeuds(s) faire
            miseAJourCoordonnées(n_i,V[i]) ;

    AppliquerCoordonnéesDuSytèmePhysique(g,s) ;
fin
```

Calcul de la force de répulsion

- Vecteur distance entre un noeud source et un noeud cible



Si $\vec{r} = \vec{0}$, on déplace le noeud source.



- Calcul des vecteurs de force de répulsion

Description générale de l'algorithme

```
Procédure Layout
Paramètres : Graphe : g ;
Variables : SystèmePhysique : s,
            Vecteur(taille 2) : force, ft, fi,
            TableauDeVecteurs(taille 2) : V ;
début
    // créerSystèmePhysique crée le système physique correspondant
    // au graphe donné en paramètre
    s ← créerSystèmePhysique(g) ;
    // creation et initialisation du tableau contenant les vitesses
    V ← nouveau TableauDeVecteurs(taille 2)(nombreDeNoeuds(s)) ;
    // boucle principale

    tant que nonEquilibre(s) faire
        pour chaque n_i ∈ noeuds(s) faire
            // initialisation des forces
            force ← (0,0) ;
            fi ← (0,0) ;
            ft ← (0,0) ;

            pour chaque n_j ∈ noeuds(s) tel que n_j relié à n_i par un
            ressort faire
                ft ← ft + calculForceTension(n_i,n_j) ;

            pour chaque n_j ∈ noeuds(s) tel que n_j ≠ n_i faire
                fi ← fi + calculForceRépulsion(n_i,n_j) ;

            // calcul de la force résultante
            force ← ft + fi ;
            // mise à jour de la vitesse du noeud n_i
            V[i] ← V[i] + calculVitesseNoeud(force,n_i) ;
        pour chaque n_i ∈ noeuds(s) faire
            miseAJourCoordonnées(n_i,V[i]) ;

    AppliquerCoordonnéesDuSytèmePhysique(g,s) ;
fin
```

Description générale de l'algorithme

```
Procédure Layout
Paramètres : Graphe : g ;
Variables : SystèmePhysique : s,
            Vecteur(taille 2) : force, ft, fi,
            TableauDeVecteurs(taille 2) : V ;
début
    // créerSystèmePhysique crée le système physique correspondant
    // au graphe donné en paramètre
    s ← créerSystèmePhysique(g) ;
    // creation et initialisation du tableau contenant les vitesses
    V ← nouveau TableauDeVecteurs(taille 2)(nombreDeNoeuds(s)) ;
    // boucle principale

    tant que nonEquilibre(s) faire
        pour chaque n_i ∈ noeuds(s) faire
            // initialisation des forces
            force ← (0,0) ;
            fi ← (0,0) ;
            ft ← (0,0) ;

            pour chaque n_j ∈ noeuds(s) tel que n_j relié à n_i par un
            ressort faire
                ft ← ft + calculForceTension(n_i,n_j) ;

            pour chaque n_j ∈ noeuds(s) tel que n_j ≠ n_i faire
                fi ← fi + calculForceRépulsion(n_i,n_j) ;

            // calcul de la force résultante
            force ← ft + fi ;
            // mise à jour de la vitesse du noeud n_i
            V[i] ← V[i] + calculVitesseNoeud(force,n_i) ;
        pour chaque n_i ∈ noeuds(s) faire
            miseAJourCoordonnées(n_i,V[i]) ;

    AppliquerCoordonnéesDuSytèmePhysique(g,s) ;
fin
```

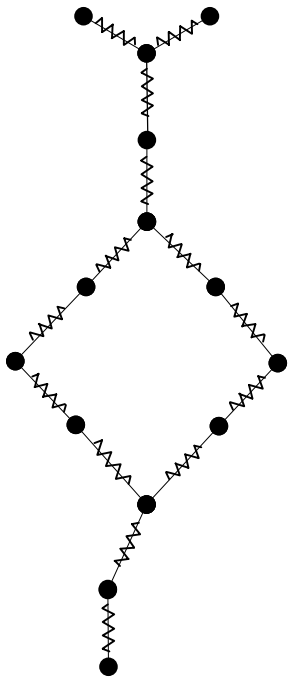
Description générale de l'algorithme

```
Procédure Layout
Paramètres : Graphe : g ;
Variables : SystèmePhysique : s,
            Vecteur(taille 2) : force, ft, fi,
            TableauDeVecteurs(taille 2) : V ;
// créerSystèmePhysique crée le système physique correspondant
// au graphe donné en paramètre
s ← créerSystèmePhysique(g) ;
// creation et initialisation du tableau contenant les vitesses
V ← nouveau TableauDeVecteurs(taille 2)(nombreDeNoeuds(s)) ;
// boucle principale

tant que nonEquilibre(s) faire
    pour chaque n_i ∈ noeuds(s) faire
        // initialisation des forces
        force ← (0,0) ;
        fi ← (0,0) ;
        ft ← (0,0) ;

        pour chaque n_j ∈ noeuds(s) tel que n_j relié à n_i par un
        ressort faire
            ft ← ft + calculForceTension(n_i,n_j) ;

        pour chaque n_j ∈ noeuds(s) tel que n_j ≠ n_i faire
            fi ← fi + calculForceRépulsion(n_i,n_j) ;

