ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ / ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΠΜΣ «ΠΛΗΡΟΦΟΡΙΚΗ»
2ος ΚΥΚΛΟΣ / Δ΄ ΕΞΑΜΗΝΟ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΑΛΓΟΡΙΘΜΟΙ ΓΙΑ ΕΥΡΕΣΗ ΤΗΣ ΜΕΓΙΣΤΗΣ ΡΟΗΣ ΣΕ ΕΝΑ ΔΙΚΤΥΟ ΡΟΗΣ

Επιβλέπων καθηγητής: κ. Ε. Φούντας

Γεώργιος Κορομηλάς: ΜΠΠΛ/07037

Πειραιάς 2011

Εισαγωγή

Το πρόβλημα μέγιστης ροής είναι να βρεθεί μία εφικτή ροή μέσω ενός δικτύου ροής μονής-πηγής, μονού-βυθού που να είναι μέγιστη. Η απλούστερη μορφή που η δήλωση μπορεί να πάρει θα ήταν κατά μήκος των γραμμών του: «Δίνεται μία λίστα σωλήνων, με διαφορετικές χωρητικότητες ροής. Αυτοί οι σωλήνες ενώνονται στα καταληκτικά σημεία τους. Ποια είναι η μέγιστη ποσότητα νερού που μπορείτε να κατευθύνετε από ένα δεδομένο σημείο εκκίνησης σε ένα δεδομένο σημείο κατάληξης;» ή ισοδύναμα «Μία εταιρεία έχει ένα εργοστάσιο που βρίσκεται στην πόλη X όπου κατασκευάζονται προϊόντα που χρειάζεται να μεταφερθούν σε ένα κέντρο διανομής στην πόλη Y. Σας δίνονται οι μονόδρομες οδοί που ενώνουν τα ζευγάρια πόλεων της χώρας, και ο μέγιστος αριθμός φορτηγών που μπορούν να κινηθούν κατά μήκος κάθε οδού. Ποιος είναι ο μέγιστος αριθμός φορτηγών που η εταιρεία μπορεί να στείλει στο κέντρο διανομής;»

Δίνεται ένα κατευθυνόμενο γράφημα G = (V, E) με ακέραιες χωρητικότητες, c : E, και έναν κόμβο πηγής s και έναν κόμβο βυθού t στο V. Σκοπός η εύρεση μίας συνάρτησης ροής στα τόξα που υπόκεινται σε «περιορισμούς διατήρησης» και «περιορισμούς χωρητικότητας». Ο περιορισμός διατήρησης για έναν κόμβο είναι ότι το άθροισμα της ροής των εισερχόμενων τόξων ισούται με το άθροισμα των εξερχόμενων τόξων για όλους τους κόμβους εκτός από την πηγή και το βυθό. Ο περιορισμός χωρητικότητας για ένα τόξο είναι ότι η ροή δεν είναι μεγαλύτερη από τη χωρητικότητα. Ζητούμενο είναι η μεγιστοποίηση της ροής από την πηγή (στο βυθό).

Ένα δίκτυο ροής G = (V, E) είναι ένα κατευθυνόμενο γράφημα στο οποίο κάθε άκρο (u,v) ϵ E έχει μη αρνητική χωρητικότητα c (u, v) ≥ 0.
Μία ροή στο G είναι μία συνάρτηση με πραγματικές τιμές f: V x V → R που ικανοποιεί τις ακόλουθες τρεις ιδιότητες:

Περιορισμός χωρητικότητας: Για όλα τα u, v ϵ V, απαιτείται να είναι f(u, v) ≤ c (u, v).

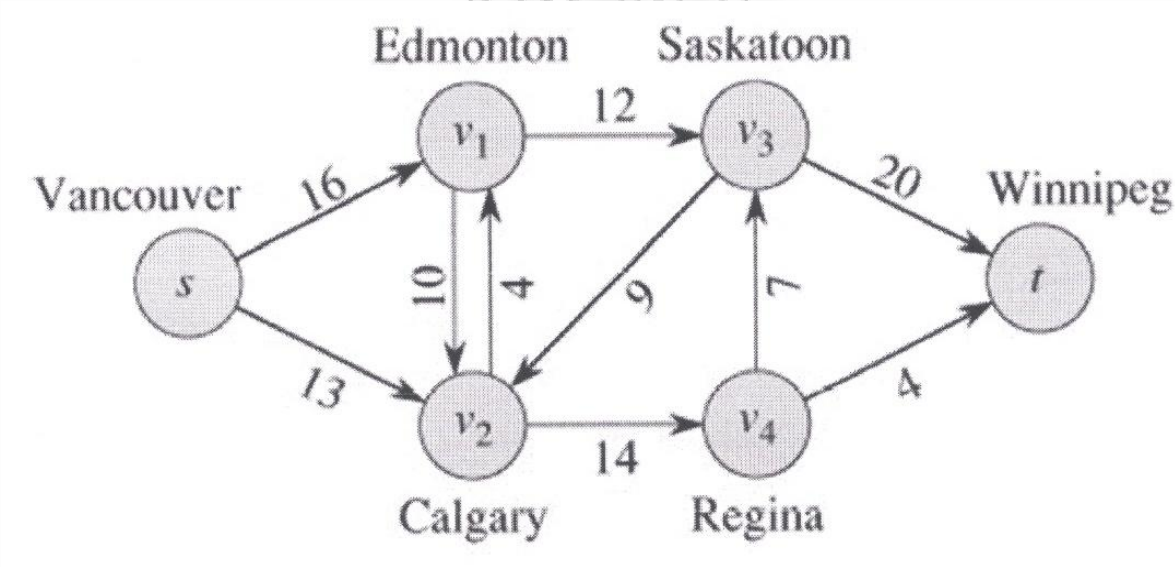
Λοξή συμμετρία: Για όλα τα u, v ϵ V, απαιτείται f(u, v) = - f(v, u).

Διατήρηση ροής: Για όλα τα u ϵ V – {s, t}, απαιτείται

$$\sum_{v \in V} f(u, v) = 0 .$$

Η ποσότητα f(u, v), η οποία μπορεί να είναι θετική, μηδέν, ή αρνητική, καλείται η **ροή** από την κορυφή u έως την κορυφή

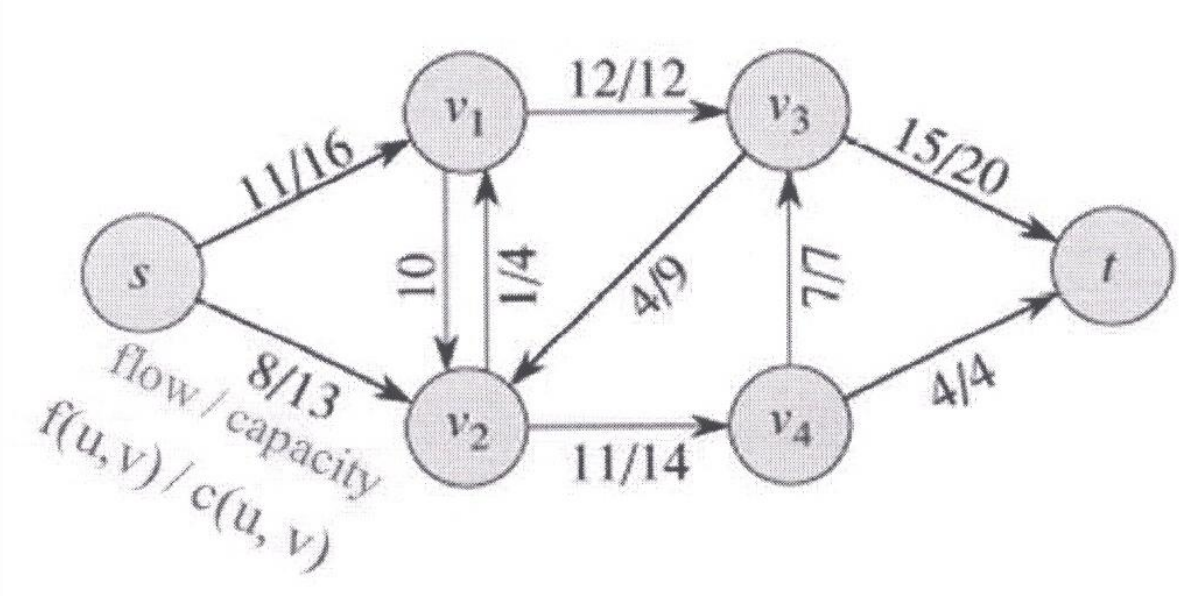
Ας μελετήσουμε το παρακάτω Γράφημα.



Εδώ, κάθε άκρο (u, v) έχει μία μη αρνητική χωρητικότητα c(u, v). Έχουμε μία πηγή s, και ένα βυθό t.

Μία ροή στο δίκτυο είναι μία συνάρτηση με ακέραιες τιμές **f** ορισμένη στα άκρα ως εξής

f(u, v) <= c(u, v).



Εικ. : Ένα παράδειγμα ενός δικτύου ροής με μία μέγιστη ροή. Η πηγή είναι s, και ο βυθός είναι t. Οι αριθμοί δηλώνουν ροή και χωρητικότητα. Παίρνουμε συνολικά την ποσότητα 19 ως τη μέγιστη ροή.

Μέθοδος Ford – Fulkerson

- Η μέθοδος Ford – Fulkerson υπολογίζει τη μέγιστη ροή σε ένα δίκτυο ροής. Εκδόθηκε το1955 από τους L. R. Ford, Jr. και D. R. Fulkerson.

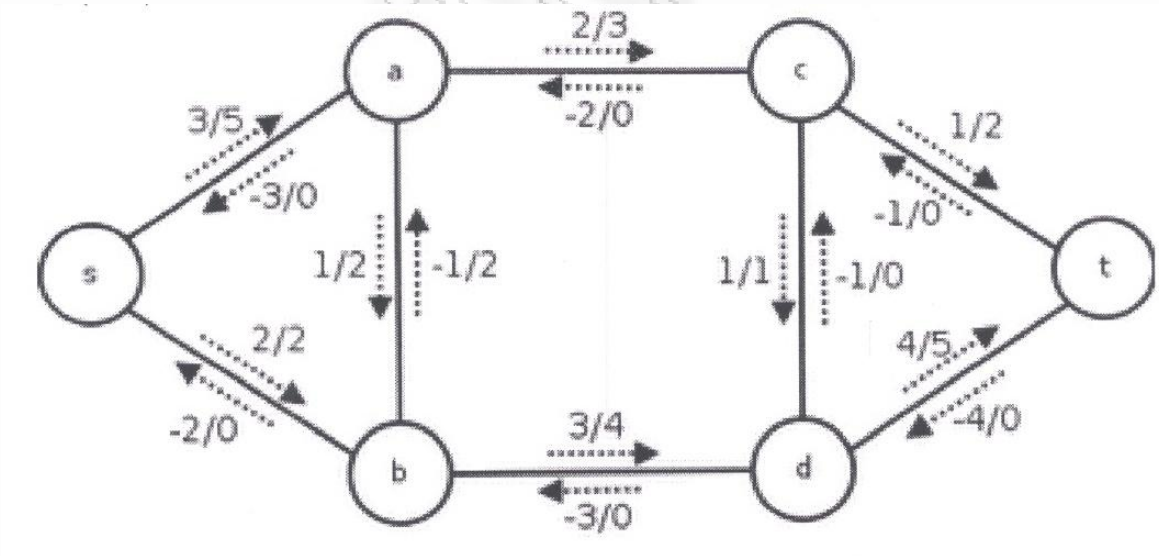
- Μπορούμε να την καλέσουμε μία «μέθοδο» παρά έναν «αλγόριθμο» επειδή περιλαμβάνει διάφορες εφαρμογές με διαφορετικούς χρόνους λειτουργίας. Η μέθοδος Ford – Fulkerson βασίζεται σε τρεις σημαντικές ιδέες που υπερβαίνουν τη μέθοδο και είναι σχετιζόμενες με πολλούς αλγόριθμους και προβλήματα ροής, **εναπομείναν δίκτυο**, **βέλτιστες διαδρομές,** και **τομές.**

