A multi-level and Parametric Approach to Max-Flow Analysis for Networked Manufacturing and Logistics

R. Bertin

J.-P. Bourrieres

LAP/GRAI University Bordeaux1-ENSEIRB- CNRS UMR 5131,

33405 Talence Cedex

FRANCE

Abstract: In this paper, an analytical formulation of the minimum-cut theorem is proposed for Max-Flow determination in networks modeled as weighted and directed acyclic s-t graphs with both parametric arc and node capacities. Such organizations are commonly found in manufacturing and logistics. A parametric Max-Flow approach avoiding redundant cuts is presented, then a recursive aggregation mechanism is provided to allow multi-level Max-Flow assessment in the case of complex networks.

Keywords: graph theory, networks flows, maximum flow, minimum cut, manufacturing.

1. Introduction

Max-Flow analysis in graphs is a generic issue with many applications in transportation, manufacturing and logistic networks. The literature provides many results based on numerical algorithms or analytical solving [1], [6].

Most of numerical approaches are derived from the famous Ford-Fulkerson algorithm [4] with the objective of minimizing algorithm's time, *i.e.* maximizing the size of graphs that can be analyzed in a reasonable time [13]. Such approaches quickly lead to Max-Flow computation but unfortunately do not provide any understanding of the structural factors which limit the flow and consequently cannot be used in network design or re-engineering problems as bottlenecks searching, capacities balancing, etc.

Analytical methods [3],[12],[14] are more suitable to network design problems since the result is expressed as a function of all arc and node capacities, which clearly depicts the effect of both network structure and local capacities on the global throughput.

The problem addressed in this paper is the determination of the maximum flow through an arc and node capacitated network under the assumption that the network is acyclic and moreover has only one source and one sink. Notice that such situations are common in the manufacturing context as, most of the time, the material flow enters an enterprise through a single input store and leaves it through a single output store.

Whatever the nature of the flow (material or data flow), networks generic activities are:

To transport the flow from a node to another.

To transform (adding value) the flow while crossing nodes.

Clearly, the transportation and transformation resources have limited performances which respectively define the arc and node capacities of the network.

In terms of graphs, the issue is to analyze the maximum-flow problem in weighted directed acyclic *s-t* graphs (*s-t wdag*) with both arc and node capacities. As justified above, the approach presented in this paper is analytical to simplify the exploitation of Max-Flow calculation for network design purpose. The organization of this paper is as follows. Sections 1 and 2 remind the basic notions and introduce the notations. Section 3 states the problem addressed. An analytical approach to Max-Flow calculation in an *s-t wdag* with arc capacities is presented in section 4. A *bi-vertex* concept is then introduced in section 5 to facilitate *s-t* cuts enumeration in the case of an *s-t wdag* with both arc and vertex capacities (section 6). Section 7 extends the previous results to the multi-level Max-Flow analysis of complex networks using recursive aggregation mechanisms. Finally, two simple examples are presented in section 8 showing how to apply the results provided in sections 6 and 7 to solve balancing problems in manufacturing networks.

2. Preliminaries

2.1. Directed graphs

Let V and E be respectively the vertex and edge set of a graph. A vertex will be noted u or v and e will be an edge. Let $N = card\ V$ be the number of vertices.

The basic notions on graphs are first reminded [2],[5]:

1. *Simple* graph: self-loops and multiple edges are not allowed.
2. *Connected graph*: it is possible to find a path, *i.e.* a sequence of edges and vertices from any vertex to any other.
3. *Digraph*: directed graph; vertices and directed edges are called respectively nodes and arcs, both terms being used in this paper. When necessary, the head vertex and the tail vertex of an arc are mentioned. In this case, an arc is noted as *arc(i,j)* with *i* the tail and *j* the head.
4. *Direct acyclic graph (dag)* : digraph without cycle. It can be found at least one source and one sink in a *dag*.
5. *Weighted dag (wdag)* : a real number *w(e)* is associated with each edge *e*. Here it is assumed that $w(e) > 0$.

2.2. Matrix Representation

Simple digraphs can classically be represented by square matrices [6]. Matrix representation is used in this article for the convenience of the exposition. From a computing point of view, adjacency lists or incidence tables might be preferred. Both representations can be interchanged without any difficulty.

The *adjacency matrix* is defined as a Boolean $(N \times N)$ matrix in which line *i* is associated with tail vertex *i* and column *j* with head vertex *j*. Entry value is *1* if there is an arc from tail *i* to head *j*, *0* otherwise. Diagonal entries are null since self loops are not allowed. Notice that if the digraph is a *dag*, it can be found a vertex ordering which leads to a triangular adjacency matrix, and reciprocally.

The *capacity matrix* of a *wdag* is similarly defined, just replacing each nonzero entry of the adjacency matrix by the weight associated with each arc. A zero in the capacity matrix denotes the absence of arc between the pair of vertices considered. For the convenience of the presentation, a zero might be replaced by a dot (.).

