V13 Solving the Maximum-Flow Problem

We will present an algorithm that originated by Ford and Fulkerson (1962).
Idea: increase the flow in a network iteratively until it cannot be increased any further → **augmenting flow path**.

Suppose that *f* is a flow in a capacitated *s-t* network *N,* and suppose that there exists a directed *s-t* path

$$P = \langle s, e_1, v_1, e_2, \ldots, e_k, t \rangle$$

in *N*, such that $f(e_i) < cap(e_i)$ for $i=1, \ldots, k$.

Then considering arc capacities only, the flow on each arc e_i
can be increased by as much as $cap(e_i) - f(e_i)$.

But to maintain the **conservation-of-flow** property at each of the vertices v_i,
the **increases** on all of the arcs of path *P* **must be equal**.

Thus, if Δ_P denotes this **increase**,
then the largest possible value for Δ_P is $min\{cap(e_i\} - f(e_i)\}$.

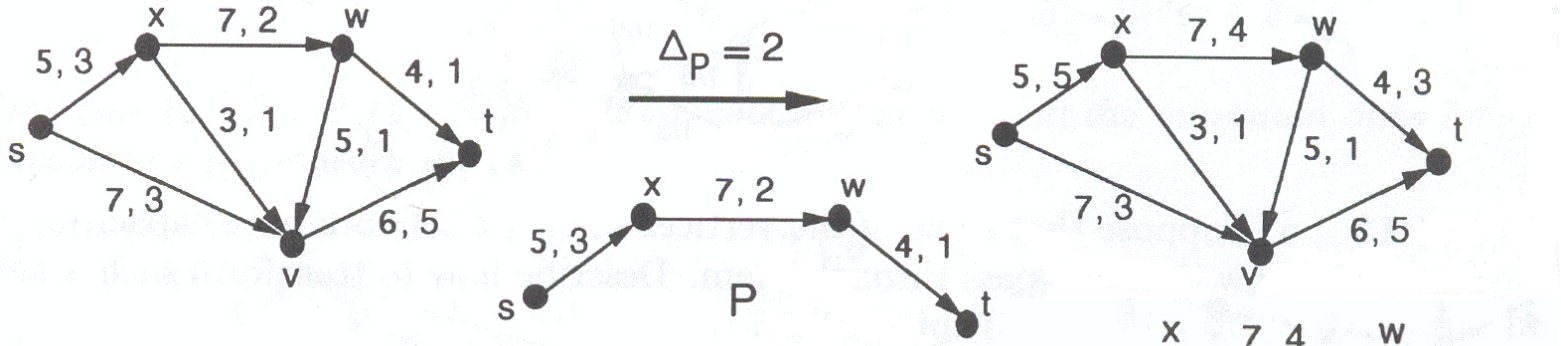
Solving the Maximum-Flow Problem

Example: Left: the value of the current flow is 6.

Consider the directed *s-t* path $P = \langle s,x,w,t \rangle$.

The flow on each arc of path *P* can be increased by $\Delta_P = 2$.

The resulting flow, which has value 8, is shown on the right side.



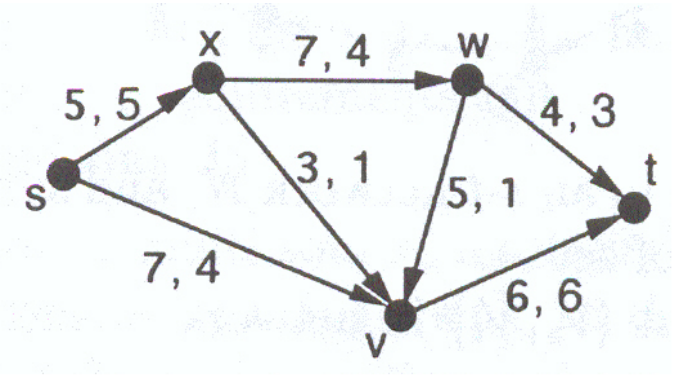
Using the directed path $\langle s,v,t \rangle$, the flow can be increased to 9. The resulting flow is shown right.

At this point, the flow cannot be increased any further along **directed *s-t paths***, because each such path must either use the arc directed from *s* to *x* or from *v* to *t*. Both arcs have flow at capacity.

Solving the Maximum-Flow Problem

However, the flow can be increased further.

E.g. increase the flow on the arc from source *s* to vertex *v* by one unit,



decrease the flow on the arc from *w* to *v* by one unit, and

increase the flow on the arc from *w* to *t* by one unit.

f-Augmenting Paths

Definition: An *s-t* **quasi-path** in a network *N* is an alternating sequence

$$\langle s = v_0, e_1, v_1, \ldots, v_{k-1}, e_k, v_k = t \rangle$$

of vertices and arcs that forms an *s-t* path in the underlying undirected graph of *N.*

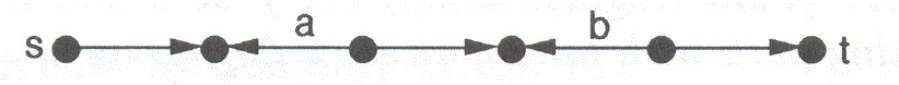
Terminology For a given *s-t* quasi-path

$$Q = \langle s = v_0, e_1, v_1, \ldots, v_{k-1}, e_k, v_k = t \rangle$$

arc e_i is called a **forward arc** if it is directed from vertex v_{i-1} to vertex v_i and
arc e_i is called a **backward arc** if it is directed from v_i to v_{i-1}.

Clearly, a directed *s-t* path is a quasi-path whose arcs are all forward.

Example. On the *s-t* quasi-path shown below, arcs *a* and *b* are backward, and the three other arcs are forward.



f-Augmenting Paths

Definition: Let *f* be a flow in an *s-t* network *N.* An ***f*-augmenting path** Q is an *s-t* quasi path in *N* such that the flow on each forward arc can be increased, and the flow on each backward arc can be decreased.

Thus, for each arc *e* on an *f*-augmenting path Q,

$f(e) < cap(e)$,	if *e* is a forward arc
$f(e) > 0$	if *e* is a backward arc.

Notation For each arc *e* on a given *f*-augmenting path Q, let Δ_e be the quantity given by

$$\Delta_e = \begin{cases} cap(e) - f(e), & \text{if } e \text{ is a forward arc} \\ f(e), & \text{if } e \text{ is a backward arc} \end{cases}$$

Terminology The quantity Δ_e is called the **slack on arc** *e.* Its value on a forward arc is the largest possible increase in the flow, and on a backward arc, the largest possible decrease in the flow, disregarding conservation of flow.

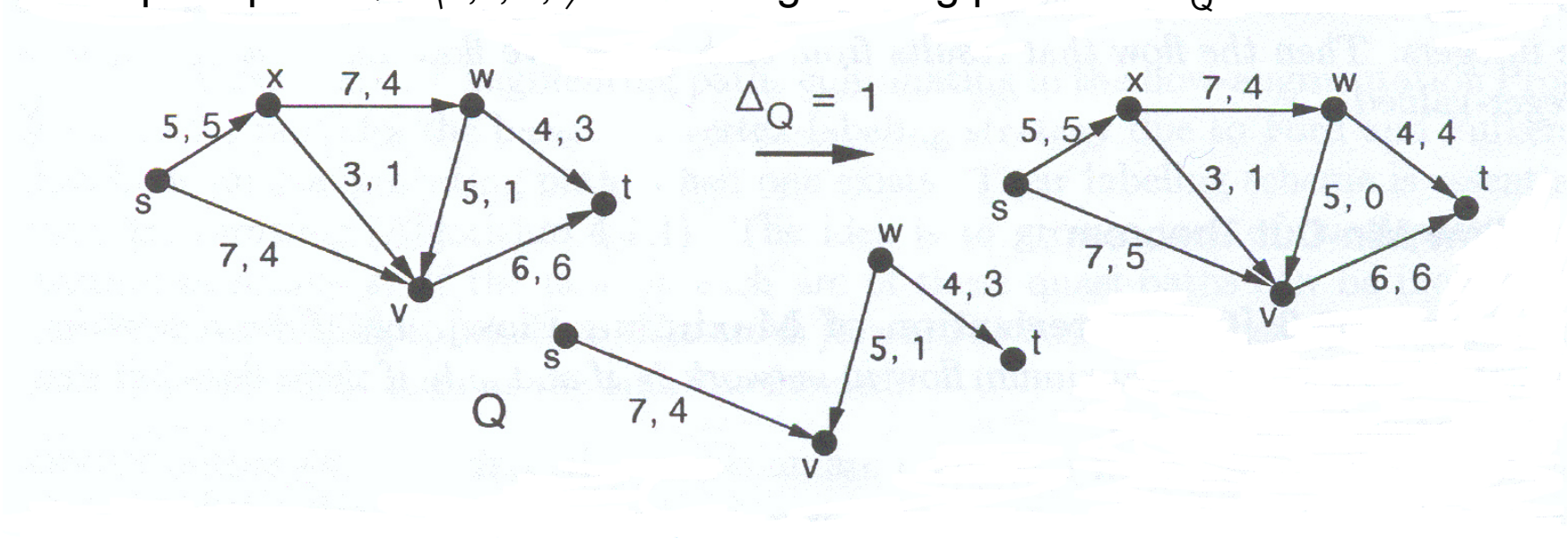
f-Augmenting Paths

Remark Conservation of flow requires that the change in the flow on the arcs of an augmenting flow path be of equal magnitude.

Thus, the maximum allowable change in the flow on an arc of quasipath Q is Δ_Q, where

$$\Delta_Q = \min_{e \in Q} \{\Delta_e\}$$

Example For the example network shown below, the current flow *f* has value 9, and the quasi-path Q = ⟨*s,v,w,t*⟩ is an *f*-augmenting path with $\Delta_Q = 1$.



flow augmentation

Proposition 12.2.1 (Flow Augmentation) Let *f* be a feasible flow in a network *N*, and let Q be an *f*-augmenting path with minimum slack Δ_Q on its arcs.
Then the augmented flow *f'* given by

$$f'(e) = \begin{cases} f(e) + \Delta_Q, & \text{if } e \text{ is a forward arc of Q} \\ f(e) - \Delta_Q, & \text{if } e \text{ is a backward arc of Q} \\ f(e) & \text{otherwise} \end{cases}$$

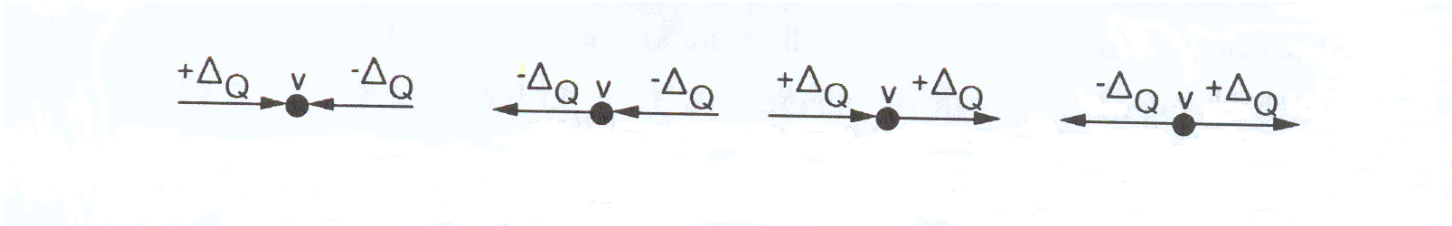
is also a feasible flow in network *N* and *val(f')* = *val(f)* + Δ_Q.

Proof. Clearly, 0 ≤ *f'(e)* ≤ *cap(e)*, by the definition of Δ_Q.

