Chapter 11
Flows

Discrete Structures for Computing on January 11, 2017

Huynh Tuong Nguyen, Nguyen An Khuong, Vo Thanh Hung
Faculty of Computer Science and Engineering
University of Technology - VNUHCM
htnguyen@hcmut.edu.vn

Contents

Contents

Huynh Tuong Nguyen, Nguyen An Khuong, Vo Thanh Hung

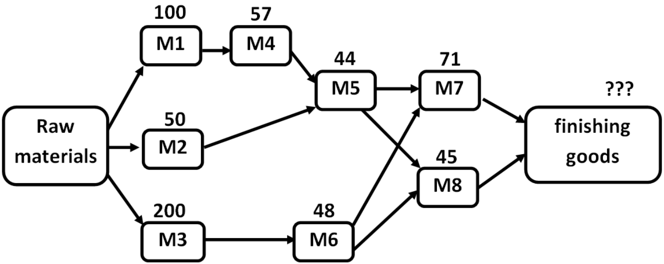
Course outcomes

	Course learning outcomes
L.O.1	Understanding of logic and discrete structures
	L.O.1.1 – Describe definition of propositional and predicate logic L.O.1.2 – Define basic discrete structures: set, mapping, graphs
L.O.2	Represent and model practical problems with discrete structures
	L.O.2.1 – Logically describe some problems arising in Computing L.O.2.2 – Use proving methods: direct, contrapositive, induction L.O.2.3 – Explain problem modeling using discrete structures
L.O.3	Understanding of basic probability and random variables
	L.O.3.1 – Define basic probability theory L.O.3.2 – Explain discrete random variables
L.O.4	Compute quantities of discrete structures and probabilities
	L.O.4.1 – Operate (compute/ optimize) on discrete structures L.O.4.2 – Compute probabilities of various events, conditional ones, Bayes theorem

Contents

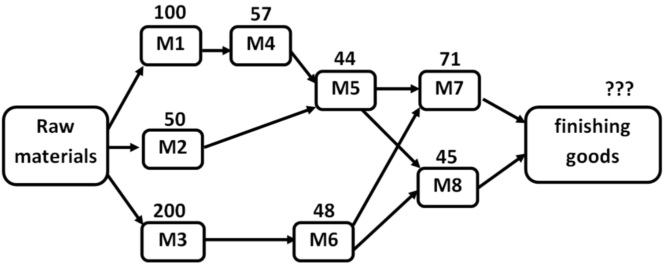
Motivation

- Distributed manufacturing system :
  $((((M1 \wedge M4) \vee M2) \wedge M5) \vee (M3 \wedge M6)) \wedge (M7 \vee M8)$
- Production capacity of each branch is defined in graph G



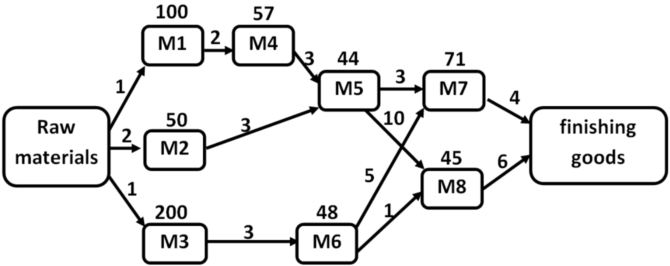
Motivation

- Distributed manufacturing system :
  $((((M1 \wedge M4) \vee M2) \wedge M5) \vee (M3 \wedge M6)) \wedge (M7 \vee M8)$
- Production capacity of each branch is defined in graph G
- How to determine the production capacity (e.g. pieces/min)?



Motivation

- Distributed manufacturing system :
  $((((M1 \wedge M4) \vee M2) \wedge M5) \vee (M3 \wedge M6)) \wedge (M7 \vee M8)$
- Production capacity of each branch is defined in graph G
- How to determine the production capacity (e.g. pieces/min)?
- How to determine the production paths with the minimum transportation cost ?



Maximum flow problem

Given data

- A directed graph $G = (V, E)$ with source node s and sink node t
- capacity function $c : E \longrightarrow \mathcal{R}$, i.e. $c(u, v) \geq 0$ for any edge $(u, v) \in E$

Objective

Send as much flow as possible with flow $f : E \longrightarrow R^+$ such that

- $f(u, v) \leq c(u, v)$, for all $(u, v) \in E$
- $\sum_{v \in V} f(v, u) = \sum_{v \in V} f(u, v)$, for $u \neq s, t$

Maximum flow problem with minimum cost

Given data

- A directed graph $G = (V, E)$ with source node s and sink node t
- capacity function $c : E \longrightarrow \mathcal{R}$, i.e. $c(u, v) \geq 0$ for any edge $(u, v) \in E$

Objective

Send as much flow as possible such that

- $f(u, v) \leq c(u, v)$, for all $(u, v) \in E$
- $\sum_{v \in V} f(v, u) = \sum_{v \in V} f(u, v)$, for $u \neq s, t$

Maximum flow problem with minimum cost

Given data

- A directed graph $G = (V, E)$ with source node s and sink node t
- capacity function $c : E \longrightarrow \mathcal{R}$, i.e. $c(u, v) \geq 0$ for any edge $(u, v) \in E$
- cost function $a : E \longrightarrow \mathcal{R}$, i.e. $a(u, v) \geq 0$ for any edge $(u, v) \in E$

Objective

Send as much flow as possible with minimum cost such that

- $f(u, v) \leq c(u, v)$, for all $(u, v) \in E$
- $\sum_{v \in V} f(v, u) = \sum_{v \in V} f(u, v)$, for $u \neq s, t$
- $\sum_{(u,v) \in E} a(u, v) f(u, v)$ should be minimized