2.3. S-t Networks

This article focuses on a particular *dag* class so-called s-t networks. Let the following definitions be added:

S-t network: a *dag* with only one source vertex noted *s* and one sink vertex noted *t* (for 'target'). In adjacency and capacity matrices, the source relates to a null column and the sink to a null line.

Minimal connected network: a complete *s-t* network; all entries in the triangular part of adjacency and capacity matrices are positive. There are $N(N-1)/2$ arcs.

Series-parallel wdag: a weighted *s-t* network is series-parallel if it can be progressively reduced to a graph with only two vertices and one arc by applying the following transformations:

If two edges e_1 and e_2 are serial, replace them by one edge *e* with weight:

$w(e) = Min\ (w(e_1), w(e_2))$

If two edges e_1 and e_2 are parallel between two vertices, replace them by one edge *e* with weight:

$w(e) = w(e_1) + w(e_2)$

Series-parallel wdag's bring much simplification in Max-Flow calculation.

2.4. Flow in s-t Networks

A flow through edge *e* is a real value $f(e)$ as:

$0 \leq f(e) \leq w(e)$

In an *s-t* network, the Kirchhoff law is verified at any node except the source and the sink:

$$\sum_{e \in In(v)} f(e) = \sum_{e \in Out(v)} f(e) \qquad \text{for } v \in V\text{-}\{s,t\}$$

where $In(v)$ is the set of arcs incoming to *v* and $Out(v)$ the set of arcs outgoing from *v*.

Moreover, the output flow of the source equals the input flow of the sink:

$$\sum_{e \in In(t)} f(e) = \sum_{e \in Out(s)} f(s) = \varphi$$

Notice that φ can be considered as the flow through a virtual external arc from t to s.

2.5. Cuts and Cut Capacities

A cut in a graph is a partition into the vertex set [7],[8], [10],[11]. Let $(X, V\text{-}X)$ be a partition (or cut) into V where X is a subset of V, with $V \neq \varnothing$ and $X \neq \varnothing$. This cut also defines the set of arcs:

$$\bigcup_{u \in X, v \in (V-X)} arc(u, v)$$

whose tails are in X and heads are in V-X.

The capacity of cut $(X, V\text{-}X)$, noted $C(X, V\text{–}X)$ (for short $C(X)$ if there is no ambiguity) is defined by:

$$C(X, V-X) = \sum_{u \in X, v \in (V-X)} c(u, v) \tag{1}$$

with $c(u,v)$ the capacity of $arc(u,v)$.

In an s-t network, a cut $(X, V\text{-}X)$ with $s \in X$ and $t \in V\text{-}X$ is called an s-t cut.

2.6. Max-Flow Min-cut Theorem

A crucial theorem has been demonstrated by Ford and Fulkerson [4]:

If a flow of an s-t capacitated network has a value equal to the capacity of the minimum s-t cut, this flow is maximum.

3. Problem Statement

3.1. Working hypotheses

The aim of this paper is to determine the maximum flow in a weighted minimal connected network, *i.e.* a *wdag* with one source and one sink, moreover complete with $N(N\text{-}1)/2$ arcs. It can be found a vertex order leading to a standard graph form called *optimal ordered s-t* network [9] as shown on Figure 1.

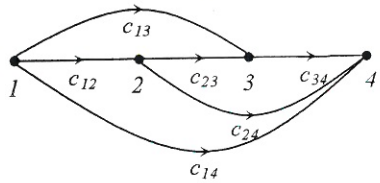


Figure 1: Optimal Ordered s-t Network

The corresponding capacity matrix has the full triangular form:

$$M = \begin{pmatrix} . & c_{12} & c_{13} & c_{14} \\ . & . & c_{23} & c_{24} \\ . & . & . & c_{34} \\ . & . & . & . \end{pmatrix}$$

Simple classical algorithms are reminded in sections 3.2 and 3.3 to check that these working hypothesis are fulfilled.

3.2. Checking Dags

The following algorithm [5] can be used to detect a directed cycle in a digraph.

Algorithm for checking dags

Input: adjacency or capacity matrix

Output: cycle detection

Repeat until last column

If column null

Then delete column, delete line with same number

Next

If last column is null the graph is a *dag*.

Else graph is not a *dag*.

3.4. Graph Ordering

A *dag* or *wdag* can be ordered using simple algorithms. The following algorithm [5] consists in modifying the previous one, both verifying that the graph is acyclic and ordering the nodes:

Algorithm for graph ordering

Input: adjacency or capacity matrix

Output: ordered dag

Repeat until no more column and line.

Step 1. Search for null columns, note vertices. When finished, allocate all vertices found to a new level. If no null column found, graph is not a dag, Stop.

Step 2. Delete lines associated with vertices provided by Step 1.

Next

Notice that if the matrix is given triangular, the graph is initially ordered. When the graph is complete *i.e.* the upper part of the triangular matrix is full, there is only one node per level. The graph is called optimal connected and ordered network.

