Optimization methods in Metabolic Engineering

Miguel Rocha

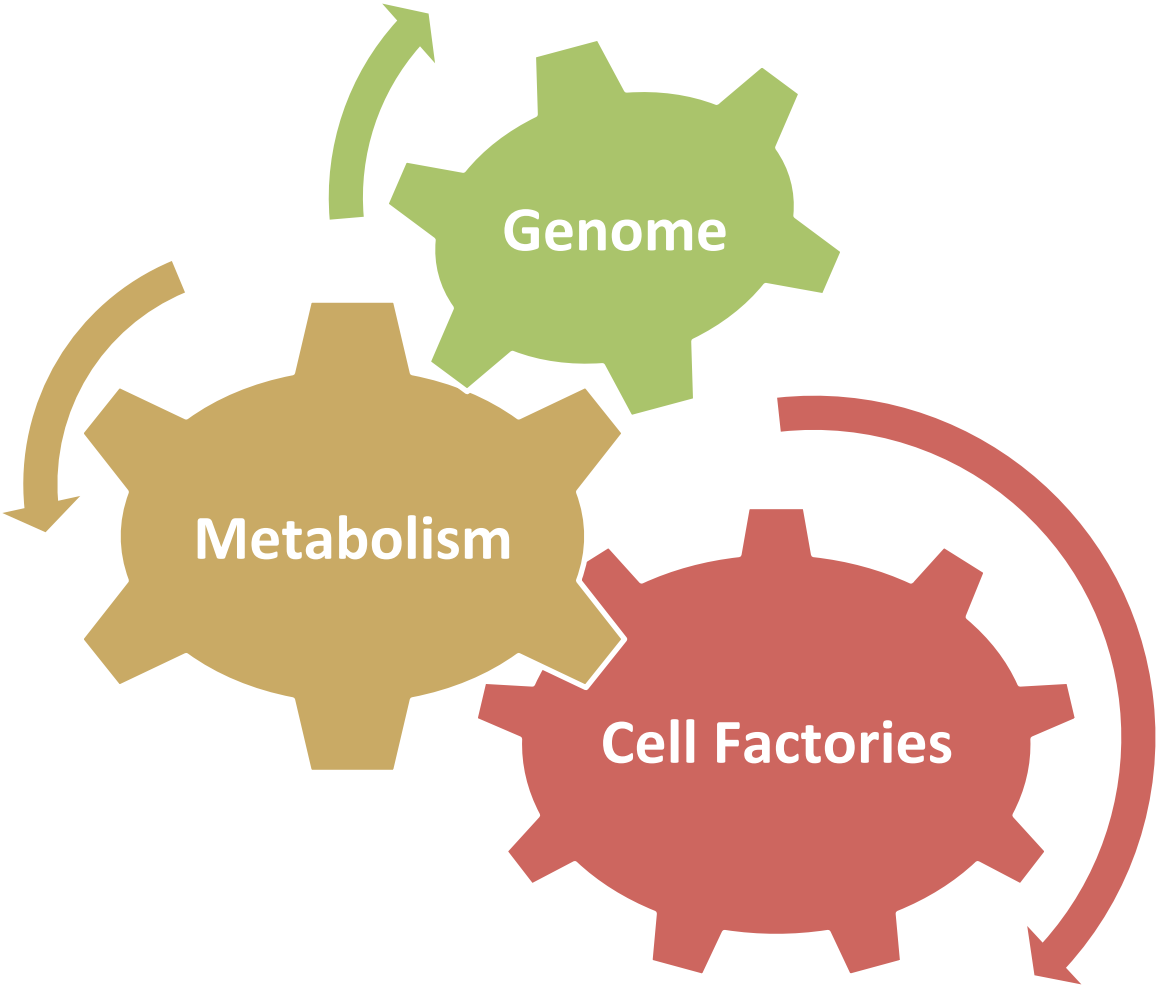
Universidade do Minho, Portugal

Metabolic Engineering

To produce **desired compounds** (e.g. antibiotics, fuels, vitamins) from **microbial cell factories** it is generally necessary to **retrofit the metabolism**

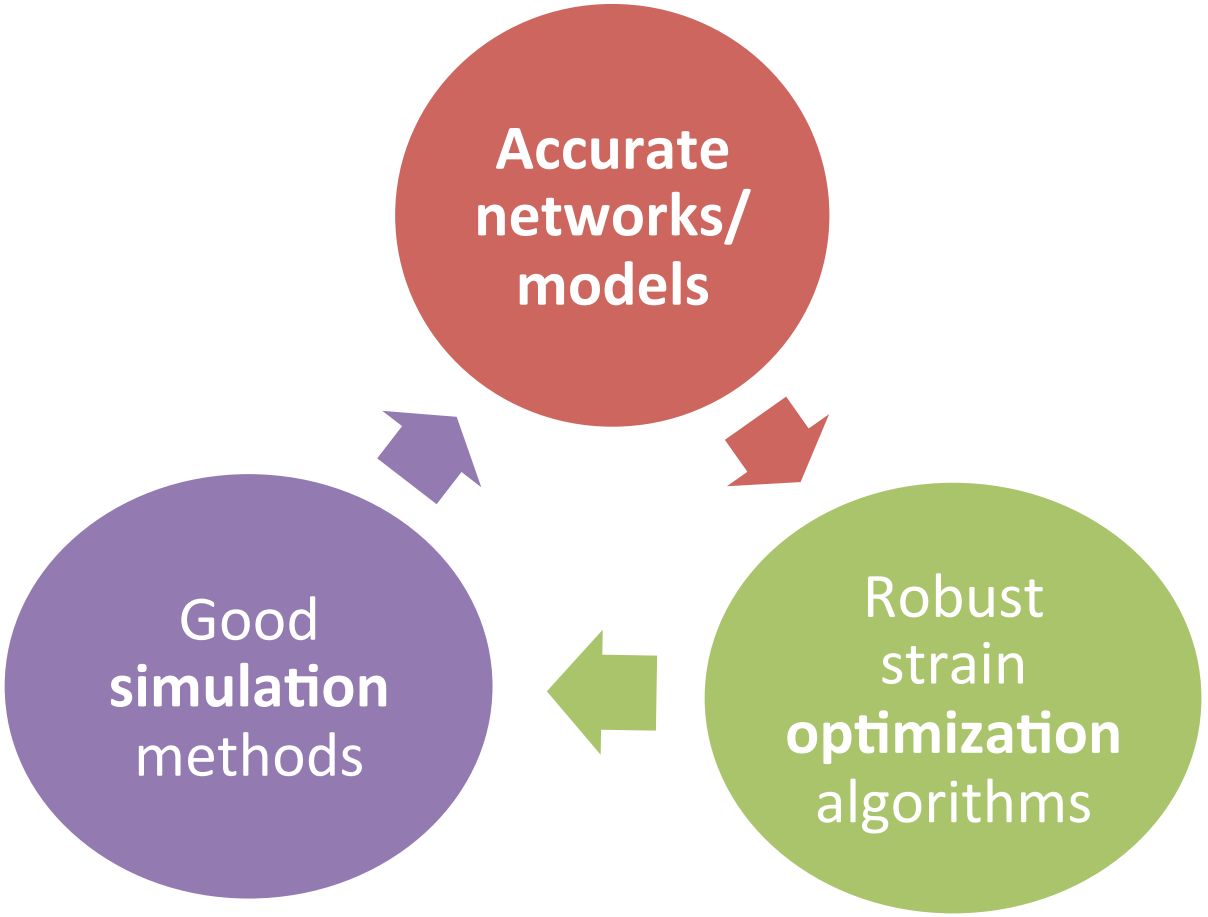
Metabolic Engineering envisages the introduction of **directed genetic modifications** leading to desirable phenotypes, as opposed to traditional methods