State-of-the-art

Flow Algorithms

- Linear programming
- Ford-Fulkerson algorithm $O(E \max |f|)$
- Edmond-Karp algorithm $O(VE^2)$
- Dinitz blocking flow algorithm $O(V^2 E)$
- General push-relabel maximum flow algorithm $O(V^2 E)$
- Push-relabel algorithm with FIFO vertex selection rule $O(V^3)$
- Dinitz blocking flow algorithm with dynamic trees $O(VE \log(V))$
- Push-relabel algorithm with dynamic trees $O(VE \log(V^2/E))$
- Binary blocking flow algorithm $O(E \min(V^{2/3}, \sqrt{E}) log(V^2/E) \log(U))$ with $U = \max c(u, v)$

Ford-Fulkerson's algorithm for solving Max Flow Problem

Input: graph G with flow capacity c, a source node s, and a sink node t

Output: a maximum flow f from s to t

Ford-Fulkerson's algorithm for solving Max Flow Problem

Input: graph G with flow capacity c, a source node s, and a sink node t

Output: a maximum flow f from s to t

$k = 0$; $G^{(0)} = G$;

$c^{(0)}(u, v) = c(u, v)$, $c^{(0)}(v, u) = 0$, $\forall (u, v) \in G^{(0)}$;

Ford-Fulkerson's algorithm for solving Max Flow Problem

Input: graph G with flow capacity c, a source node s, and a sink node t

Output: a maximum flow f from s to t

$k = 0$; $G^{(0)} = G$;

$c^{(0)}(u,v) = c(u,v)$, $c^{(0)}(v,u) = 0$, $\forall (u,v) \in G^{(0)}$;

While $\exists$ a path $\Pi^{(k)}(s,t)$ in $G^{(k)}$ such that $c^{(k)}(u,v) > 0$, $\forall (u,v) \in \Pi^{(k)}$ do

  Find $f(\Pi^{(k)}) = \min\{c^{(k)}(u,v) | (u,v) \in \Pi^{(k)}\}$;

Ford-Fulkerson's algorithm for solving Max Flow Problem

Input: graph G with flow capacity c, a source node s, and a sink node t

Output: a maximum flow f from s to t

$k = 0$; $G^{(0)} = G$;

$c^{(0)}(u, v) = c(u, v)$, $c^{(0)}(v, u) = 0$, $\forall (u, v) \in G^{(0)}$;

While $\exists$ a path $\Pi^{(k)}(s, t)$ in $G^{(k)}$ such that $c^{(k)}(u, v) > 0$, $\forall (u, v) \in \Pi^{(k)}$ do

  Find $f(\Pi^{(k)}) = \min\{c^{(k)}(u, v) | (u, v) \in \Pi^{(k)}\}$;

  For each edge $(u, v) \in \Pi^{(k)}$ do

    If $(u, v) \in G$ then

      $c^{(k+1)}(u, v) = c^{(k)}(u, v) - f(\Pi^{(k)})$;

      $c^{(k+1)}(v, u) = c^{(k)}(v, u) + f(\Pi^{(k)})$;

    Else

      $c^{(k+1)}(u, v) = c^{(k)}(u, v) + f(\Pi^{(k)})$;

      $c^{(k+1)}(v, u) = c^{(k)}(v, u) - f(\Pi^{(k)})$;

Contents

Ford-Fulkerson's algorithm for solving Max Flow Problem

Input: graph G with flow capacity c, a source node s, and a sink node t

Output: a maximum flow f from s to t

$k = 0$; $G^{(0)} = G$;

$c^{(0)}(u, v) = c(u, v)$, $c^{(0)}(v, u) = 0$, $\forall(u, v) \in G^{(0)}$;

While $\exists$ a path $\Pi^{(k)}(s, t)$ in $G^{(k)}$ such that $c^{(k)}(u, v) > 0$, $\forall(u, v) \in \Pi^{(k)}$ do

  Find $f(\Pi^{(k)}) = \min\{c^{(k)}(u, v) | (u, v) \in \Pi^{(k)}\}$;

  For each edge $(u, v) \in \Pi^{(k)}$ do

    If $(u, v) \in G$ then

      $c^{(k+1)}(u, v) = c^{(k)}(u, v) - f(\Pi^{(k)})$;

      $c^{(k+1)}(v, u) = c^{(k)}(v, u) + f(\Pi^{(k)})$;

    Else

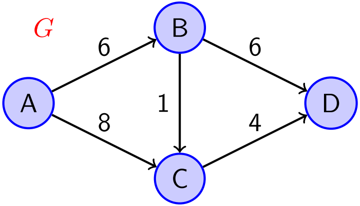
      $c^{(k+1)}(u, v) = c^{(k)}(u, v) + f(\Pi^{(k)})$;

      $c^{(k+1)}(v, u) = c^{(k)}(v, u) - f(\Pi^{(k)})$;

  $k + +$;

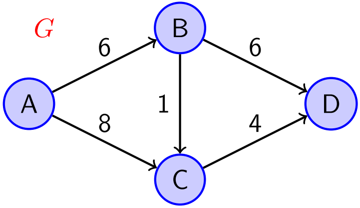
Contents

Example 1



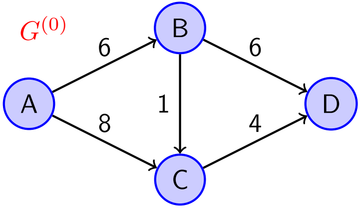
Contents

Example 1



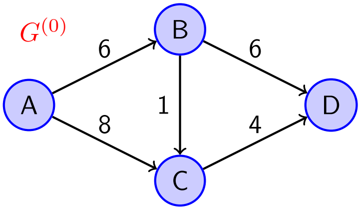
k	$\Pi^{(k)}$	(A,B)	(A,C)	(B,C)	(B,D)	(C,D)	$f(\Pi^{(k)})$

Example 1



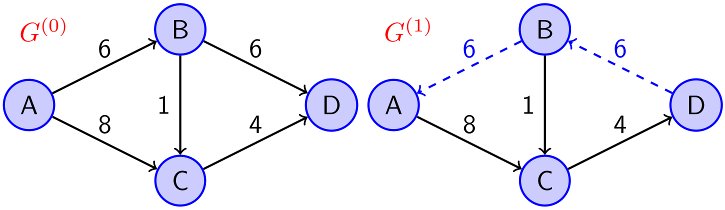
k	$\Pi^{(k)}$	(A,B)	(A,C)	(B,C)	(B,D)	(C,D)	$f(\Pi^{(k)})$