4. Min-cut Algorithms for *s-t* Networks Without Node Capacity

According to Ford-Fulkerson's theorem, the parametric Max-Flow problem is solved by searching the minimal *s-t* cut. Clearly, *s-t* cut capacities are provided by enumerating all cuts in the *s-t* network. Since the source must belong to X and the sink to V-X, the maximal number of cuts is the number of combinations among N-2 vertices, *i.e*:

$$C_{N-2}^{0}+C_{N-2}^{1}+\ldots+C_{N-2}^{N-3}+C_{N-2}^{N-2}=2^{N-2} \tag{2}$$

The issue is then to identify the capacity $C(X,V\text{-}X)$ of any possible partition of V according to definition (1). In the next sections, three alternative approaches are stated for enumerating the cuts and calculating their capacity.

4.1. Matrix Calculus

Capacity $C(X,V\text{-}X)$ is a function of entries $c(i,j)$ of capacity matrix M as:

$$C(X,\ V-X)=\sum_{i\in(X)}\sum_{j\in(V-X)}c(i,j) \tag{3}$$

Assuming an ordering with $i=1$ as the source and $i=N$ as the sink, each tail $i = 1,\ldots,N\text{-}1$ is in X and each head $j = 2,..,N$ is in V-X with $i \neq j$.

Among all cuts, cut number k $(k=0,\ \ldots,\ 2^{N-2}\text{-}1)$ is defined by set I_k of tails in X and set J_k of heads in V-X. The capacity of cut k is then:

$$C_k=\sum_{i\in Ik}\sum_{j\in Jk}c(i,j) \tag{4}$$

To compute C_k easily from matrix M the following expression is introduced :

$$C_k=P_k\ M\ (\overline{P}_k)^T \qquad k=\{0,\ \ldots,\ 2^{N-2}\text{-}1\} \tag{5}$$

where P_k and $\overline{P}_k$ are binary line matrices defined as follows:

1. Entries of P_k are from left to right associated with vertex *1* to N.
2. Any entry value of P_k is *1* when the corresponding vertex belongs to I_k otherwise the entry is *0*.
3. $\overline{P}_k$ is obtained by complementing the Boolean entries of P_k.

Notice that the form of P_k is $P_k = (1 \underbrace{x\,x\,x\,x\,x}_{\text{binary code of } k}\ 0)$.

In other words, P_k is the binary code of $2^{N-1} + 2k$ using N-bit format.

Considering for instance a graph with $N=5$ vertices, the cut number $k = 4$ is identified by matrices $P_4 = (11000)$ and $\overline{P}_4 = (00111)$. P_4 is the code of $2^4 + 2.4 = 24$ using *5*-bit format and defines partition $X = \{1,2\}$, $V\text{-}X = \{3,4,5\}$.

All partitions are generated by incrementing k from *0* to 2^{N-2}-*1* and coding P_k and $\overline{P}_k$ accordingly.

Consequently, a method for generating all *s-t* cuts is to pre-calculate the beam of matrices $\{P_k ; k=0,\ldots 2^{N-2}-1\}$ then calculate the cuts using relation (5).

As an important remark, if matrix M is defined literally (*i.e* the entries are parametric capacities instead of numerical capacities), then the cuts of an *s-t* network with N nodes are identified once and for all. The result is generic and can be instantiated on any particular case.

Algorithm #1 for cut enumeration

Input : capacity matrix M

Output : *s-t* cuts

For $k = 0$ to $2^{N-2} - 1$

Encode P_k as the code of $2^{N-1} + 2k$ using N-bit format

Compute $C_k = P_k\, M\, (\overline{P}_k)^T$

End

4.2. Set Calculus

Here is stated another approach using set theory. It is assumed that the *s-t* network is ordered so that the capacity matrix is triangular.

Let $I(i)$ be the set of capacities on line i of the capacity matrix. Let $J(j)$ be the set of capacities on column j of the capacity matrix. For any partition $(X, V\text{-}X)$, it is clear that:

$$\bigcup_{i\in X} I(i) = \bigcup_{i\in X, j\in V} c(i,j) \qquad \bigcup_{j\in X} J(j) = \bigcup_{i\in V, j\in X} c(i,j)$$

According to set theory, the following relation is true :

$$\bigcup_{i\in X, j\in (V-X)} c(i,j) = \bigcup_{i\in X} I(i) - \bigcup_{i\in X} I(i) \cap \bigcup_{j\in X} J(j) \qquad (6)$$

Proof:

$$\bigcup_{i\in X, j\in (V-X)} c(i,j) = \bigcup_{i\in X, j\in V} c(i,j) - \bigcup_{i\in X, j\in X} c(i,j)$$

Moreover, it is clear that :

$$\bigcup_{i\in X,\, j\in X} c(i,j) = \bigcup_{i\in X,\, j\in V} c(i,j) \cap \bigcup_{i\in V,\, j\in X} c(i,j)$$

Combining the two previous relations yields to (6).

Having the entries of the capacity matrix with $i \in X$ and $j \in (V\text{-}X)$, the capacity of cut $(X,V\text{-}X)$ is obtained using (6) and (1).

Min-cut algorithm #2

Input: capacity matrix

Output: s-t cuts

Step 1 Select partition X, which defines set of lines $\bigcup_{i\in X} I(i)$.