Metabolic Engineering



Metabolic Engineering

It is often difficult to identify which genetic manipulations will originate a given desired phenotype



Strain optimization problems

Possible aims

- Select appropriate reaction (gene) deletions
- Select reactions (genes) to over/under express
- Select set of reactions (from another organism) to add to a metabolic model (of a given organism)

Objective functions for the mutant strains

- Maximizing the production of a given compound
- Keeping the organism viable
- Minimizing number of changes
- etc …

Strain optimization

Optimization methods

- Depending on the type of problem, a number of methods have been proposed
- These depend on the representation chosen for the metabolic systems and on the optimization task

Complexity

- In all cases, the solution space is very large
- Problems are NP – hard
- In some cases, a solution is the use of metaheuristic optimization methods

One approach: bilevel optimization

Aim: select the appropriate genetic changes to enable the production of a desired product

Bi-level optimization problem:

Maximize (compound) – bioengineering objective

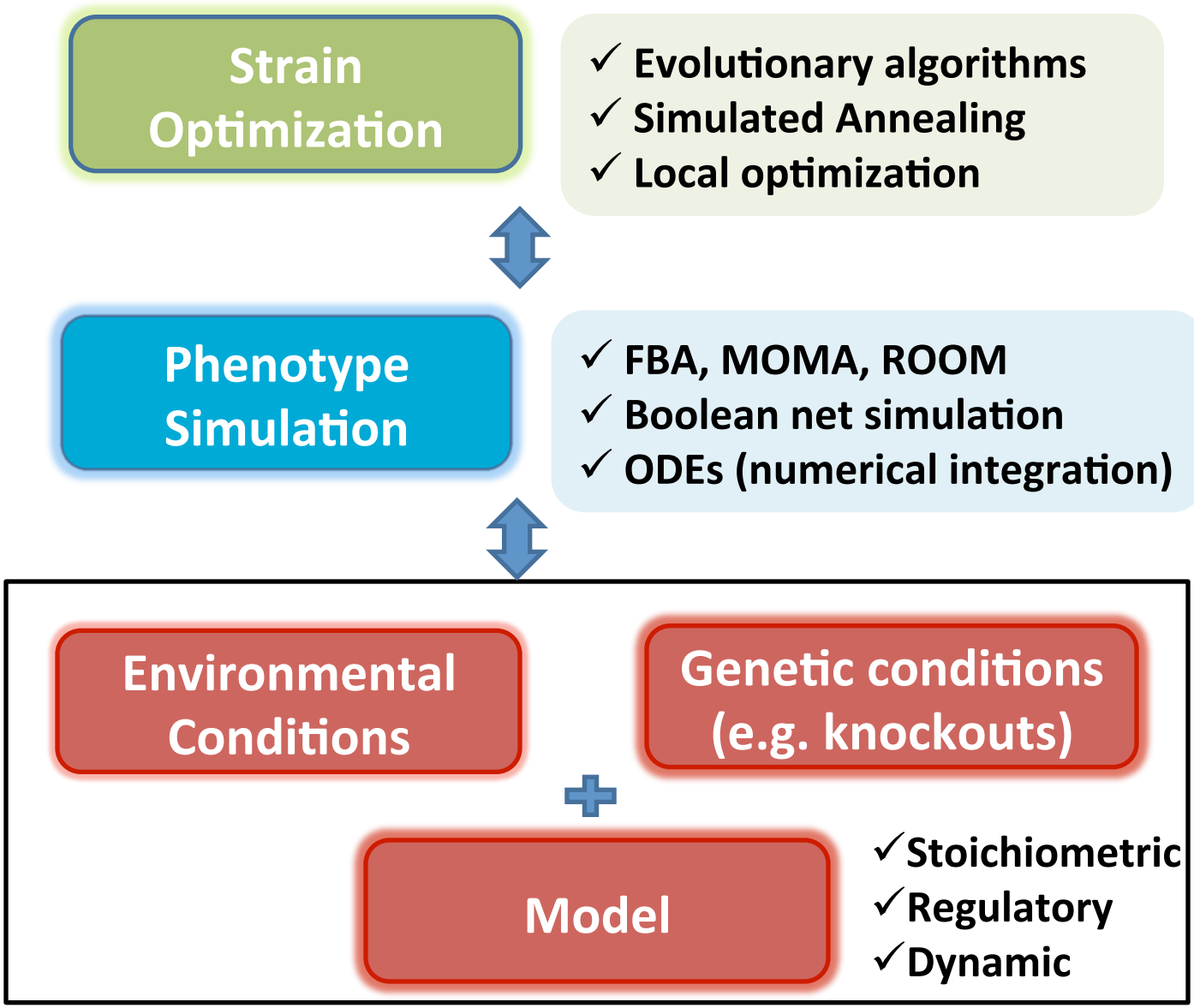
↳ Candidate solution evaluation

Maximize (biomass) – cellular objective

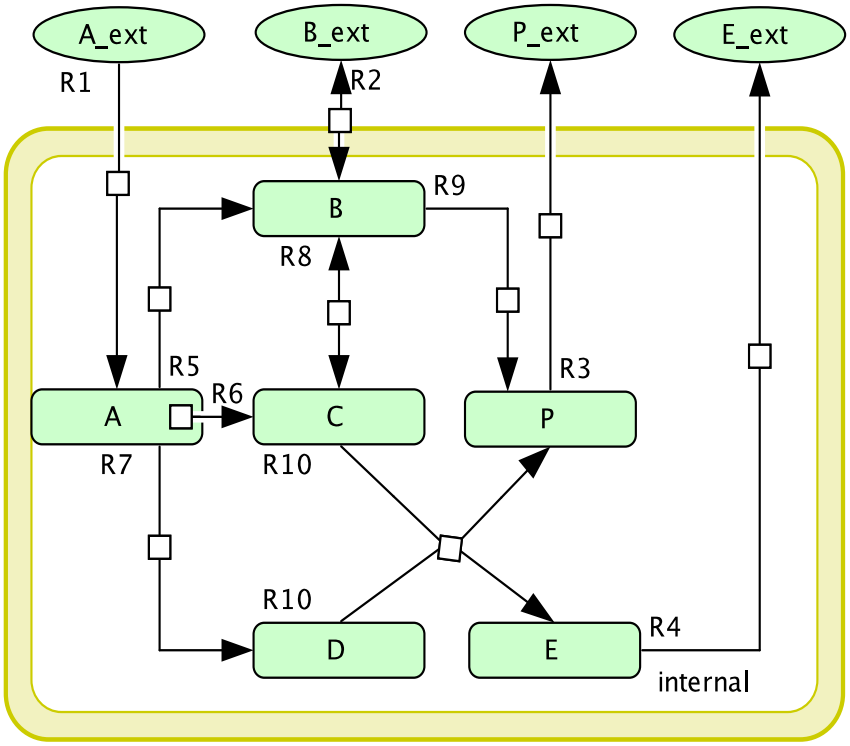
constraints:
- steady state
- reversibility
(…)

SIMULATION

Conceptual overview



Metabolic models: example



Simplified metabolic system

$$N = \begin{bmatrix} 1 & 0 & 0 & 0 & -1 & -1 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & -1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \begin{matrix} \leftarrow A \\ \leftarrow B \\ \leftarrow C \\ \leftarrow D \\ \leftarrow E \\ \leftarrow P \end{matrix}$$

$$\begin{matrix} \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow \\ R1 & R2 & R3 & R4 & R5 & R6 & R7 & R8 & R9 & R10 \end{matrix}$$