Εναπομείναντα Δίκτυα

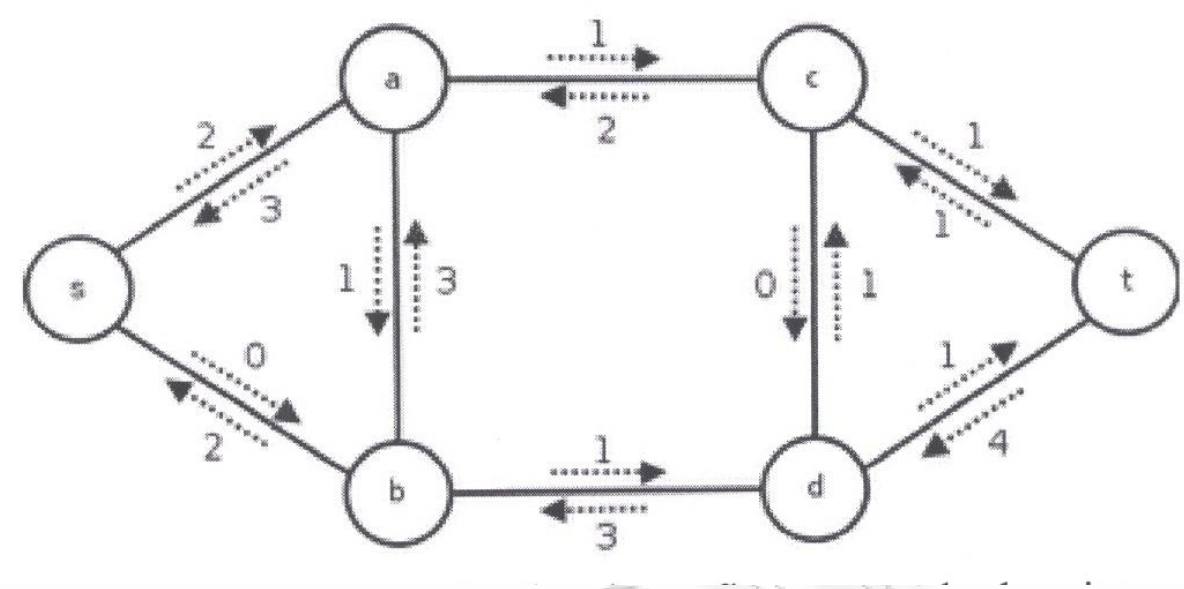
Υποθέστε ότι έχουμε ένα δίκτυο ροής G = (V, E) με πηγή s και βυθό t. Ας υποθέσουμε ότι f είναι μία ροή στο G, και ας συλλογιστούμε ένα ζευγάρι κορυφών u, v ϵ V. Η ποσότητα της πρόσθετης ροής που μπορούμε να ωθήσουμε από το u στο v προτού ξεπεράσουμε τη χωρητικότητα c(u, v) είναι η **εναπομείνασα χωρητικότητα** του (u, v), που δίνεται από:

$$c_f(u, v) = c(u, v) - f(u, v)$$

Για παράδειγμα, εάν c(u, v) = 16 και f(u, v) = 11, τότε μπορούμε να αυξήσουμε το f(u, v) κατά
c_f(u, v) = 5 μονάδες προτού ξεπεράσουμε τον περιορισμό χωρητικότητας στο άκρο (u, v).



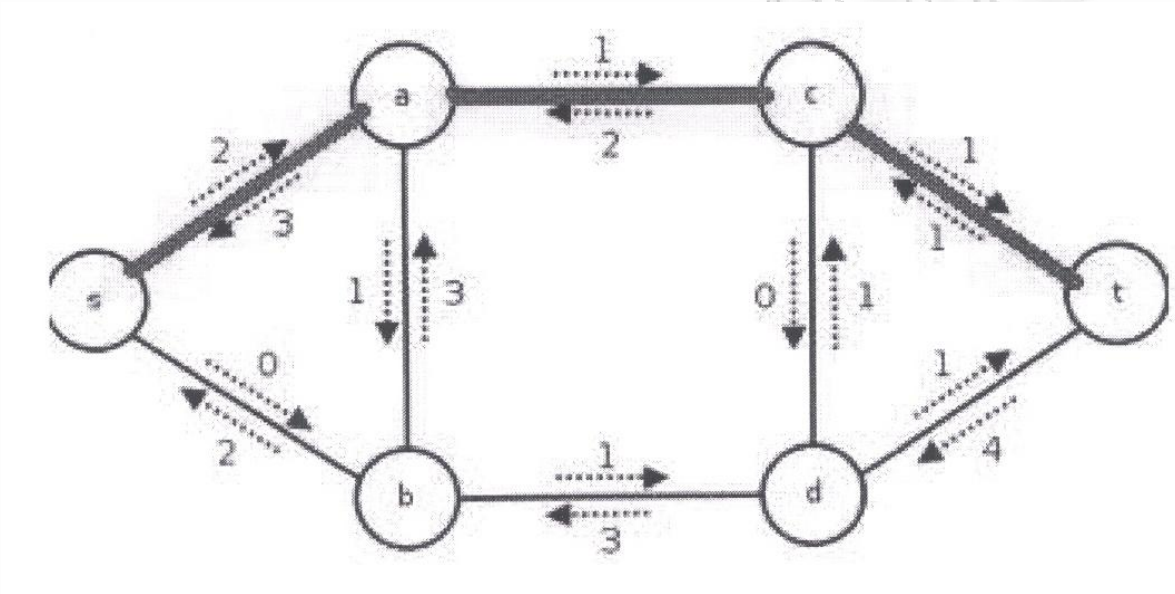
Εικ. : Ένα δίκτυο ροής που δείχνει ροή και χωρητικότητα.



Εικ: Εναπομείναν δίκτυο για το παραπάνω δίκτυο ροής που δείχνει τις εναπομείνασες χωρητικότητες.

Βέλτιστη διαδρομή

Με δεδομένα ένα δίκτυο ροής G = (V, E) και μία ροή f, μία βέλτιστη διαδρομή **p** είναι μία απλή διαδρομή από το s στο t στο εναπομείναν δίκτυο G_f.



Η σκιασμένη διαδρομή στην εικόνα δείχνει μία βέλτιστη διαδρομή σε ένα εναπομείναν δίκτυο.
Υπάρχει διαθέσιμη χωρητικότητα κατά μήκος των διαδρομών (s, a, c, t), (s, a, b, d, t) και (s, a, b, d, c, t), που είναι τότε οι βέλτιστες διαδρομές.

Η εναπομείνασα χωρητικότητα της πρώτης διαδρομής είναι:

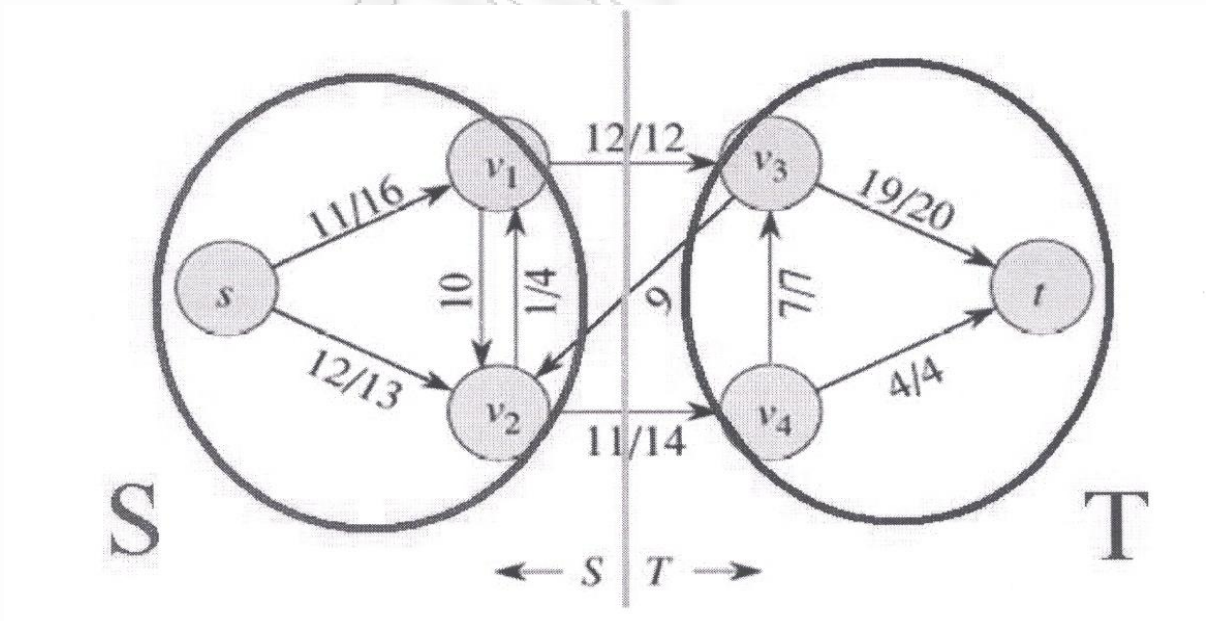
$$\begin{aligned} &\min(\, c(s,a) - f(s,a),\, c(a,c) - f(a,c),\, c(c,t) - f(c,t)\,) \\ &= \min(\, 5 - 3,\, 3 - 2,\, 2 - 1) \\ &= \min(\, 2,\, 1,\, 1) \\ &= 1 \end{aligned}$$

Τομή:

Στη θεωρία γραφημάτων, μία τομή είναι ένας διαχωρισμός των κορυφών ενός γραφήματος σε δύο ομάδες. Πιο επίσημα, ας επιτρέψουμε στο G(V, E) να δηλωθεί με ένα γράφημα.
Μία τομή είναι ένας διαχωρισμός των κορυφών V σε δύο ομάδες S και T.
Κάθε άκρο (u, v) ϵ E με u ϵ S και v ϵ T (ή u ϵ T και v ϵ S, στην περίπτωση ενός κατευθυνόμενου γραφήματος) λέγεται ότι διασχίζει την τομή και είναι ένα άκρο τομής.

Σε δίκτυο ροής, το μέγεθος μίας τομής ορίζεται ως το σύνολο των βαρών των άκρων που διασχίζουν την τομή από την πλευρά της πηγής προς την πλευρά του βυθού (αλλά όχι εκείνων που πηγαίνουν προς την άλλη κατεύθυνση).

Εδώ μπορούμε να διαιρέσουμε το γράφημα στον ακόλουθο διαχωρισμό έτσι ώστε:



$(s, v_1, v_2) \in S$
$(v_3, v_4, t) \in T$

Το **δίκτυο ροής** μέσω μίας τομής (S, T) ορίζεται ως:

$$f(S,T) = \sum f(u,v) - \sum f(v,u)$$

όπου u ϵ S και v ϵ T.

Η **Χωρητικότητα Τομής** (S, T) ορίζεται ως:

$$c(S,T) = \sum c(u,v)$$

όπου u ϵ S και v ϵ T.

Απλοϊκός Αλγόριθμος

```
FORD-FULKERSON-METHOD(G, s, t)
1   initialize flow f to 0
2       while there exists an augmenting path p
3           do augment flow f along p
4   return f
```

Η βασική ιδέα της μεθόδου Ford – Fulkerson είναι η επαναλαμβανόμενη εύρεση της βέλτιστης διαδρομής από το δίκτυο ροής μέχρι να μην υπάρχει διαδρομή.