The only vertices through which the net flow may have changed are those vertices on the augmenting path Q. Thus, to verify that *f'* satisfies conservation of flow, only the internal vertices of Q need to be checked.

f-Augmenting Paths

For a given vertex *v* on augmenting path Q, the two arcs of Q that are incident on *v* are configured in one of four ways, as shown below. In each case, the net flow into or out of vertex *v* does not change, thereby preserving the conservation-of-flow property.



It remains to be shown that the flow has increased by Δ_Q.

The only arc incident on the source *s* whose flow has changed is the first arc e_1 of augmenting path Q.

If e_1 is a forward arc, then $f'(e_1) = f(e_1) + \Delta_Q$, and

if e_1 is a backward arc, then $f'(e_1) = f(e_1) - \Delta_Q$. In either case,

$$val(f') = \sum_{e \in Out(s)} f'(e) - \sum_{e \in In(s)} f'(e) = \Delta_Q + val(f) \quad \square$$

Max-Flow Min-Cut

Theorem 12.2.3 [Characterization of Maximum Flow]

Let *f* be a flow in a network *N.*

Then *f* is a maximum flow in network *N* if and only if there does not exist an *f*-augmenting path in *N.*

Proof: Necessity (⇒) Suppose that *f* is a maximum flow in network *N*.

Then by Proposition 12.2.1, there is no *f*-augmenting path.

Proposition 12.2.1 (Flow Augmentation) Let *f* be a flow in a network *N*, and let Q be an *f*-augmenting path with minimum slack ΔQ on its arcs. Then the augmented flow *f'* given by

$$f'(e) = \begin{cases} f(e) + \Delta_Q, & \text{if } e \text{ is a forward arc of Q} \\ f(e) - \Delta_Q, & \text{if } e \text{ is a backward arc of Q} \\ f(e) & \text{otherwise} \end{cases}$$

is a feasible flow in network *N* and ***val(f') = val(f) + ΔQ***.

→ assuming an *f*-augmenting path existed, we could construct a flow *f'* with *val(f') > val(f)* contradicting the maximality of *f.*

Max-Flow Min-Cut

Sufficiency ($\Leftarrow$) Suppose that there does not exist an *f*-augmenting path in network *N.*

Consider the collection of all quasi-paths in network *N* that begin with source *s*, and have the following property: each forward arc on the quasi-path has positive slack, and each backward arc on the quasi-path has positive flow.

Let V_s be the union of the vertex-sets of these quasi-paths.

Since there is no *f*-augmenting path, it follows that sink $t \notin V_s$.

Let $V_t = V_N - V_s$.

Then $\langle V_s, V_t \rangle$ is an *s*-*t* cut of network *N*. Moreover, by definition of the sets V_s and V_t ,

$$f(e) = \begin{cases} cap(e) & \text{if } e \in \langle V_s, V_t \rangle \\ 0 & \text{if } e \in \langle V_t, V_s \rangle \end{cases}$$

(if the flow along these edges *e* were not cap(e) or 0, these edges would belong to V_s!)

Hence, *f* is a maximum flow, by Corollary 12.1.8. □

Max-Flow Min-Cut

Theorem 12.2.4 [Max-Flow Min-Cut] For a given network, the value of a maximum flow is equal to the capacity of a minimum cut.

Proof: The *s-t* cut $\langle V_s, V_t \rangle$ that we just constructed in the proof of Theorem 12.2.3 (direction $\Leftarrow$) has capacity equal to the maximum flow. □

The outline of an algorithm for maximizing the flow in a network emerges from Proposition 12.2.1 and Theorem 12.2.3.

Algorithm 12.2.1: Outline for Maximum Flow

Input: an *s-t* network N.
Output: a maximum flow f^* in network N.

[*Initialization*]
For each arc e in network N
  $f^*(e) := 0$
[*Flow Augmentation*]
While there exists an f^*-augmenting path in network N
  Find an f^*-augmenting path Q.
  Let $\Delta_Q = \min_{e \in Q}\{\Delta_e\}$.
  For each arc e of augmenting path Q
    If e is a forward arc
      $f^*(e) := f^*(e) + \Delta_Q$
    Else (e is a backward arc)
      $f^*(e) := f^*(e) - \Delta_Q$
Return flow f^*.

Finding an *f*-Augmenting Path

The discussion of *f*-augmenting paths culminating in the flow-augmenting Proposition 12.2.1 provides the basis of a vertex-labeling strategy due to Ford and Fulkerson that finds an *f*-augmenting path, when one exists.

Their labelling scheme is essentially **basic tree-growing**.

The idea is to grow a tree of quasi-paths, each starting at source *s*.

If the flow on each arc of these quasi-paths can be increased or decreased, according to whether that arc is **forward** or **backward**, then an *f*-augmenting path is obtained as soon as the sink *t* is labelled.

Finding an *f*-Augmenting Path

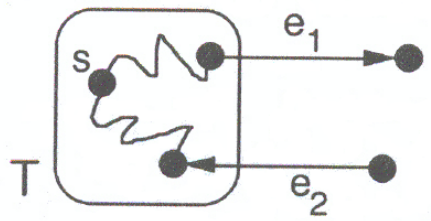
A **frontier arc** is an arc *e* directed from a **labeled** endpoint *v* to an **unlabeled** endpoint *w*.

For constructing an *f*-augmenting path, the frontier path *e* is allowed to be backward (directed from vertex *w* to vertex *v*), and it can be added to the tree as long as it has slack $\Delta_e > 0$.

Finding an *f*-Augmenting Path

Terminology: At any stage during tree-growing for constructing an *f*-augmenting path, let *e* be a frontier arc of tree *T*, with endpoints *v* and *w.*
The arc *e* is said to be **usable** if, for the current flow *f*, either

e is directed from vertex *v* to vertex *w* and $f(e) < cap(e)$, or
e is directed from vertex *w* to vertex *v* and $f(e) > 0$.



Frontier arcs e_1 and e_2 are usable if
$f(e_1) < cap(e_1)$ and $f(e_2) > 0$

Remark From this vertex-labeling scheme, any of the existing *f*-augmenting paths could result. But the efficiency of Algorithm 12.2.1 is based on being able to find „good“ augmenting paths.
If the arc capacities are irrational numbers, then an algorithm using the Ford&Fulkerson labeling scheme might not terminate (strictly speaking, it would not be an algorithm).

Finding an *f*-Augmenting Path

Even when flows and capacities are restricted to be integers, problems concerning efficiency still exist.

E.g., if each flow augmentation were to increase the flow by only one unit, then the number of augmentations required for maximization would equal the capacity of a minimum cut.

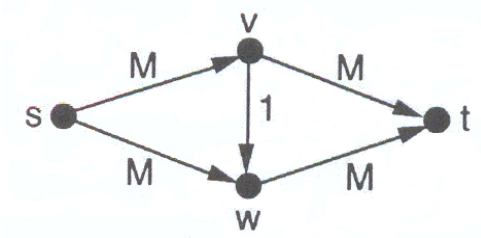
Such an algorithm would depend on the size of the arc capacities instead of on the size of the network.

Finding an *f*-Augmenting Path

Example: For the network shown below, the arc from vertex *v* to vertex *w* has flow capacity 1, while the other arcs have capacity *M*, which could be made arbitrarily large.

If the choice of the augmenting flow path at each iteration were to alternate between the directed path $\langle s,v,w,t \rangle$ and the quasi path $\langle s,w,v,t \rangle$, then the flow would increase by only one unit at each iteration.

Thus, it could take as many as $2M$ iterations to obtain the maximum flow.



Finding an *f*-Augmenting Path

Edmonds and Karp avoid these problems with this algorithm.

It uses **breadth-first search** to find an *f*-augmenting path with the smallest number of arcs.

Algorithm 12.2.2: Finding an Augmenting Path

Input: a flow f in an s-t network N.
Output: an f-augmenting path Q or a minimum s-t cut with capacity $val(f)$.

Initialize vertex set $V_s := \{s\}$.
Write label 0 on vertex s.
Initialize label counter $i := 1$
While vertex set V_s does not contain sink t
  If there are usable arcs
    Let e be a usable arc whose labeled endpoint v has the smallest possible label.
    Let w be the unlabeled endpoint of arc e.
    Set $backpoint(w) := v$.
    Write label i on vertex w.
    $V_s := V_s \cup \{w\}$
    $i := i + 1$
  Else
    Return s-t cut $\langle V_s, V_N - V_s \rangle$.
Reconstruct the f-augmenting path Q by following backpointers, starting from sink t.
Return f-augmenting path Q.

FFEK algorithm: Ford, Fulkerson, Edmonds, and Karp

Algorithm 12.2.3 combines Algorithms 12.2.1 and 12.2.2

Algorithm 12.2.3: FFEK - Maximum Flow

Input: an s-t network N.
Output: a maximum flow f^* in network N.

[*Initialization*]
For each arc e in network N
  $f^*(e) := 0$
[*Flow Augmentation*]
Repeat
  Apply Algorithm 12.2.2 to find an f^*-augmenting path Q.
  Let $\Delta_Q = \min_{e \in Q}\{\Delta_e\}$.
  For each arc e of augmenting path Q
    If e is a forward arc
      $f^*(e) := f^*(e) + \Delta_Q$
    Else (e is a backward arc)
      $f^*(e) := f^*(e) - \Delta_Q$
Until an f^*-augmenting path cannot be found in network N.
Return flow f^*.

FFEK algorithm: Ford, Fulkerson, Edmonds, and Karp

Example: the figures illustrate algorithm 12.2.3.

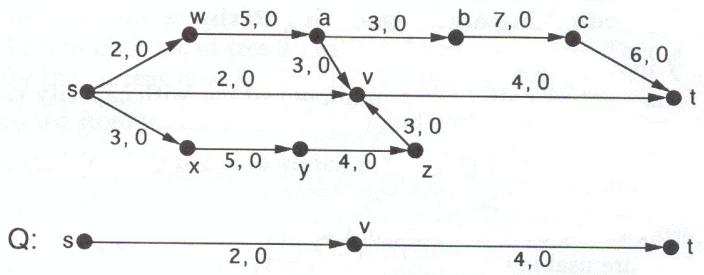


Figure 12.2.8 Iteration 0: $val(f) = 0$; augmenting path Q has $\Delta_Q = 2$.

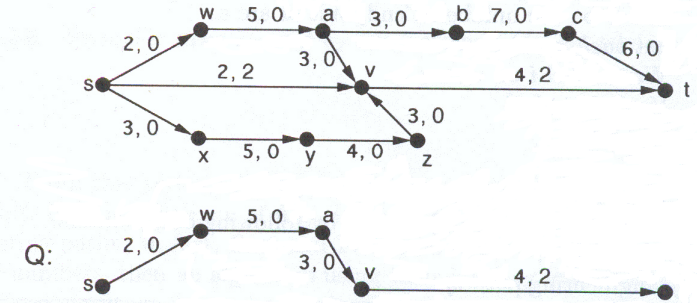


Figure 12.2.9 Iteration 1: $val(f) = 2$; augmenting path Q has $\Delta_Q = 2$.