Example 1



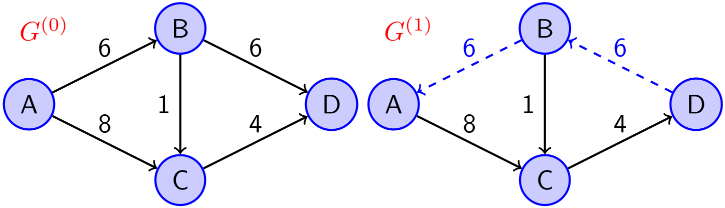
k	$\Pi^{(k)}$	(A,B)	(A,C)	(B,C)	(B,D)	(C,D)	$f(\Pi^{(k)})$
0	{(A,B),(B,D)}	6	-	-	6	-	6

Example 1



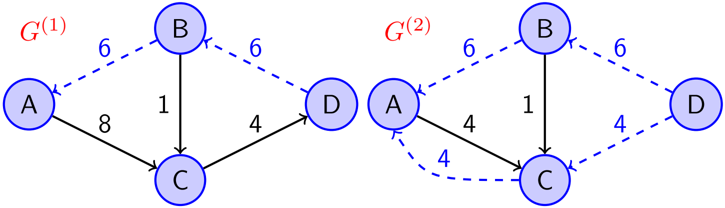
k	$\Pi^{(k)}$	(A,B)	(A,C)	(B,C)	(B,D)	(C,D)	$f(\Pi^{(k)})$
0	{(A,B),(B,D)}	6	-	-	6	-	6

Example 1



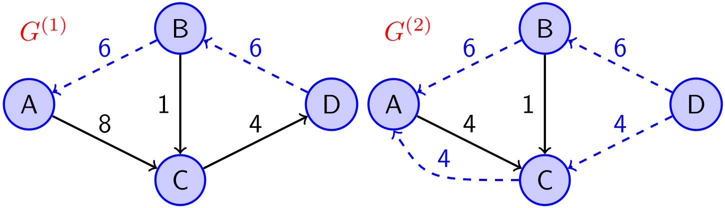
k	$\Pi^{(k)}$	(A,B)	(A,C)	(B,C)	(B,D)	(C,D)	$f(\Pi^{(k)})$
0	{(A,B),(B,D)}	6	-	-	6	-	6
1	{(A,C),(C,D)}	-	4	-	-	4	4

Example 1



k	$\Pi^{(k)}$	(A,B)	(A,C)	(B,C)	(B,D)	(C,D)	$f(\Pi^{(k)})$
0	{(A,B),(B,D)}	6	-	-	6	-	6
1	{(A,C),(C,D)}	-	4	-	-	4	4

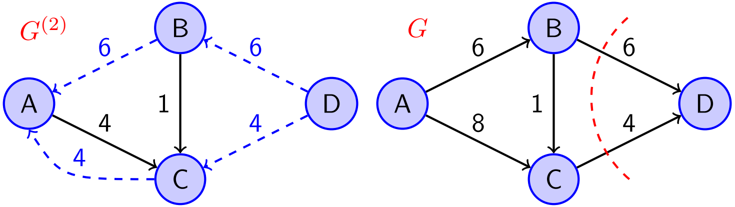
Example 1



k	$\Pi^{(k)}$	(A,B)	(A,C)	(B,C)	(B,D)	(C,D)	$f(\Pi^{(k)})$
0	{(A,B),(B,D)}	6	-	-	6	-	6
1	{(A,C),(C,D)}	-	4	-	-	4	4
	Stop		with $f_{max} = 10$				

Huynh Tuong Nguyen, Nguyen An Khuong, Vo Thanh Hung

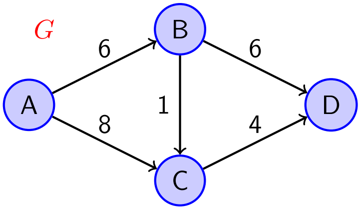
Example 1



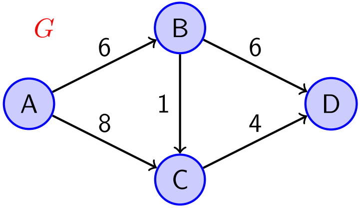
k	$\Pi^{(k)}$	(A,B)	(A,C)	(B,C)	(B,D)	(C,D)	$f(\Pi^{(k)})$
0	{(A,B),(B,D)}	6	-	-	6	-	6
1	{(A,C),(C,D)}	-	4	-	-	4	4
		6	4	-	6	4	10