Όταν έχουμε ένα άκρο u → v και έχουμε ελάχιστη ροή f(u, v), η εναπομείνασα χωρητικότητα ορίζεται ως:

$$c_f(u, v) = c(u, v) - f(u, v)$$

Για κάθε ροή, ορίζουμε μία εναπομείνασα χωρητικότητα σε αντίθετη κατεύθυνση (v → u) ως

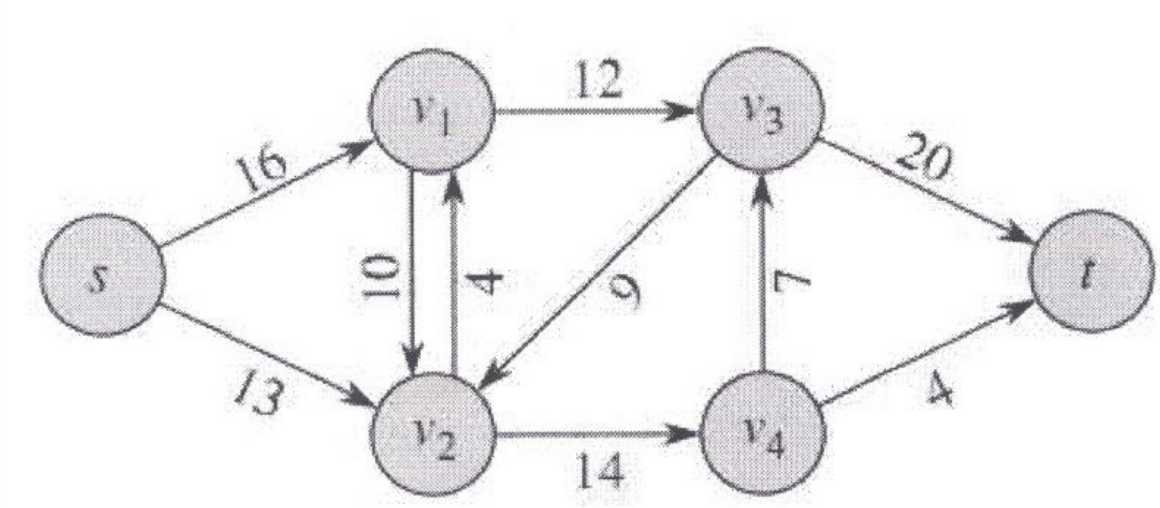
$$c_f(v, u) = c(v, u) - f(u, v)$$

$$\Rightarrow \; c_f(v, u) = c(v, u) + f(v, u)$$

(καθώς f (u, v) = - f (v, u) από την ιδιότητα λοξής συμμετρίας)

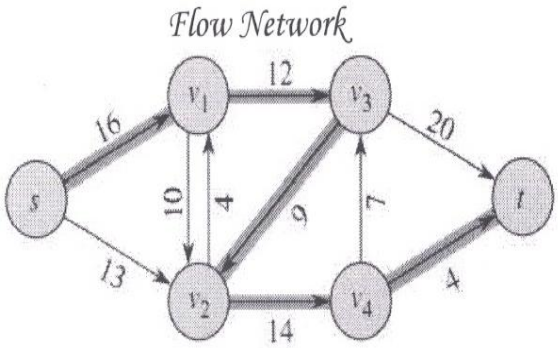
Το ακόλουθο παράδειγμα με εικόνα θα διαλευκάνει τυχόν απορίες:

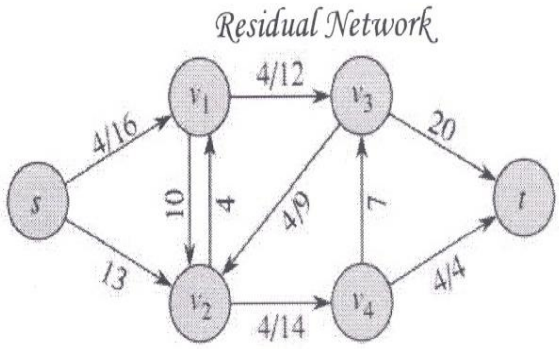
Ας υποθέσουμε, ότι θέλουμε να βρούμε τη μέγιστη ροή στο ακόλουθο γράφημα:



1ο Βήμα

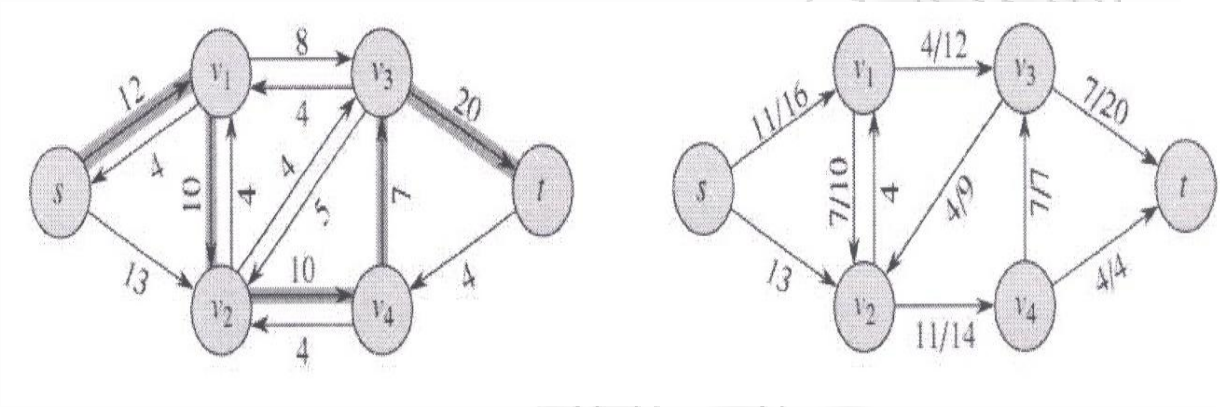
Η βέλτιστη διαδρομή **s** → **v1** → **v3** → **v2** → **v4** → **t** και το ελάχιστο-Κόστος από όλα τα άκρα της διαδρομής είναι **min**(16, 12, 9, 14, 4), και έτσι η τρέχουσα μέγιστη-ροή = 4.





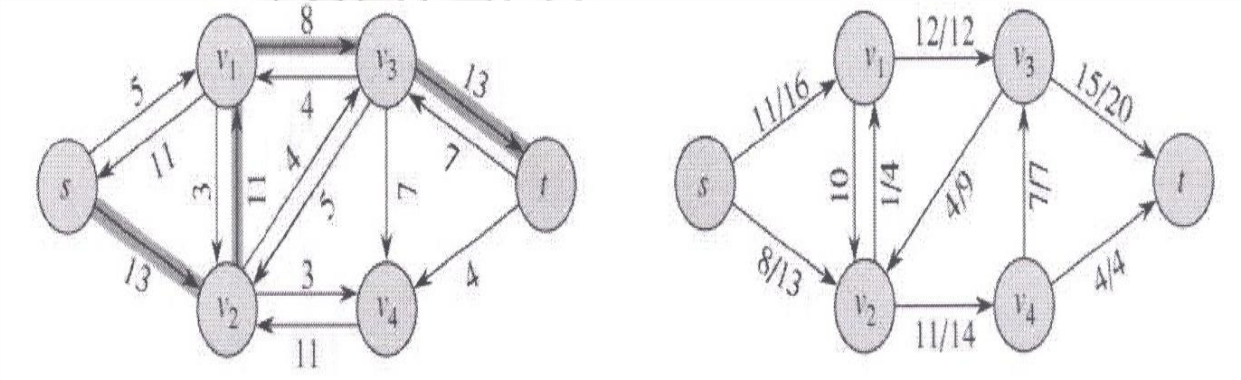
2ο Βήμα

Η βέλτιστη διαδρομή **s** → **v1** → **v2** → **v4** → **v3** → **t** από το εναπομείναν δίκτυο και έχουμε τη μέγιστη-ροή = 4 + 7 = 11



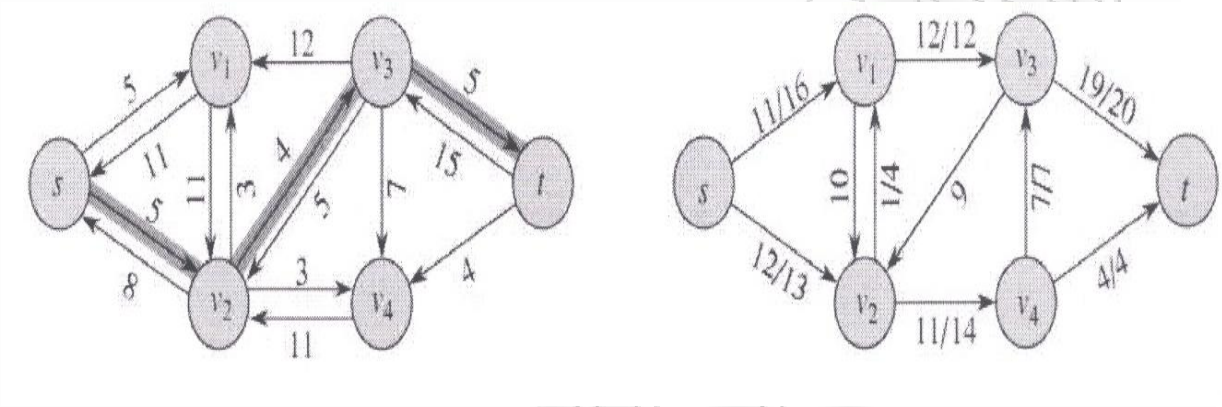
3ο Βήμα

Η βέλτιστη διαδρομή **s** → **v2** → **v1** → **v3** → **t** από το εναπομείναν δίκτυο και έχουμε τη μέγιστη-ροή = 11 + 8 = 19

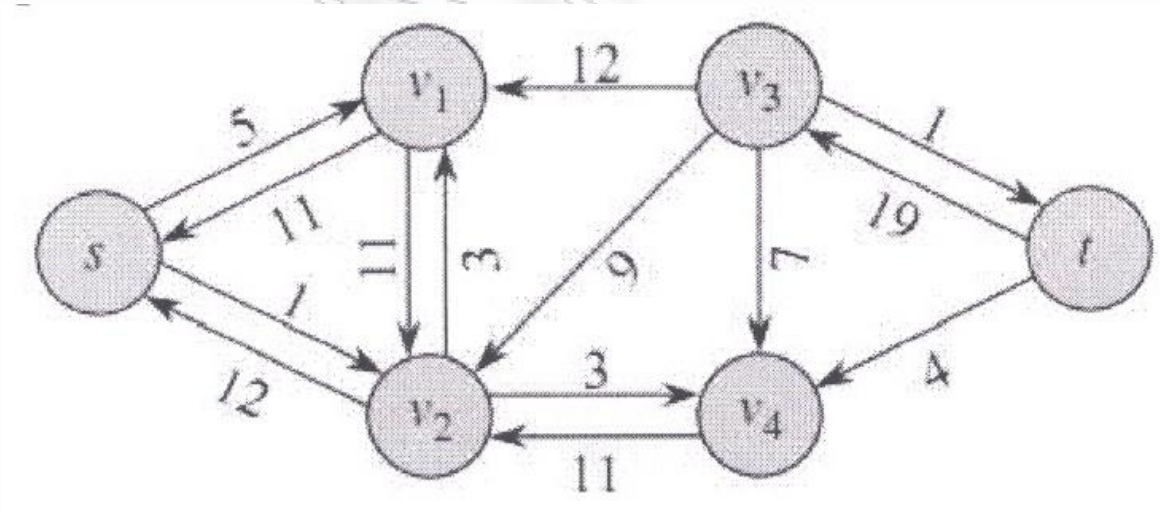


4ο Βήμα

Η βέλτιστη διαδρομή **s** → **v2** → **v3** → **t** από το εναπομείναν δίκτυο και έχουμε τη μέγιστη-ροή = 19 + 4 = 23



Αυτή είναι η τελευταία περίπτωση όπου μπορούμε να σταματήσουμε επειδή δεν υπάρχει βέλτιστη διαδρομή στο ακόλουθο εναπομείναν δίκτυο:



Έτσι παίρνουμε 23 ως τη μέγιστη ροή για αυτό το δίκτυο ροής.

Ο βασικός αλγόριθμος Ford – Fulkerson:

```
FORD-FULKERSON(G, s, t)
1  for each edge (u, v) ∈ E[G]
2       do f[u, v] ← 0
3          f[v, u] ← 0
4  while there exists a path p from s to t in the residual network G_f
5       do c_f(p) ← min {c_f(u, v) : (u, v) is in p}
6          for each edge (u, v) in p
7               do f[u, v] ← f[u, v] + c_f(p)
8                  f[v, u] ← −f[u, v]
```

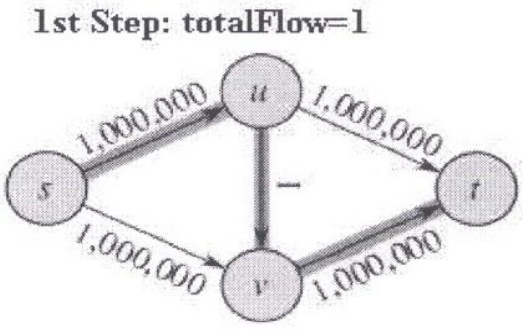
Μπορούμε να βρούμε διαδρομή από την πηγή (s) έως το βυθό (t) από κάθε γνωστό αλγόριθμο, όπως οι DFS, BFS κλπ.