        // calcul de la force résultante
        force ← ft + fi ;
        // mise à jour de la vitesse du noeud n_i
        V[i] ← V[i] + calculVitesseNoeud(force,n_i) ;
    pour chaque n_i ∈ noeuds(s) faire
        miseAJourCoordonnées(n_i,V[i]) ;

AppliquerCoordonnéesDuSytèmePhysique(g,s) ;
fin
```

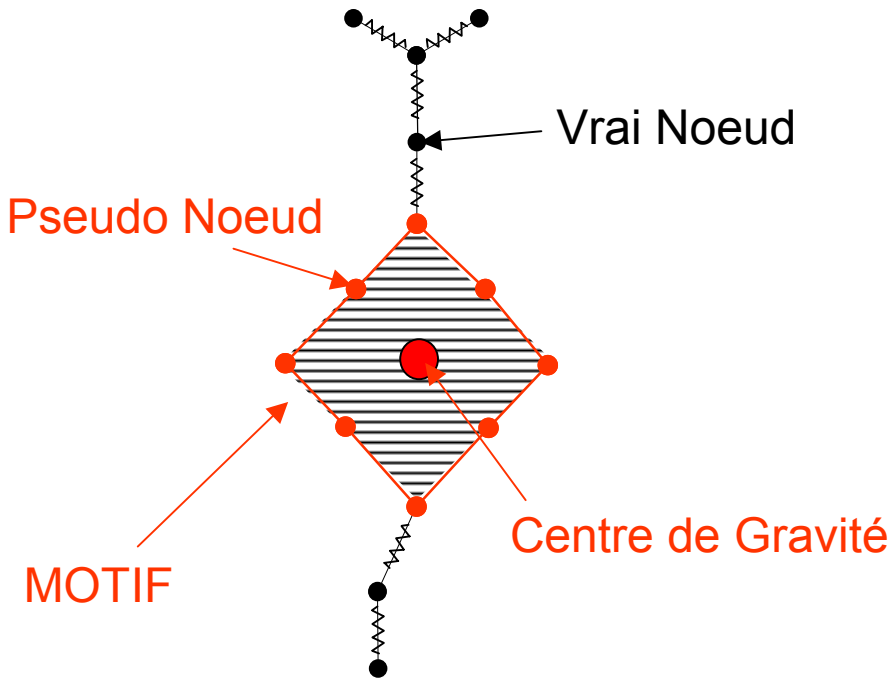
Extension: prise en compte de motif

- Motif : ensemble de noeuds fixes les uns par rapport aux autres

Système physique simple



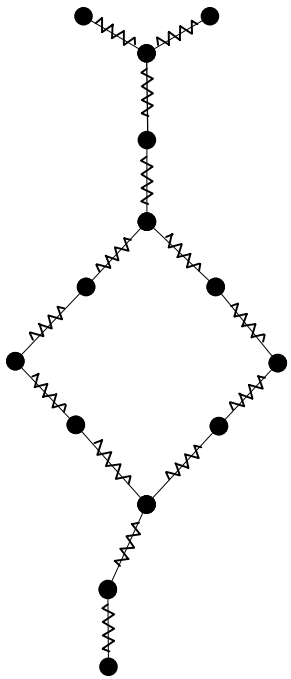
Système physique avec motif



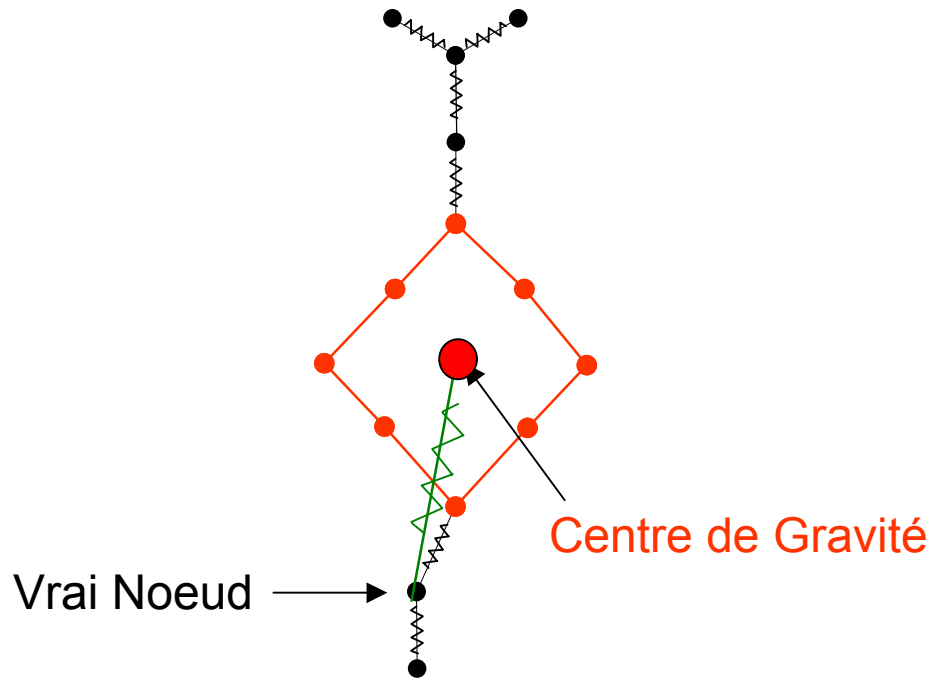
Intégration de la notion de motif

- Motif : ensemble de noeuds fixes les uns par rapport aux autres