Stoichiometric matrix

Steady state simulation approaches

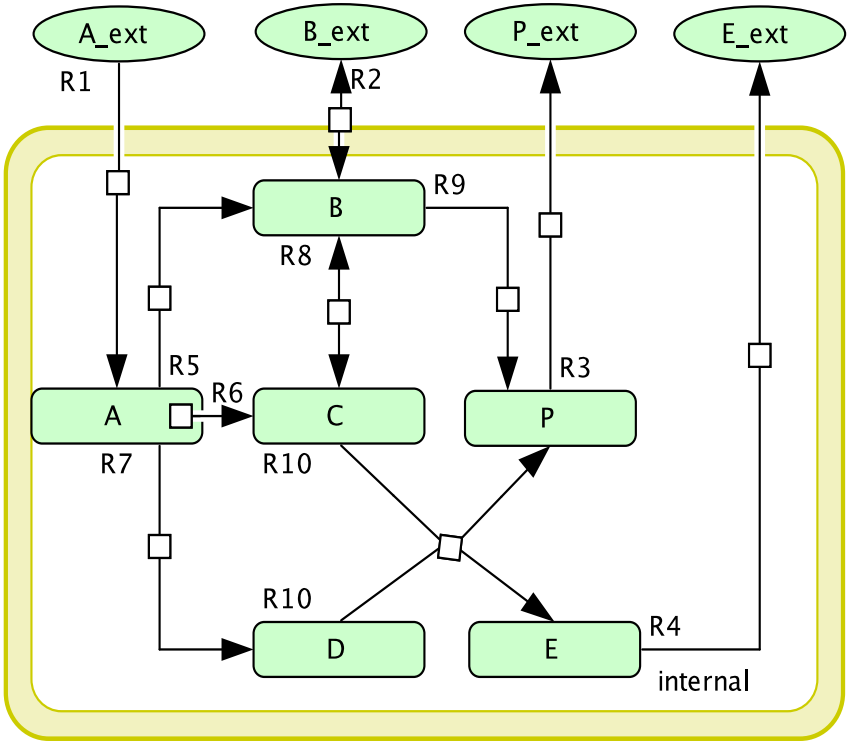
Calculating intracellular reaction fluxes based on:

- Mass balance over intra-cellular metabolites
- Assumption of pseudo-steady state
- Sum of all fluxes from reactions that consume the metabolite equals the sum of all fluxes for reaction that produce it (no net accumulation)

Constraint based approach

- Result: linear equation system described by stoichiometry
- Constraints added for reaction reversibility
- Resulting equation system typically undetermined

Steady state simulation: example



N=

1	0	0	0	-1	-1	-1	0	0	0	← A
0	1	0	0	1	0	0	-1	-1	0	← B
0	0	0	0	0	1	0	1	0	-1	← C
0	0	0	0	0	0	1	0	0	-1	← D
0	0	0	-1	0	0	0	0	0	1	← E
0	0	-1	0	0	0	0	0	1	1	← P
↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	
R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	

Steady State Constraints

A: R1 – R5 – R6 – R7 = 0
B: R2 + R5 – R8 – R9 = 0
C: R6 + R8 – R10 = 0
D: R7 – R10 = 0
E: - R4 + R10 = 0
P: -R3 + R9 + R10 = 0

Reversibility Constraints

R1 >= 0 R6 >= 0
R3 >= 0 R7 >= 0
R4 >= 0 R9 >= 0
R5 >= 0 R10 >= 0

Undetermined system

Steady state simulation

Optimization problem

- The system is undetermined - solution is to transform into an **optimization problem**
- **Flux Balance Analysis**: assumes organisms have evolved "perfectly" to maximize their growth
- **Objective function**: maximize biomass flux – artificial flux determined experimentally including all biomass precursors
- Linear OF; linear constraints – **Linear Programming** problem
- Easy to solve (e.g. simplex algorithm)
- Other methods assume distinct objective functions and constraints

Knockout optimization
(using steady state simulation)

Combinatorial optimization

- Aim is to find the best of reactions/ genes to remove from the original strain to redirect fluxes to desired products
- Solutions can be represented as subsets of elements (genes, reactions) taken from a large set
- Knockouts are represented as extra constraints in the simulation (flux of reaction constrained to 0)

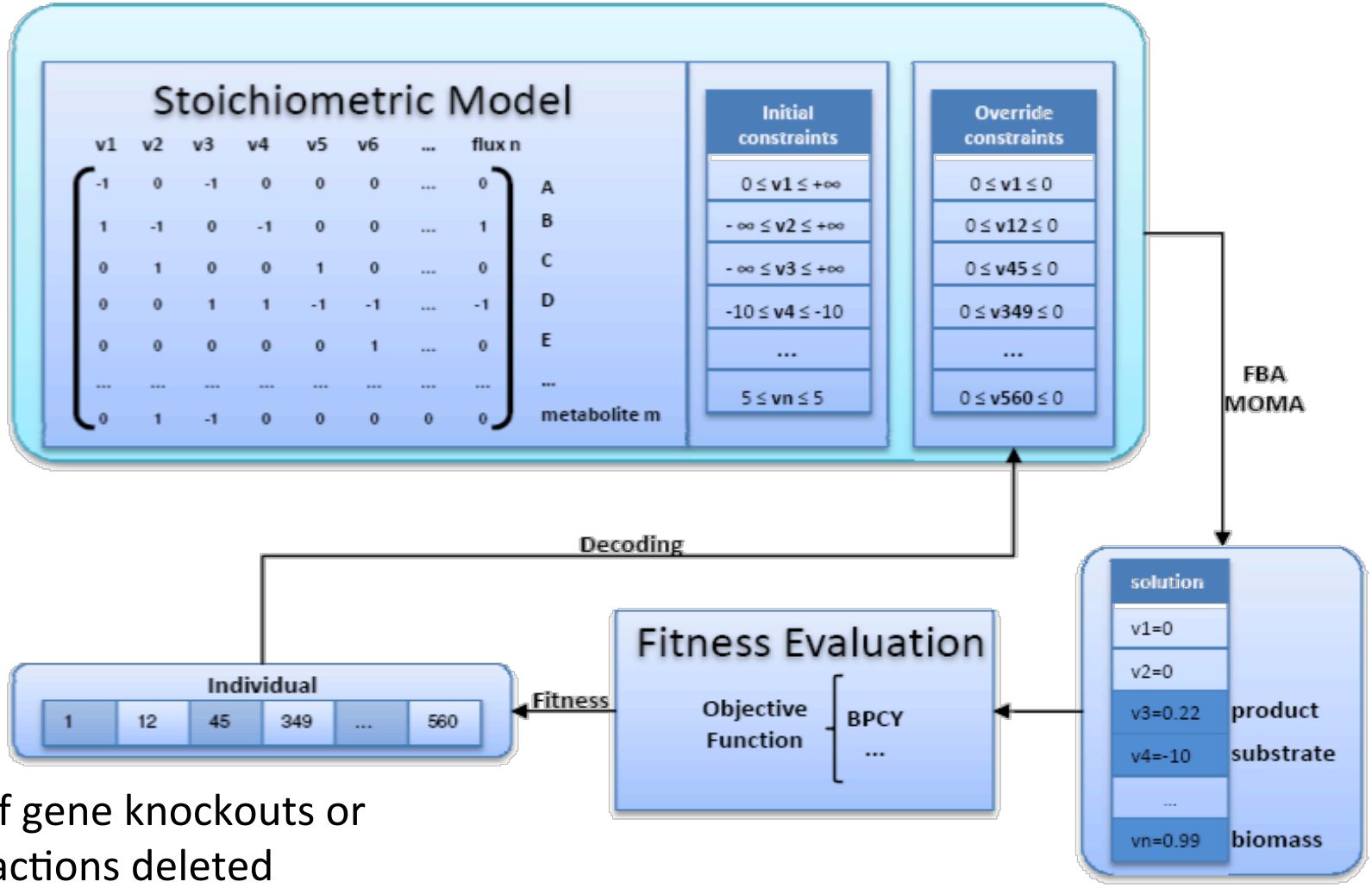
Methods

- Approaches based on MILP (e.g. OptKnock) cannot handle non-linear objective functions and are restricted to small models
- Solution: use of metaheuristic optimization methods