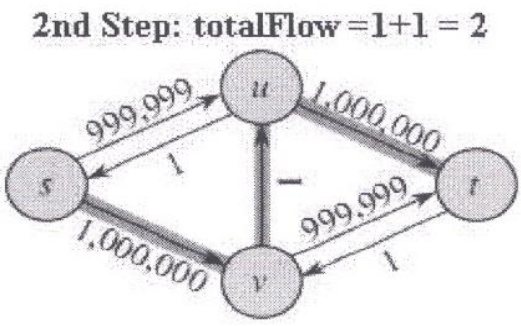
Ένας απλός κώδικας με χρήση του DFS για μέγιστη-ροή μπορεί να είναι όπως ακολούθως:

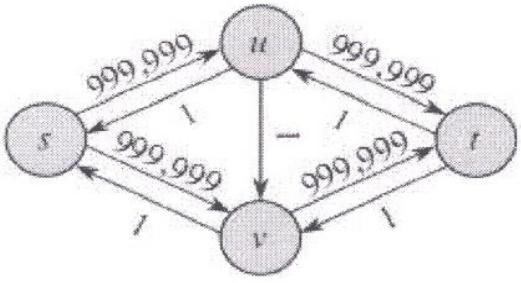
```
// Code of Max-Flow | Using DFS

#define MIN(a,b) ((a<b)?a:b)
#define INF 2147483647

int flow[101][101],prev[101],flowfound,no_of_Node;
void DFS(int source, int destination, int c)
{
        int i;
        if(source==destination)
                flowfound=c;

        if(flowfound!=0)
                return;

        for(i=1;i<=no_of_Node;i++)
        {
                if(prev[i]==0 && flow[source][i]!=0)
                {
                        prev[i]=source;
                        c=MIN(flow[source][i],c);       //#define MIN(a,b) ((a<b)?a:b)
                        DFS( i, destination, c );
                }
        }
}

int ford_fulkerson_using_DFS(int source, int destination)
{
        memset(prev,0,(no_of_Node+1)*sizeof(prev[0]));
        flowfound=0;
        prev[source]=-1;
        DFS( source, destination, INF );        //#define INF 2147483647
        return flowfound;
}

int maxflow(int source, int destination)
{
        int f,i,maxflow=0;

        while( (f=ford_fulkerson_using_DFS( source, destination )) != 0 )
        {
                i=destination;
                while(prev[i]!=-1)
                {
                        flow[prev[i]][i] -= f;
                        flow[i][prev[i]] += f;
                        i=prev[i];
                }
                maxflow+=f;
        }
        return maxflow;
}

int main()
{
```

```
        //freopen("in.txt","rt",stdin);

        int source,destination,no_of_Edge,i,j,a,b,c;
                scanf("%d",&no_of_Node);
                scanf("%d %d %d",&source,&destination,&no_of_Edge);

                for(i=1;i<=no_of_Node;i++)
                        for(j=1;j<=no_of_Node;j++)
                                flow[i][j]=0;

                for(i=1;i<=no_of_Edge;i++)
                {
                        scanf("%d %d %d",&a,&b,&c);
                        flow[a][b]+=c;
                        flow[b][a]+=c;
                }
                printf("Maximum Flow is : %d\n",maxflow(source,destination) );
return 0;
}
```

Ο χρόνος λειτουργίας του FORD-FULKERSON εξαρτάται από τον καθορισμό της βέλτιστης διαδρομής p. Εάν επιλεγεί ανεπαρκώς, ο αλγόριθμος πιθανόν να μην τερματιστεί: η τιμή της ροής θα αυξηθεί με επιτυχείς βελτιστοποιήσεις, αλλά δε χρειάζεται καν να συγκλίνει στην τιμή μέγιστης ροής. Δείτε το παράδειγμα στο ακόλουθο γράφημα:







Από αυτό το γράφημα μπορούμε να συνεχίσουμε όπως ενεργούμε στο $1^{ο}$ βήμα και συνέχεια στο $2^{ο}$ βήμα. Έτσι θα βρούμε 2.000.000 διαδρομές (!) και η συνολική ροή θα είναι = 2000000.

Αλλά εάν απλά επιλέξουμε τη διαδρομή **s** → **u** → **t** στο $1^{ο}$ βήμα και **s** → **v** → **t** στο $2^{ο}$ βήμα, παίρνουμε τη μέγιστη ροή από 2 μόνο διαδρομές!

Καλύτερη Εφαρμογή

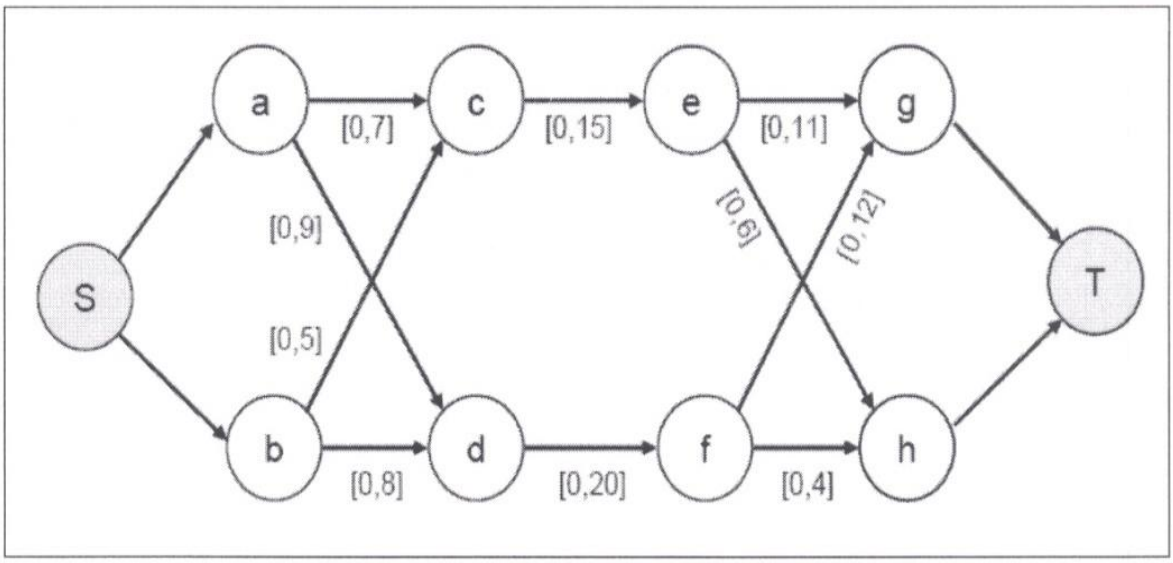
Το όριο στη μέθοδο FORD – FULKERSON μπορεί να βελτιωθεί εάν εφαρμόσουμε τον υπολογισμό της βέλτιστης διαδρομής p με μία πρώτη αναζήτηση κατά πλάτος, που σημαίνει ότι, εάν η βέλτιστη διαδρομή είναι μία συντομότερη διαδρομή από το s στο t σε ένα εναπομείναν δίκτυο, όπου κάθε άκρο έχει μονάδα απόστασης (βάρους). Καλούμε τη μέθοδο Ford – Fulkerson όπως υλοποιήθηκε στον αλγόριθμο Edmonds – Karp. Αυτός ο αλγόριθμος τρέχει σε πολυωνυμικό χρόνο.

Κώδικας με χρήση του BFS:

```
int bfs(int source,int destination)
{
	int u,v;
	queue<int>Q;
	memset(visited,0,sizeof(visited));
	visited[source]=1;
	pr[source]=0;
	Q.push(source);
	while(!Q.empty())
	{
		u=Q.front();
		Q.pop();
		if(u==destination) return 1;
		for(v=1;v<=V;v++)
		{
			if( (!visited[v]) && (capacity[u][v]-flow[u][v]>0) )
			{
				Q.push(v);
				pr[v]=u;
				visited[v]=1;
			}
		}
	}
	return 0;
}
int ford_fulkerson(int source,int destination)
{
	int u,increment,v,max_flow=0;
	while(bfs(source,destination))
	{
		increment=inf;
		for(v=destination;pr[v]>0;v=pr[v])
		{
			u=pr[v];
			increment=min(increment,capacity[u][v]-flow[u][v]);
		}
		for(v=destination;pr[v]>0;v=pr[v])
		{
			u=pr[v];
			flow[u][v]+=increment;
			flow[v][u]-=increment;
		}
		max_flow+=increment;
	}
	return max_flow;
}
```

Παράδειγμα Προβλήματος Μέγιστης Ροής

Ας μελετήσουμε το πρόβλημα μέγιστης ροής που απεικονίζεται στην κάτωθι εικόνα . Η μέγιστη ροή μεταξύ των κόμβων S και T χρειάζεται να καθοριστεί. Η ελάχιστη ροή τόξου και οι χωρητικότητες τόξου καθορίζονται ως τα χαμηλότερα και υψηλότερα όρια σε αγκύλες, αντιστοίχως.



Εικόνα: Παράδειγμα Προβλήματος Μέγιστης Ροής

Μπορείτε να επιλύσετε το πρόβλημα χρησιμοποιώντας είτε EXCESS=ARCS ή EXCESS=SLACKS. Συλλογιστείτε πρώτα τη χρήση της επιλογής EXCESS=ARCS. Μπορείτε να χρησιμοποιήσετε τον ακόλουθο κώδικα SAS για να δημιουργήσετε το σετ δεδομένων εισόδου.

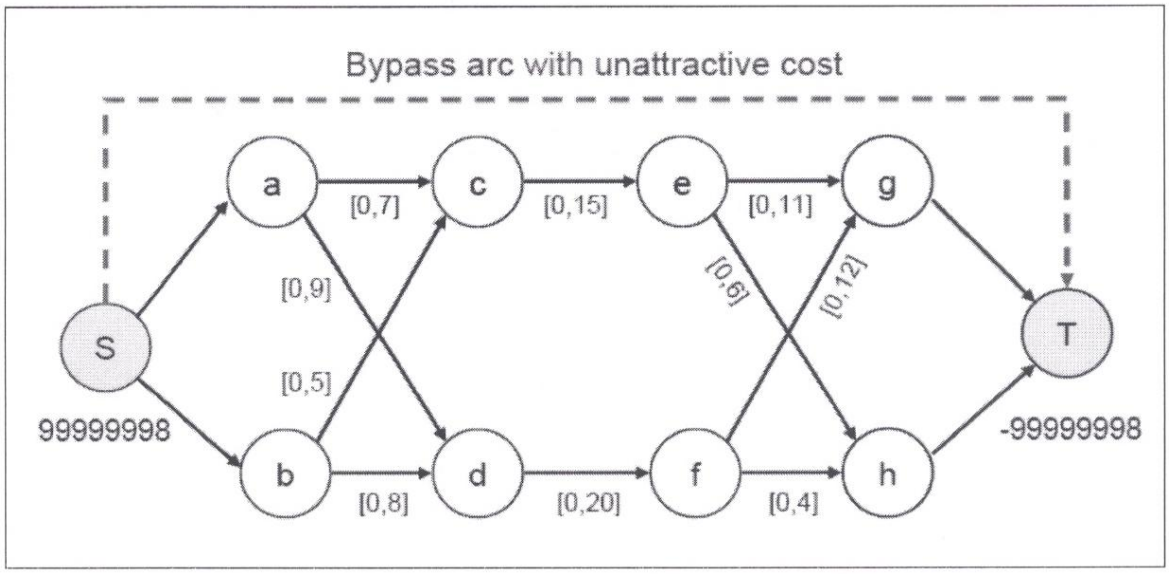
Κώδικας SAS

```
data arcs;
   input _from_ $ _to_ $ _cost_ _capac_;
datalines;
S a  .  .
S b  .  .
a c  1  7
b c  2  9
a d  3  5
b d  4  8
c e  5 15
d f  6 20
e g  7 11
f g  8  6
e h  9 12
f h 10  4
g T  .  .
h T  .  .
;
```

Μπορείτε να χρησιμοποιήσετε το ακόλουθο κάλεσμα της PROC NETFLOW για να επιλύσετε το πρόβλημα:

```
title1 'The NETFLOW Procedure';
proc netflow
   intpoint
   maxflow
   excess = arcs
   arcdata = arcs
   source  = S    sink = T
   conout  = gout3;
run;
```

Με την επιλογή EXCESS=ARCS καθορισμένη, το πρόβλημα μετατρέπεται εσωτερικά σε εκείνο που απεικονίζεται κάτωθι. Σημειώστε ότι υπάρχει ένα πρόσθετο τόξο από τον κόμβο πηγής στον κόμβο βυθού.