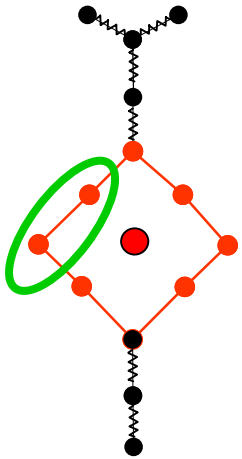
Système physique simple



Système physique avec motif

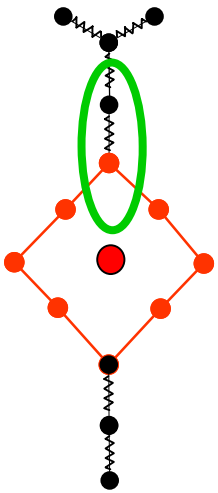


Calcul de la force de répulsion (motif)



Entre 2 pseudo noeuds

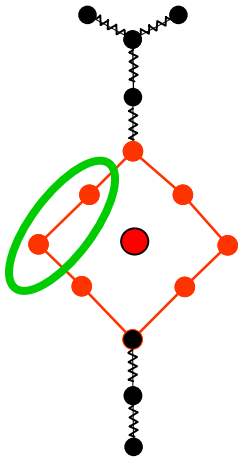
Aucune force d'interaction ne s'applique



Entre 1 vrai noeud et 1 pseudo noeud

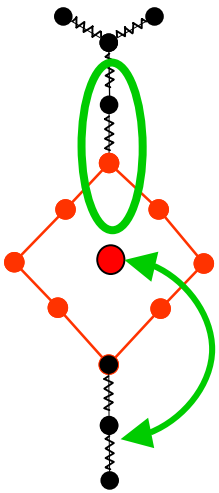
Calcul identique à celui décrit pour 2 vrais noeuds

Calcul de la force de tension (motif)



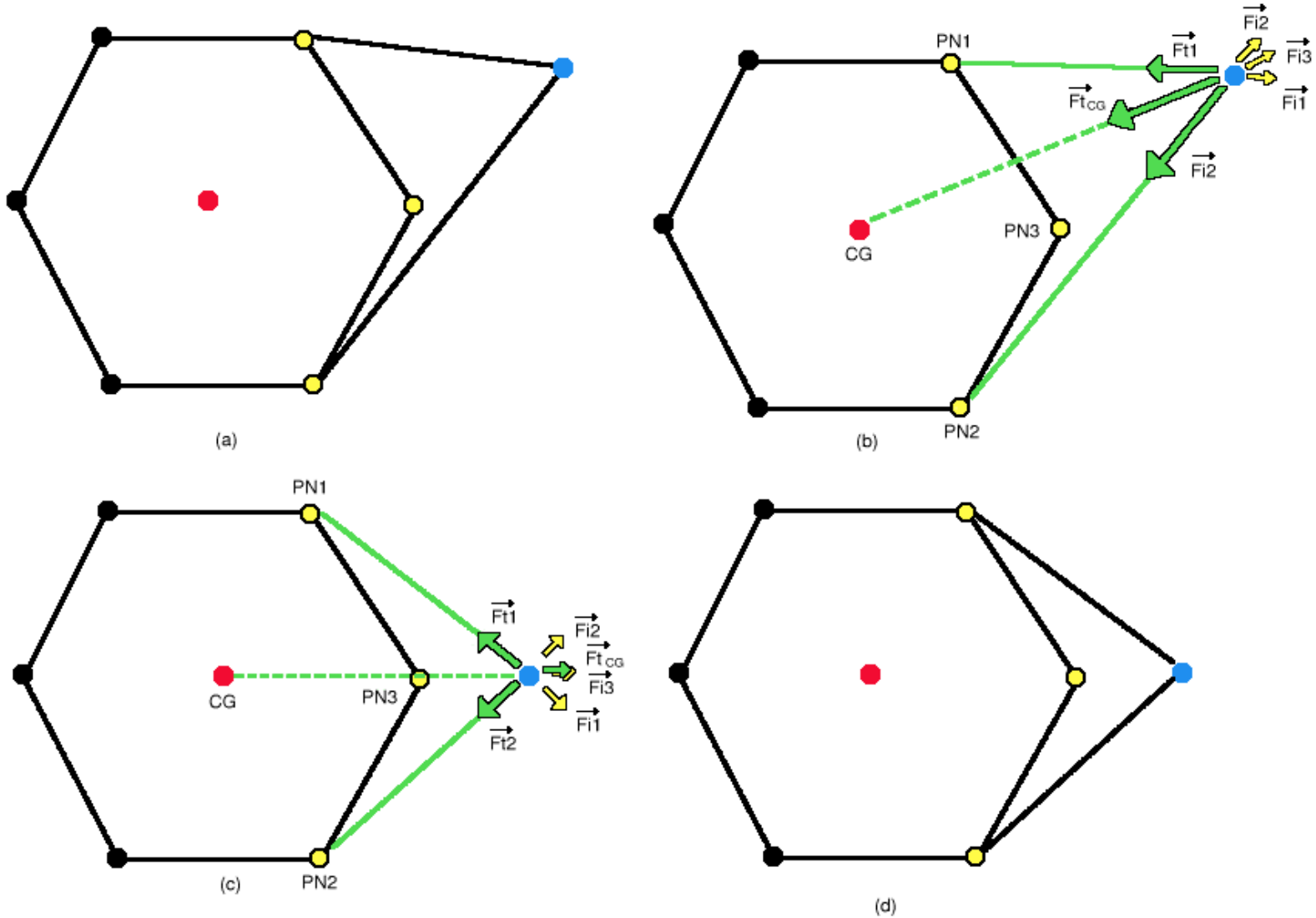
Entre 2 pseudo noeuds

Aucune force de tension ne s'applique



Vrai Noeud / Pseudo Noeud :
ressort entre une masse légère (VN) et une masse infinie (PN)