Stop with $f_{max} = 10$

Example 2

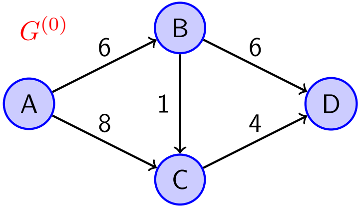


Example 2



k	$\Pi^{(k)}$	(A,B)	(A,C)	(B,C)	(B,D)	(C,D)	$f(\Pi^{(k)})$

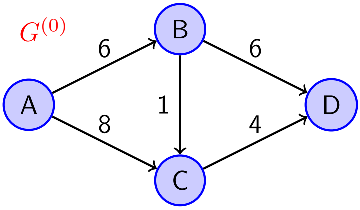
Example 2



k	$\Pi^{(k)}$	(A,B)	(A,C)	(B,C)	(B,D)	(C,D)	$f(\Pi^{(k)})$

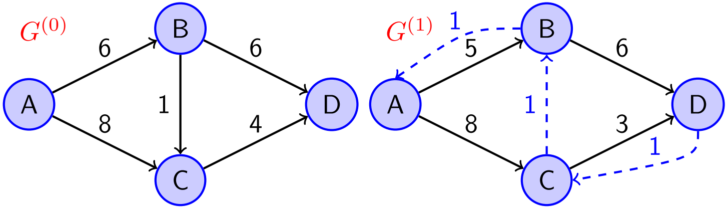
Contents

Example 2



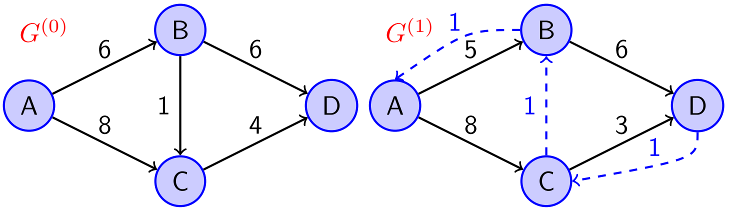
k	$\Pi^{(k)}$	(A,B)	(A,C)	(B,C)	(B,D)	(C,D)	$f(\Pi^{(k)})$
0	{(A,B),(B,C),(C,D)}	1	-	1	-	1	1

Example 2



k	$\Pi^{(k)}$	(A,B)	(A,C)	(B,C)	(B,D)	(C,D)	$f(\Pi^{(k)})$
0	{(A,B),(B,C),(C,D)}	1	-	1	-	1	1

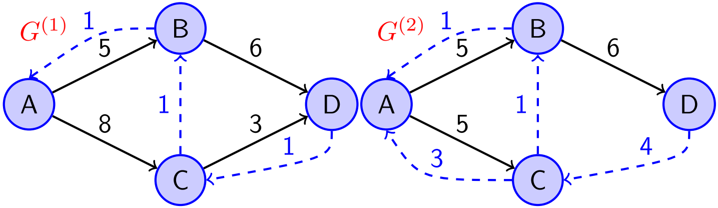
Example 2



k	$\Pi^{(k)}$	(A,B)	(A,C)	(B,C)	(B,D)	(C,D)	$f(\Pi^{(k)})$
0	{(A,B),(B,C),(C,D)}	1	-	1	-	1	1
1	{(A,C),(C,D)}	-	3	-	-	3	3

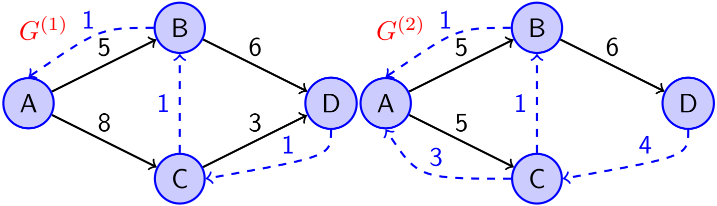
Contents

Example 2



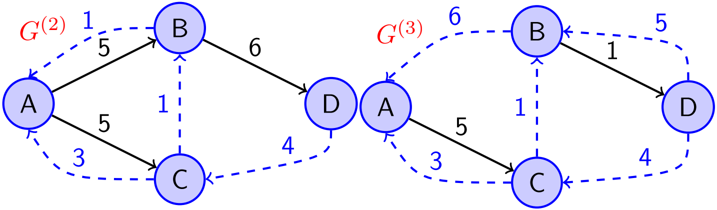
k	$\Pi^{(k)}$	(A,B)	(A,C)	(B,C)	(B,D)	(C,D)	$f(\Pi^{(k)})$
0	{(A,B),(B,C),(C,D)}	1	-	1	-	1	1
1	{(A,C),(C,D)}	-	3	-	-	3	3

Example 2



k	$\Pi^{(k)}$	(A,B)	(A,C)	(B,C)	(B,D)	(C,D)	$f(\Pi^{(k)})$
0	{(A,B),(B,C),(C,D)}	1	-	1	-	1	1
1	{(A,C),(C,D)}	-	3	-	-	3	3
2	{(A,B),(B,D)}	5	-	-	5	-	5

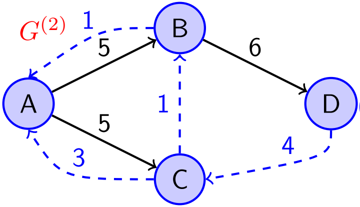
Example 2

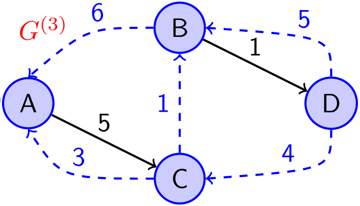


k	$\Pi^{(k)}$	(A,B)	(A,C)	(B,C)	(B,D)	(C,D)	$f(\Pi^{(k)})$
0	{(A,B),(B,C),(C,D)}	1	-	1	-	1	1
1	{(A,C),(C,D)}	-	3	-	-	3	3
2	{(A,B),(B,D)}	5	-	-	5	-	5

Contents

Example 2



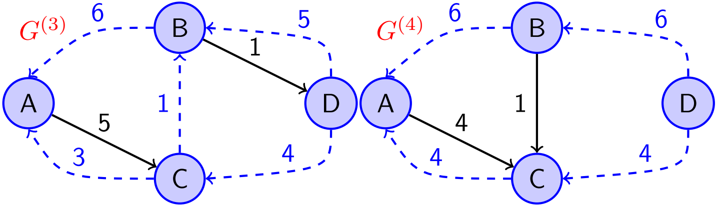


k	$\Pi^{(k)}$	(A,B)	(A,C)	(B,C)	(B,D)	(C,D)	$f(\Pi^{(k)})$
0	{(A,B),(B,C),(C,D)}	1	-	1	-	1	1
1	{(A,C),(C,D)}	-	3	-	-	3	3
2	{(A,B),(B,D)}	5	-	-	5	-	5
3	{(A,C),(C,B),(B,D)}	-	1	-1	1	-	1