Evolutionary Algorithms

- **Evolutionary Algorithms (EAs)** are based on an analogy with the evolution of living beings by means of natural selection;
- EAs evolve a **population**, i.e. a set of individuals, each encoding a possible solution to the target problem;
- New individuals created applying **reproduction operators** to the existing ones (e.g. mutation, crossover);
- Individuals are assigned a **fitness**, numerical value given according to the quality of the solution;
- Fitness based **stochastic selection** used to choose parents.

Optimization algorithms – solution encoding and evaluation

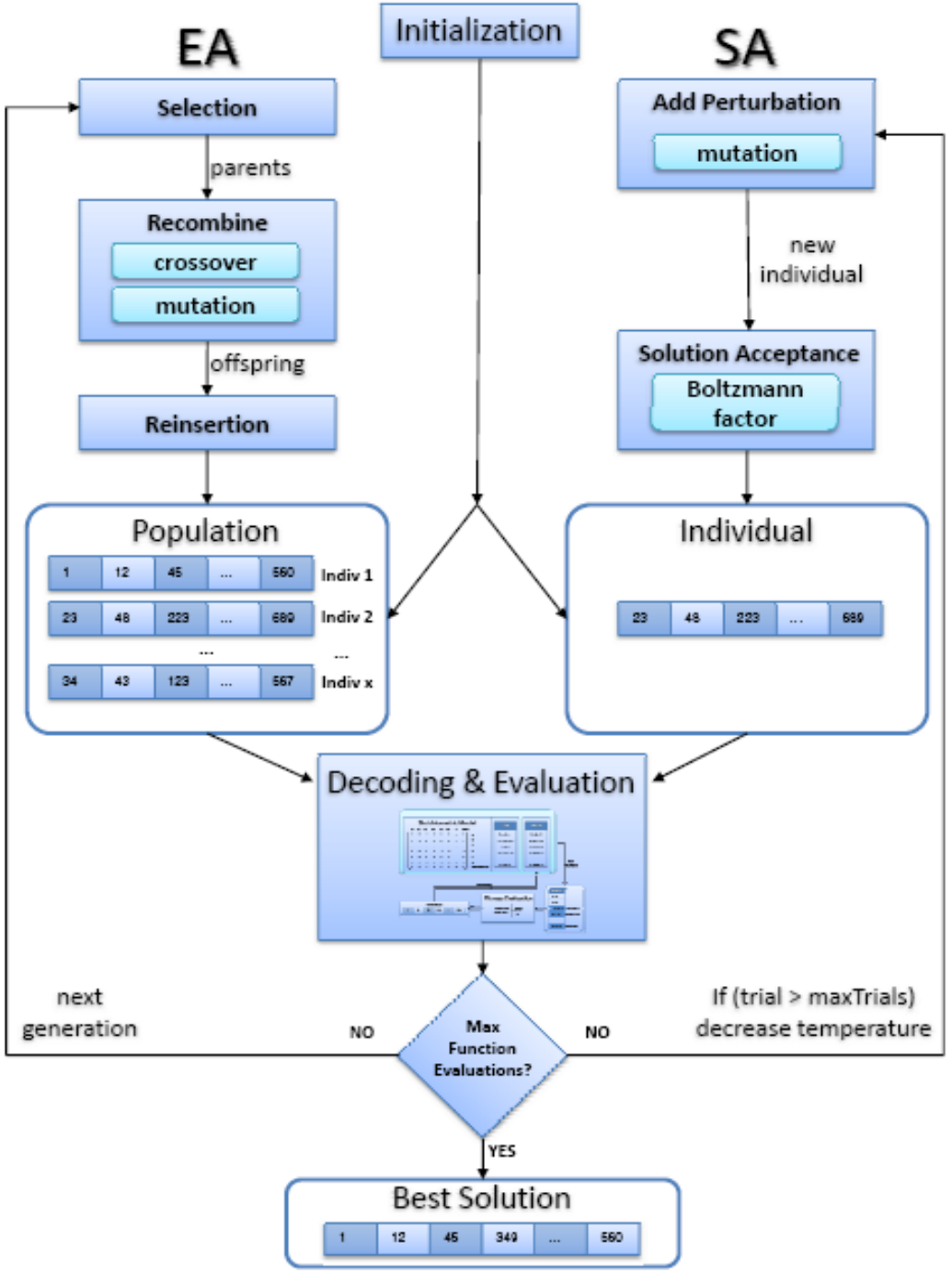


List of gene knockouts or of reactions deleted

Optimization algorithms

Evolutionary Algorithms (EA)

Simulated Annealing (SA)