Vrai Noeud / Centre de Gravité :
ressort entre le vrai noeud et le centre du motif


(a)
PN1
Ft1
Fi2
Fi3
Fi1
Ftcg
Fi2
CG
PN3
PN2
(b)
PN1
Ft1
Fi2
Ftcg
Fi3
Fi1
CG
PN3
Ft2
PN2
(c)
(d)
pseudo noeud
vrai noeud
pseudo noeud lié à un vrai noeud
centre de gravité du motif
arête
ressort
ressort (arête fictive)
force de tension
force d'interaction électromagnétique

Déplacement des nœuds

- Déplacement des vrais noeuds

$$q_x(t+h) = q_x(t) + \dot{q}_x(t)*h$$
$$q_y(t+h) = q_y(t) + \dot{q}_y(t)*h$$

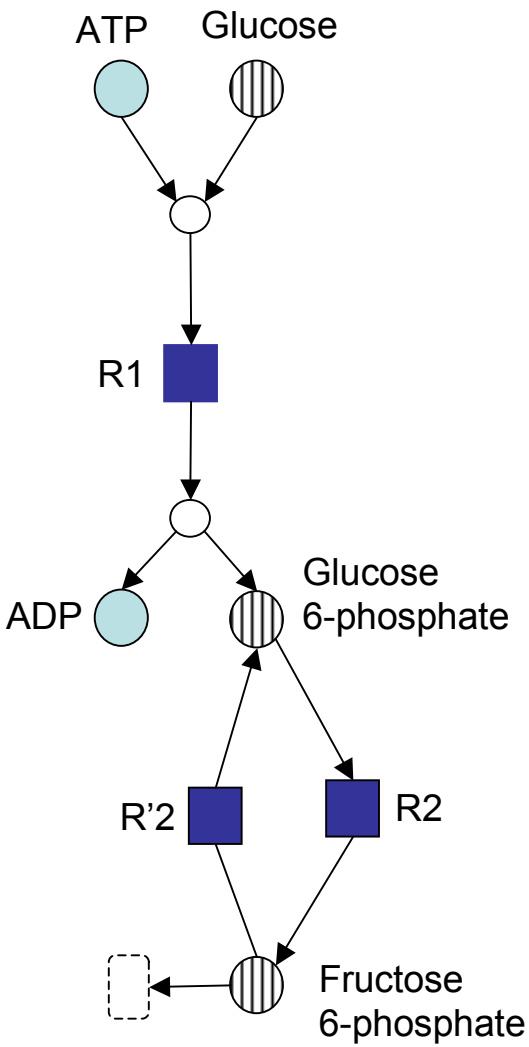
Avec: q_x, l'abscisse du nœud
q_y, l'ordonnée du nœud
$\dot{q}_x$, la vitesse du nœud selon l'axe des abscisses