Step 2 Delete other lines.

Step 3 Delete the columns corresponding to the nodes of X, these columns define set $\bigcup_{j\in X} J(j)$.

Step 4 Find cut capacity by summing the remaining entries which define set $\bigcup_{i\in X,\, j\in (V-X)} c(i,j)$.

Step 5 Continue till all 2^{N-2} partitions are done.

4.3. Iterative Combinatory Approach

A third way of enumerating *s-t* cuts is here provided by an iterative method presented through an example with dimension $N = 6$. The capacity matrix is as follows:

$$M = \begin{pmatrix} . & c_{12} & c_{13} & c_{14} & c_{15} & c_{16} \\ . & . & c_{23} & c_{24} & c_{25} & c_{26} \\ . & . & . & c_{34} & c_{35} & c_{36} \\ . & . & . & . & c_{45} & c_{46} \\ . & . & . & . & . & c_{56} \\ . & . & . & . & . & . \end{pmatrix}$$

Vertex set is $V = \{1,2,3,4,5,6\}$. There are $2^{N-2} = 16$ cuts.

Let Γ_i be the sum of capacities on line i $(i=1,..,N-1)$ of matrix M, *i.e.* the output capacity of node i. Notice that Γ_i is also the capacity of the cut defined by node i alone in X. Consequently:

$$\Gamma_i = \sum_j c(i,j) = C(\{i\}; V - \{i\}) \tag{7}$$

While enumerating the *s-t* cuts, the source $\{1\}$ is always in X and the sink $\{N\}$ always in $V\text{-}X$. Here the capacities of the cuts $C(X, V\text{-}X)$, for short $C(X)$, are:

C({1}), C({1,2}), C({1,3}), C({1,4}), C({1,5}), C({1,2,3}), C({1,2,4)}, C({1,2,5}), C({1,3,4}), C({1,3,5}), C({1,4,5}), C({1,2,3,4}), C({1,2,3,5}), C({1,3,4,5}), C({1,2,4,5}), C({1,2,3,4,5}).

Now it will be shown that these *s-t* cut capacities are linked to the capacities Γ_i introduced above.

Let two independent subsets (X,Y) of V be considered with the source in X and $Y = \{y\}$ a singleton set (only one node in Y). Let moreover $C_m(X,Y)$ be the sum of all mutual capacities between X and Y, that is from any node in X to singleton Y and vice versa. The following relationship is true:

$C(X \cup Y) = C(X) + \Gamma_y - C_m(X,Y)$ (8)

Proof:

C(X,V-X) = C(X,V-X-Y) + C(X,Y)

$C(Y,V-Y) = C(Y,V-X-Y) + C(Y,X)$

Adding the two last formulae leads to:

$C(X,V-X) + C(Y,V-Y) =$

$C(X,V-X-Y) + C(Y,V-X-Y) + C_m(X,Y)$

Moreover

C(X ∪ Y, V-X-Y) = C(X,V-X-Y) + C(Y,V-X-Y)

which finally yields to:

$C(X \cup Y, V-X-Y) = C(X,V-X) + C(Y,V-Y) - Cm(X,Y)$

for short C(X ∪ Y)= C(X) + Γ_y – Cm(X,Y)

Relation (8) provides an iterative method to enumerate all s-t cuts, starting with X = {1}, then successively adding one node to X.

Algorithm #3 for cut enumeration

The method is here presented for N=6.

Step 1 add line capacities:

$\Gamma_1 = c_{12} + c_{13} + c_{14} + c_{15} + c_{16}$

$\Gamma_2 = c_{23} + c_{24} + c_{25} + c_{26}$

$\Gamma_3 = c_{34} + c_{35} + c_{36}$

$\Gamma_4 = c_{45} + c_{46}$

$\Gamma_5 = c_{56}$

Step 2 Calculate the 16 *s-t* cut capacities iteratively:

$C(\{1\}) = \Gamma_1$

$C(\{1\} \cup \{2\}) = C(\{1\}) + \Gamma_2 - C_m(\{1\};\{2\})$

etc.

$C(\{1,2\} \cup \{3\}) = C(\{1,2\}) + \Gamma_3 - C_m(\{1,2\};\{3\})$

etc.

$C(\{1,2,4\} \cup \{3\}) = C(\{1,2,4\}) + \Gamma_3 - C_m(\{1,2,4\};\{3\})$

etc.

$C(\{1,2,4,5\} \cup \{3\}) = C(\{1,2,4,5\}) + \Gamma_3 - C_m(\{1,2,4,5\};\{3\})$

Relation (8) is of major interest to detect redundant cuts in case of non complete graphs, in particular series-parallel graphs, or graphs with bridges as shown in next section. Assuming for example $C_m(\{1\};\{3\}) = 0$, then $C(\{1,3\}) = C(\{1\}) + \Gamma_3$. The consequence is that capacity $C(\{1,3\})$ is greater than $C(\{1\})$ thus cannot be the min cut.