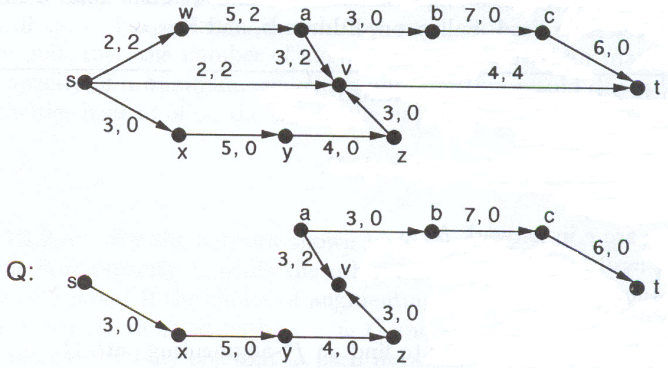
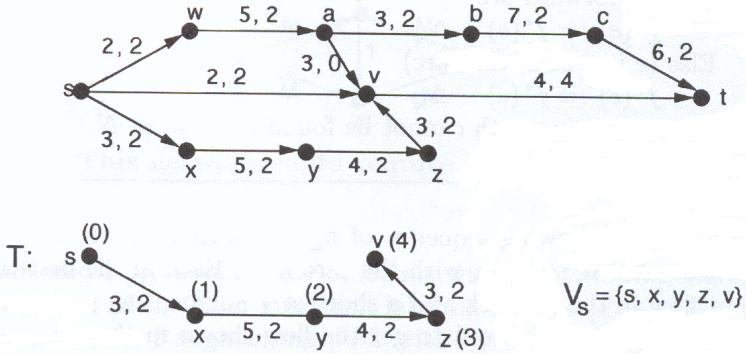


Figure 12.2.10 Iteration 2: $val(f) = 4$; augmenting path Q has $\Delta_Q = 2$.



e 12.2.11 Final iteration: $val(f) = 6 = cap\langle\{s, x, y, z, v\}, \{w, a, b, c, t\}\rangle$.

<{s, x, y, z, v}, {w, a, b, c, t}> is the *s-t* cut with capacity equal to the current flow, establishing optimality.

FFEK algorithm: Ford, Fulkerson, Edmonds, and Karp

At the end of the final iteration, the two arcs from source *s* to vertex *w* and the arc directed from vertex *v* to sink *t* form the minimum cut $\langle$ {*s,x,y,z,v* }, {*w,a,b,c,t*} $\rangle$. Neither of them is usable, i.e. the flow(e) = cap(e).

This illustrates the *s-t* cut that was constructed in the proof of theorem 12.2.3.

From Graph connectivity to Metabolic networks

We will now use the theory of network flows to give constructive proofs of Menger‘s theorem.

These proofs lead directly to algorithms for determining the edge-connectivity and vertex-connectivity of a graph.

The strategy to prove Menger‘s theorems is based on properties of certain **networks** whose arcs all have **unit capacity**.

These **0-1 networks** are constructed from the original graph.

Determining the connectivity of a graph

Lemma 12.3.1. Let *N* be an *s-t* network such that

outdegree(s) > *indegree(s),*

indegree(t) > *outdegree (t),* and

outdegree(v) = *indegree(v)* for all other vertices *v.*

Then, there exists a directed *s-t* path in network *N.*

Proof. Let *W* be a longest directed trail (trail = walk without repeated edges; path = trail without repeated vertices) in network *N* that starts at source *s*, and let *z* be its terminal vertex.

If vertex *z* were not the sink *t,* then there would be an arc not in trail *W* that is directed from *z* (since *indegree(z) = outdegree(z)).*

But this would contradict the maximality of trail *W.*

Thus, *W* is a directed trail from source *s* to sink *t.*

If *W* has a repeated vertex, then a part of *W* determines a directed cycle, which can be deleted from *W* to obtain a shorter directed *s-t* trail.

This deletion step can be repeated until no repeated vertices remain, at which point, the resulting directed trail is an *s-t* path. □

Determining the connectivity of a graph

Proposition 12.3.2. Let *N* be an *s*-*t* network such that

outdegree(s) – indegree(s) = m = indegree(t) – outdegree (t),

and *outdegree(v) = indegree(v)* for all vertices $v \neq s,t$.

Then, there exist *m* disjoint directed *s*-*t* path in network *N.*

Proof. If *m* = 1, then there exists an open eulerian directed trail *T* from source *s* to sink *t* by Theorem 6.1.3.

Review: An eulerian trail in a graph is a trail that visits every edge of that graph exactly once.

Theorem 6.1.3. A connected digraph *D* has an open eulerian trail from vertex *x* to vertex *y* if and only if *indegree(x)* + 1 = *outdegree(x), indegree(y)* = *outdegree(y)* + 1, and all vertices except *x* and *y* have equal indegree and outdegree.
Euler proved that a necessary condition for the existence of Eulerian circuits is that all vertices in the graph have an even degree.

Theorem 1.5.2. Every open *x*-*y* walk *W* is either an *x*-*y* path or can be reduced to an *x*-*y* path.

Therefore, trail *T* is either an *s*-*t* directed path or can be reduced to an *s*-*t* path.

Determining the connectivity of a graph

By way of induction, assume that the assertion is true for $m = k$, for some $k \geq 1$, and consider a network N for which the condition holds for $m = k + 1$.
There does exist at least one directed s-t path P by Lemma 12.3.1.

If the arcs of path P are deleted from network N, then the resulting network $N - P$ satisfies the condition of the proposition for $m = k$.

By the induction hypothesis, there exist k arc-disjoint directed s-t paths in network $N - P$. These k paths together with path P form a collection of $k + 1$ arc-disjoint directed s-t paths in network N. □

Basic properties of 0-1 networks

Definition A **0-1 network** is a capacitated network whose arc capacities are either 0 or 1.

Proposition 12.3.3. Let *N* be an *s-t* network such that *cap(e)* = 1 for every arc *e.* Then the value of a maximum flow in network *N* equals the maximum number of arc-disjoint directed *s-t* paths in *N.*

Proof: Let *f** be a maximum flow in network *N*, and let *r* be the maximum number of arc-disjoint directed *s-t* paths in *N.*
Consider the network *N** obtained by deleting from *N* all arcs *e* for which *f*(e)* = 0.
Then *f*(e)* = 1 for all arcs *e* in network *N*.*
It follows from the definition that for every vertex *v* in network *N*,*

$$\sum_{e \in Out(v)} f^*(e) = |Out(v)| = outdegree(v)$$

and

$$\sum_{e \in In(v)} f^*(e) = |In(v)| = indegree(v)$$

Basic properties of 0-1 networks

Thus by the definition of *val(f*)* and by the conservation-of-flow property,

$$outdegree(s) - indegree(s) = val(f^*) = indegree(t) - outdegree(t)$$

and $outdegree(v) = indegree(v)$, for all vertices $v \neq s,t$.

By Proposition 12.3.2., there are *val(f*)* arc-disjoint *s-t* paths in network *N*,* and hence, also in *N*, which implies that $val(f^*) \leq r$.

To obtain the reverse inequality, let $\{P_1, P_2, \ldots, P_r\}$ be the largest collection of arc-disjoint directed *s-t* paths in *N,* and consider the function $f: E_N \rightarrow R^+$ defined by

$$f(e) = \begin{cases} 1, & \text{if some path } P_i \text{ uses arc } e \\ 0, & \text{otherwise} \end{cases} .$$

Then *f* is a feasible flow in network *N,* with *val(f)* = *r.*
It follows that $val(f^*) \geq r$. □

Separating Sets and Cuts

Review from §5.3

Let *s* and *t* be distinct vertices in a graph *G.* An *s-t* **separating edge set** in *G* is a set of edges whose removal destroys all *s-t* paths in *G.*

Thus, an *s-t* separating edge set in *G* is an edge subset of E_G that contains at least one edge of every *s-t* path in *G.*

Definition: Let *s* and *t* be distinct vertices in a digraph *D.*
An *s-t* **separating arc set** in *D* is a set of arcs whose removal destroys all directed *s-t* paths in *D.*

Thus, an *s-t* separating arc set in *D* is an arc subset of E_D that contains at least one arc of every directed *s-t* path in digraph *D.*

Remark: For the degenerate case in which the original graph or digraph has no *s-t* paths, the empty set is regarded as an *s-t* separating set.

Separating Sets and Cuts

Proposition 12.3.4 Let N be an s-t network such that $cap(e) = 1$ for every arc e. Then the capacity of a minimum s-t cut in network N equals the minimum number of arcs in an s-t separating arc set in N.

Proof: Let $K^* = \langle V_s , V_t \rangle$ be a minimum s-t cut in network N, and let q be the minimum number of arcs in an s-t separating arc set in N.
Since K^* is an s-t cut, it is also an s-t separating arc set. Thus $cap(K^*) \geq q$.

To obtain the reverse inequality, let S be an s-t separating arc set in network N containing q arcs, and let R be the set of all vertices in N that are reachable from source s by a directed path that contains no arc from set S.

Then, by the definitions of arc set S and vertex set R, $t \notin R$, which means that $\langle R, V_N - R \rangle$ is an s-t cut.

Moreover, $\langle R, V_N - R \rangle \subseteq S$. Therefore

Separating Sets and Cuts

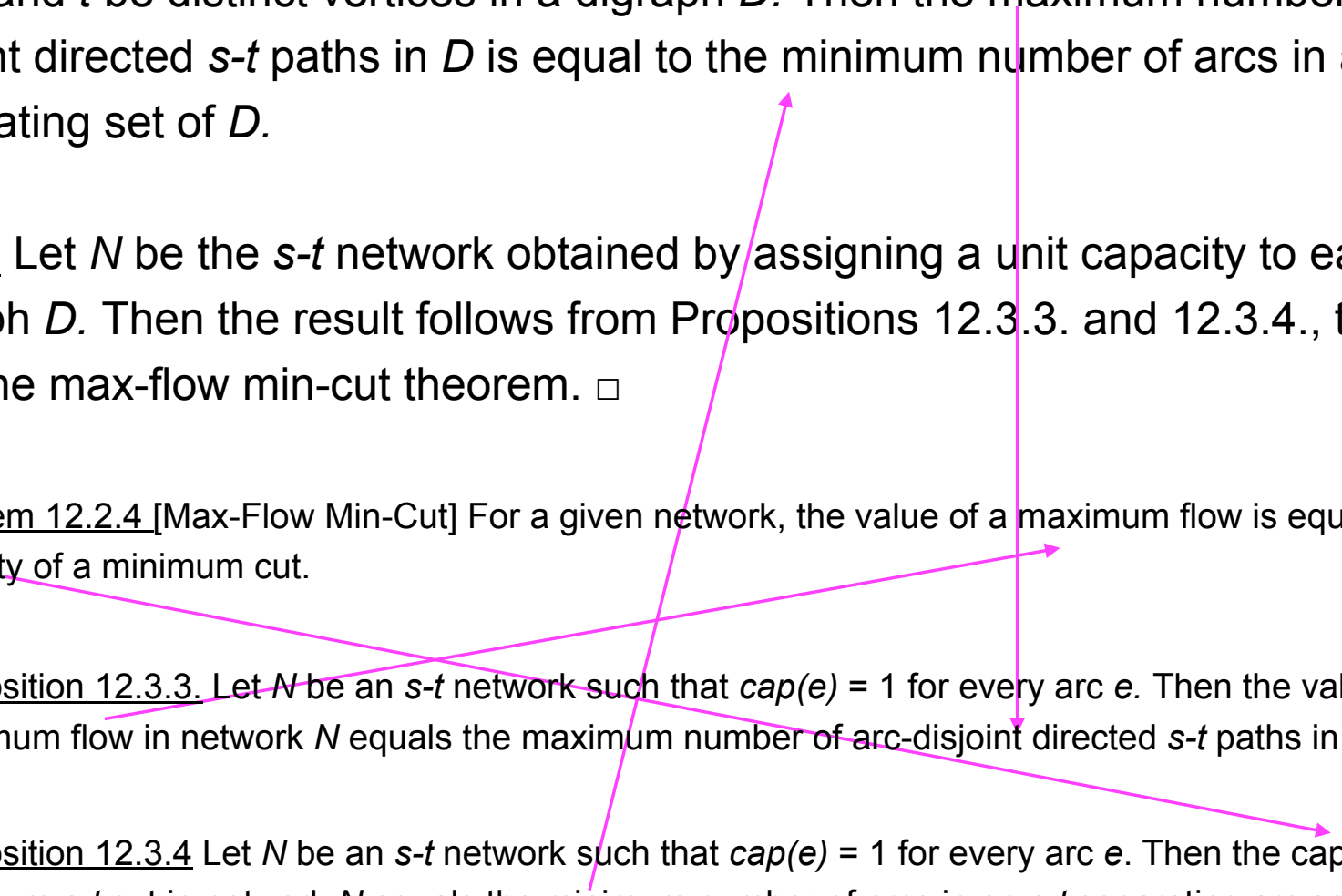
$$
\begin{aligned}
cap(K^*) &\leq cap\langle R, V_N - R\rangle && \text{since } K^* \text{ is a minimum } s-t \text{ cut} \\
&= \left|\langle R, V_N - R\rangle\right| && \text{since all capacities are 1} \\
&\leq |S| && \text{since } \langle R, V_N - R\rangle \subseteq S \\
&= q
\end{aligned}
$$

which completes the proof. □

Arc and Edge Versions of Menger's Theorem Revisited

Theorem 12.3.5 [Arc form of Menger's theorem]
Let *s* and *t* be distinct vertices in a digraph *D.* Then the maximum number of arc-disjoint directed *s-t* paths in *D* is equal to the minimum number of arcs in an *s-t* separating set of *D.*