$\dot{q}_y$, la vitesse du nœud selon l'axe des ordonnées
h, longueur du pas de temps

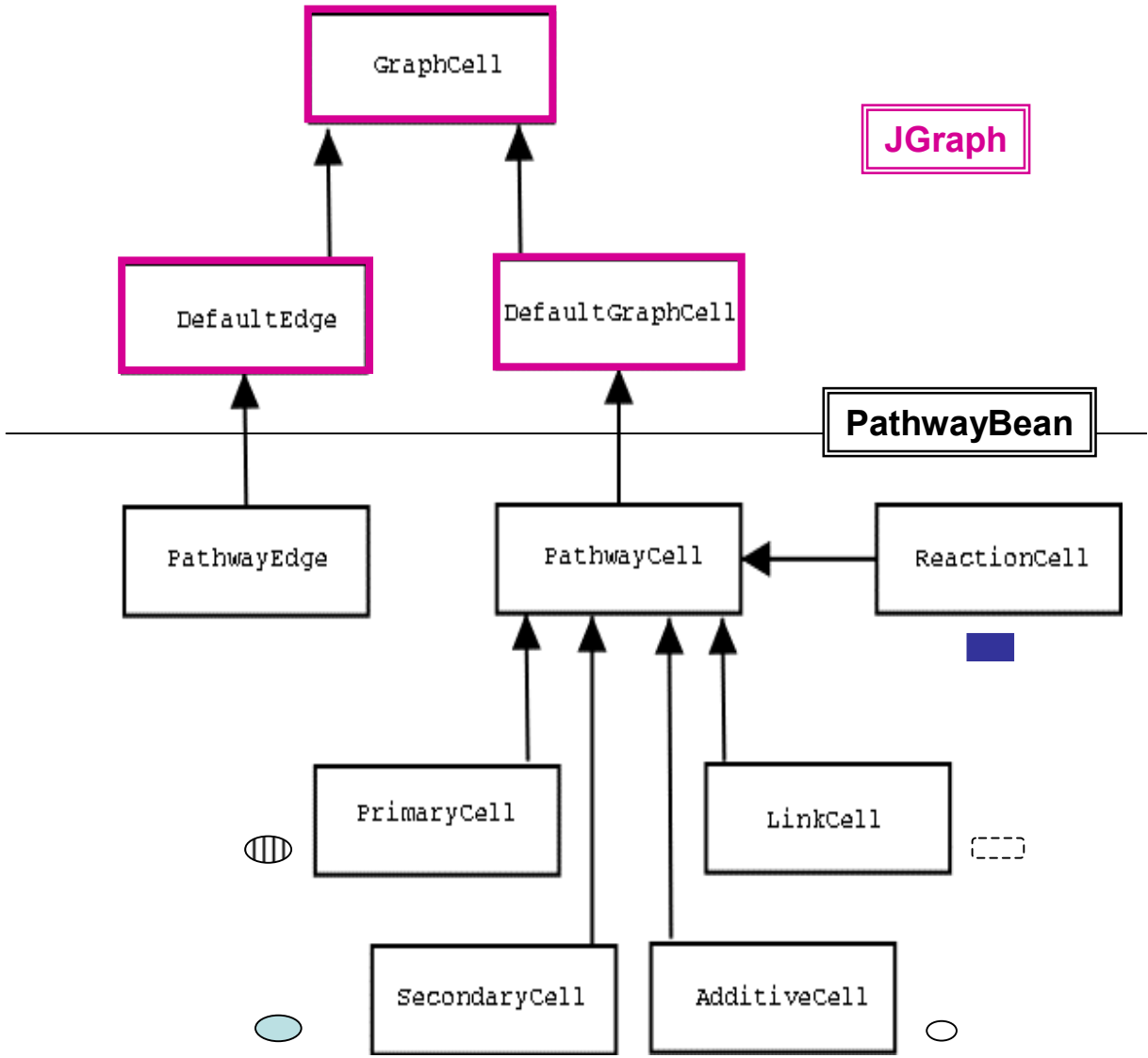
- Déplacement du motif

La position de chaque noeud du motif est calculée en fonction de la position du centre de gravité du motif.

Implémentation

- Composant graphique sous forme de JavaBean
- Extension de la librairie JGraph


ATP
Glucose
R1
ADP
Glucose
6-phosphate
R'2
R2
Fructose
6-phosphate


GraphCell
JGraph
DefaultEdge
DefaultGraphCell
PathwayBean
PathwayEdge
PathwayCell
ReactionCell
PrimaryCell
LinkCell
SecondaryCell
AdditiveCell

Implémentation

- Composant graphique sous forme de JavaBean
- Extension de la librairie JGraph