5. Bi-vertex Concept for Digraphs with Bridges

As a preliminary to the Max-Flow problem solving for networks with both arc and node capacities, the purpose is here to detect the redundancies due to the existence of bridges in a graph (Figure 2).

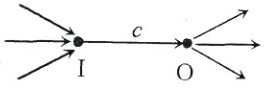
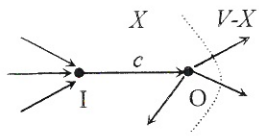


Figure 2. Bi-vertex

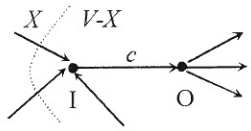
Let I and O be the Input and Output nodes of a bridge in a digraph, *i.e.* the tail and head vertices of the bridge. The capacity of the bridge is c. In this section, it is called *bi-vertex* the triplet composed of vertex I, vertex O and the directed bridge from I to O.

Remark that any *s-t* cut $(X, V\text{-}X)$ involving a definite bi-vertex is of four types:

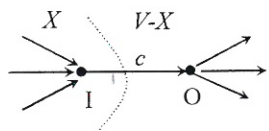
1. The bi-vertex is in X.



2. The bi-vertex is in $V\text{-}X$.



3. I is in X whereas O is in V-X.



4. is in X and I is in $V\text{-}X$: such cuts are redundant.

The redundancies mentioned in case iv are demonstrated below.

Proof:

Let case ii be considered. The cut capacity is $C(X)$ and the bi-vertex is in $V\text{-}X$.

Let now transfer node O from $V\text{-}X$ to X. The new situation is as case iv with cut $(X \cup \{O\}, V\text{-}X\text{-}\{O\})$.

According to relation (8), the cut capacity is :

$$C(X \cup \{O\}) = C(X) + \Gamma_O - C_m(X, \{O\})$$

with mutual capacity

$$C_m(X, \{O\}) = C(X, \{O\}) + C(\{O\}, X)$$

Here it is clear that:

$$C(X, \{O\}) = 0 \qquad C(\{O\}, X) \leq \Gamma_O$$

and consequently:

$$\Gamma_O - C_m(X, \{O\}) \geq 0$$

It finally can be seen that the cut capacity in case iv is greater than or equal to the cut capacity in case ii. Therefore cuts in case iv are redundant with regard to cuts in case ii.

6. Enumerating non Redundant Cuts for *s-t* Networks with arc and Node Capacities

An *s-t* network with N capacitated nodes can obviously be converted into an *s-t* network without node capacities. The transformation consists in replacing each node by a bi-vertex in which the internal bridge has the capacity of the original node (Figure 3).

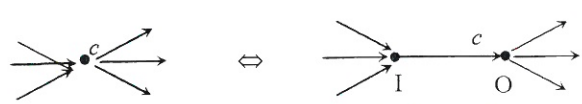


Figure 3. Converting a Capacitated Node Into a bi-vertex

Applying such a transformation to each node of the initial digraph leads to N bi-vertices *i.e.* $2N$ nodes without capacity.

Since the maximum flow is necessarily limited by the capacity of the source bi-vertex as well as of the sink bi-vertex, the input node of the source bi-vertex and the output node of the sink bi-vertex will be first removed, which leads to a sub-graph with *2N-2* non capacitated nodes, *i.e. N-2* bi-vertices plus the output node of the source and the input node of the sink. Assuming that the Max-Flow of the sub-graph is calculated, the capacity C_s of the source and the capacity C_t of the sink will be reintroduced. The Max-Flow m is finally provided by relation (5) corrected as:

$$m = Min\left[C_s, C_t, \underset{k}{Min}\left(P_k M_p^{\lambda}\left(\overline{P_k}\right)^T\right)\right] \qquad (5 \text{ bis})$$

with $k = 0, \ldots, 2^{2(N-2)} - 1$

Nevertheless, many of the $2^{2(N-2)}$ cuts involved in relation (5 bis) are redundant and the corresponding calculation can be spared.

6.1. Enumerating non Redundant Minimum s-t cuts

Having removed the input node of the source bi-vertex and the output node of the sink bi-vertex, the issue is to enumerate the cuts of the remaining graph with *N-2* bi-vertices. Each of the bi-vertex can be considered as a unique vertex with two possible situations respectively characterized by case ii and case iii as seen in section 5. Enumerating the *s-t* cuts must therefore take into account these two possibilities while combining vertices. The number of non redundant cuts is consequently:

$$C_{N-2}^0 2^0 + C_{N-2}^1 2^1 + \ldots + C_{N-2}^{N-2} 2^{N-2} = 3^{N-2} \qquad (9)$$

Reintroducing the capacities of the source and the sink, the number of cuts in an *s-t* network with capacitated arcs and N capacitated nodes is finally $3^{N-2} + 2$.

Table 1 shows the number of cuts as a function of vertex number N in the case of a network without node capacities (Line 2) and with node capacities (Line 3). Most of the cuts shown in line 3 are redundant. Line 4 provides the number of non redundant cuts which is considerably lower.