Proof: Let *N* be the *s-t* network obtained by assigning a unit capacity to each arc of digraph *D.* Then the result follows from Propositions 12.3.3. and 12.3.4., together with the max-flow min-cut theorem. □



Theorem 12.2.4 [Max-Flow Min-Cut] For a given network, the value of a maximum flow is equal to the capacity of a minimum cut.

Proposition 12.3.3. Let *N* be an *s-t* network such that *cap(e)* = 1 for every arc *e.* Then the value of a maximum flow in network *N* equals the maximum number of arc-disjoint directed *s-t* paths in *N.*

Proposition 12.3.4 Let *N* be an *s-t* network such that *cap(e)* = 1 for every arc *e*. Then the capacity of a minimum *s-t* cut in network *N* equals the minimum number of arcs in an *s-t* separating arc set in *N.*

Metabolic Networks - Overview

There exist different levels of computational methods for describing metabolic networks:

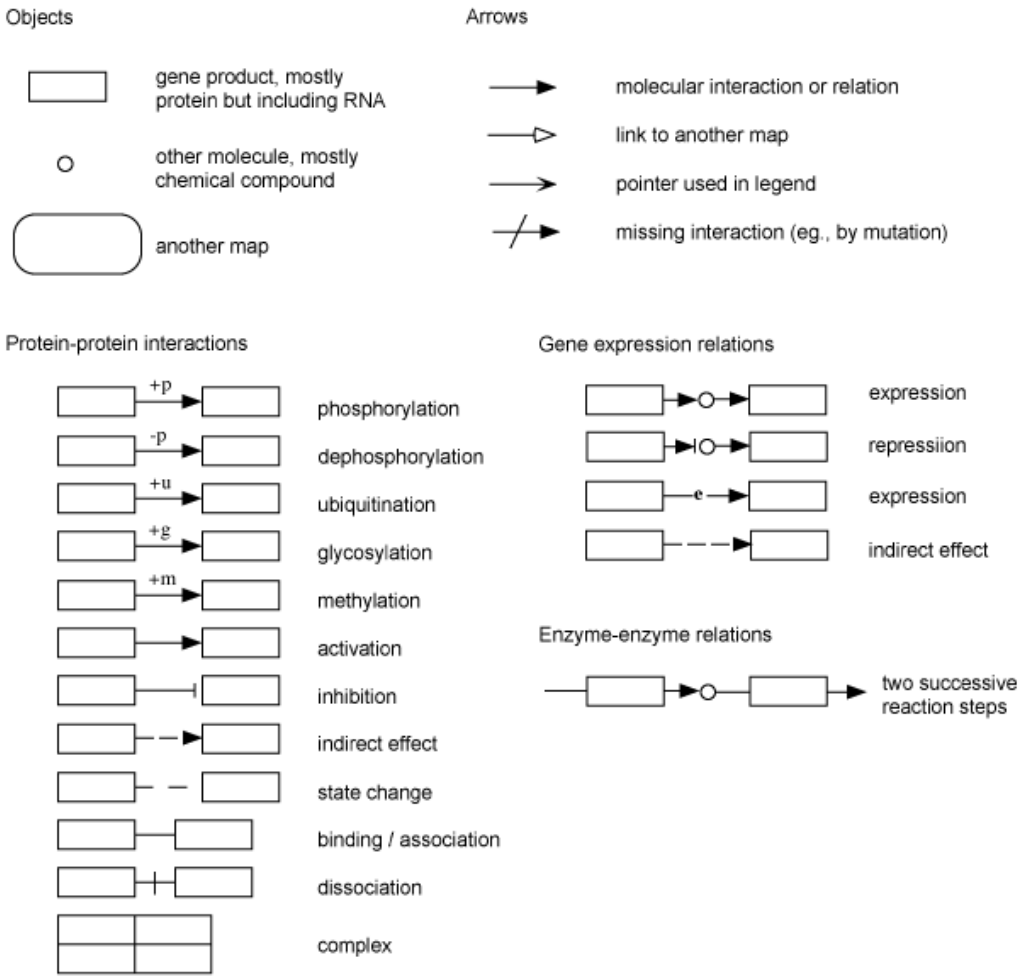
- stoichiometry/kinetics of classical biochemical **pathways** (glycolysis, TCA cycle, ...

- stoichiometric modelling (**flux balance analysis**): theoretical capabilities of an integrated cellular process, feasible metabolic flux distributions

- automatic decomposition of metabolic networks (elementary nodes, extreme pathways ...)

- **kinetic modelling** of coupled cellular pathways (E-Cell ...)
General problem: lack of kinetic information on the dynamics and regulation of cellular metabolism

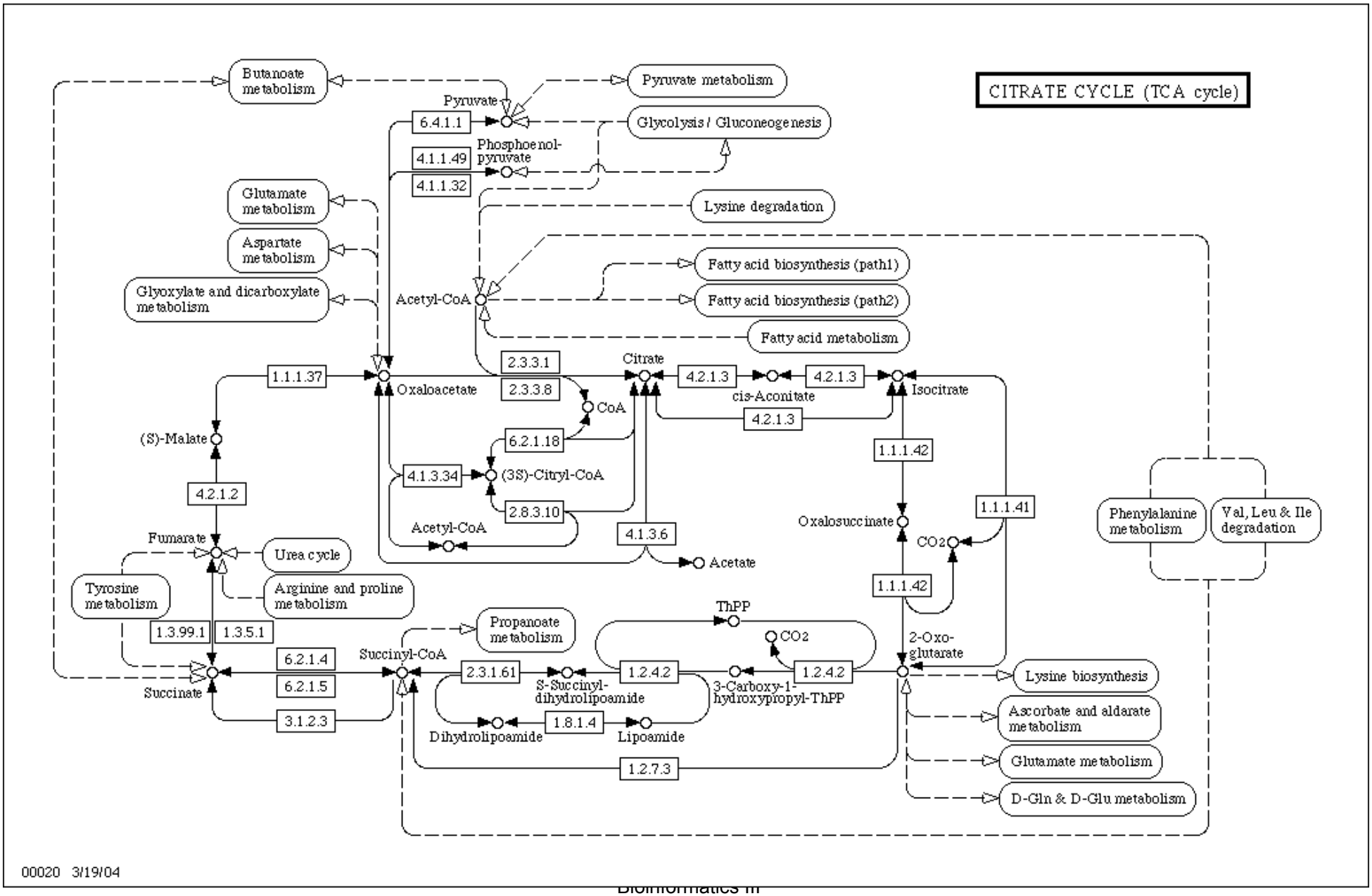
KEGG database



The KEGG PATHWAY database (http://www.genome. jp/kegg/pathway.html) is a collection of graphical diagrams (KEGG pathway maps) representing molecular interaction networks in various cellular processes. Each reference pathway is manually drawn and updated with the notation shown left.

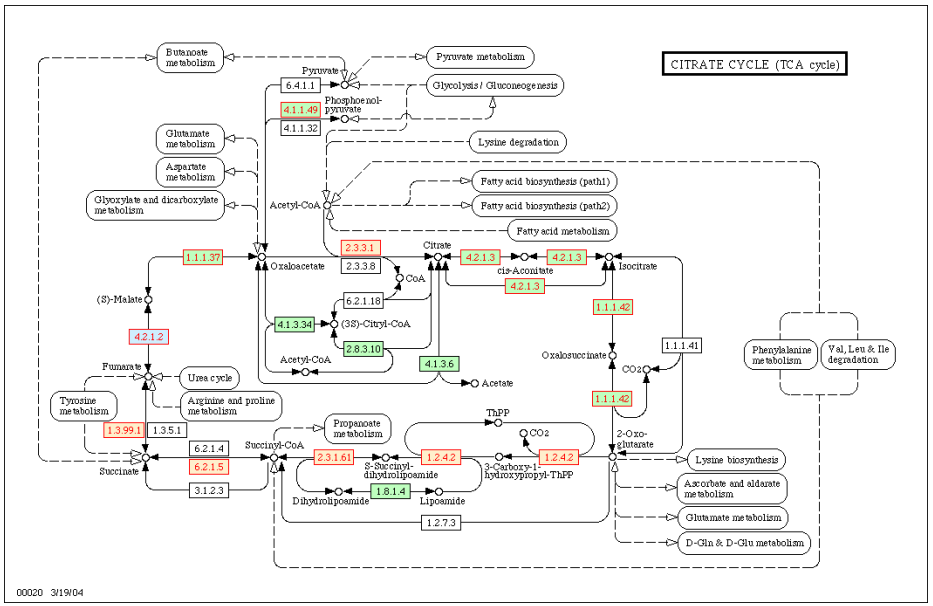
Organism-specific pathways (green-colored pathways) are computationally generated based on the KO assignment in individual genomes.