Huynh Tuong Nguyen, Nguyen An Khuong, Vo Thanh Hung

Contents

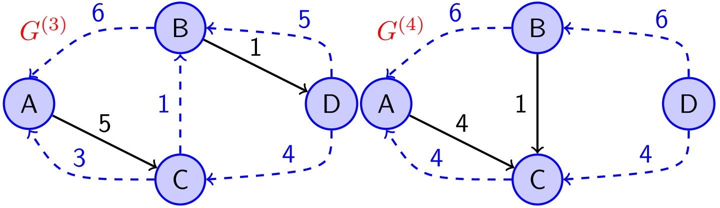
Example 2



k	$\Pi^{(k)}$	(A,B)	(A,C)	(B,C)	(B,D)	(C,D)	$f(\Pi^{(k)})$
0	{(A,B),(B,C),(C,D)}	1	-	1	-	1	1
1	{(A,C),(C,D)}	-	3	-	-	3	3
2	{(A,B),(B,D)}	5	-	-	5	-	5
3	{(A,C),(C,B),(B,D)}	-	1	-1	1	-	1

Huynh Tuong Nguyen, Nguyen An Khuong, Vo Thanh Hung

Example 2

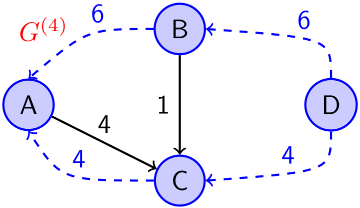


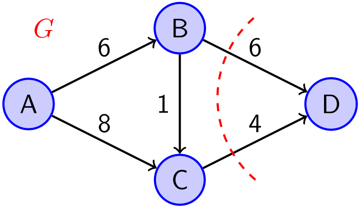
k	$\Pi^{(k)}$	(A,B)	(A,C)	(B,C)	(B,D)	(C,D)	$f(\Pi^{(k)})$
0	{(A,B),(B,C),(C,D)}	1	-	1	-	1	1
1	{(A,C),(C,D)}	-	3	-	-	3	3
2	{(A,B),(B,D)}	5	-	-	5	-	5
3	{(A,C),(C,B),(B,D)}	-	1	-1	1	-	1
	Stop		with $f_{max} = 10$				

Contents

Huynh Tuong Nguyen, Nguyen An Khuong, Vo Thanh Hung

Example 2





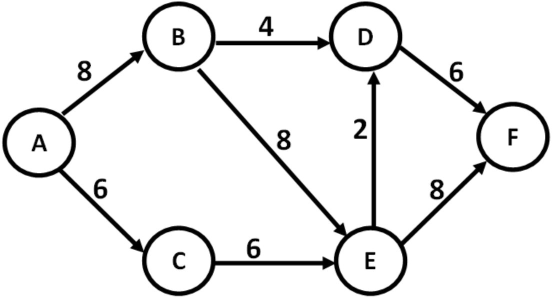
k	$\Pi^{(k)}$	(A,B)	(A,C)	(B,C)	(B,D)	(C,D)	$f(\Pi^{(k)})$
0	{(A,B),(B,C),(C,D)}	1	-	1	-	1	1
1	{(A,C),(C,D)}	-	3	-	-	3	3
2	{(A,B),(B,D)}	5	-	-	5	-	5
3	{(A,C),(C,B),(B,D)}	-	1	-1	1	-	1
		6	4	-	6	4	10

Stop with $f_{max} = 10$

Contents

Problem

Find the maximum flow and the min-cut in the following network.



G_1

Resolution for Max Flow and Min Cost Problem

Input: graph G with flow capacity c, a source node s, and a sink node t

Output: a maximum flow f from s to t

Huynh Tuong Nguyen, Nguyen An Khuong, Vo Thanh Hung

Resolution for Max Flow and Min Cost Problem

Input: graph G with flow capacity c, a source node s, and a sink node t

Output: a maximum flow f from s to t

$k = 0$; $G^{(0)} = G$; $c^{(0)}(u, v) = c(u, v)$, $c^{(0)}(v, u) = 0$, $\forall (u, v) \in G^{(0)}$;

Resolution for Max Flow and Min Cost Problem

Input: graph G with flow capacity c, a source node s, and a sink node t

Output: a maximum flow f from s to t

$k = 0$; $G^{(0)} = G$; $c^{(0)}(u,v) = c(u,v)$, $c^{(0)}(v,u) = 0$, $\forall (u,v) \in G^{(0)}$;

While $\exists$ a shortest path $\Pi^{(k)}(s,t)$ in $G^{(k)}$ such that $c^{(k)}(u,v) > 0$, $\forall (u,v) \in \Pi^{(k)}$ do

Find $f(\Pi^{(k)}) = \min\{c^{(k)}(u,v) | (u,v) \in \Pi^{(k)}\}$;

Contents

Resolution for Max Flow and Min Cost Problem

Input: graph G with flow capacity c, a source node s, and a sink node t

Output: a maximum flow f from s to t

$k = 0$; $G^{(0)} = G$; $c^{(0)}(u, v) = c(u, v)$, $c^{(0)}(v, u) = 0$, $\forall (u, v) \in G^{(0)}$;

While $\exists$ a shortest path $\Pi^{(k)}(s, t)$ in $G^{(k)}$ such that $c^{(k)}(u, v) > 0$, $\forall (u, v) \in \Pi^{(k)}$ do

Find $f(\Pi^{(k)}) = \min\{c^{(k)}(u, v) | (u, v) \in \Pi^{(k)}\}$;

For each edge $(u, v) \in \Pi^{(k)}$ do

If $(u, v) \in G$ then

$c^{(k+1)}(u, v) = c^{(k)}(u, v) - f(\Pi^{(k)})$;

$c^{(k+1)}(v, u) = c^{(k)}(v, u) + f(\Pi^{(k)})$;

Else

$c^{(k+1)}(u, v) = c^{(k)}(u, v) + f(\Pi^{(k)})$;

$c^{(k+1)}(v, u) = c^{(k)}(v, u) - f(\Pi^{(k)})$;