Table 1: Counting Cuts in *s-t* Networks

N	3	4	5	6	7
2^{N-2}	2	4	8	16	32
$2^{2(N-2)}+2$	6	18	66	258	1026
$3^{N-2}+2$	5	11	29	83	245

6.2. Application Example

The aim of this section is to identify all non redundant cuts in an *s-t* network with $N = 4$ capacitated nodes, *i.e.* in an *s-t* network with $2N = 8$ non capacitated nodes.

Removing first the input node of the source bi-vertex and the output node of the sink bi-vertex leads to a sub-graph with $2N-2 = 6$ nodes (Figure 4).

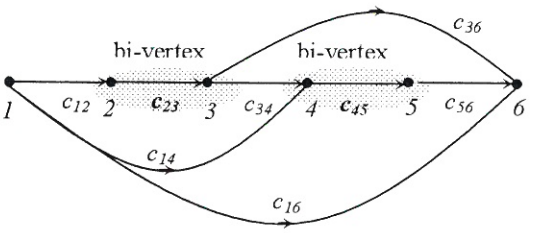


Figure 4. Equivalent *s-t* Network Without Node Capacities

The associated capacity matrix is:

$$M = \begin{pmatrix} \cdot & c_{12} & \cdot & c_{14} & \cdot & c_{16} \\ \cdot & \cdot & c_{23} & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & c_{34} & \cdot & c_{36} \\ \cdot & \cdot & \cdot & \cdot & c_{45} & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & c_{56} \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{pmatrix}$$

The notations are generalized to take simultaneously into account different sub-graphs:

G_p^λ	*s-t* network p at level λ
V_p^λ	vertex set of G_p^λ
$N_p^\lambda = card V_p^\lambda$	vertex number of G_p^λ
$\{m_{p,q}^\lambda \quad q = 1,...,N_p^\lambda\}$	vertex capacities of G_p^λ
$M_p^\lambda \quad (N_p^\lambda \times N_p^\lambda)$	adjacency matrix of G_p^λ

A recursive Max-Flow aggregation process can be formalized as the composition of two applications [16]:

$$L^{N_p^\lambda} \times R^{N_p^\lambda} \xrightarrow{\quad f \quad} \mathcal{M} \xrightarrow{\quad g \quad} L \times R$$

with L the set of natural numbers for labeling the nodes, R the set of real numbers for expressing node capacities and $\mathcal{M}$ the set of matrices with real entries for defining network structures.

Application f (network building) groups a set of labeled nodes with real capacities and generates the topology of network G_p^λ through capacity matrix M_p^λ.

Application g (network reduction) computes the maximum flow through network G_p^λ as mentioned in section 6, *i.e* :

$$m_p^{\lambda+1} = Min\left[C_{p,s}^\lambda, C_{p,t}^\lambda, \underset{k}{Min}\left(P_k M_p^\lambda \left(\overline{P_k}\right)^T\right)\right] \tag{10}$$

for $k \in \hat{K}_{N_p^\lambda}$

with $C_{p,s}^\lambda$ and $C_{p,t}^\lambda$ the capacity of the source and of the sink in G_p^λ.

Finally, two consecutive levels of capacities are linked by application $g \circ f$ as:

$$m_p^{\lambda+1} = g \circ f(\ m_{p,q}^\lambda \ \ q = 1,...,N_p^\lambda) \tag{11}$$

Clearly, expression (11) is recursive and can be used to compute step by step the Max-Flow of large networks, which limits the number of vertices to be considered simultaneously as shown hereafter:

Consider a network G divided into n *s-t* networks $\{G_p \quad p=1,...,n\}$. Each network G_p has N_p vertices. The number of cuts required to compute directly the maximum flow (see section 6) through G would be:

$$3^\alpha + 2 \quad \text{with} \quad \alpha = \sum_{p=1}^{n} N_p - 2 \tag{12}$$

whereas a two-level approach consists in reducing each of the n graphs, which leads to a new graph with n nodes to be reduced in a second step. The global number of cuts is then:

$$\sum_{p=1}^{n} (3^{N_p - 2} + 2) + 3^{n-2} + 2 \tag{13}$$

which considerably limits the combinatory explosion. A three-level network break-up would still decrease the number of cuts to analyze. Actually, the higher the number of graph levels is, the more the gain is high.

8. Examples in Manufacturing

In this section, two simple examples are presented to apply the results stated in sections 6 and 7 to the analysis and reengineering of manufacturing workshops.

The first example shows the relevance of the analytical (*vs* numerical) approach to Max-Flow calculation with regard to engineering issues. The second example is an illustration of multi-level reasoning for Max-Flow determination.

8.1. Searching for Workshop Bottleneck

Let consider a workshop with $N=4$ workstations and a part routing network. Workstation capacities $(\sigma_1, \sigma_2, \sigma_3, \sigma_4)$ and transportation capacities (a,b,c,d,e) between workstations are known (Figure 5). Flow units are part amounts per time unit.