Citrate Cycle (TCA cycle) in E.coli

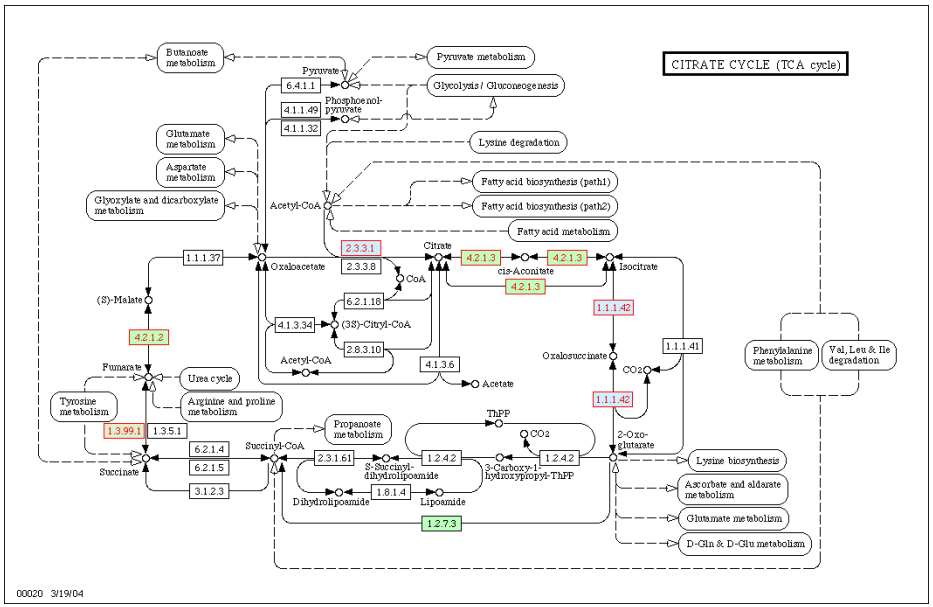


Citrate Cycle (TCA cycle) in different organisms

Citrate cycle (TCA cycle) - Escherichia coli K-12 MG1655



Citrate cycle (TCA cycle) - Helicobacter pylori 26695



Green/red: enzyme annotated in this organism

EcoCyc Database

E.coli genome contains 4.7 million DNA bases.

How can we characterize the functional complement of *E.coli* and according to what criteria can we compare the biochemical networks of two organisms?

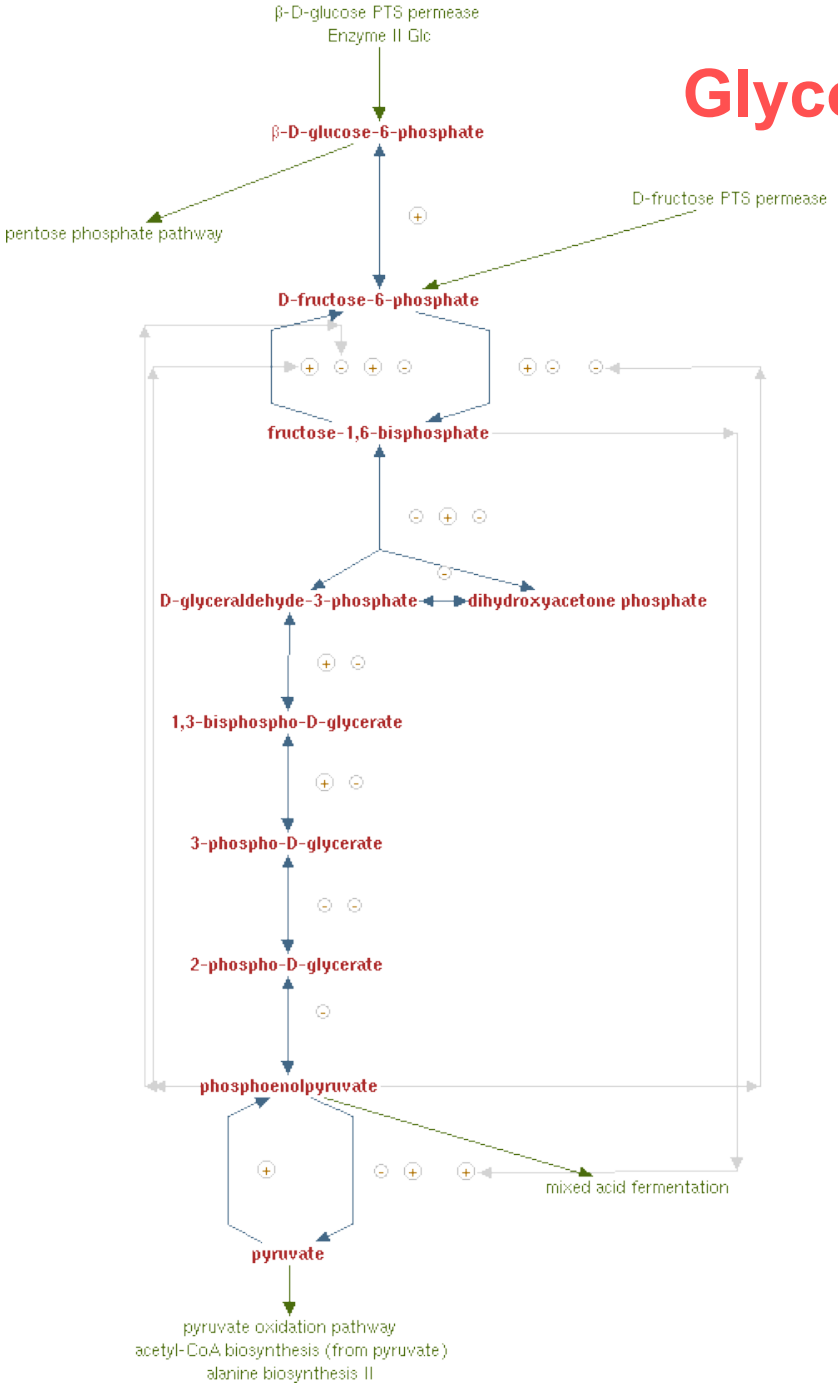
EcoCyc contains the metabolic map of *E.coli* defined as the set of all known pathways, reactions and enzymes of *E.coli* small-molecule metabolism.

Analyze

- the connectivity relationships of the metabolic network
- its partitioning into pathways
- enzyme activation and inhibition
- repetition and multiplicity of elements such as enzymes, reactions, and substrates.

Ouzonis, Karp, Genome Res. 10, 568 (2000)

Glycolysis in E.coli

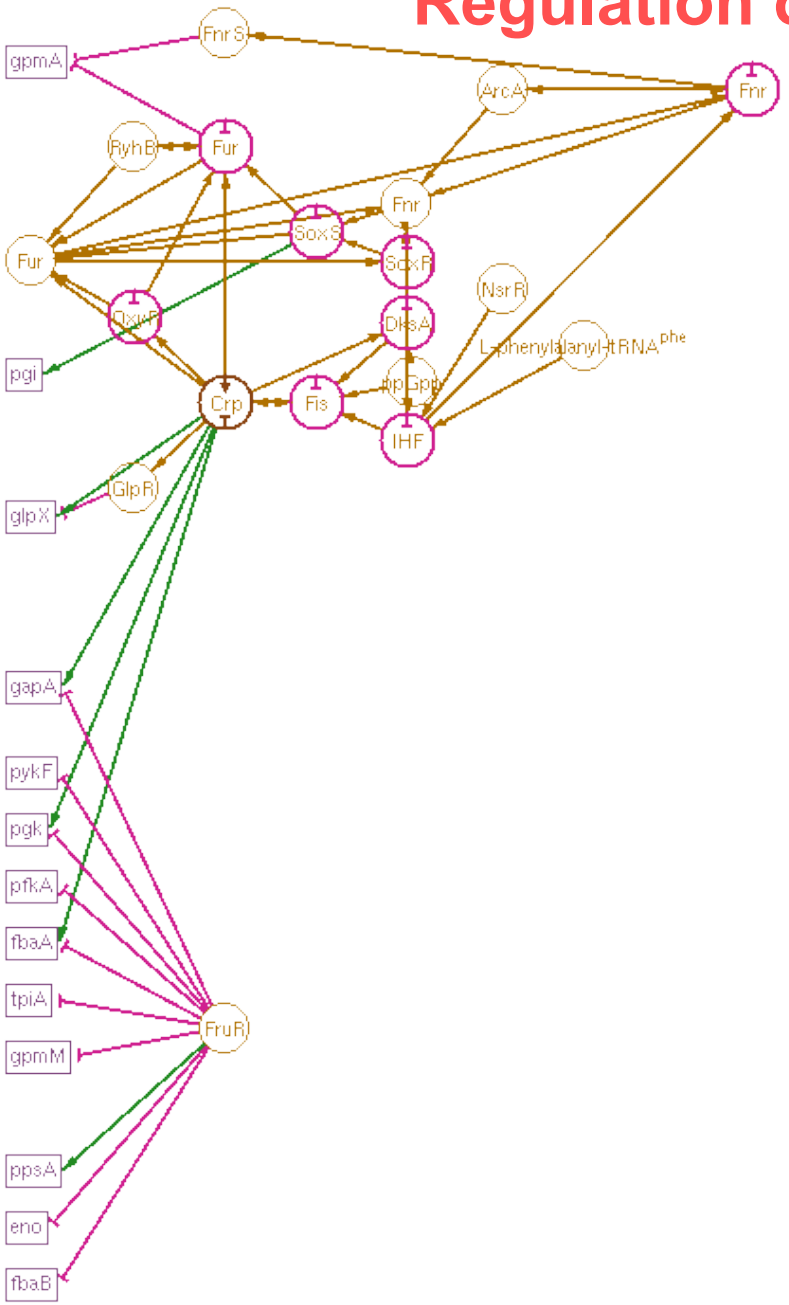


Blue arrows: biochemical reactions
clicking on arrow shows responsible enzyme

\+ and - : activation and inhibition of enzymes

www.ecocyc.org

Regulation of Glycolysis in E.coli



Boxed genes on the left are enzymes of glycolysis pathway

pgi: phosphoglucose isomerase
pgk: phosphoglycerate kinase
pfk: 6-phosphofructo kinase ...