Strain optimization: gene over-under expression

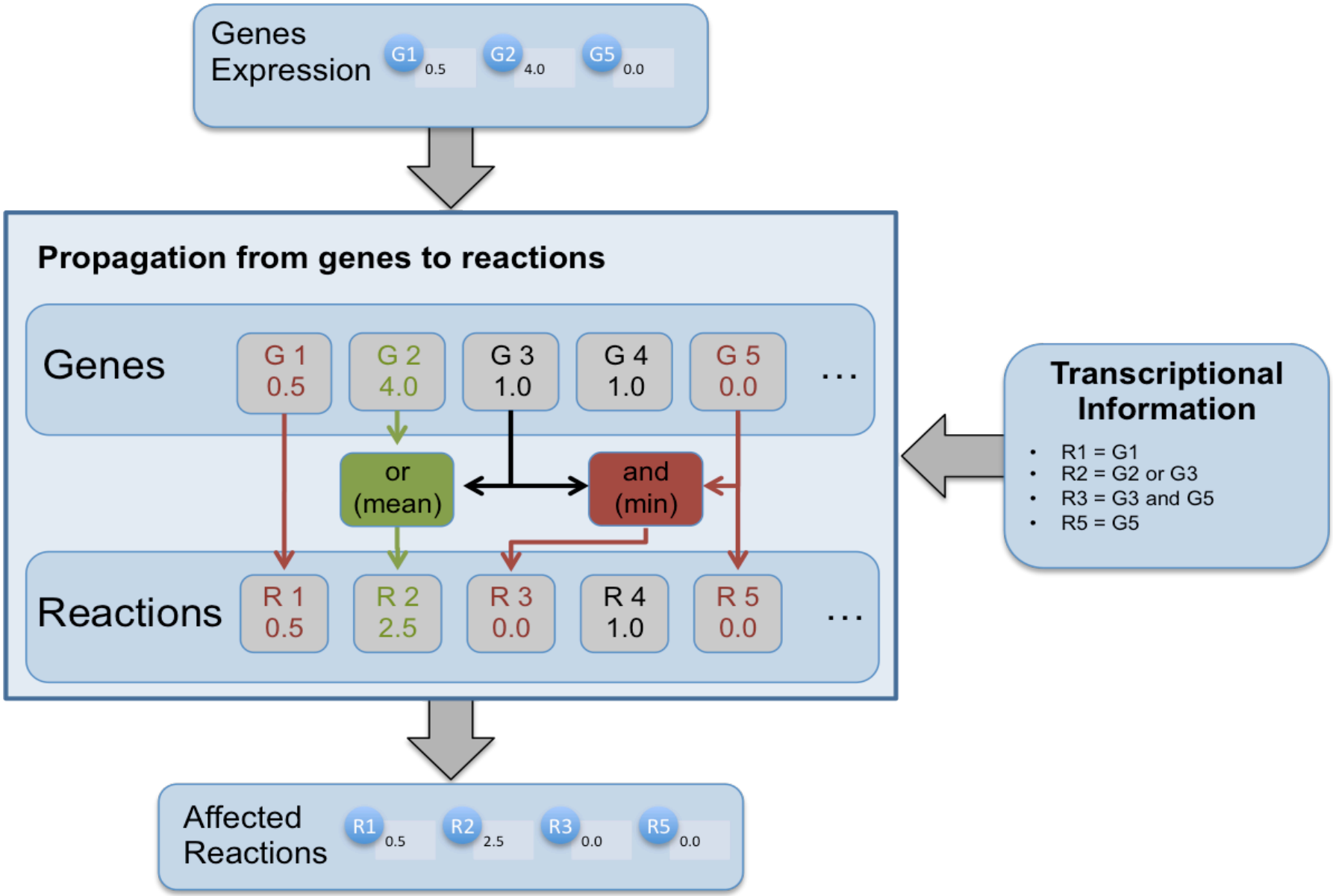
Aim

- Select the best set of genes to **over/under express** and their relative levels

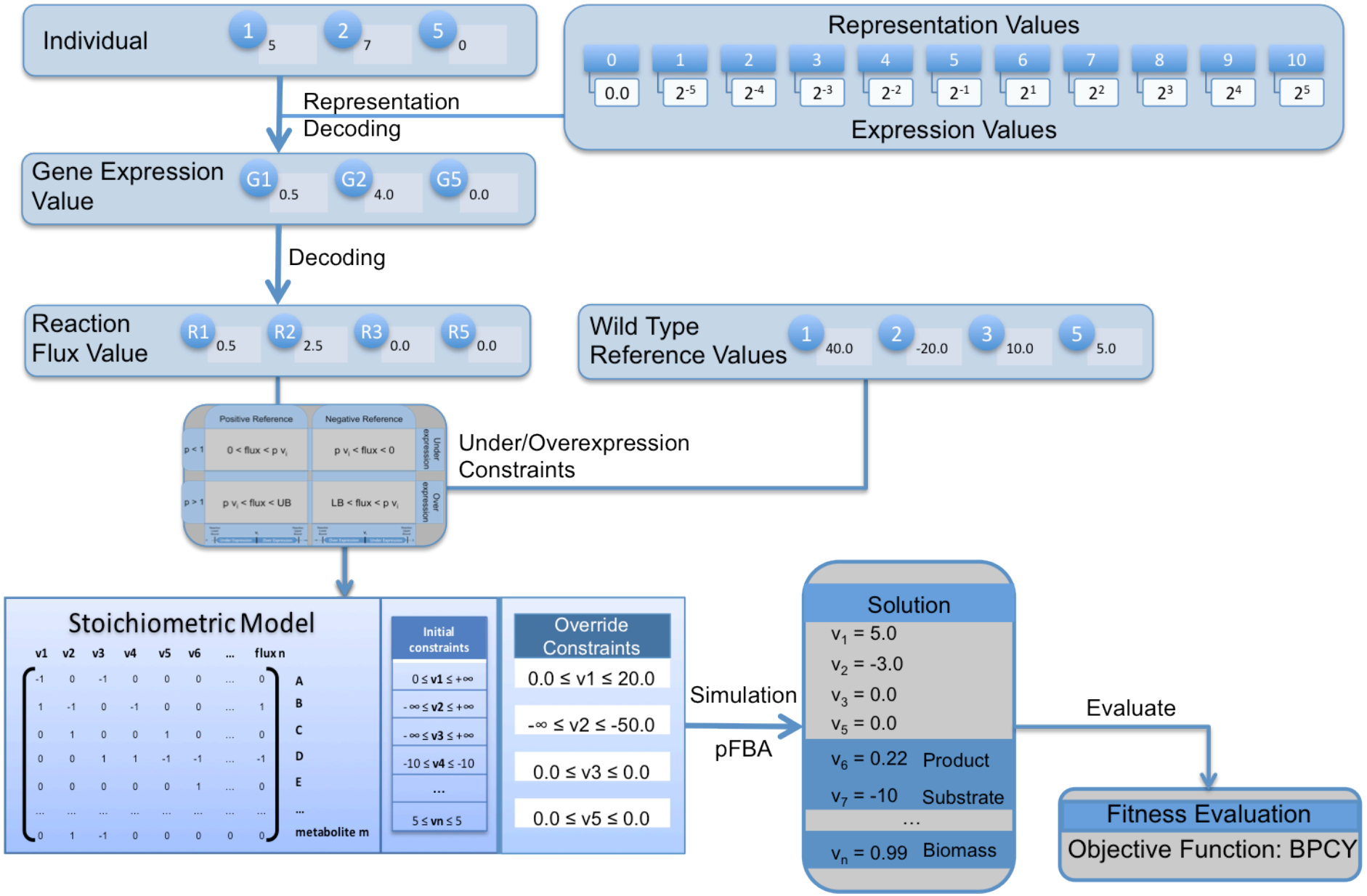
A possible objective function

- Maximizing the production of a compound
- Keeping the organism viable
- **Biomass product coupled yield (BPCY)**: multiplies biomass and compound production fluxes and divides by substrate intake flux

From genes to reactions



Solution decoding and evaluation



Implementation of EAs

- EAs used in this work are implemented in the JECoLi (darwin.di.uminho.pt/jecoli)

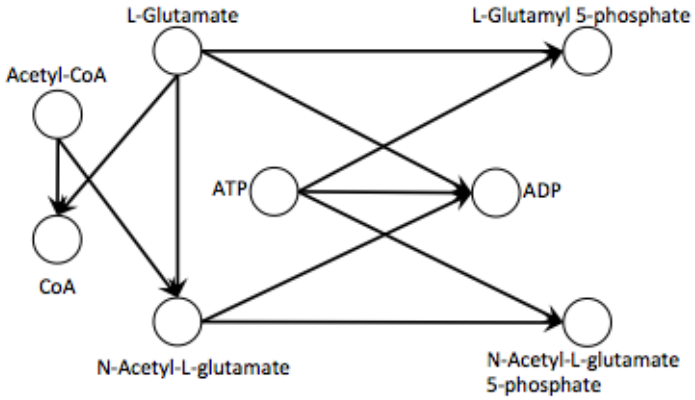
- Previous work on the **parallelization** of these algorithms http://www.cmpbjournal.com/article/S0169-2607(12)00243-X/abstract

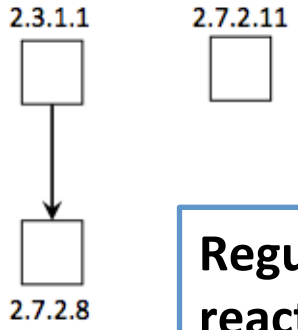
- Parallel EAs were applied to the knockout optimization problem with good results

Optimization of "heterologous" pathways

- **Problem statement**: having a host organism (**chassis**), determine the best set of **reactions** from other organisms (called ***heterologous***) to **add** in order to allow the production of an **exogenous compound**
- Chassis represented by the set of reactions and metabolites (typically metabolic **model**/ network)
- Need to have a larger **"universal" metabolic database** where to select the heterologous reactions
- Different criteria to evaluate solutions: production capability, size of added pathway, easiness of insertion, etc.