- Implémentation de l’algorithme de placement automatique

Application: PathwayBean

Entrée = liste de réactions

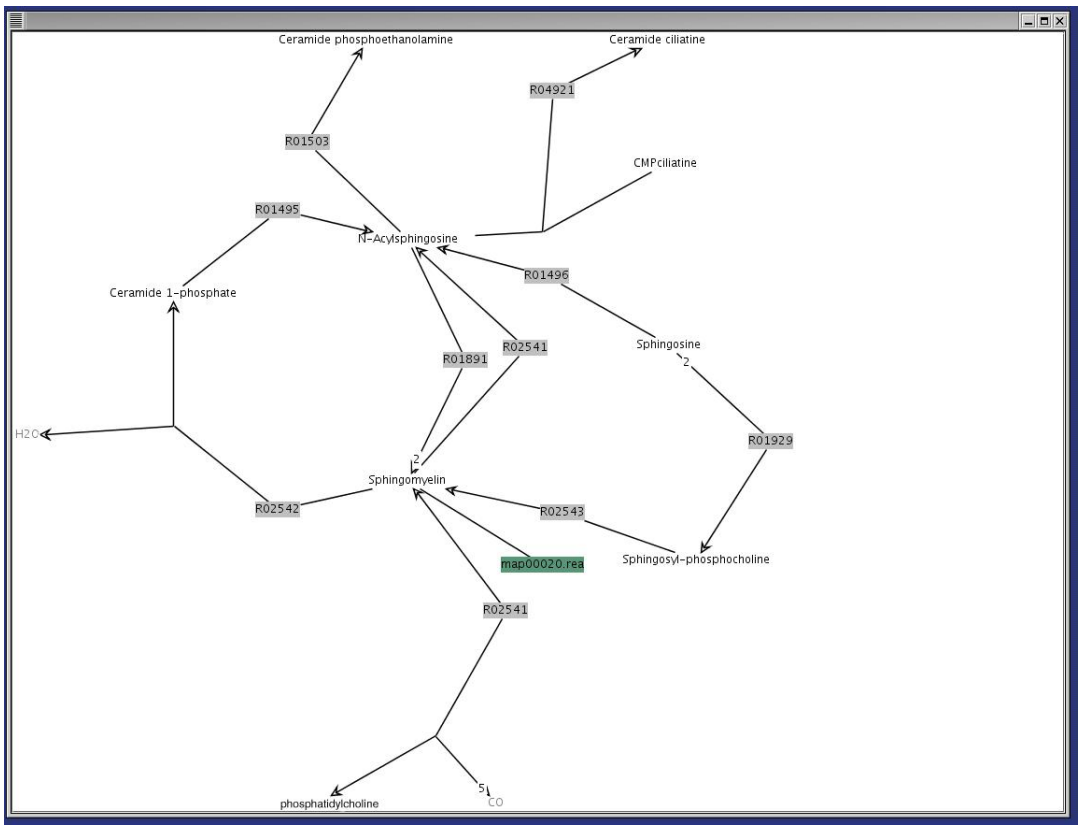
R1: Ceramide-1-phosphate => N-Acylsphingosine

R2: Sphingomyeline => Ceramide-1-phosphate + H20

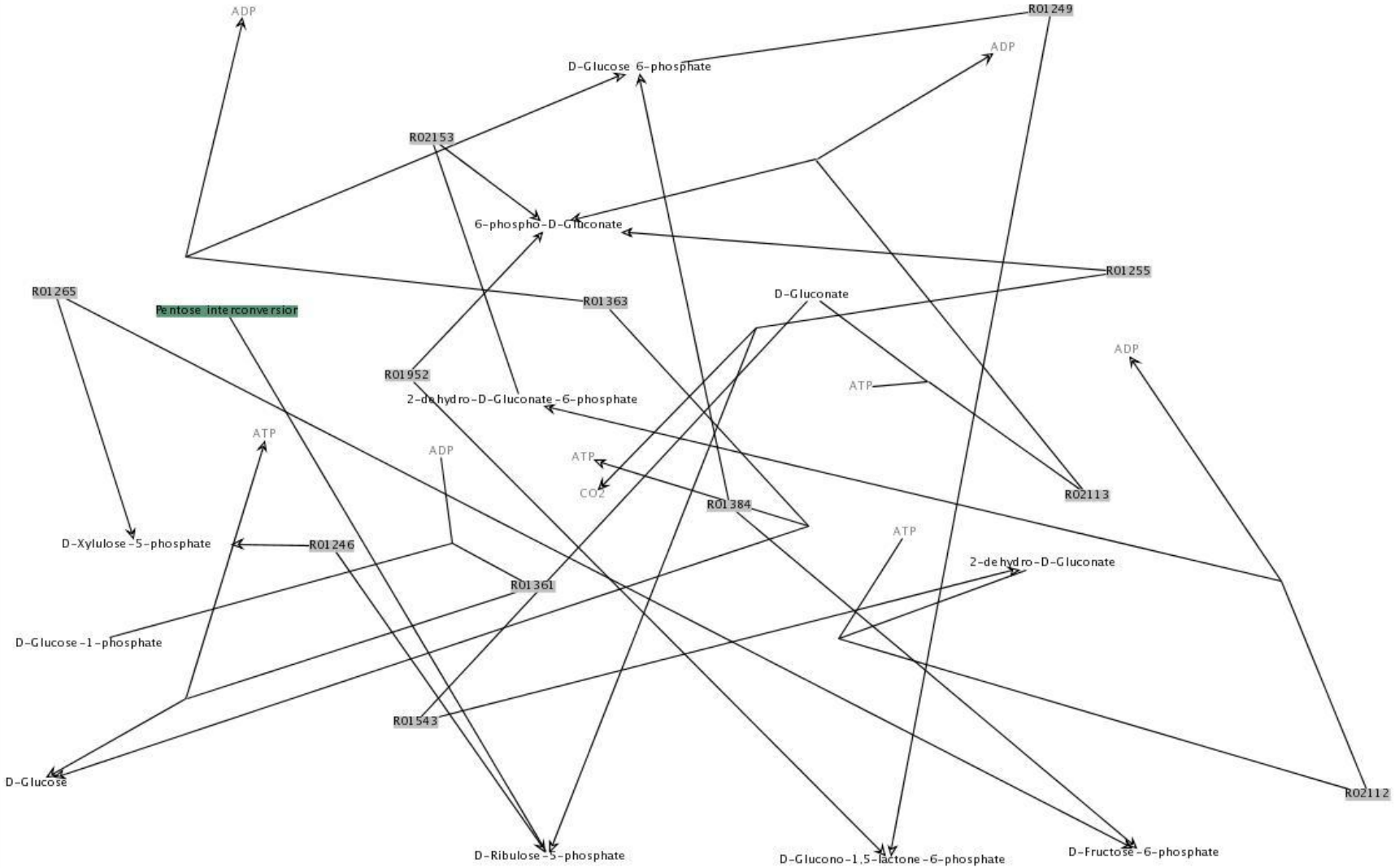
R3: N-Acylsphingosine => Sphingomyeline

R4: N-Acylsphingosine => Ceramide ciliatine + CMP-Ciliatine

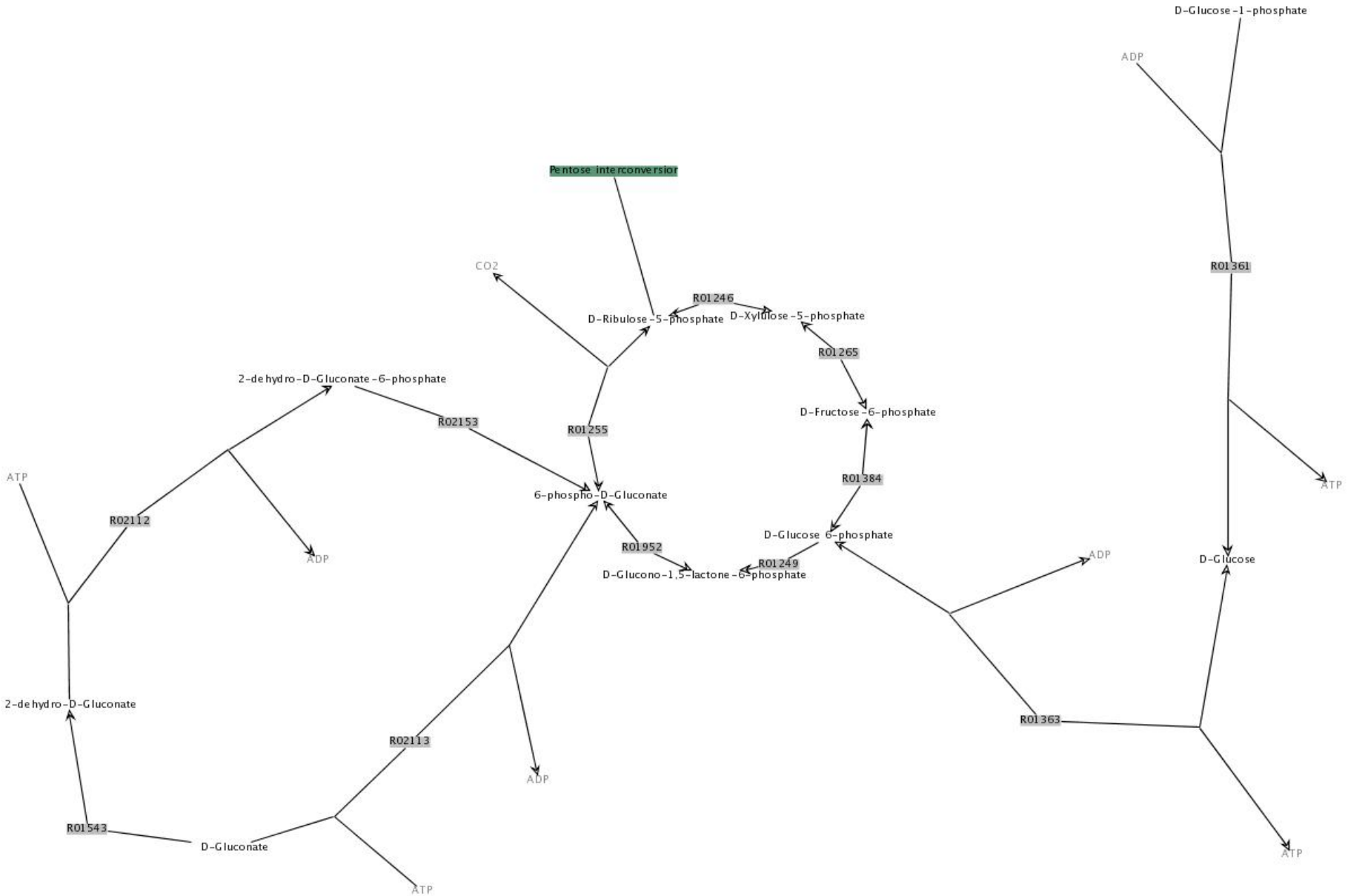
Sortie = représentation graphique



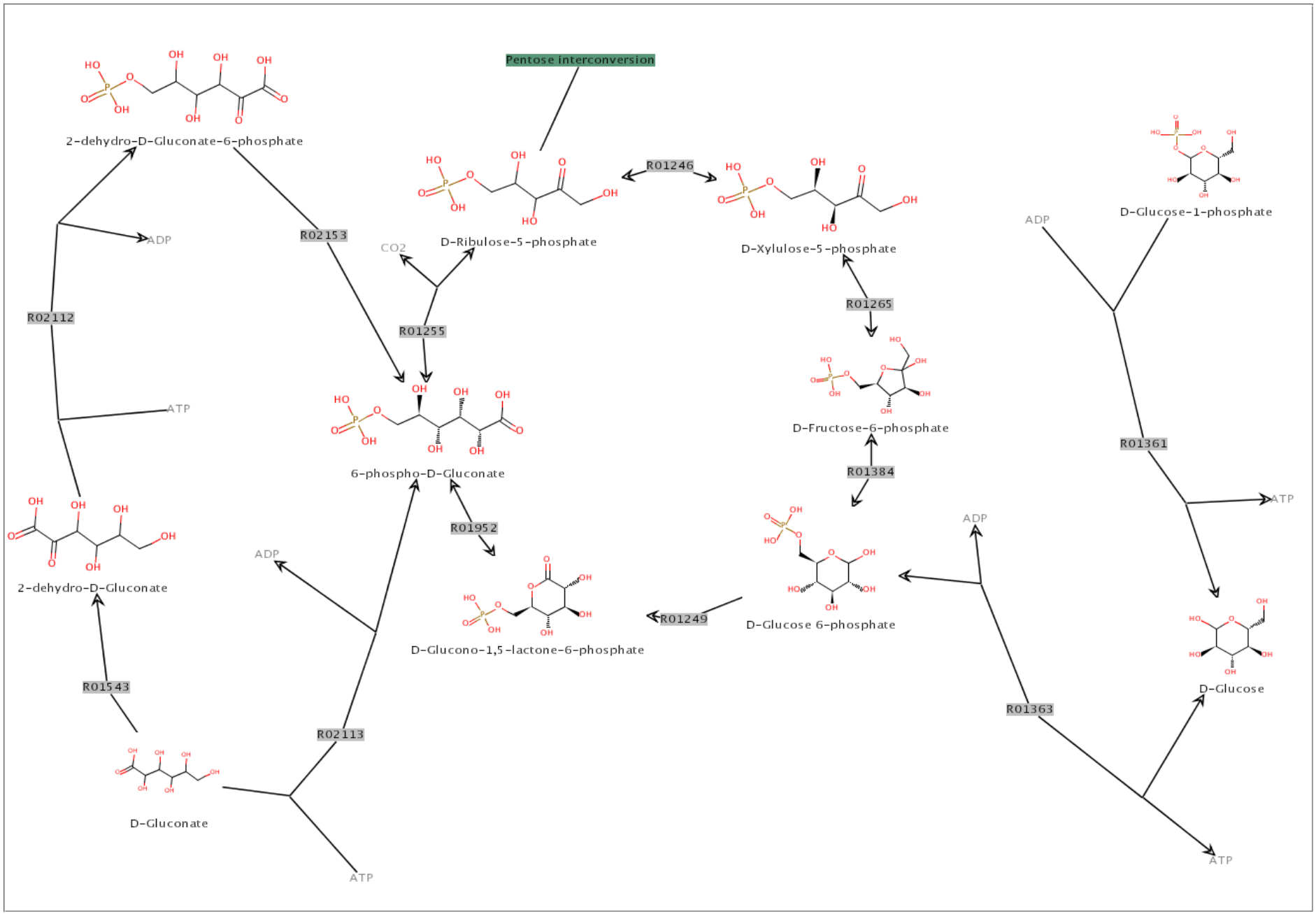
AVANT


ADP
R01249
ADP
D-Glucose 6-phosphate
R02153
6-phospho-D-Gluconate
R01255
R01265
R01363
D-Gluconate
Pentose interconversior
ADP
R01952