Circled FruR, CRP etc. on the right : transcription factors

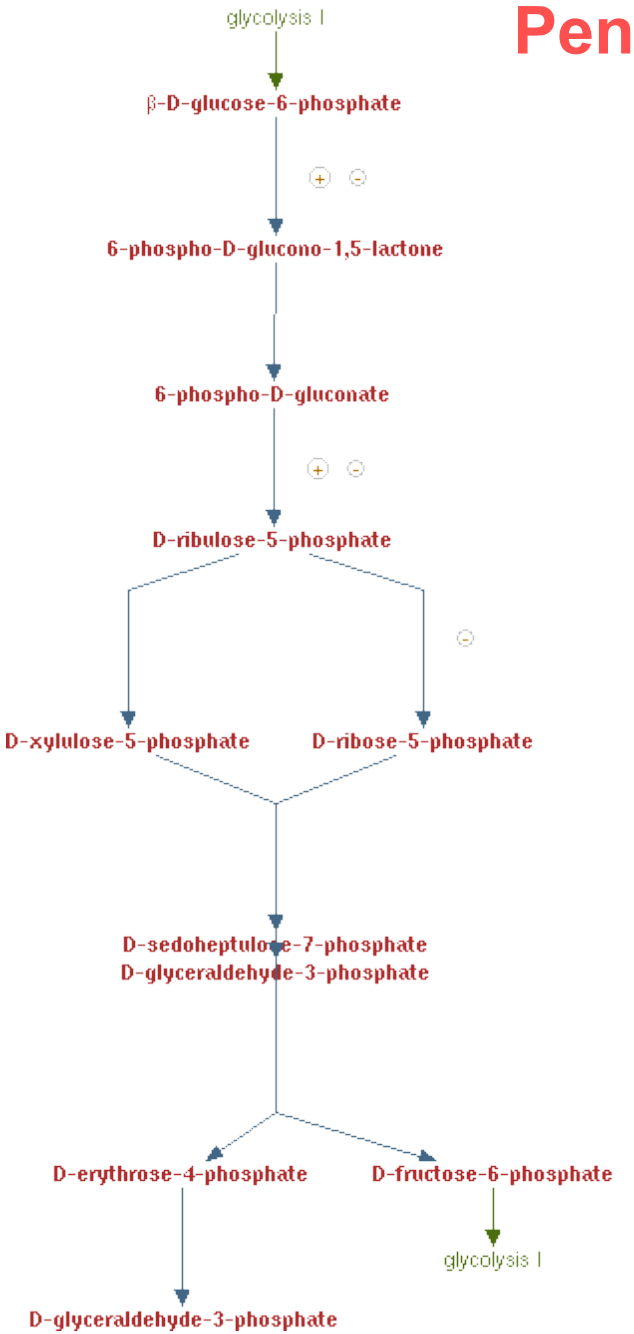
Green pointed arrows: activation of transcription;

Violet blunt arrow : repression;

Brown circle-ended arrow indicates that the factor can activate or repress, depending on circumstances.

www.ecocyc.org

Pentose Phosphate pathway

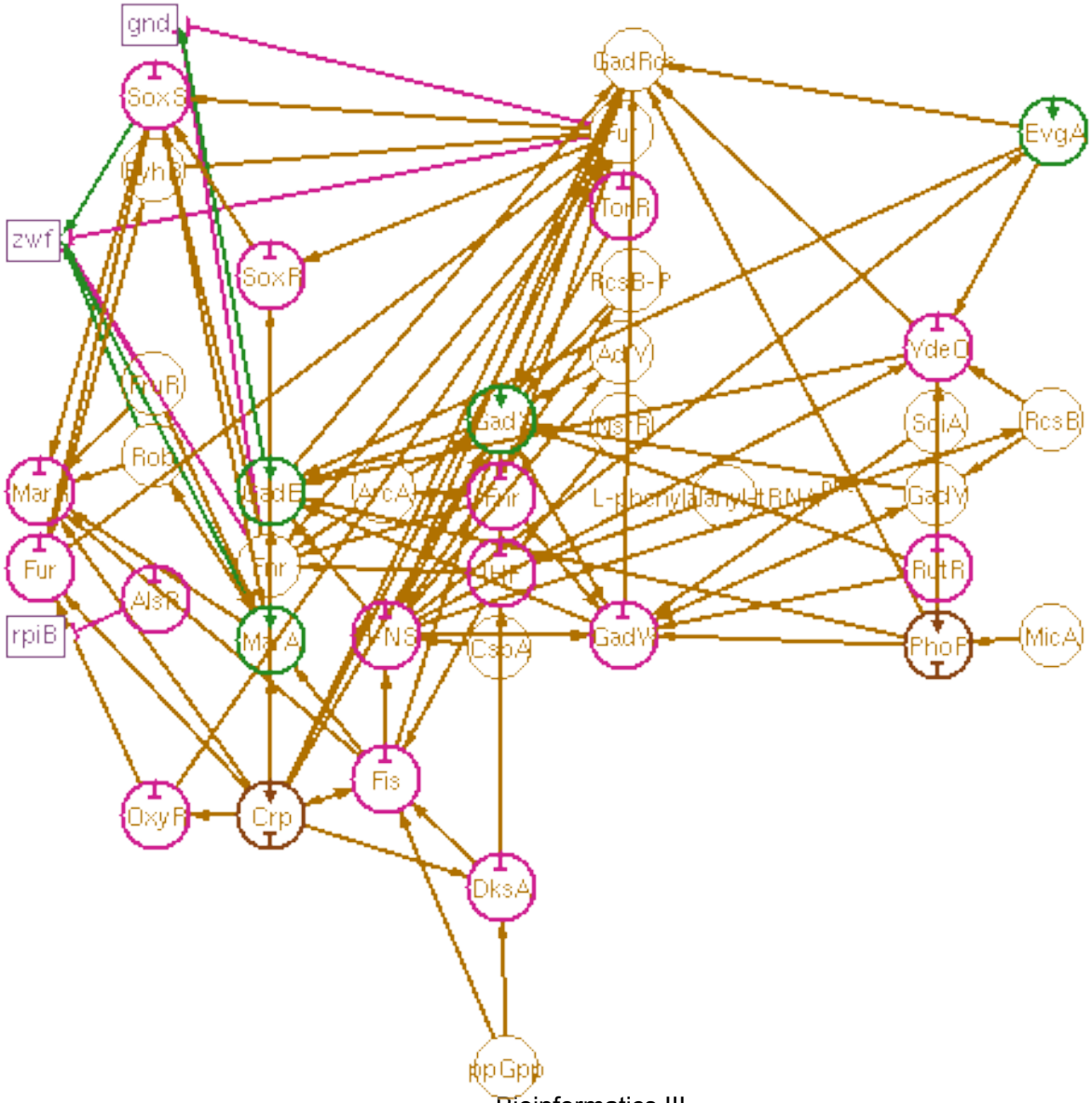


Blue arrows: biochemical reactions
clicking on arrow shows responsible enzyme

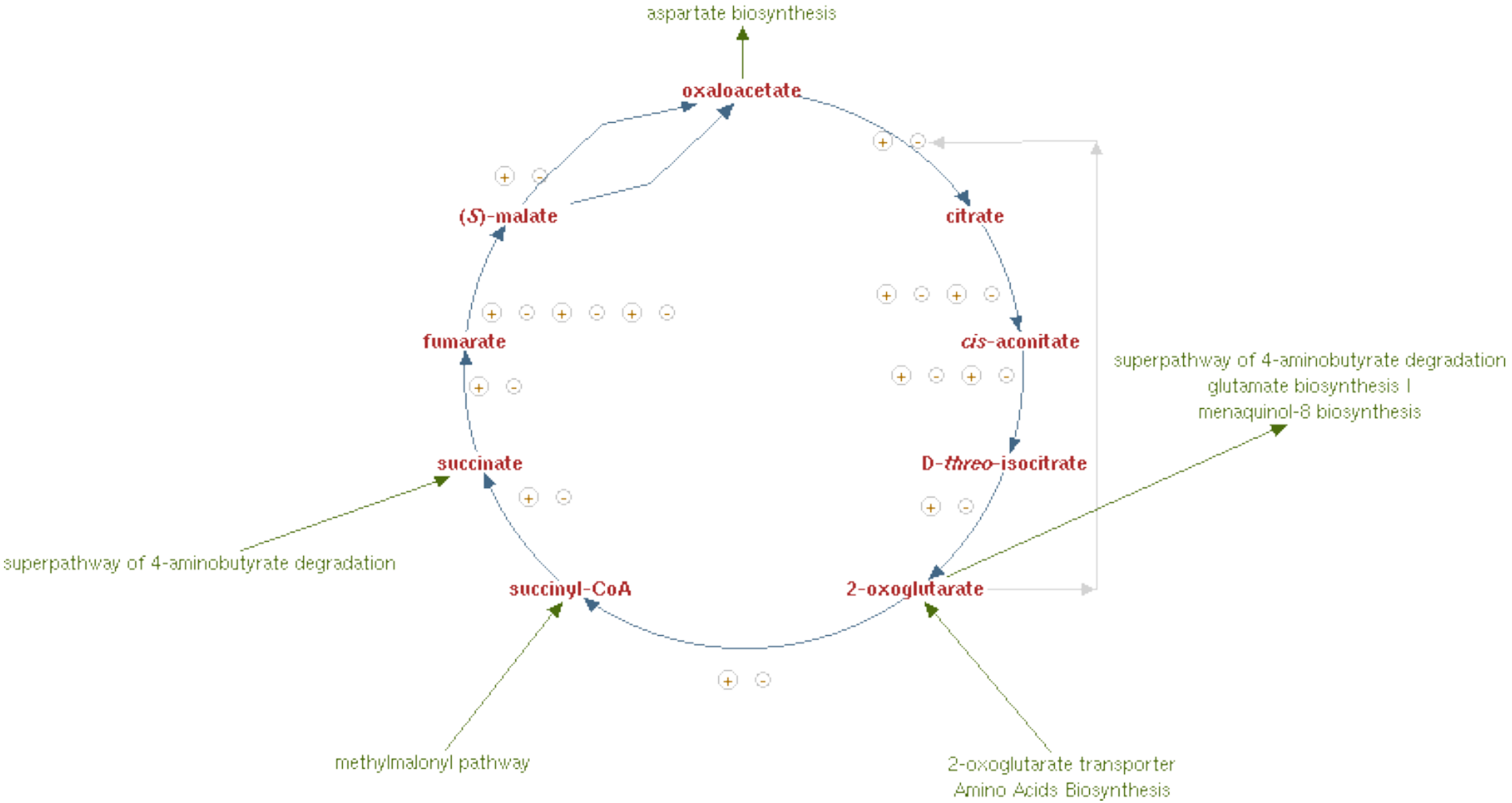
+ and - : activation and inhibition of enzymes