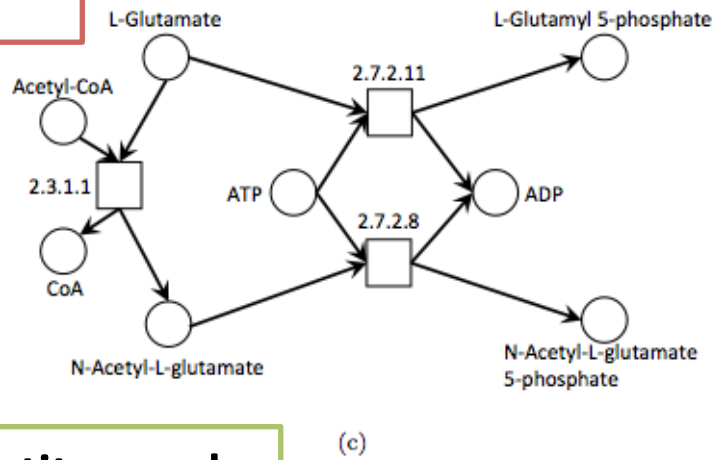
Topological approach: graph representations





Regular graphs: reactions

Regular graphs: metabolites



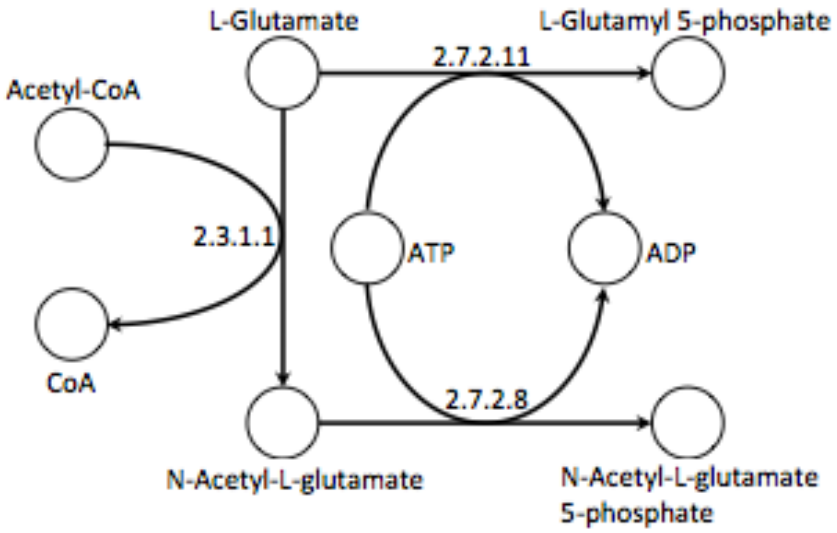
Bipartite graphs

- Nodes are reactions and/ or metabolites
- Allows topological analysis of metabolic networks
- Do not support simulation, but only topological analysis

Optimization of "heterologous" pathways with regular graphs

- A number of well known **shortest path algorithms** have been developed for graphs that can be used in this problem (Dijkstra, BFS, …)
- However, these have important **disadvantages** since they do not capture the nature of the metabolic systems, where a reaction can have >1 substrate and >1 product
- Existing algorithms (e.g. Pathway Hunter) only take into account main substrates and products missing important solutions

Hypergraph representations



Vertices represent **metabolites**

Hyperarcs represent **reactions**, containing their sets of substrates and products

A directed hypergraph is a pair $\mathcal{H} = (V, E)$ *where* $V = \{v_1, v_2 \ldots, v_n\}$ *is the set of vertices and* $E = \{e_1, e_2, \ldots, e_m\}$ *is the set of hyperarcs. A hyperarc* e_i *is an ordered pair* $e_i = (X_i, Y_i)$ *of disjoint subsets of* V.

The set X_i is also called the tail of e_i and the set Y_i is called the head, with reference to the graphical representation of arcs (directed edges) and hyperarcs as arrows.

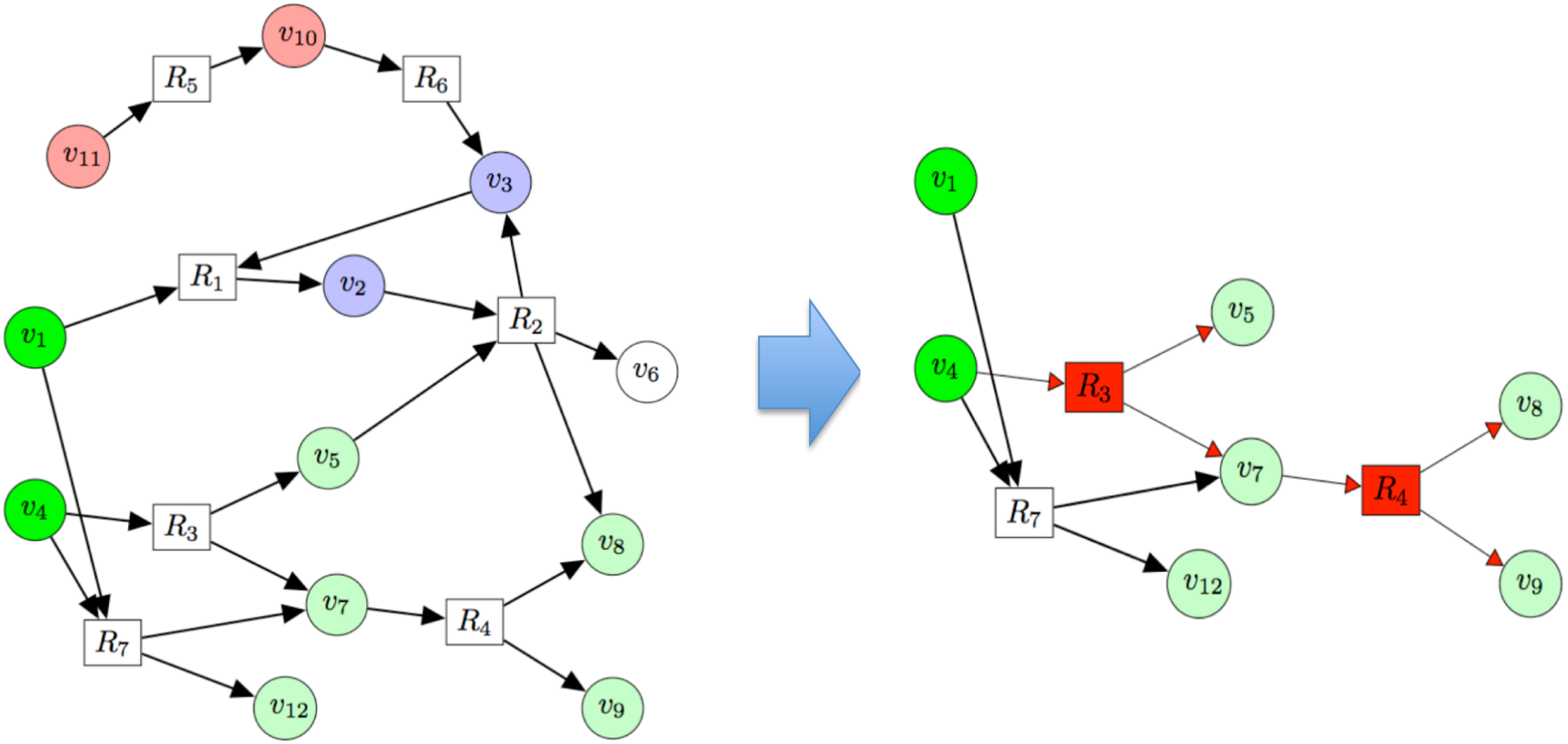
Algorithms over hypergraphs

- Carbonell et al (2012) have proposed a number of algorithms over hypergraphs to support **pathway enumeration**
- Three major steps:
  1. Reactions producing a target compound are iteratively **searched backwards** until the set of precursors includes only source metabolites (endogenous to the chassis)
  2. **Enumeration of all the minimal pathways** that allow to produce the target metabolite from the source ones
  3. **Evaluation** of the different pathways according to different criteria