Εικόνα : Πρόβλημα Μέγιστης Ροής, Επιλογή EXCESS=ARCS καθορισμένη

Απεικόνιση σετ δεδομένων εξόδου SAS.

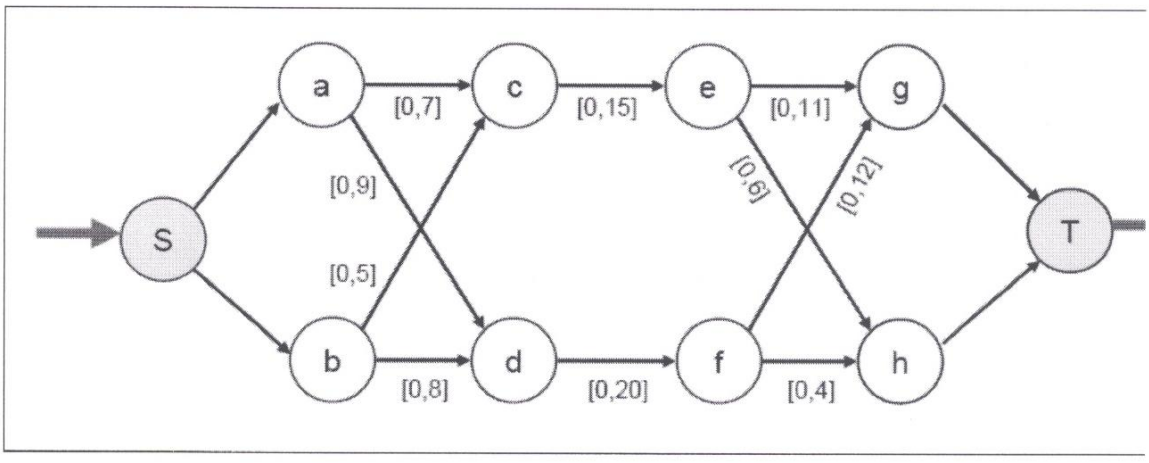
The NETFLOW Procedure

Obs	_from_	_to_	_cost_	_capac_	_LO_	_SUPPLY_	_DEMAND_	_FLOW_	_FCOST_
1	g	T	0	99999999	0	.	99999998	16.9996	0.0000
2	h	T	0	99999999	0	.	99999998	8.0004	0.0000
3	S	a	0	99999999	0	99999998	.	11.9951	0.0000
4	S	b	0	99999999	0	99999998	.	13.0049	0.0000
5	a	c	1	7	0	.	.	6.9952	6.9952
6	b	c	2	9	0	.	.	8.0048	16.0097
7	a	d	3	5	0	.	.	4.9999	14.9998
8	b	d	4	8	0	.	.	5.0001	20.0002
9	c	e	5	15	0	.	.	15.0000	75.0000
10	d	f	6	20	0	.	.	10.0000	60.0000
11	e	g	7	11	0	.	.	10.9996	76.9975
12	f	g	8	6	0	.	.	6.0000	48.0000
13	e	h	9	12	0	.	.	4.0004	36.0032
14	f	h	10	4	0	.	.	4.0000	40.0000

Μπορείτε να επιλύσετε το ίδιο πρόβλημα μέγιστης ροής, αλλά αυτή τη φορά με την EXCESS=SLACKS καθορισμένη. Ο κώδικας SAS είναι όπως ακολούθως:

```
title1 'The NETFLOW Procedure';
proc netflow
   intpoint
   excess  = slacks
   arcdata = arcs
   source  = S    sink = T
   maxflow
   conout  = gout3b;
run;
```

Με την επιλογή EXCESS=SLACKS καθορισμένη, το πρόβλημα μετατρέπεται εσωτερικά σε εκείνο που απεικονίζεται κάτωθι. Σημειώστε ότι ο κόμβος πηγής και ο κόμβος βυθού έχουν ο καθένας ένα “excess” τόξο μονού τερματισμού ενωμένο με αυτούς.



Εικόνα: Πρόβλημα Μέγιστης Ροής, Επιλογή EXCESS=SLACKS καθορισμένη

Η λύση, όπως απεικονίζεται στην Έξοδο, είναι η ίδια όπως πριν. Σημειώστε ότι η τιμή _SUPPLY_ του κόμβου πηγής Y έχει αλλάξει από 99999998 σε απών S, και η τιμή _DEMAND_ του κόμβου βυθού Z έχει αλλάξει από -99999998 σε απών D.

Έξοδος: Πρόβλημα Μέγιστης Ροής, Επιλογή EXCESS=SLACKS καθορισμένη

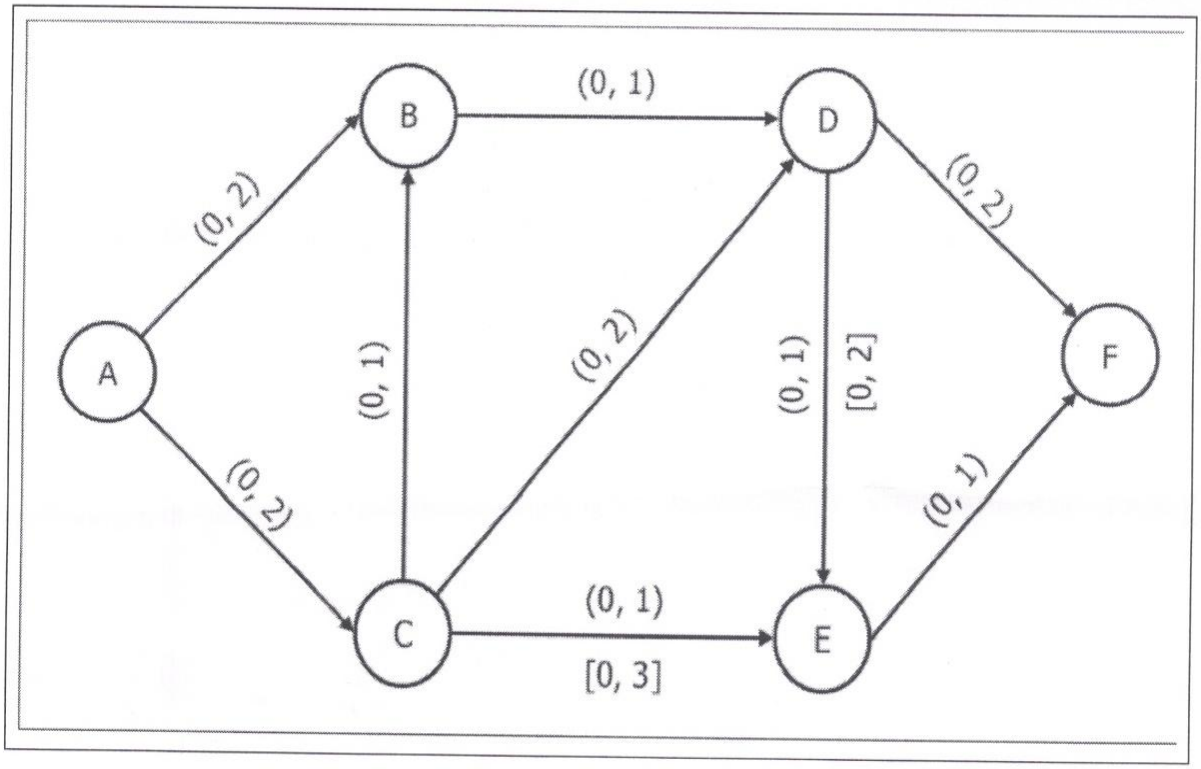
The NETFLOW Procedure

Obs	_from_	_to_	_cost_	_capac_	_LO_	_SUPPLY_	_DEMAND_	_FLOW_	_FCOST_
1	g	T	0	99999999	0	.	D	16.9993	0.0000
2	h	T	0	99999999	0	.	D	8.0007	0.0000
3	S	a	0	99999999	0	S	.	11.9867	0.0000
4	S	b	0	99999999	0	S	.	13.0133	0.0000
5	a	c	1	7	0	.	.	6.9868	6.9868
6	b	c	2	9	0	.	.	8.0132	16.0264
7	a	d	3	5	0	.	.	4.9999	14.9998
8	b	d	4	8	0	.	.	5.0001	20.0002
9	c	e	5	15	0	.	.	15.0000	75.0000
10	d	f	6	20	0	.	.	10.0000	60.0000
11	e	g	7	11	0	.	.	10.9993	76.9953
12	f	g	8	6	0	.	.	6.0000	48.0000
13	e	h	9	12	0	.	.	4.0007	36.0061
14	f	h	10	4	0	.	.	4.0000	40.0000

Παράδειγμα 5.12:

Γενικευμένα Δίκτυα: Πρόβλημα Μέγιστης Ροής

Σκεφτείτε το γενικευμένο δίκτυο που απεικονίζεται στην Εικόνα. Χαμηλότερα και υψηλότερα όρια της ροής απεικονίζονται στις παρενθέσεις πάνω από το τόξο, και το κόστος και ο πολλαπλασιαστής, όπου υπάρχουν, δηλώνονται σε αγκύλες κάτω από το τόξο.



Εικόνα 5.49: Γενικευμένο Πρόβλημα Μέγιστης Ροής

Μπορείτε να εισάγετε τα δεδομένα για το πρόβλημα χρησιμοποιώντας τον ακόλουθο κώδικα SAS:

```
data garcsM;
   input _from_ $ _to_ $ _upper_ _mult_;
datalines;
A B 2 .
A C 2 .
C B 1 .
B D 1 .
C D 2 .
C E 1 3
D E 1 2
E F 5 .
D F 2 .
;
```

Χρησιμοποιήστε την ακόλουθη κλήση της PROC NETFLOW:

```
title1 'The NETFLOW Procedure';
proc netflow
   arcdata = garcsM
   maxflow
   source  = A  sink = F
   conout  = gmfpout;
run;
```

Έξοδος: Γενικευμένο Πρόβλημα Μέγιστης Ροής: Βέλτιστη Λύση

The NETFLOW Procedure

Obs	_from_	_to_	_COST_	_upper_	_LO_	_mult_	_SUPPLY_	_DEMAND_	_FLOW_	_FCOST_
1	A	B	0	2	0	1	S	.	1.00000	0
2	C	B	0	1	0	1	.	.	0.00000	0
3	A	C	0	2	0	1	S	.	2.00000	0
4	B	D	0	1	0	1	.	.	1.00000	0
5	C	D	0	2	0	1	.	.	1.00000	0
6	C	E	0	1	0	3	.	.	1.00000	0
7	D	E	0	1	0	2	.	.	1.00000	0
8	E	F	0	5	0	1	.	D	5.00000	0
9	D	F	0	2	0	1	.	D	1.00000	0

Ένας Αναγνωριστικός Αλγόριθμος για το Πρόβλημα Μέγιστης Ροής Δικτύου

Παράρτημα Γ

Τα προβλήματα ροής δικτύου μπορούν να επιλυθούν με διάφορες μεθόδους. Στο Κεφάλαιο 8 εισάγαμε αυτό το θέμα διερευνώντας την ιδιαίτερη δομή των προβλημάτων ροής δικτύου και εφαρμόζοντας τη μέθοδο simplex σε αυτή τη δομή. Αυτό το παράρτημα συνεχίζει την ανάλυση των προβλημάτων δικτύου περιγράφοντας την εφαρμογή του αναγνωριστικού αλγόριθμου στο πρόβλημα μέγιστης ροής δικτύου. Οι αναγνωριστικές μέθοδοι παρέχουν μία εναλλακτική προσέγγιση για την επίλυση προβλημάτων δικτύου. Η βασική ιδέα πίσω από τη διαδικασία αναγνώρισης είναι η συστηματική απόδοση ετικετών στους κόμβους ενός δικτύου έως ότου η βέλτιστη λύση να επιτευχθεί.

