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 \rightarrow **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

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

in *N*, such that $f(e_i) < cap(e_i)$ for i=1, ..., 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 side: 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, <u>decrease</u> 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.



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

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

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

<u>Terminology</u> For a given *s-t* quasi-path

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

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.



<u>Definition</u>: 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 <i>e</i> is a backward arc.

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

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

<u>Terminology</u> 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, <u>disregarding conservation of flow</u>.

<u>Remark</u> 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

$$\Delta_{\mathcal{Q}} = \min_{\boldsymbol{\theta} \in \mathcal{Q}} \left\{ \Delta_{\boldsymbol{\theta}} \right\}$$

<u>Example</u> 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 $\Delta_{\Omega} = 1$.



flow augmentation

<u>Proposition 12.2.1</u> (Flow Augmentation) Let *f* be a 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

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

is a feasible flow in network N and $val(f) = val(f) + \Delta_Q$.

<u>Proof.</u> Clearly, $0 \le f'(e) \le cap(e)$, 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.

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.

$$+\Delta_{Q} \vee -\Delta_{Q} -\Delta_{Q} \vee -\Delta_{Q} +\Delta_{Q} \vee +\Delta_{Q} -\Delta_{Q} \vee +\Delta_{Q}$$

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,

$$Val(f') = \sum_{\theta \in Out(s)} f'(\theta) - \sum_{\theta \in In(s)} f'(\theta) = \Delta_{Q} + Val(f) \square$$

Max-Flow Min-Cut

<u>Theorem 12.2.3</u> [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*.

<u>Proof</u>: Necessity (\Rightarrow) Suppose that *f* is a maximum flow in network *N*. Then by Proposition 12.2.1, there is no *f*-augmenting path.

<u>Proposition 12.2.1</u> (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

$$f'(\theta) = \begin{cases} f(\theta) + \Delta_{\varrho}, & \text{if } \theta \text{ is a forward arc of } Q \\ f(\theta) - \Delta_{\varrho}, & \text{if } \theta \text{ is a backward arc of } Q \\ f(\theta) & \text{otherwise} \\ \text{ow in network } N \text{ and } val(f') = val(f) + \Delta Q. \end{cases}$$

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

is a feasible fl

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. \Box
```

Max-Flow Min-Cut

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

<u>Proof</u>: 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^*.
```

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.

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 $\Delta_{\rm e} > 0$.

<u>Terminology</u>: 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 $f(e_2) > 0$

<u>Remark</u> 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).

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.

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 2*M* iterations to obtain the maximum flow.



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 := 1While vertex set V_s does not contain sink t If there are usable arcs Let e be a usable arc whose labeled endpoint vhas 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 + 1Else 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 12.2.10 Iteration 2: val(f) = 4; augmenting path Q has $\Delta_Q = 2$.



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



3 12.2.11 Final iteration: $val(f) = 6 = cap(\{s, x, y, z, v\}, \{w, a, b, c, t\})$.

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

13. Lecture WS 2015/16

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*.

<u>Proof</u>. 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. \Box

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*.

<u>Proof</u>. 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.

13. Lecture WS 2015/16

Bioinformatics III

Determining the connectivity of a graph

By way of induction, assume that the assertion is true for m = k, for some $k \ge 1$, 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

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

<u>Proposition 12.3.3.</u> 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*.

<u>Proof</u>: 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

$$\sum_{\boldsymbol{\theta} \in ln(\boldsymbol{\nu})} \boldsymbol{f}^{*}(\boldsymbol{\theta}) = |ln(\boldsymbol{\nu})| = indegree(\boldsymbol{\nu})$$

Bioinformatics III

Basic properties of 0-1 networks

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

and $outdegree(s) - indegree(s) = val(f^*) = indegree(t) - outdegree(t)$ 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 $val(f^*) \le r$.

To obtain the reverse inequality, let $\{P_1, P_2, ..., P_r\}$ be the largest collection of arcdisjoint 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 $val(f^*) \ge r$. \Box

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*.

<u>Definition</u>: 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*.

<u>Remark</u>: 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

<u>Proposition 12.3.4</u> 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*.

<u>Proof</u>: 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 $cap(K^*) \ge q$.

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

 $cap(K^*) \le cap\langle R, V_n - R \rangle \quad \text{since } K^* \text{ is a minimum } s - t \text{ cut}$ $= |\langle R, V_n - R \rangle| \quad \text{since all capacities are 1}$ $\le |S| \quad \text{since } \langle R, V_n - R \rangle \subseteq S$ = q

which completes the proof. \Box

Arc and Edge Versions of Menger's Theorem Revisited

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

<u>Proof:</u> 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. \Box

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

<u>Proposition 12.3.3.</u> 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*.

<u>Proposition 12.3.4</u> 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*.

Idea – extreme pathways



A torch is directed at an open door and shines into a dark room ...

What area is lighted ?

Instead of marking all lighted points individually, it would be sufficient to characterize the "extreme rays" that go through the corners of the door.

The lighted area is the area between the extreme rays = linear combinations of the extreme rays.

Stoichiometric matrix - Flux Balance Analysis

Stoichiometric matrix S:

 $m \times n$ matrix with stochiometries of the *n* reactions as columns and participations of *m* metabolites as rows.

The stochiometric matrix is an important part of the *in silico* model.

With the matrix, the methods of extreme pathway and elementary mode analyses can be used to generate a unique set of pathways P1, P2, and P3 that allow to express all steady-state fluxes as linear combinations of P1 – P3.



Extreme Pathways

introduced into metabolic analysis by the lab of Bernard Palsson (Dept. of Bioengineering, UC San Diego). The publications of this lab are available at http://gcrg.ucsd.edu/publications/index.html

The extreme pathway technique is based on the stoichiometric matrix representation of metabolic networks.

All external fluxes are defined as pointing outwards.

Schilling, Letscher, Palsson, J. theor. Biol. 203, 229 (2000) 13. Lecture WS 2015/16



Mass	bala	nce co	onstra	ints						v ₂			Internal flux constraints
$\begin{bmatrix} -1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$	0 -1 1 0 0	0 1 -1 0 0	0 0 -1 1 0	0 0 1 -1 0 (S	$0 \\ 0 \\ -1 \\ 0 \\ 1 \\ \cdot v =$	-1 0 0 0 0	0 -1 0 0	0 0 -1 0	0 0 0 -1	$ \begin{array}{c} \upsilon_3\\ \upsilon_4\\ \upsilon_5\\ \upsilon_6\\ b_1\\ b_2\\ b_3\\ b_4 \end{array} $	=	0 0 0 0 0	$v_j \ge 0$, $j = 1,,6$ Exchange flux constraints $-\infty \le b_j \le +\infty$, $j = 1,,4$

Idea – extreme pathways



Edwards & Palsson PNAS 97, 5528 (2000)

Bioinformatics III