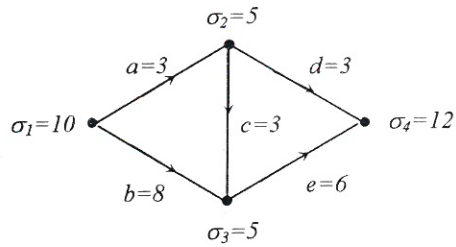
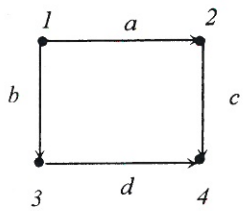


Figure 5. Capacitated Manufacturing Network

Searching for workshop bottleneck

The capacity matrix associated with the *s-t* network is:



The maximum flow is found as follows:

$$m = Min(\sigma_1, \sigma_4, c_k) \qquad c_k = \underset{k}{Min}\left(P_k M_p \left(\overline{P_k}\right)^T\right)$$

$$k \in \hat{K}_4 = \{0,2,3,8,10,11,12,14,15\}$$

$P_0 = 1\ 0000\ 0$	$P_2 = 1\ 0010\ 0$	$P_3 = 1\ 0011\ 0$
$P_8 = 1\ 1000\ 0$	$P_{10} = 1\ 1010\ 0$	$P_{11} = 1\ 1011\ 0$
$P_{12} = 1\ 1100\ 0$	$P_{14} = 1\ 1110\ 0$	$P_{15} = 1\ 1111\ 0$

$c_0 = a+b=11$	$c_2 = a+\sigma_3=8$	$c_3 = a+e=9$
$c_8 = b+\sigma_2=13$	$c_{10} = \sigma_2+\sigma_3=10$	$c_{11} = e+\sigma_2=11$
$c_{12} = b+c+d=14$	$c_{14} = d+\sigma_3=8$	$c_{15} = d+e=9$

m =Min[10,12,Min(11,8,9,13,10,11,14,8,9)]= 8

The result is that the maximum production flow is 8 due to the following bottlenecks:

$c_2 = a+\sigma_3$ $\qquad$ $c_{14} = d+\sigma_3$

Bottleneck solving

Assuming that workstation capacities cannot be modified for economical reasons, the issue is now to maximize workshop's capacity by investing in new transportation resources.

Since the maximum throughput m cannot surpass $Min\ (\sigma_1, \sigma_2 + \sigma_3, \sigma_4) = 10$ the improvements required are as follows:

$$M = \begin{pmatrix} . & a & . & b & . & . \\ . & . & \sigma_2 & . & . & . \\ . & . & . & c & . & d \\ . & . & . & . & \sigma_3 & . \\ . & . & . & . & . & e \\ . & . & . & . & . & . \end{pmatrix}$$

$c_2 \geq 10 \qquad c_3 \geq 10 \qquad c_{14} \geq 10 \qquad c_{15} \geq 10$

These requirements are fulfilled from the moment that:

$a \geq 5 \qquad d \geq 5$

8.2. Multi-level Max-Flow

Consider at level $\lambda=2$ a supply chain consisting in a network of $N_1^2 = 4$ enterprises interconnected as shown hereafter:

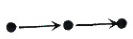
Supply Chain global description at level $\lambda=2$

The part routing between the enterprises is submitted to transportation capacity constraints a to d. Each enterprise has its own production capacity m_p^2 $p=1,\ldots,4$. Once the source input node and sink output are removed, the capacity matrix associated to the partial s-t network has the form:

$$M_1^2 = \begin{pmatrix} . & a & . & b & . & . \\ . & . & m_2^2 & . & . & . \\ . & . & . & . & . & c \\ . & . & . & . & m_3^2 & . \\ . & . & . & . & . & d \\ . & . & . & . & . & . \end{pmatrix}$$

Lower level $\lambda=1$ provides the internal description of flow capacities within each enterprise, under s-t network assumption:

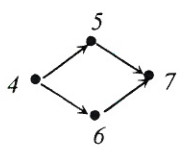
Site 1



$$M_1^1 = \begin{pmatrix} . & 10 & . & . \\ . & . & 7 & . \\ . & . & . & 9 \\ . & . & . & . \end{pmatrix}$$

source capacity: 6
sink capacity: 6

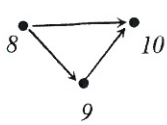
Site 2



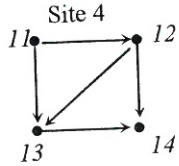
$$M_2^1 = \begin{pmatrix} . & 4 & . & 2 & . & . \\ . & . & 2 & . & . & . \\ . & . & . & . & . & 3 \\ . & . & . & . & 1 & . \\ . & . & . & . & . & 1 \\ . & . & . & . & . & . \end{pmatrix}$$

source capacity: 3
sink capacity: 4

Site 3



Site 4



$$M_3^1 = \begin{pmatrix} . & 2 & . & 1 \\ . & . & 2 & . \\ . & . & . & 1 \\ . & . & . & . \end{pmatrix} \qquad M_4^1 = \begin{pmatrix} . & 3 & . & 5 & . & . \\ . & . & 4 & . & . & . \\ . & . & . & 2 & . & 2 \\ . & . & . & . & 4 & . \\ . & . & . & . & . & 2 \\ . & . & . & . & . & . \end{pmatrix}$$