Resolution for Max Flow and Min Cost Problem

Input: graph G with flow capacity c, a source node s, and a sink node t

Output: a maximum flow f from s to t

$k = 0$; $G^{(0)} = G$; $c^{(0)}(u, v) = c(u, v)$, $c^{(0)}(v, u) = 0$, $\forall (u, v) \in G^{(0)}$;

While $\exists$ a shortest path $\Pi^{(k)}(s, t)$ in $G^{(k)}$ such that $c^{(k)}(u, v) > 0$, $\forall (u, v) \in \Pi^{(k)}$ do

  Find $f(\Pi^{(k)}) = \min\{c^{(k)}(u, v) | (u, v) \in \Pi^{(k)}\}$;

  For each edge $(u, v) \in \Pi^{(k)}$ do

    If $(u, v) \in G$ then

      $c^{(k+1)}(u, v) = c^{(k)}(u, v) - f(\Pi^{(k)})$;

      $c^{(k+1)}(v, u) = c^{(k)}(v, u) + f(\Pi^{(k)})$;

    Else

      $c^{(k+1)}(u, v) = c^{(k)}(u, v) + f(\Pi^{(k)})$;

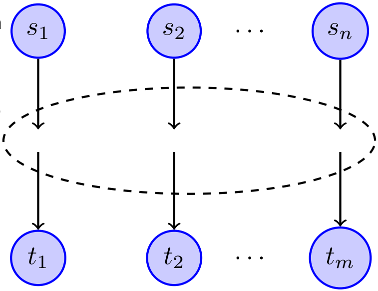
      $c^{(k+1)}(v, u) = c^{(k)}(v, u) - f(\Pi^{(k)})$;

  $k++$;

Contents

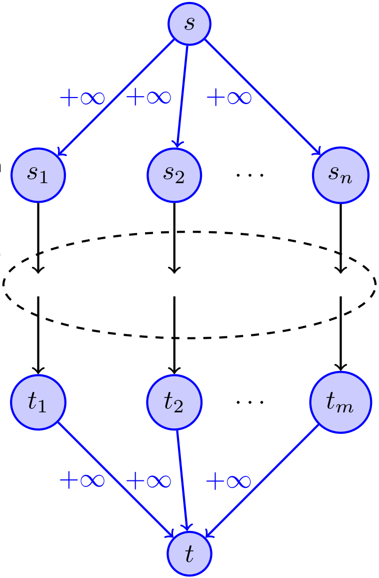
Multi-source Multi-sink Maximum Flow Problem

- Given a network $\mathcal{N} = (V, E)$ with a set of sources $S = s_1, \ldots, s_n$ and a set of sinks $T = t_1, \ldots, t_m$
- find the maximum flow across $\mathcal{N}$.



Multi-source Multi-sink Maximum Flow Problem

- Given a network $\mathcal{N} = (V, E)$ with a set of sources $S = s_1, \ldots, s_n$ and a set of sinks $T = t_1, \ldots, t_m$
- find the maximum flow across $\mathcal{N}$.
- $\Longrightarrow$ transform into a maximum flow problem by adding a super source connecting to each vertex in S and a super sink connected by each vertex in T with infinite capacity on each edge



Maximum Cardinality Bipartite Matching

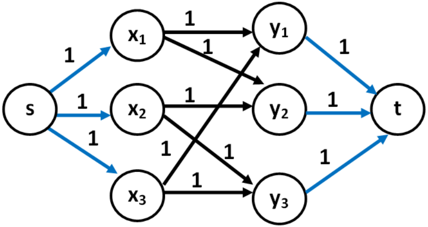
- Given a bipartite graph $G = (X \cup Y, E)$
- find a maximum cardinality matching in G, that is a matching that contains the largest possible number of edges.

Huynh Tuong Nguyen, Nguyen An Khuong, Vo Thanh Hung

Contents

Maximum Cardinality Bipartite Matching

- Given a bipartite graph $G = (X \cup Y, E)$
- find a maximum cardinality matching in G, that is a matching that contains the largest possible number of edges.
- $\Longrightarrow$ transform into a maximum flow problem by constructing a network $\mathcal{N} = (X \cup Y \cup \{s, t\}, E')$:
  1. E' contains the edges in G directed from X to Y.
  2. $(s, x) \in E'$ for each $x \in X$ and $(y, t) \in E'$ for each $y \in Y$.
  3. $c(e) = 1$ for each $e \in E'$.



Minimum Path Cover in Directed Acyclic Graph

- Given a directed acyclic graph $G = (V, E)$, we are to find the minimum number of paths to cover each vertex in V. We can construct a bipartite graph $G' = (Vout \cup Vin, E')$ from G, where
  1. $Vout = \{v \in V$: v has positive out-degree $\}$.
  2. $Vin = \{v \in V$: v has positive in-degree $\}$.
  3. $E' = \{(u, v) \in (Vout, Vin)$: $(u, v) \in E\}$.
- Then it can be shown that G' has a matching of size m iif there exists $n - m$ paths that cover each vertex in G, where n is the number of vertices in G.
- Therefore, the problem can be solved by finding the maximum cardinality matching in G' instead.

Maximum Flow Problem with Vertex Capacities

- Given a network $\mathcal{N} = (V, E)$, in which there is capacity at each node in addition to edge capacity, that is, a mapping $c : V \to R+$, denoted by $c(v)$, such that the flow f has to satisfy not only the capacity constraint and the conservation of flows, but also the vertex capacity constraint $\sum_{i \in V} f_{i,v} \leq c(v), \forall v \in V \setminus s, t$
- $\Longrightarrow$ the amount of flow passing through a vertex cannot exceed its capacity.
- To find the maximum flow across $\mathcal{N}$, we can transform the problem into the maximum flow problem in the original sense by expanding $\mathcal{N}$.
  - each $v \in V$ is replaced by v_{in} and v_{out}
    - v_{in} is connected by edges going into v
    - v_{out} is connected to edges coming out from v,
  - assign capacity $c(v)$ to the edge connecting v_{in} and v_{out}
- In this expanded network, the vertex capacity constraint is removed and therefore the problem can be treated as the original maximum flow problem.