Algorithms over hypergraphs: *FindAll*

Returns all hyperarcs that are part of an hyperpath from source to target

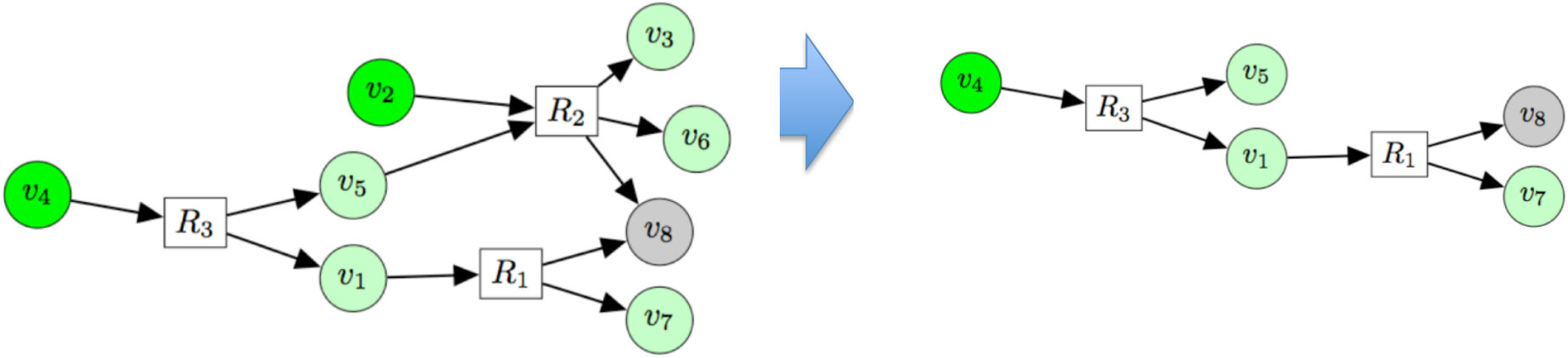
Filters the universe of “interest”



Algorithms over hypergraphs: *Minimize*

Finds minimal hyperpaths: can not be reduced !

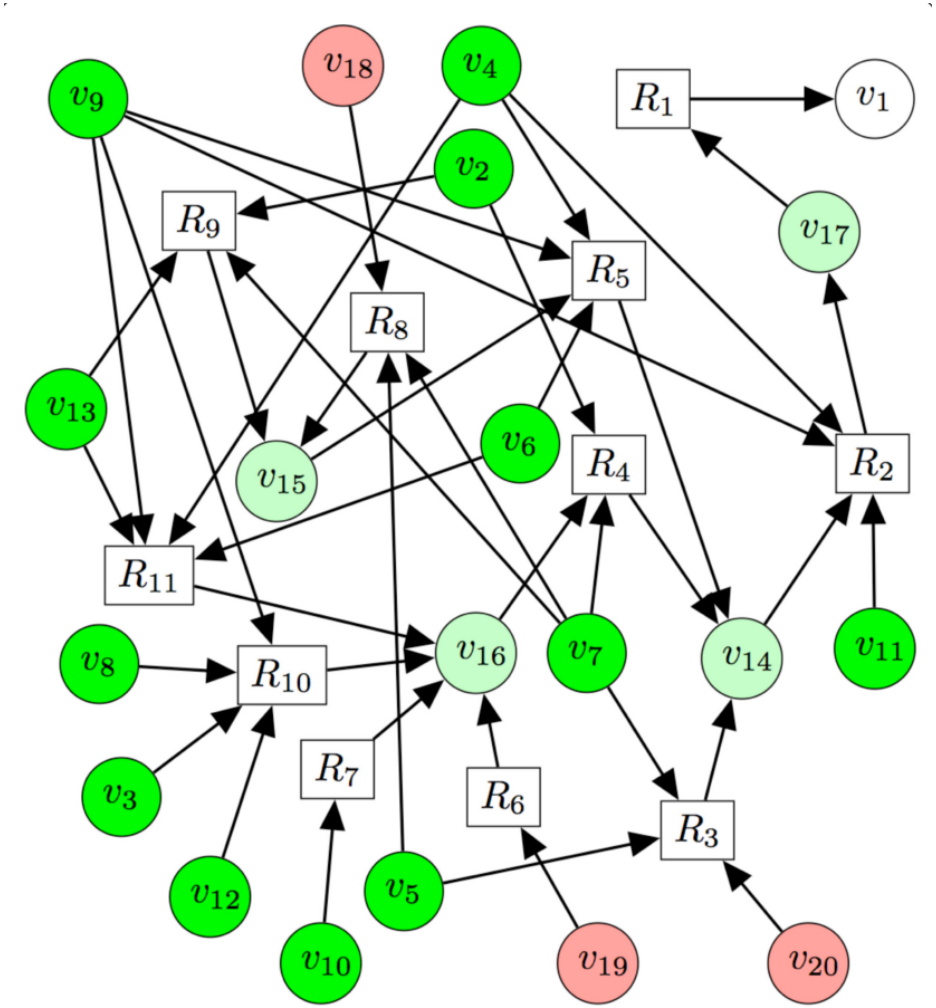
$$R_1\colon\ v_1 \rightarrow v_8 + v_7$$
$$R_2\colon\ v_2 + v_5 \rightarrow v_3 + v_6 + v_8$$
$$R_3\colon\ v_4 \rightarrow v_1 + v_5$$