www.ecocyc.org

Regulation of Pentose Phosphate Pathway

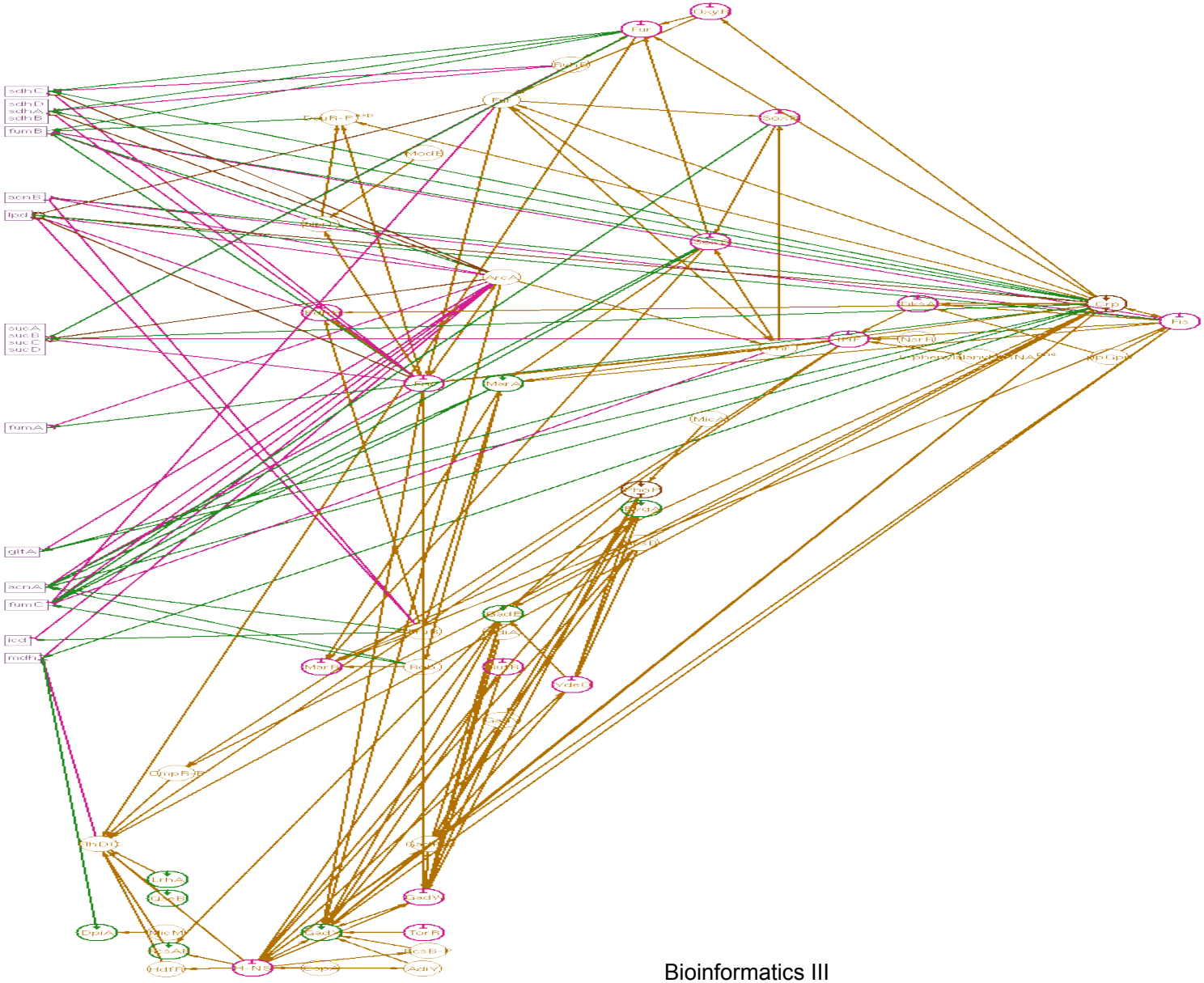


TCA cycle



www.ecocyc.org

Regulation of TCA cycle



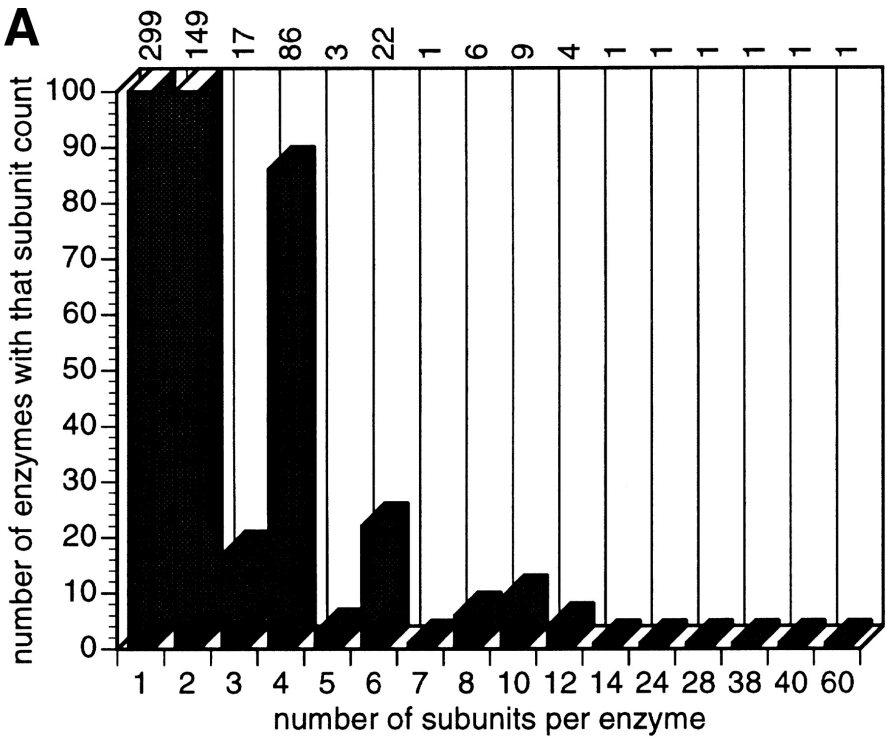
EcoCyc Analysis of *E.coli* Metabolism

In 2000, *E.coli* genome contained 4391 predicted genes, of which 4288 coded for proteins (4503 genes in Dec. 2011, 209 RNAs).

676 of these genes form 607 enzymes of the *E.coli* small-molecule metabolism.

Of those enzymes, 311 are protein complexes, 296 are monomers.

Organization of protein complexes.
Distribution of subunit counts for all EcoCyc protein complexes.
The predominance of monomers, dimers, and tetramers is obvious



Ouzonis, Karp, Genome Res. 10, 568 (2000)

Reactions

EcoCyc describes 905 metabolic reactions that are catalyzed by *E. coli.*
(1991 in Dec. 2011)

Of these reactions, 161 are not involved in small-molecule metabolism,
e.g. they participate in macromolecule metabolism
such as DNA replication and tRNA charging.

Of the remaining 744 reactions, 569 have been assigned to at least one pathway.

Ouzonis, Karp, Genome Res. 10, 568 (2000)

Reactions

The number of reactions (744) and the number of enzymes (607) differ ...

WHY??

(1) there is no one-to-one mapping between enzymes and reactions –
some enzymes catalyze multiple reactions, and some reactions are catalyzed by multiple enzymes.

(2) for some reactions known to be catalyzed by *E.coli*,
the enzyme has not yet been identified.

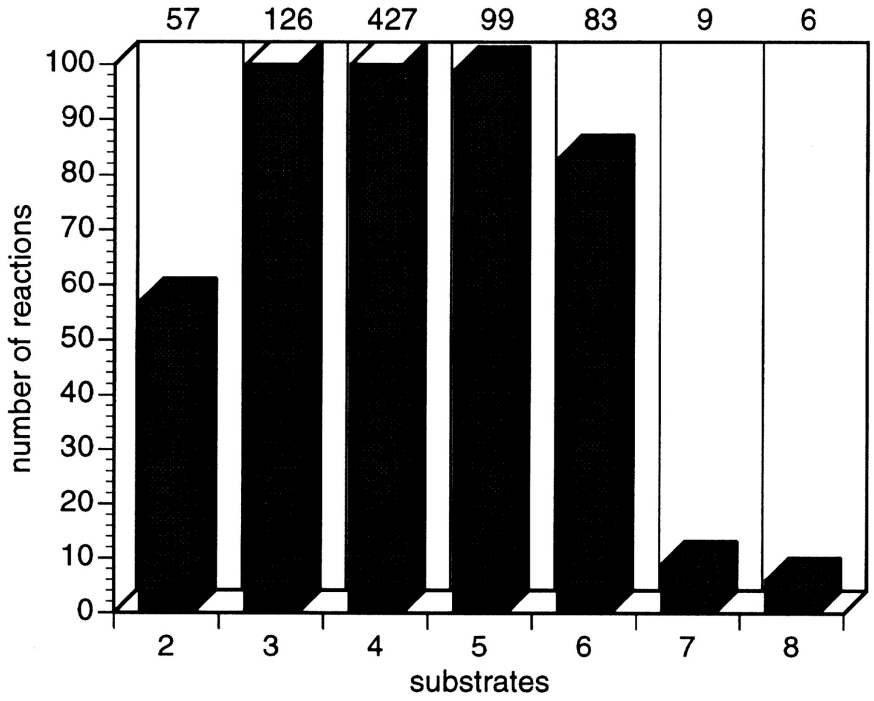
Ouzonis, Karp, Genome Res. 10, 568 (2000)

Compounds

The 744 reactions of *E.coli* small-molecule metabolism involve a total of 791 different substrates.

On average, each reaction contains 4.0 substrates, (think of A + B <-> C + D)

Number of reactions containing varying numbers of substrates (reactants plus products).



Ouzonis, Karp, Genome Res. 10, 568 (2000)

Compounds

Each distinct substrate occurs in an average of 2.1 reactions.

Table 1. Most Frequently Used Metabolites in *E. coli* Central Metabolism

Occurrence	Name of metabolite
205	H_2O
152	ATP
101	ADP
100	phosphate
89	pyrophosphate
66	NAD
60	NADH
54	CO_2
53	H^+
49	AMP
48	NH_3
48	NADP
45	NADPH
44	Coenzyme A
43	L-glutamate
41	pyruvate
29	acetyl-CoA
26	O_2
24	2-oxoglutarate
23	S-adenosyl-L-methionine
18	S-adenosyl-homocysteine
16	L-aspartate
16	L-glutamine
15	H_2O_2
14	glucose
13	glyceraldehyde-3-phosphate
13	THF
13	acetate
12	PRPP
12	[acyl carrier protein]
12	oxaloacetic acid
11	dihydroxy-acetone-phosphate
11	GDP
11	glucose-1-phosphate
11	UMP
10	e^-
10	phosphoenolpyruvate
10	acceptor
10	reduced acceptor
10	GTP
10	L-serine
10	fructose-6-phosphate
9	L-cysteine
9	reduced thioredoxin
9	oxidized thioredoxin
9	reduced glutathione
8	acyl-ACP
8	L-glycine
8	GMP
8	formate

Metabolites were used either as reactants or products.

Ouzonis, Karp, Genome Res. 10, 568 (2000)

11	glucose-1-phosphate

Pathways

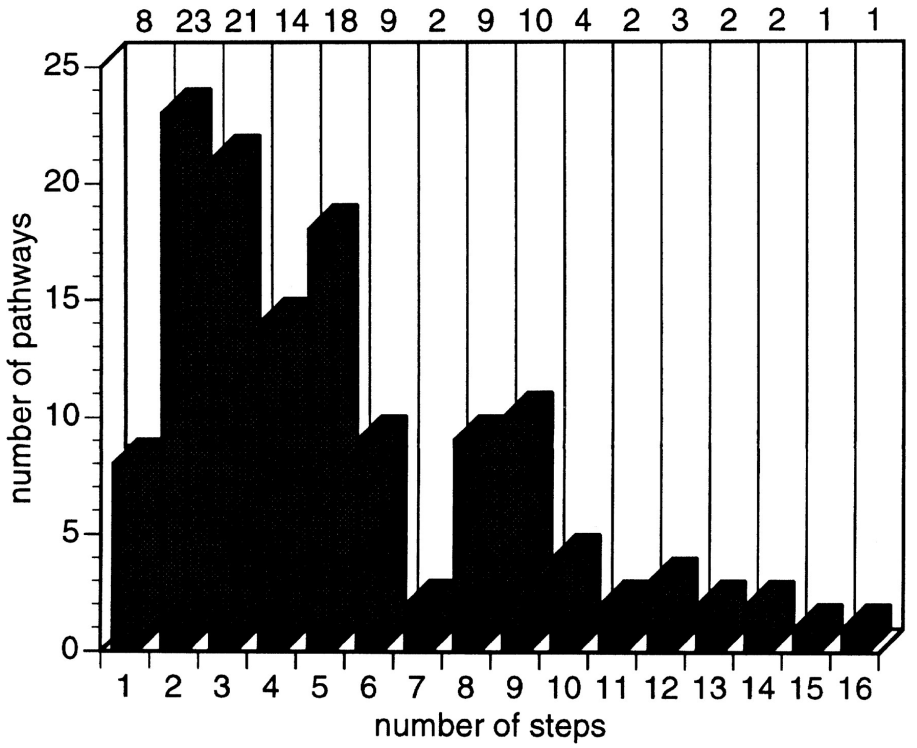
EcoCyc describes 131 pathways (347 in Dec. 2011):

- energy metabolism
- nucleotide and amino acid biosynthesis
- secondary metabolism

Pathways vary in length from a single reaction step to 16 steps with an average of 5.4 steps.