Maximum Independent Path

- Given a directed graph $G = (V, E)$ and two vertices s and t,
- Find the maximum number of independent paths from s to t.

Maximum Independent Path

- Given a directed graph $G = (V, E)$ and two vertices s and t,
- Find the maximum number of independent paths from s to t.
- Two paths are said to be independent if they do not have a vertex in common apart from s and t.
- We can construct a network $\mathcal{N} = (V, E)$ from G with vertex capacities, where
  1. s and t are the source and the sink of $\mathcal{N}$ respectively.
  2. $c(v) = 1$ for each $v \in V$.
  3. $c(e) = 1$ for each $e \in E$.
- Then the value of the maximum flow is equal to the maximum number of independent paths from s to t.

Maximum Edge-disjoint Path

- given a directed graph $G = (V, E)$ and two vertices s and t
- find the maximum number of edge-disjoint paths from s to t.

Maximum Edge-disjoint Path

- given a directed graph $G = (V, E)$ and two vertices s and t
- find the maximum number of edge-disjoint paths from s to t.
- This problem can be transformed to a maximum flow problem by constructing a network $\mathcal{N} = (V, E)$ from G with s and t being the source and the sink of $\mathcal{N}$ respectively and assign each edge with unit capacity.

Contents

Problem

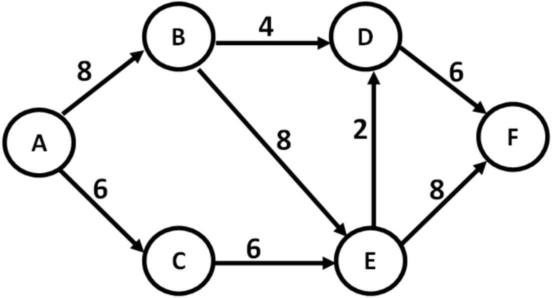
Find the maximum flow in the following networks

Huynh Tuong Nguyen, Nguyen An Khuong, Vo Thanh Hung

Contents

Problem

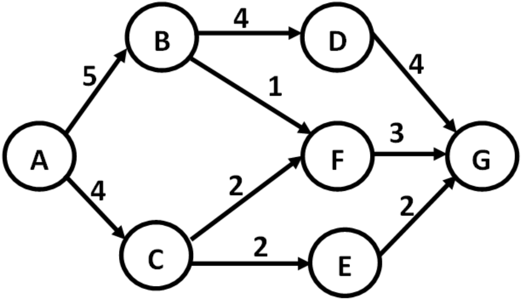
Find the maximum flow in the following networks



G_1

Problem

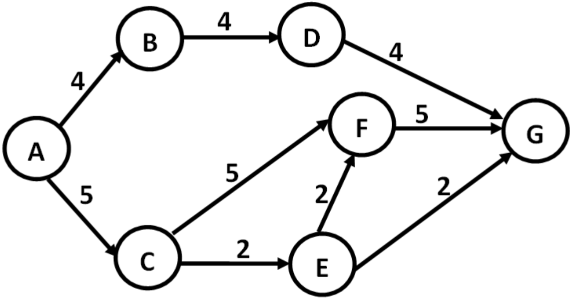
Find the maximum flow in the following networks



G_2

Problem

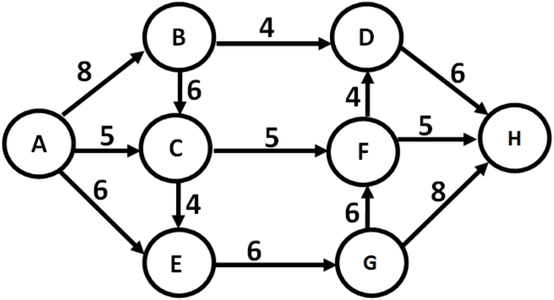
Find the maximum flow in the following networks



G_3

Problem

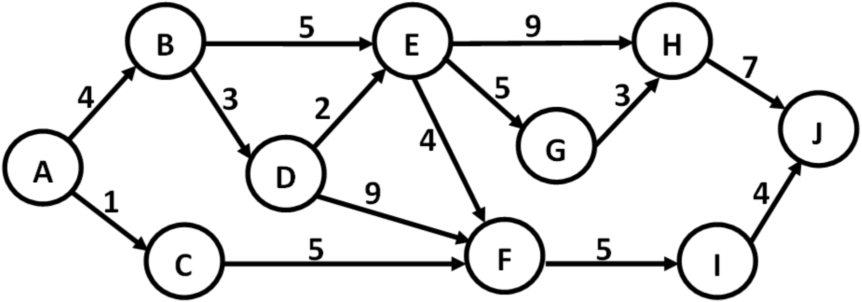
Find the maximum flow in the following networks



G_4

Problem

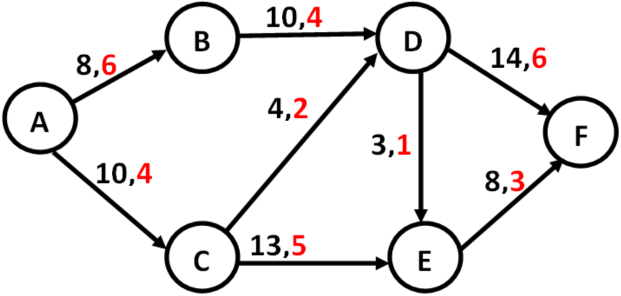
Find the maximum flow in the following networks



G_5

Problem

Find the maximum flow in the following networks



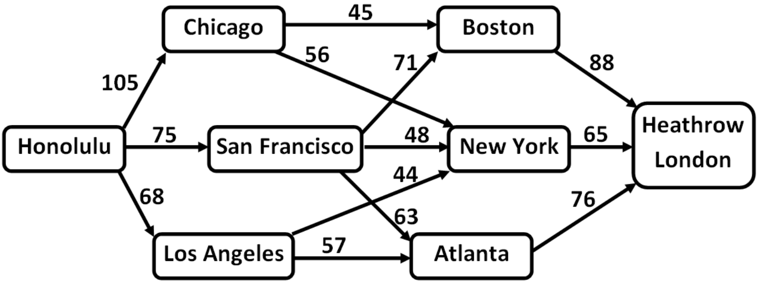
G_6

Restaurant management

- Whole pineapples are served in a restaurant in London.
- To ensure freshness, the pineapples are purchased in Hawaii and air freighted from Honolulu to Heathrow in London.
- The following network diagram outlines the different routes that the pineapples could take.