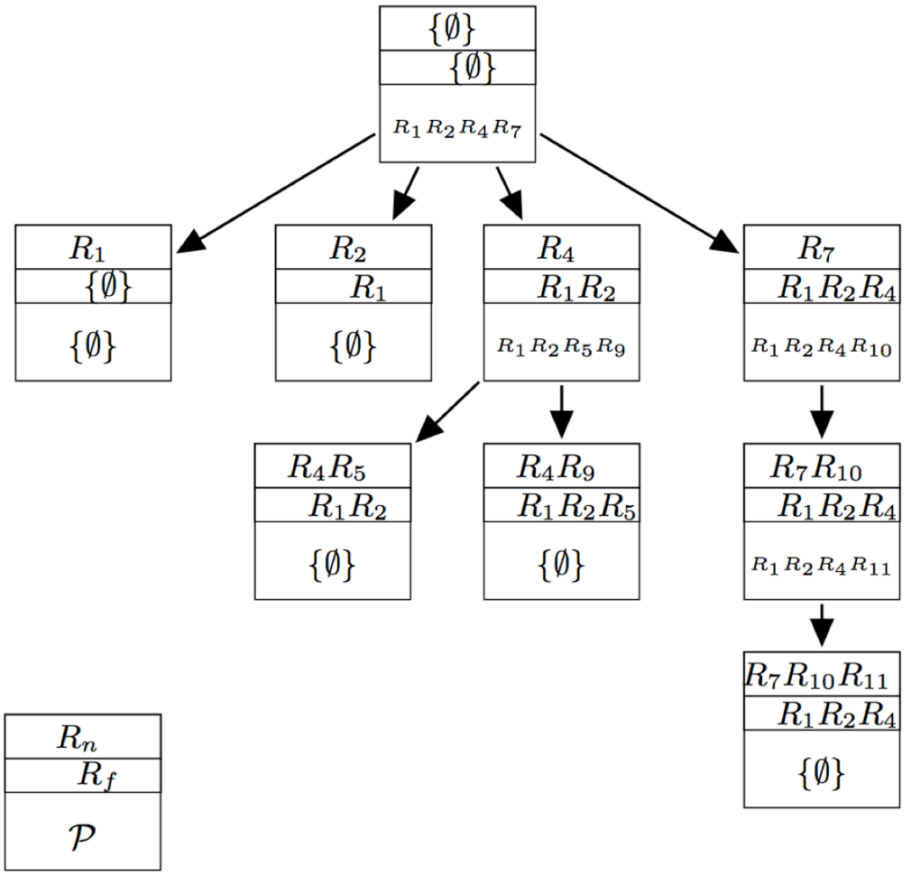
Algorithms over hypergraphs: *FindPath*

- Main algorithm: attempts to **enumerate** all minimal hyperpaths from the sources to the target
- Does not provide a guarantee that all hyperpaths are minimal; problem is **NP complete**; approximate algorithm
- Based on a **recursive partitioning** of the search space (divide-and-conquer algorithm), by defining sets of hyperarcs (reactions) that can not be/ must be in the solutions
- Uses *FindAll* and *Minimize* in its steps

Algorithms over hypergraphs: *FindPath*

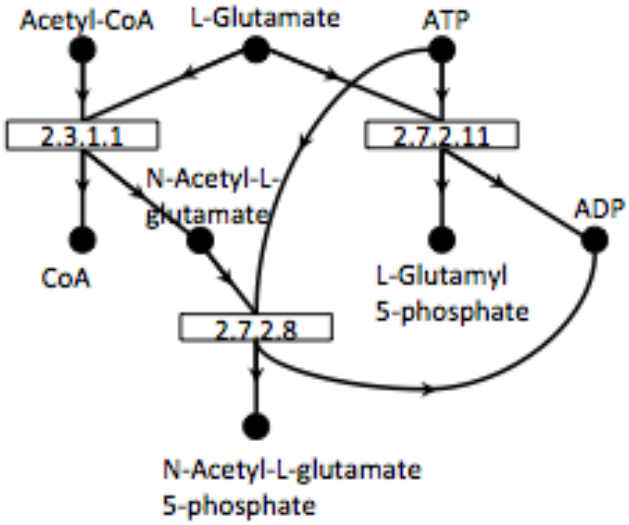


Production of liquiritigenin by E. coli



Rn (top) – reactions that “can not be”
Rf (middle) – reactions that “must be”
P – solutions found (minimal)

Alternative: p-graph representations



Materials represent **metabolites**

Operational untis represent **reactions**

A process graph is a pair $\mathcal{P} = \langle M, O \rangle$, where $M = \{m_0, m_1, \ldots, m_i\}$ is the set of materials (or species) and $O = \{o_0, o_1, \ldots, o_j\}$ is the set of operational units. An operational unit o is a tuple such that $o \subseteq \wp(M) \times \wp(M)$. Moreover, the vertices of the process graph are the elements of:

$$\mathcal{V} = M \cup O$$

and the arcs of the process graph are the elements of:

$$\mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2$$

where $\mathcal{A}_1$ contains the arcs that point towards the operating units:

$$\mathcal{A}_1 = \{\langle x, y \rangle \mid y = \langle \alpha, \beta \rangle \in o \text{ and } x \in \alpha\}$$

and $\mathcal{A}_2$ keeps the arcs that point outwards of the operating units:

$$\mathcal{A}_2 = \{\langle y, x \rangle \mid y = \langle \alpha, \beta \rangle \in o \text{ and } x \in \beta\}$$

Algorithms to address tasks similar to the ones in hypergraphs have been developed (MSG, SSG, etc) – not presented here

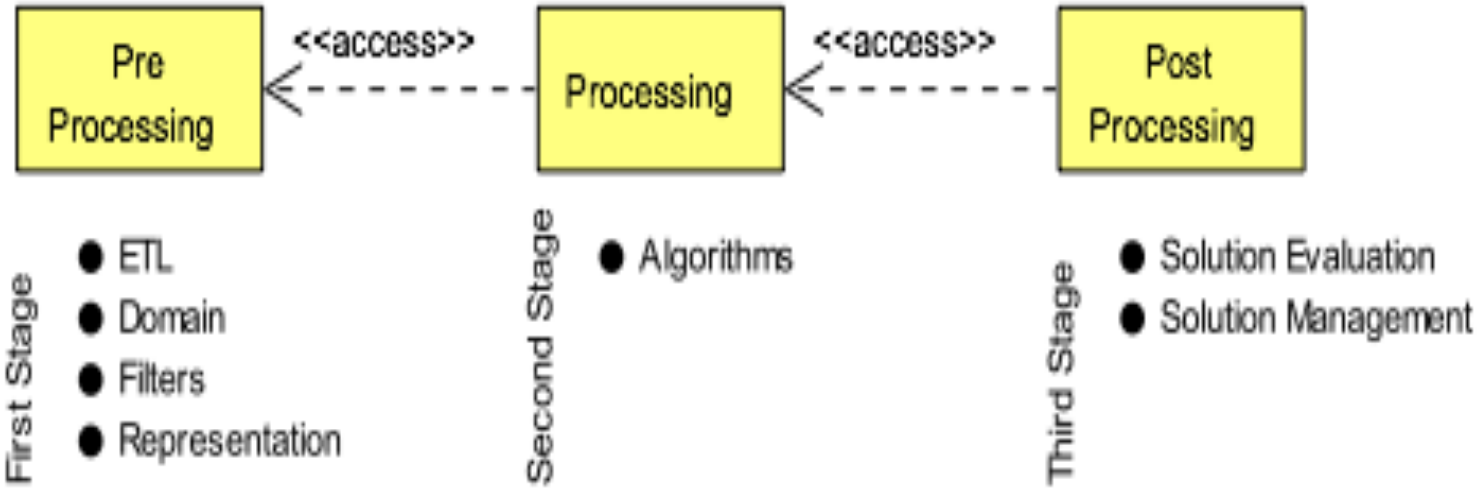
The Biosynth framework

- Existing algorithms over hypergraphs and p-graphs have been implemented and improved – Master thesis work of Filipe Liu @ MEI (defense in the Dec 11th)

- All algorithms were implemented in Java without any parallelization

The Biosynth framework

- The framework has 3 distinct layers:



Algorithms implemented:
- Over hypergraphs: FindAll, Minimize, FindPath
- Over p-graphs: MSG, SSG
- Several evaluators for solutions
- Creation of domains from KEGG and BioCyc databases
- Filters of the domains

Future objectives

- Implement parallel versions of the main algorithms: starting with *FindPath*
- Study the characteristics of the algorithms regarding their efficient paralellization
- Develop new versions of the algorithms more suited to parallelism