Οι τεχνικές αναγνώρισης μπορούν να χρησιμοποιηθούν για την επίλυση πολυποίκιλων δικτυακών προβλημάτων, όπως τα προβλήματα συντομότερης διαδρομής, τα προβλήματα μέγιστης ροής, τα γενικά προβλήματα ροής δικτύου ελαχίστου κόστους και τα προβλήματα ελάχιστου spanning-tree. Ο σκοπός αυτού του παραρτήματος είναι να προβάλλει τη γενική φύση των αναγνωριστικών αλγορίθμων περιγράφοντας μία μέθοδο αναγνώρισης για το πρόβλημα μέγιστης ροής.

Γ.1 ΤΟ ΠΡΟΒΛΗΜΑ ΜΕΓΙΣΤΗΣ ΡΟΗΣ

Το πρόβλημα μέγιστης ροής παρουσιάστηκε στην Ενότητα 8.2 του συγγράματος. Εάν το u υποδηλώνει το ποσό της ύλης που στάλθηκε από τον κόμβο s, που καλούμε πηγή, στον κόμβο t, που καλούμε βυθό, το πρόβλημα μπορεί να διατυπωθεί ως εξής:

Μεγιστοποίηση του u, που υπόκειται σε:

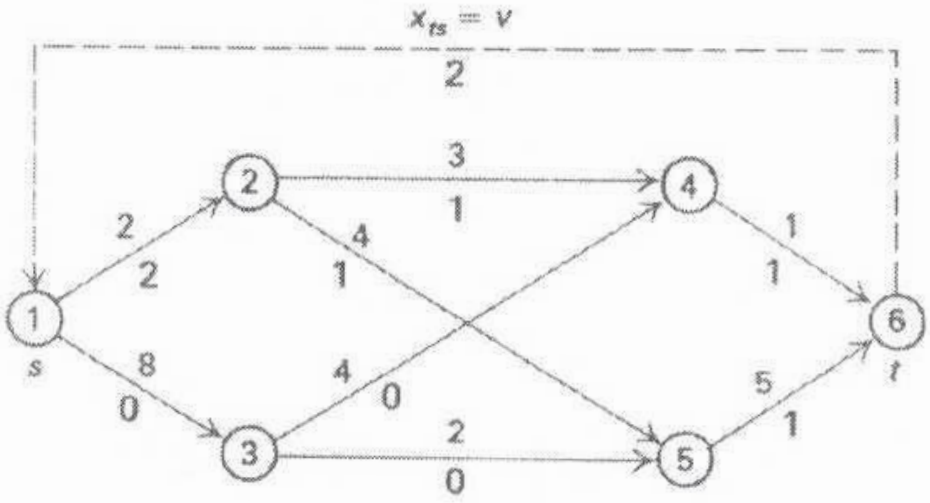
$$\sum_j x_{ij} - \sum_k x_{ki} = \begin{cases} v & \text{if } i = s, \\ -v & \text{if } i = t, \\ 0 & \text{otherwise,} \end{cases}$$
$$0 \le x_{ij} \le u_{ij}, \qquad (i = 1, 2, \ldots, n; \quad j = 1, 2, \ldots, n).$$

Υποθέτουμε ότι δεν υπάρχει τόξο από το t έως το s. Επίσης, το $u_{ij} = +\infty$ if arc i–j έχει απεριόριστη χωρητικότητα. Η ερμηνεία είναι ότι οι μονάδες του u προμηθεύονται στο s και καταναλώνονται στο t. Επιτρέποντας στο x_{ts}να δείχνει στη μεταβλητή u και αναδιατάσσοντας, βλέπουμε πως το πρόβλημα υποθέτει τη ακόλουθη ιδιαίτερη μορφή του γενικού προβλήματος δικτύου:

Μεγιστοποίηση του x_{ts}, που υπόκειται σε:

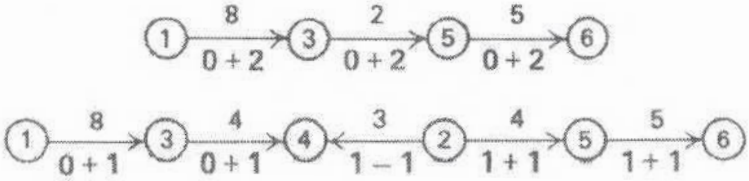
$$\sum_j x_{ij} - \sum_k x_{ki} = 0 \qquad (i = 1, 2, \ldots, n)$$
$$0 \le x_{ij} \le u_{ij} \qquad (i = 1, 2, \ldots, n; \quad j = 1, 2, \ldots, n).$$

Εδώ το τόξο t-s έχει εισαχθεί στο δίκτυο με το u_{ts}εξ ορισμού να είναι +∞, το x_{ts}απλά επιστρέφει τις μονάδες του u από τον κόμβο t πίσω στον κόμβο s, έτσι ώστε να μην υπάρχει τυπικός εξωτερικός εφοδιασμός ύλης. Ας ανακαλέσουμε το παράδειγμα (βλέπε Εικ.) που είχε τεθεί στο Κεφάλαιο 8 αναφορικά με ένα σύστημα αγωγών νερού. Οι αριθμοί πάνω από τα τόξα υποδηλώνουν τη χωρητικότητα ροής και οι έντονοι αριθμοί κάτω από τα τόξα προσδιορίζουν ένα δοκιμαστικό πλάνο ροής.



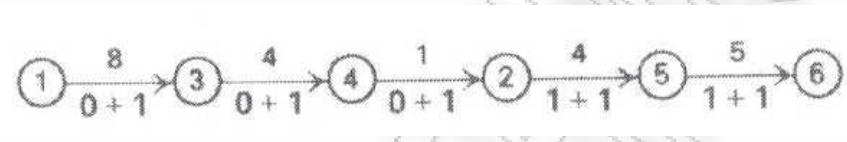
Εικόνα : Σύστημα αγωγών νερού

Ο αλγόριθμος εύρεσης της μέγιστης ροής βασίζεται στην παρατήρηση δύο τρόπων βελτίωσης της ροής σε αυτό το παράδειγμα. Οι ακόλουθες δύο «διαδρομές» φαίνονται στην ανωτέρω εικόνα.



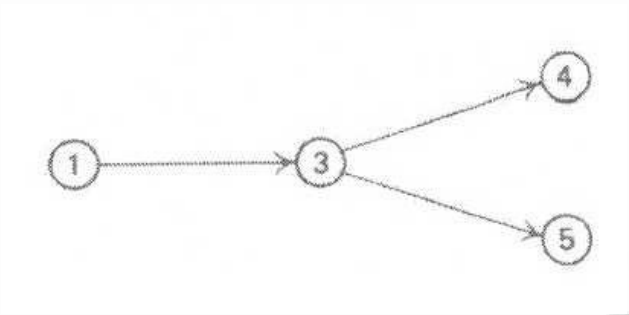
Στην πρώτη περίπτωση, η κατευθυνόμενη διαδρομή 1-3-6 έχει τη χωρητικότητα να μεταφέρει 2 επιπλέον μονάδες από την πηγή στο βυθό, όπως φαίνεται από τη χωρητικότητα του ασθενέστερου συνδέσμου της, τόξου 3-5. Σημειώστε ότι η προσθήκη αυτής της ροής δίνει ένα εφικτό πρότυπο ροής, καθώς 2 μονάδες έχουν προσθεθεί τόσο στην είσοδο όσο και στην έξοδο και των δύο κόμβων 3 και 5.Η δεύτερη περίπτωση δεν είναι μία κατευθυνόμενη διαδρομή από την πηγή στο βυθό καθώς το τόξο 2-4 εμφανίζεται με τον λάθος προσανατολισμό.

Σημειώστε, ωστόσο, ότι η προσθήκη μίας μονάδας ροής στα «μπροστινά τόξα» από 1 έως 6 και η αφαίρεση μίας μονάδας από το «αντίστροφο τόξο» 2-4 παρέχει μία εφικτή λύση, με αυξημένη ροή πηγής-προς-βυθό. Ισορροπία μάζας διατηρείται στον κόμβο 4, καθώς η μία επιπλέον μονάδα που στέλνεται από τον κόμβο 3 ακυρώνεται με τη μία λιγότερη μονάδα που στέλνεται από τον κόμβο 2. Παρομοίως, στον κόμβο 2 η μία επιπλέον μονάδα που στέλνεται στον κόμβο 5 ακυρώνεται με τη μία λιγότερη μονάδα που στέλνεται στον κόμβο 4. Η δεύτερη περίπτωση είναι εννοιολογικά ισοδύναμη με την πρώτη εάν δούμε φθίνουσα τη ροή κατά μήκος του τόξου 2-4 καθώς στέλνουμε ροή από τον κόμβο 4 πίσω στον κόμβο 2 κατά μήκος του αντίστροφου τόξου 4-2. Αυτό σημαίνει ότι, η μονάδα ροής από το 2 στο 4 αυξάνει την ενεργή χωρητικότητα από το τόξο «επιστροφής» 4-2 από 0 στο αρχικό δίκτυο σε 1. Την ίδια στιγμή, η δεύτερη περίπτωση γίνεται:

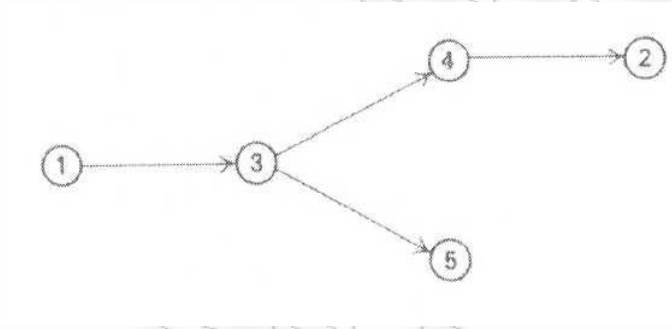


Τώρα και τα δύο παραδείγματα έχουν καθορίσει μία κατευθυνόμενη διαδρομή μεταφοράς ροής από την πηγή στο βυθό, που είναι, μία κατευθυνόμενη διαδρομή με τη χωρητικότητα να μεταφέρει πρόσθετη ροή. Ο αλγόριθμος μέγιστης ροής εισάγει επιστροφή, τόξα, όπως το 4-2 εδώ, και αναζητά διαδρομές μεταφοράς ροής. Χρησιμοποιεί μια διαδικασία κοινή στους αλγόριθμους δικτύου με «fanning our» από τον κόμβο πηγής μέσω ενός μόνο τόξου. Για το παράδειγμα των αγωγών νερού, το τόξο 1-2 είναι κορεσμένο και έχει μηδενική ενεργή χωρητικότητα, ενώ το τόξο 1-3 μπορεί να μεταφέρει 8 επιπλέον μονάδες, έτσι, μόνο ο κόμβος 3 αναγνωρίζεται από την πηγή.