However, there is no precise biological definition of a pathway.

Length distribution of EcoCyc pathways



Ouzonis, Karp, Genome Res. 10, 568 (2000)

Enzyme Modulation

An `enzymatic reaction` is a type of EcoCyc object that represents the pairing of an enzyme with a reaction catalyzed by that enzyme.

EcoCyc contains extensive information on the modulation of *E.coli* enzymes with respect to particular reactions:
- activators and inhibitors of the enzyme,
- cofactors required by the enzyme
- alternative substrates that the enzyme will accept.

Of the 805 enzymatic-reaction objects within EcoCyc, physiologically relevant activators are known for 22, physiologically relevant inhibitors are known for 80.

327 (almost half) require a cofactor or prosthetic group.

Ouzonis, Karp, Genome Res. 10, 568 (2000)

Enzyme Modulation

Table 3. Most Common Modulators, cofactors, and prosthetic groups of *E. coli* enzymes and Their Frequencies

A. Modulators (activators and inhibitors)

Occurrence	Name of modulator	Activator	Inhibitor
35	Cu^{2+}		•
32	ATP	•	•
30	Zn^{2+}	•	•
29	AMP	•	•
26	ADP	•	•
25	EDTA	•	•
23	*p*-chloromercuribenzoate		•
23	pyrophosphate	•	•
22	K^{+}	•	•
22	phosphate	•	•
20	Hg^{2+}		•
20	Ca^{2+}	•	•
19	*N*-ethylmaleimide	•	•
16	NAD	•	•
16	iodoacetamide		•
16	coenzyme A		•
15	Co^{2+}	•	•
15	Mg^{2+}	•	•
15	phosphoenolpyruvate	•	•
14	Fe^{2+}	•	•
14	GTP	•	•
14	pyruvate	•	•
13	*p*-hydroxymercuribenzoate		•
13	NADP		•
12	Mn^{2+}	•	•

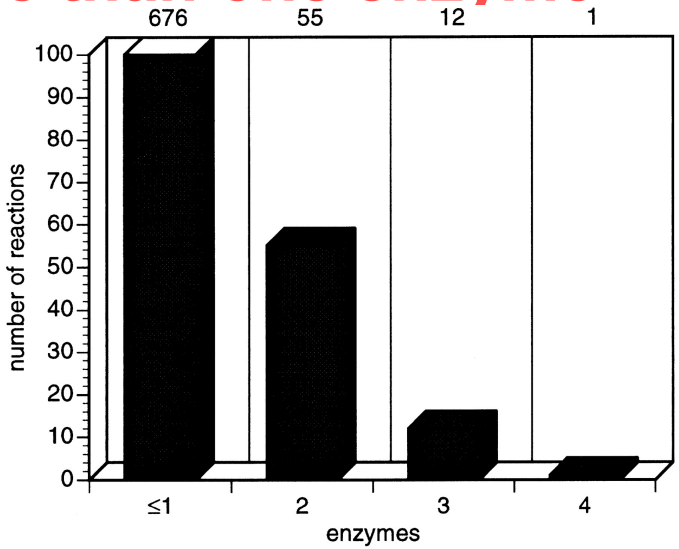
B. Cofactors and prosthetic groups

Occurrence	Name of compound	Cofactor	Prosthetic group
145	Mg^{2+}	•	•
48	pyridoxal 5'-phosphate	•	•
33	Mn^{2+}	•	
31	FAD	•	•
21	Fe^{2+}	•	•
18	Zn^{2+}	•	•
16	thiamine-pyrophosphate		•
11	FMN	•	•
10	Co^{2+}	•	
9	K^{+}	•	
6	Mo^{2+}		•
5	NAD	•	•
4	protoheme		•
4	Ni^{2+}	•	•
4	Ca^{2+}	•	
4	4Fe-4S center		•
3	NH_4^{+}	•	
3	pyruvate		•
3	siroheme		•
3	cytochrome *c*		•
2	heme C		•
2	B_{12}		•
2	NADP	•	
2	Cu^{2+}		•
2	biotin		•
2	Cd^{2+}	•	

Ouzonis, Karp, Genome Res. 10, 568 (2000)

Reactions catalyzed by more than one enzyme

Diagram showing the **number of reactions** that are **catalyzed** by **one or more enzymes**. Most reactions are catalyzed by one enzyme, some by two, and very few by more than two enzymes.



For 84 reactions, the corresponding enzyme is not yet encoded in EcoCyc.

What may be the reasons for isozyme redundancy?

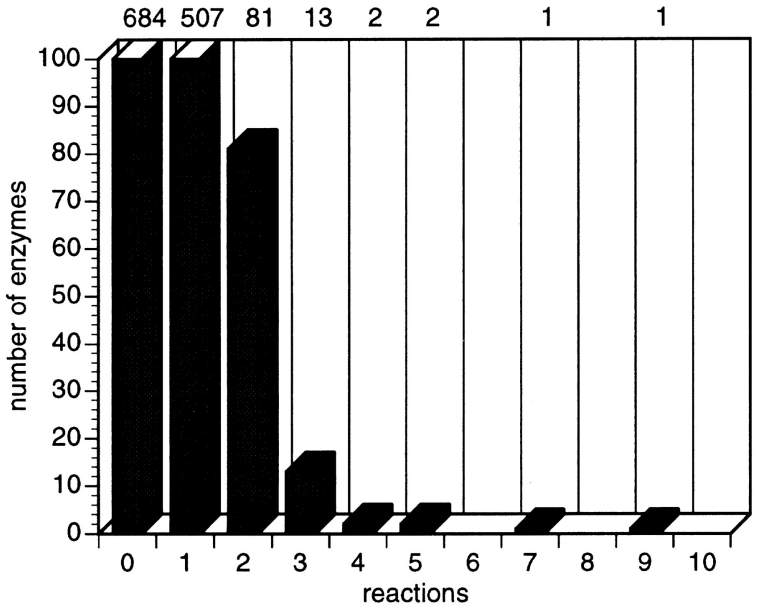
(1) the enzymes that catalyze the same reaction are **paralogs** (homologs) and have duplicated (or were obtained by horizontal gene transfer), acquiring some specificity but retaining the same mechanism (**divergence**)

(2) the reaction is easily „invented“; therefore, there is more than one protein family that is independently able to perform the catalysis (**convergence**).

Ouzonis, Karp, Genome Res. 10, 568 (2000)

Enzymes that catalyze more than one reaction

Of the 607 *E.coli* enzymes, 100 are multifunctional, either having the same active site and different substrate specificities or different active sites.

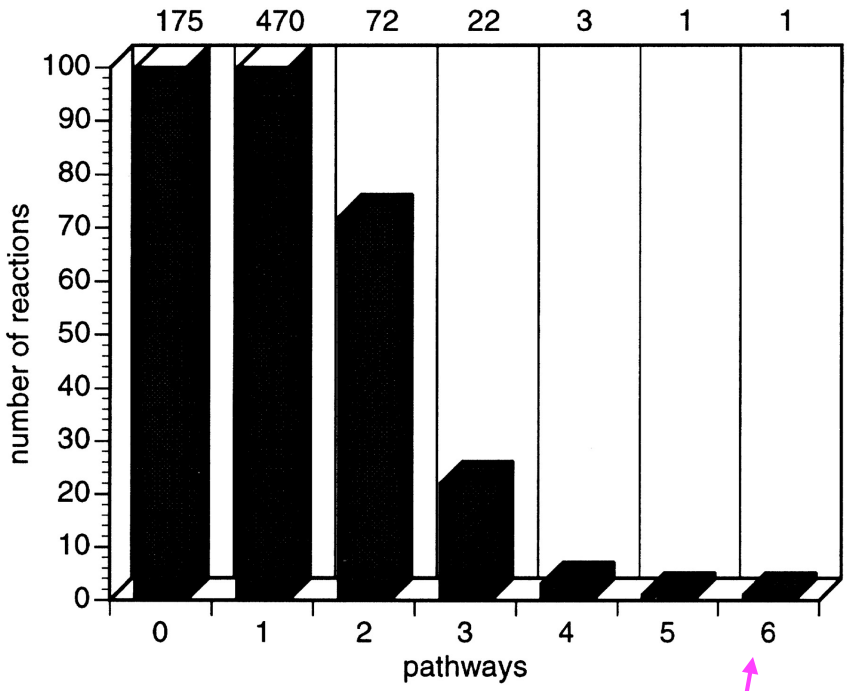


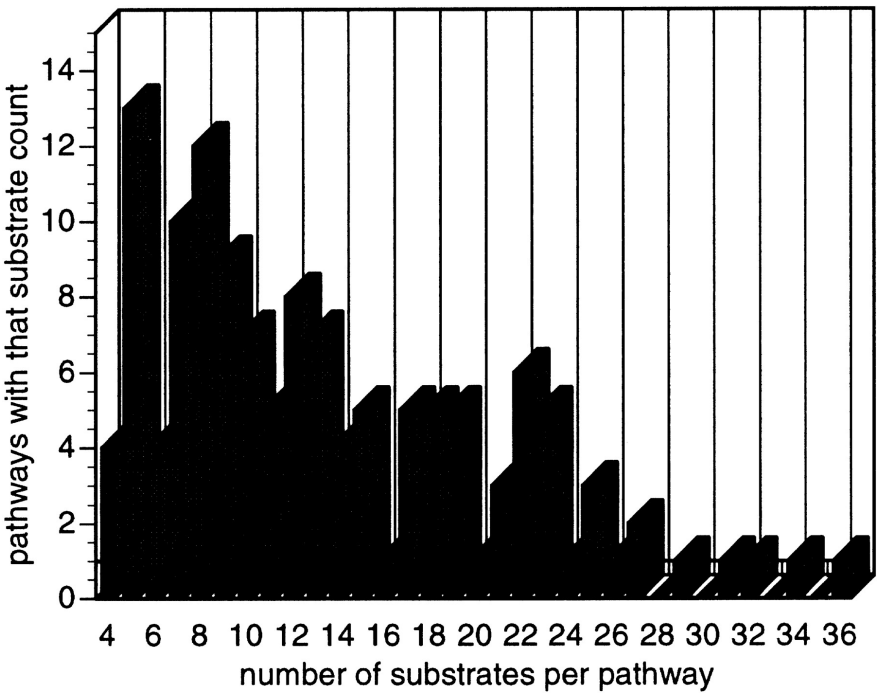
Number of enzymes that catalyze one or more reactions. Most enzymes catalyze one reaction; some are multifunctional.

The enzymes that catalyze 7 and 9 reactions are purine nucleoside phosphorylase and nucleoside diphosphate kinase.

Ouzonis, Karp, Genome Res. 10, 568 (2000)

Reactions participating in more than one pathway





The 99 reactions belonging to multiple pathways appear to be the **intersection points** in the complex network of chemical processes in the cell.

Ouzonis, Karp,
Genome Res. 10, 568 (2000)

E.g. the reaction present in 6 pathways corresponds to the reaction catalyzed by malate dehydrogenase, a central enzyme in cellular metabolism.