Restaurant management

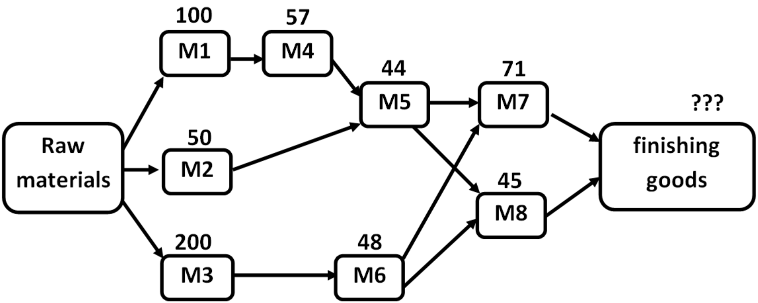
- Whole pineapples are served in a restaurant in London.
- To ensure freshness, the pineapples are purchased in Hawaii and air freighted from Honolulu to Heathrow in London.
- The following network diagram outlines the different routes that the pineapples could take.



Huynh Tuong Nguyen, Nguyen An Khuong, Vo Thanh Hung

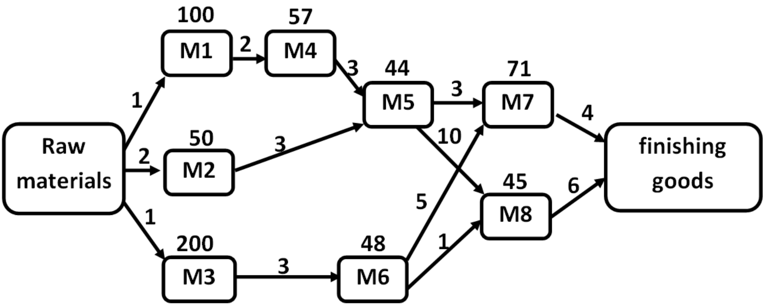
Production quantity measuring

- Distributed manufacturing system : $((((M1 \wedge M4) \vee M2) \wedge M5) \vee (M3 \wedge M6)) \wedge (M7 \vee M8)$
- Production capacity of each branch is defined in graph G
- How to determine the production capacity (e.g. pieces/min)?



Production quantity measuring

- Distributed manufacturing system :
  $((((M1 \wedge M4) \vee M2) \wedge M5) \vee (M3 \wedge M6)) \wedge (M7 \vee M8)$
- Production capacity of each branch is defined in graph G
- How to determine the production capacity (e.g. pieces/min)?
- How to determine the production paths with the minimum transportation cost ?



Travelling problem

- The table below gives the expenses for persons W, X, Y and Z to travel to places A, B, C and D.
- The objective is to send each person to one of the four places such that all places will be visited, whilst the total costs are as small as possible.
- Translate this problem into a maximum flow problem and solve it with the maximum flow algorithm.

	A	B	C	D
W	16	12	11	12
X	13	11	8	14
Y	10	6	7	9
Z	11	15	10	8

Contents

Seminar assignment problem

- Consider the problem of assigning student to writing seminars.
- In class, we modeled a version of the problem where the total number of students exactly equals the number of available spots.
- In real applications, there are fewer students than available spots so some writing seminars are assigned fewer than 15 students.
- Model this problem as a minimum cost flow problem.
- Explain (in words and/or pictures) what are the vertices, supplies and demands, edges, and edge weights.

Blood donation problem

- Enthusiastic celebration of a sunny day at a prominent northeastern university has resulted in the arrival at the university's medical clinic of 169 students in need of emergency treatment.
- Each of the 169 students requires a transfusion of one unit of whole blood. The clinic has supplies of 170 units of whole blood.
- The number of units of blood available in each of the four major blood groups and the distribution of patients among the groups is summarized below.
  - type A patients can only receive type A or O;
  - type B patients can receive only type B or O;
  - type O patients can receive only type O;
  - type AB patients can receive any of the four types.

Blood type	A	B	O	AB
Supply	46	34	45	45
Demand	39	38	42	50

Give a max flow formulation that determines a distribution that satisfies the demands of a maximum number of patients.

Energy supplying problem

Dining Services wonders how little money they can spend on food while still supplying sufficient energy (2000 kcal), protein (55g), and calcium (800mg) to meet the minimum Federal guidelines and avert a potential lawsuit. A limited selection of potential menu items along with their nutrient content and maximum tolerable quantities per day is given in the table below.

	Energy (kcal)	Protein (g)	Calcium (mg)	Cost per serving (cents)
Oatmeal	110	4	2	3
Chicken	205	32	12	24
Eggs	160	13	54	13
Whole milk	160	8	285	9
Cherry pie	420	4	22	20
Pork with beans	260	14	80	19

Formulate a linear program to find the most economical menu.

Circulation problem

Given data

- A directed graph $G = (V, E)$ with source node s and sink node t
- lower bound $l(u, v)$ and upper bound $u(u, v) \geq 0$ for any edge $(u, v) \in E$
- cost function $a : E \longrightarrow \mathcal{R}$, i.e. $a(u, v) \geq 0$ for any edge $(u, v) \in E$

Objective

Send as much flow as possible with minimum cost such that

- $l(u, v) \leq f(u, v) \leq u(u, v)$, for all $(u, v) \in E$
- $\sum_{v \in V} f(v, u) = \sum_{v \in V} f(u, v)$, for $u \neq s, t$
- $\sum_{(u,v) \in E} a(u, v) f(u, v)$ should be minimized

Contents