source capacity: 3 source capacity: 4
sink capacity: 4 sink capacity: 5

Local site description at level $\lambda=1$

Site capacities

The capacity of each site is first calculated using (10):

$$m_1^1 = Min\left[6, 6, \underset{k\in\hat{K}_3}{Min}\left(P_k M_1^1 \left(\overline{P_k}\right)^T\right)\right] = 6$$

$$m_2^1 = Min\left[3, 4, \underset{k\in\hat{K}_4}{Min}\left(P_k M_2^1 \left(\overline{P_k}\right)^T\right)\right] = 3$$

$$m_3^1 = Min\left[3, 4, \underset{k\in\hat{K}_3}{Min}\left(P_k M_3^1 \left(\overline{P_k}\right)^T\right)\right] = 3$$

$$m_4^1 = Min\left[4, 5, \underset{k\in\hat{K}_4}{Min}\left(P_k M_4^1 \left(\overline{P_k}\right)^T\right)\right] = 4$$

Supply chain capacity

Having aggregated the local production capacities, the issue is then to calculate the capacity of the whole supply chain at level $\lambda=2$:

$$m_1^2 = Min\,[m_1^1, m_4^1, Min(a, m_2^1, c) + Min(b, m_3^1, d)]$$

$$= Min[6, 4, Min(a, 3, c) + Min(b, 3, d)]$$

The Max-Flow cannot surpass *4*. A technical or economical criterion will guide the instantiation of the transportation resources capacities. Assuming for example a balancing criterion, the transportation capacities might be set to minimal values $a=c=b=d=2$ to provide the supply chain with maximum throughput.

9. Conclusions

This study was motivated by the need for assessing the maximum production throughput in manufacturing and logistic distributed systems under both transportation and transformation capacity constraints. As required for engineering purpose, the capacities are parametric. Assumed that network's topology is acyclic and with only one input node and one output node, the analytical method presented here allows to calculate all non redundant cuts ($3^{N-2}+2$ cuts for N nodes) using matrix calculus. Thanks to the parametric approach, the results of Max-Flow calculation are pre-calculated once and for all for complete graphs, to be then instantiated on any particular network with same node number.

A recursive multi-level calculation process is also presented for complex networks, avoiding combinatory explosion through hierarchical analysis. The multilevel approach can be made bottom-up (starting from local data to assess the global flow capacity) or conversely top-down (starting from the global capacities expected to specify afterwards the local organizations).

REFERENCES

1. K. AHUJA, T. L. MAGNANTI, J. B. ORLIN, **Networks flows**, Prentice-Hall, 1993.

2. C. BERGE, **Graphs and hypergraphs**, North Holland, 1973.

3. Y. L. CHEN, **A parametric maximum flow algorithm for bipartite graphs with applications**, European Journal of Operational Research, 80 (1995) pp. 226-235.

4. L. R. FORD, D. R. FULKERSON, **Flows in networks**, Princeton University Press, 1962.

5. M. GONDRAN, M. MINOUX, **Graphes et algorithmes**, Eyrolles, 1995 (in French).

6. J. GROSS, J. YELLEN, **Graph theory and its applications**, CRC Press, 1999.

7. D. HARTVIGSEN, F. MARGOT, **Multiterminal flows and cuts**, Operations Research Letters, 17 (1995) pp. 201-204.

8. M. HENZINGER, D. P. WILLIAMSON, **On the number of small cuts in a graph**, Information Processing Letters, 59 (1996) 41-44.

9. X. JIANG, H. BUNKE, **Optimal vertex ordering of graph**, Information Processing Letters, 72 (1999) pp. 149-154.

10. H. NAGAMOCHI, K. NISHIMURA, T. IBARAKI, **Computing all small cuts in an undirected network**, SIAM Journal of Discrete Mathematics, 10 (3) (1997) pp. 469-481.

11. J. C. PICARD, M. QUEYRANNE, **On the structure of all minimum cuts in a network and applications**, Math. Program. Study, 13 (1980) pp. 8-16.

12. F. A. SHARIFOV, **Minimum cuts in parametric networks**, Cybern. Syst. Anal., 30 (6) (1994) pp. 885-890.

13. Y. SHEN, **A new simple algorithm for enumerating all minimal paths and cuts of a graph**, Microelectron. Reliab., 35 n° 6 (1995) pp. 973-976.

14. J. SOMERS, I. ADLER, **The max-flow problem with parametric capacities**, Discrete Appl. Math., 1 (1979) pp. 287-300.

15. S. XU, **The line index and minimum cut of weighed graphs**, European Journal of Operational Research, 109 (1998) pp. 672-685.

16. M. ZOLGHADRI, J.P. BOURRIERES, **A data aggregation framework for multi-level production system control**, Proc. of the IEEE Int. Conf. on Systems, Man, and Cybernetics, Intelligent Systems For Humans In A Cyberworld October 11-14, 1998, San Diego.