Η διαδικασία επαναλαμβάνεται αναγνωρίζοντας όλους τους κόμβους με τη χωρητικότητα για παραλαβή ροής απευθείας από τον κόμβο 3 μέσω ενός μόνο τόξου. Σε αυτή την περίπτωση, οι κόμβοι 4 και 5 αναγνωρίζονται και οι αναγνωρισμένοι κόμβοι γίνονται:



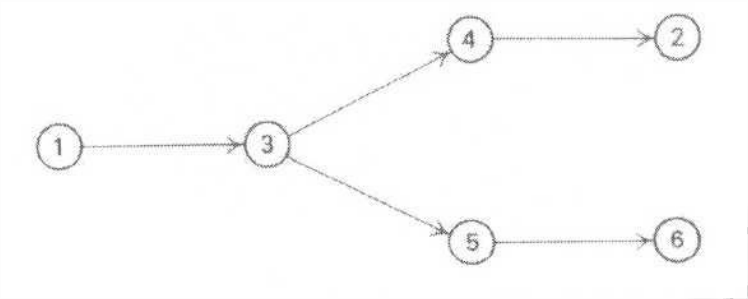
Πρόσθετοι κόμβοι μπορούν να αναγνωριστούν τώρα είτε από τον κόμβο 4 ή από τον κόμβο 5. Εάν κατόπιν επιλεγεί ο κόμβος 4, τότε ο κόμβος 2 αναγνωρίζεται, καθώς η ενεργή χωρητικότητα (χωρητικότητα επιστροφής) του τόξου 4-2 είναι θετική (ο κόμβος 6 δε μπορεί να αναγνωριστεί από τον κόμβο 4 καθώς το τόξο 4-6 είναι κορεσμένο). Οι αναγνωρισμένοι κόμβοι είναι:



Σε κάθε σημείο μέσα στον αλγόριθμο, μερικοί από τους αναγνωρισμένους κόμβους, εδώ οι κόμβοι 1, 3, και 4, θα έχουν ήδη χρησιμοποιηθεί για την αναγνώριση επιπλέον κόμβων. Θα πούμε ότι αυτοί οι κόμβοι έχουν σαρωθεί. Οποιοσδήποτε ασάρωτος κόμβος μπορεί να επιλεγεί και να σαρωθεί στη συνέχεια.

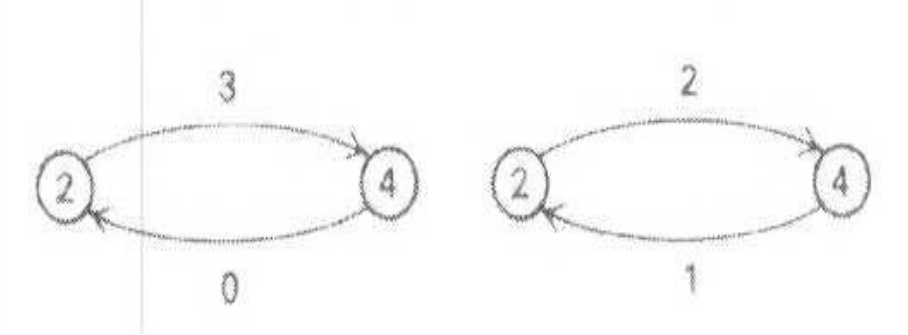
Η σάρωση του κόμβου 2 δεν παράγει επιπλέον αναγνωρίσεις, καθώς ο κόμβος 2 μπορεί να στείλει ροή απ'ευθείας μόνο στους κόμβους 1, 4 και 5, και αυτοί έχουν ήδη αναγνωριστεί. Ο κόμβος 6 αναγνωρίζεται όταν σαρώνεται ο κόμβος

5, και οι αναγνωρισμένοι κόμβοι είναι:



Μία διαδρομή μεταφοράς ροής έχει προσδιοριστεί, καθώς ο βυθός έχει αναγνωριστεί. Σε αυτή την περίπτωση, η εντοπισμένη διαδρομή είναι 1-3-5-6, που ήταν η πρώτη διαδρομή που δόθηκε παραπάνω. Η ροή κατά μήκος αυτής της διαδρομής ενημερώνεται αυξάνοντας τη ροή κατά μήκος κάθε τόξου της διαδρομής με τη χωρητικότητα ροής της διαδρομής, εδώ 2 μονάδες. Οι ενεργές χωρητικότητες τώρα υπολογίζονται εκ νέου και η διαδικασία αναγνώρισης επαναλαμβάνεται ξεκινώντας με αναγνωρισμένο μόνο τον κόμβο της πηγής.

Ακολουθεί μία τυπική περιγραφή του αλγόριθμου και μία λύση του παραδείγματος των αγωγών νερού. Παρατηρήστε ότι η τιμή της ροής στα τόξα δεν χρειάζεται να καταγραφεί βήμα-βήμα στον αλγόριθμο καθώς αυτή η πληροφορία υπολογίζεται εύκολα από τις ενεργές (ή αξιοποιήσιμες) χωρητικότητες των τόξων και τα τόξα επιστροφής τους. Για παράδειγμα παραπάνω, η ενεργή χωρητικότητα στο τόξο 2-4 είναι λιγότερη από την αρχική του χωρητικότητα.



Η διαφορά 3=2-1 πρέπει να είναι η τιμή ροής στο τόξο που κατέληξε με τη μειωμένη χωρητικότητα. Η ενεργή χωρητικότητα στο τόξο 4-2 έχει αυξηθεί από 0 σε 1 με προσθήκη της χωρητικότητας της επιστρεφόμενης ροής στο τόξο, όχι με φυσική προσθήκη ροής στο τόξο. Γενικά, αυτό συμβαίνει οποτεδήποτε η ενεργή χωρητικότητα υπερβαίνει την αρχική χωρητικότητα. Αυτά τα τόξα, κατά συνέπεια, δε μεταφέρουν ροή.

Όταν έχει βρεθεί μία διαδρομή μεταφοράς ροής από την πηγή έως το τέρμα, η οποία έχει τη δυνατότητα να μεταφέρει θ επιπλέον μονάδες, η ενεργή χωρητικότητα σε κάθε τόξο i-j αυτής της διαδρομής μειώνεται κατά θ. Την ίδια στιγμή, η ενεργή χωρητικότητα κάθε αντίστροφου τόξου j-I αυξάνεται κατά θ, καθώς τώρα μπορούν να χρησιμοποιηθούν για την εκτροπή (ή την αναδίπλωση) αυτών των μονάδων ροής.

Γ.2 ΑΛΓΟΡΙΘΜΟΣ ΜΕΓΙΣΤΗΣ ΡΟΗΣ-ΤΥΠΙΚΗ ΔΗΛΩΣΗ

Αρχικοποίηση

Υποθέστε ένα δοσμένο πλάνο εφικτής ροής x_{ij}(εάν κανένα δεν έχει δοθεί, χρησιμοποιήστε το εφικτό πλάνο με όλα τα x_{ij}=0). Η αρχική εφικτή χωρητικότητα u_{ij}^*στο τόξο i-j δίνεται υπολογίζοντας $u_{ij}^* = u_{ij} - x_{ij} + x_{ji}$ (δηλαδή, την αχρησιμοποίητη χωρητικότητα u_{ij}-x_{ij}συν τη χωρητικότητα επιστροφής x_{ji}).

Αναζήτηση Διαδρομής

Ξεκινήστε με τον κόμβο πηγής s και αναγνωρίστε (σημειώστε) κάθε κόμβο k με u_{sk}^*>0. Τότε, με τη σειρά, επιλέξτε κάθε αναγνωρισμένο κόμβο i που δεν έχει ήδη σαρωθεί (δηλαδή, δεν έχει χρησιμοποιηθεί για την αναγνώριση άλλων κόμβων) και αναγνωρίστε κάθε κόμβο j με u_{ij}^*>0 μέχρι είτε το t να έχει αναγνωριστεί είτε να μην είναι δυνατή περαιτέρω αναγνώριση.

Ενημέρωση Χωρητικότητας

Εάν το t έχει αναγνωρισθεί, τότε μία διαδρομή μεταφοράς ροής P θα έχει βρεθεί από την πηγή έως τον βυθό ($u_{ij}^*>0$ για κάθε τόξο i-j στη διαδρομή), και

$$\theta = \text{Min}\ \{u_{ij}^* \mid i-j \text{ in } P\}$$

είναι η χωρητικότητα ροής της διαδρομής. Για κάθε τόξο i-j της P, αλλάζει το u_{ij}^* σε u_{ij}^*-θ, και αλλάζει το u_{ji}^* σε u_{ji}^*+θ, δηλαδή, αυξάνεται η ενεργή χωρητικότητα της διαδρομής επιστροφής. (Προσθέτοντας ή αφαιρώντας τον πεπερασμένο θ σε οποιοδήποτε $u_{ij}^*=+\infty$ διατηρεί το u_{ij}^* στο $+\infty$. Εάν κάθε u_{ij}^* στο P είναι $+\infty$, τότε είναι θ=+∞ και η βέλτιστη ροή είναι άπειρη.)

Τερματισμός

Εάν η αναζήτηση διαδρομής ολοκληρωθεί χωρίς αναγνώριση του t, τότε τερματίστε. Το πρότυπο βέλτιστης ροής δίνεται από:

$$x_{ij} = \begin{cases} u_{ij} - u_{ij}^* & \text{if } u_{ij}^* > u_{ij}^*, \\ 0 & \text{if } u_{ij} \le u_{ij}^*. \end{cases}$$

Γ.3 ΕΠΙΛΥΣΗ ΠΑΡΑΔΕΙΓΜΑΤΟΣ

Η Εικόνα Γ.2 λύνει το παράδειγμα με τους σωλήνες νερού στην Εικ. Γ.1 με αυτό τον αλγόριθμο. Οι έλεγχοι δίπλα στις γραμμές δηλώνουν ότι ο κόμβος που αντιστοιχεί σε αυτή τη γραμμή έχει ήδη σαρωθεί. Οι πρώτοι τρεις πίνακες προσδιορίζουν λεπτομερώς την πρώτη εφαρμογή του βήματος αναζήτησης διαδρομής. Στον Πίνακα 1, ο κόμβος 1 έχει χρησιμοποιηθεί για να αναγνωριστεί ο κόμβος 3. Στον Πίνακα 2, ο κόμβος 3 έχει χρησιμοποιηθεί για να αναγνωριστούν οι κόμβοι 4 και 5 (καθώς $u_{34}^*>0$, $u_{35}^*>0$). Σε αυτό το σημείο μπορεί να χρησιμοποιηθεί είτε ο κόμβος 4 είτε ο 5 για την επόμενη αναγνώριση.

Η επιλογή είναι αυθαίρετη, και στον Πίνακα 3, ο κόμβος 5 έχει χρησιμοποιηθεί για να αναγνωριστεί ο κόμβος 6. Καθώς ο 6 είναι ο βυθός, η ροή ενημερώνεται. Η τελευταία στήλη του πίνακα καταγράφει πόσοι κόμβοι έχουν αναγνωριστεί.[1]Με υπαναχώρηση παίρνουμε μία διαδρομή μεταφοράς ροής P από την πηγή στο τέρμα. Για παράδειγμα, από τον Πίνακα 3, γνωρίζουμε ότι το 6 έχει αναγνωριστεί από το 5, το 5 από το 3, και το 3 από το 1, έτσι ώστε η διαδρομή είναι 1-3-5-6.

Initial capacity u_{ij}

		1	2	3	4	5	6
Source	1		2	8			
	2				3	4	
	3				4	2	
	4						1
	5						5
Destination	6						

Tableau 1

		1	2	3	4	5	6	*Labeled from*
Source	1			8				*Start*
	2	2			2	3		
	3				4	2		1
	4		1					
	5		1				4	
Destination	6				1	1		

[1]Για εφαρμογή στον υπολογιστή, μία άλλη στήλη μπορεί να διατηρηθεί στον πίνακα για καταγραφή της χωρητικότητας των διαδρομών μεταφοράς ροής καθώς αυτά επεκτείνονται. Με αυτό τον τρόπο, το θ μπορεί να μη χρειάζεται να υπολογιστεί τελικά.

Tableau 2

		1	2	3	4	5	6	Labeled from	
Source	1			8				Start	✓
	2	2			2	3			
	3				4	2		1	✓
	4		1					3	
	5		1				4	3	
Destination	6				1	1			

Εικόνα Γ.2Πίνακες για τον αλγόριθμο μέγιστης ροής

Tableau 3

	1	2	3	4	5	6	Labeled from		Path P
1			8				Start	✓	①
2	2			2	3				↓ ③
3				4	2	1	1	✓	↓
4		1				3	3		⑤
5		1				4	3	✓	↓ ⑥
6				1	1		5		

$\theta = \underset{i,j \text{ in } P}{\text{Min}} \{u^*_{ij}\} = \text{Min}\{8, 2, 4\} = 2$

Subtract 2 from u^*_{13}, u^*_{35}, and u^*_{56}.

Add 2 to u^*_{31}, u^*_{53}, and u^*_{65}.