2-dehydro-D-Gluconate-6-phosphate
ATP
ATP
ADP
ATP
CO2
R01384
R02113
D-Xylulose-5-phosphate
R01246
ATP
2-dehydro-D-Gluconate
R01361
D-Glucose-1-phosphate
R01543
D-Glucose
R02112
D-Ribulose-5-phosphate
D-Glucono-1,5-lactone-6-phosphate
D-Fructose-6-phosphate

APRES


D-Glucose-1-phosphate
ADP
Pentose interconversion
CO2
R01361
R01246
D-Ribulose-5-phosphate
D-Xylulose-5-phosphate
R01265
2-dehydro-D-Gluconate-6-phosphate
D-Fructose-6-phosphate
R02153
R01255
R01384
ATP
6-phospho-D-Gluconate
ATP
R02112
D-Glucose 6-phosphate
R01952
ADP
R01249
D-Glucono-1,5-lactone-6-phosphate
ADP
D-Glucose
2-dehydro-D-Gluconate
R01363
R02113
ADP
R01543
D-Gluconate
ATP
ATP

Représentation de la formule développée des composés primaires (MolBean)

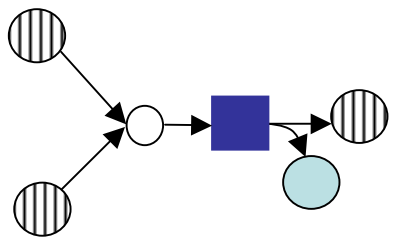


Conclusion

- Travail sur toutes les phases de développement de PathwayBean
  - ❑ Réflexion sur le projet
  - ❑ Conception et modélisation
  - ❑ Implémentation

Perspectives de développement

- Respect des conventions de représentation
- Appliquer une viscosité générale
- Appliquer un mouvement brownien général



Bilan du stage

- Méthodes de travail, maîtrise des outils utilisés dans l'équipe Helix (Ant, CVS, XML...)
- Travail en équipe
- Travail pluridisciplinaire (biochimie, physique, algorithmique, développement logiciel)

Des questions ?