Path 1–3–5–6

Tableau 4

	1	2	3	4	5	6	Labeled from		Path P
1			6				Start	✓	
2	2			2	3		4	✓	
3	2			4			1	✓	①
4		1					3	✓	↓
5		1	2			2	2	✓	③
6				1	3		5		↓ ⑤ ↓ ⑥

① $\theta = \underset{i,j \text{ in } P}{\text{Min}} \{u^*_{ij}\} = \text{Min}\{6,4,1,5,2\} = 1$

③ Subtract 1 from u^*_{13}, u^*_{34}, u^*_{42}, u^*_{25}, u^*_{56}.

Add 1 to u^*_{31}, u^*_{43}, u^*_{24}, u^*_{52}, u^*_{65}.

Path 1–3–4–2–5–6

Tableau 5

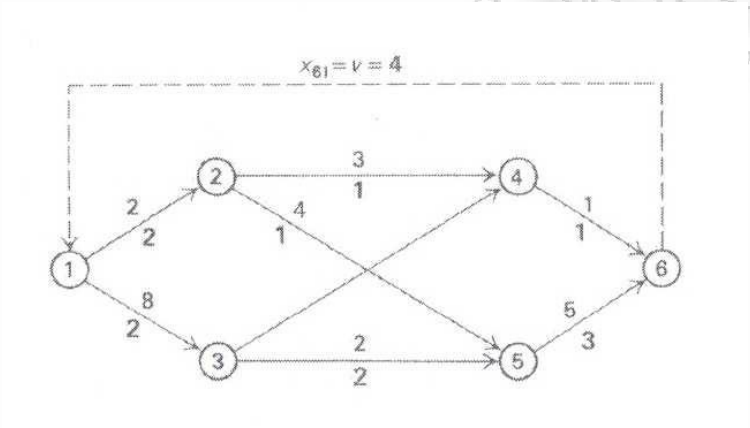
	1	2	3	4	5	6	Labeled from	
1			5				Start	✓
2	2			3	2			
3	3			3			1	✓
4			1				3	✓
5		2	2			2		
6				1	4			

Key: Entry in ith row and jth column of each tableau is u^*_{ij}. Blank entries denote zeros.

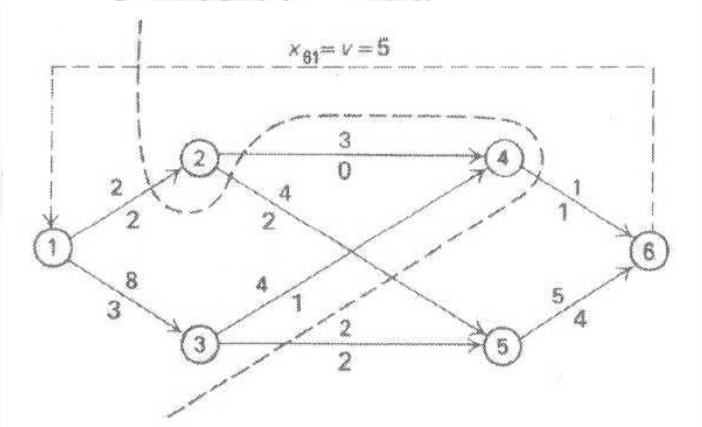
Εικόνα Γ.2 (Συνεχ.)

Ο Πίνακας 4 περιέχει τις ενημερωμένες χωρητικότητες και μία σύνοψη της επόμενης αναζήτησης διαδρομής, που χρησιμοποίησε τους κόμβους 1, 3, 4, 2, και 5 για αναγνώριση. Ο πέμπτος πίνακας περιέχει τις τελικές ενημερωμένες χωρητικότητες και την τελική αναζήτηση διαδρομής. Μόνο οι κόμβοι 1, 3, και 4 μπορούν να αναγνωριστούν σε αυτό τον πίνακα, οπότε ο αλγόριθμος έχει ολοκληρωθεί.

Η ροή σε οποιοδήποτε σημείο στον αλγόριθμο έχει υπολογισθεί αφαιρώντας τις ενεργές χωρητικότητες από τις αρχικές χωρητικότητες, u_{ij}-u_{ij}^*, και απορρίπτοντας τους αρνητικούς αριθμούς. Για τους Πίνακες 1, 2, ή 3, η ροή δίνεται στην Εικ. Γ.1. Αφότου έχετε κάνει τις δύο αλλαγές ροής που υποδεικνύονται για να καταλήξετε στον Πίνακα 4 και κατόπιν στον Πίνακα 5, οι λύσεις δίνονται σε δύο δίκτυα που εμφανίζονται στις Εικ. Γ.3 και Γ.4. Η βέλτιστη λύση στέλνει δύο μονάδες κατά μήκος καθεμίας από τις διαδρομές 1-2-5-6 και 1-3-5-6 και μία μονάδα κατά μήκος της διαδρομής 1-3-4-6.



Εικόνα Γ.3



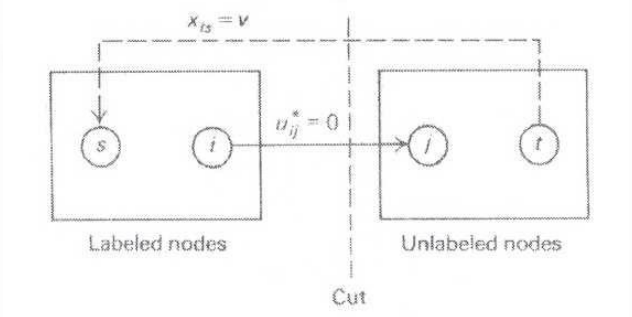
Εικόνα Γ.4

Γ.4 ΕΠΑΛΗΘΕΥΣΗ ΤΟΥ ΑΛΓΟΡΙΘΜΟΥ ΜΕΓΙΣΤΗΣ ΡΟΗΣ/ΕΛΑΧΙΣΤΗΣ ΤΟΜΗΣ

Η τελική λύση στο πρόβλημα παράδειγμα που φαίνεται στην Εικ. Γ.4 απεικονίζει ένα σημαντικό εννοιολογικό χαρακτηριστικό του προβλήματος μέγιστης ροής. Η έντονη διακεκομμένη γραμμή σε αυτή την εικόνα «τέμνει» το δίκτυο στα δύο, με την έννοια ότι χωρίζει τους κόμβους σε δύο ομάδες, η μία ομάδα περιλαμβάνει τον κόμβο πηγής και η άλλη ομάδα περιλαμβάνει τον κόμβο βυθού. Σημειώστε ότι το μέγιστο που μπορεί ποτέ να σταλλεί διαμέσου αυτής της τομής στο βυθό είναι δύο μονάδες από τον κόμβο 1 στον κόμβο 2, μία μονάδα από τον 4 στον 6, και δύο μονάδες από τον 3 στον 5, για ένα σύνολο από 5 μονάδες (το τόξο 2-5 συνδέει κόμβους που είναι και οι δύο στα δεξιά της τομής και δεν προσμετράται). Εφόσον κανένα πρότυπο ροής δε μπορεί ποτέ να στείλει περισσότερες από αυτές τις 5 μονάδες και η τελική λύση επιτυγχάνει αυτή την τιμή, πρέπει να είναι βέλτιστη.

Παρομοίως, όταν η τομή διαχωρίζει αναγνωρισμένους από μη αναγνωρισμένους κόμβους όταν ο αλγόριθμος τερματίζει (βλέπε Εικ. Γ.5) αυτό δείχνει ότι η τελική λύση θα είναι βέλτιστη.

Όσον αφορά τη μάζα, το u ισούται με τη ροή δικτύου από αριστερά προς τα δεξιά διαμέσου αυτής της τομής, έτσι ώστε το u δε μπορεί να είναι μεγαλύτερο από τη συνολική χωρητικότητα όλων των τόξων i-j που εικονίζονται. Στην πραγματικότητα, αυτή η παρατήρηση εφαρμόζεται σε οποιαδήποτε τομή διαχωρίζει



Εικόνα Γ.5

τον κόμβο πηγής s από τον κόμβο βυθού t.

≤ [*Κάθε εφικτή ροή u από τον κόμβο s έως τον κόμβο t*] [*Χωρητικότητα κάθε τομής που διαχωρίζει τον κόμβο s και τον κόμβο t*]

Όπως στο παράδειγμα που συζητήθηκε παραπάνω, η χωρητικότητα κάθε τομής ορίζεται ως η συνολική χωρητικότητα όλων των τόξων που κατευθύνονται «από αριστερά προς τα δεξιά» διαμέσου της τομής.

Εξ ορισμού από το σχεδιάγραμμα αναγνώρισης, ωστόσο, κάθε τόξο i-j που κατευθύνεται από αριστερά προς τα δεξιά διαμέσου της τομής που διαχωρίζει τους αναγνωρισμένους από τους μη αναγνωρισμένους κόμβους πρέπει να έχει $u_{ij}^{*}=0$, ειδάλλως το j θα είχε αναγνωριστεί από το i. Αυτή η παρατήρηση συνεπάγεται ότι

$$x_{ij} = u_{ij} \quad \text{and} \quad x_{ij} = 0,$$

καθώς εάν είναι είτε $x_{ij}<u_{ij}$ ή $x_{ji}<0$, τότε το $u_{ij}^{*}=(u_{ij}-x_{ij})+x_{ji}$ θα είναι θετικό. Κατά συνέπεια, η ροή δικτύου διαμέσου της τομής, και έτσι το u, ισούται με τη χωρητικότητα της τομής. Καθώς καμία ροή δε μπορεί να είναι καλύτερη, αυτό το πρότυπο ροής είναι βέλτιστο. Μπορούμε να αναδιατυπώσουμε αυτή την παρατήρηση με έναν λίγο διαφορετικό τρόπο. Η ανισότητα που δηλώθηκε παραπάνω δείχνει ότι κάθε εφικτή ροή u από την πηγή στο βυθό περιορίζεται προς τα πάνω από τη χωρητικότητα της κάθε τομής.

Κατά συνέπεια, η μέγιστη ροή από την πηγή στο βυθό περιορίζεται από τη χωρητικότητα οποιασδήποτε διαχωριστικής τομής που διαχωρίζει την πηγή από

το βυθό. Ειδικότερα, η μέγιστη ροή από την πηγή στο βυθό περιορίζεται προς τα πάνω από την ελάχιστη χωρητικότητα οποιασδήποτε τομής. Έχουμε μόλις δείξει, ωστόσο, ότι η μέγιστη ροή u ισούται με τη χωρητικότητα της τομής που διαχωρίζει τους αναγνωρισμένους και τους μη αναγνωρισμένους κόμβους όταν ο αλγόριθμος τερματίζει, επομένως, έχουμε επαληθεύσει το διάσημο θεώρημα μέγιστης ροής/ελάχιστης τομής.

Η μέγιστη ροή u από τον κόμβο s στον κόμβο t = *Η ελάχιστη χωρητικότητα οποιασδήποτε τομής διαχωρίζει τον κόμβο s και τον κόμβο t*

Υπάρχουν πολλές σημαντικές διακλαδώσεις σε αυτό το θεώρημα. Μπορεί να χρησιμοποιηθεί, για παράδειγμα, για την καθιέρωση κομψών αποτελεσμάτων στη συνδυαστική θεωρία. Παρόλο που τέτοια αποτελέσματα οδηγούν σε πολλές χρήσιμες εφαρμογές συνδυαστικής ανάλυσης, η ανάπτυξη τους είναι πέραν του σκοπού της κάλυψης μας εδώ. Έχουμε απλά δείξει πώς προκύπτει το θεώρημα μέγιστης ροής/ελάχιστης τομής, και χρησιμοποιήσαμε το θεώρημα για να δείξουμε ότι όταν ο αλγόριθμος αναγνώρισης μέγιστης ροής τερματίζει, έχει βρει τη μέγιστη δυνατή ροή από τον κόμβο πηγής στον κόμβο βυθού.

Τελικά, θα πρέπει να σημειώσουμε ότι η διαδικασία μέγιστης ροής κάποια στιγμή τερματίζει, όπως στο παραπάνω παράδειγμα, και δε συνεχίζει να βρίσκει διαδρομές μεταφοράς ροής από το s στο t για πάντα. Για ορισμένες περιπτώσεις αυτό είναι εύκολο να απεικονιστεί. Εάν η χωρητικότητα δεδομένων u_{ij}είναι ακέραιος και κάθε x_{ij}στο αρχικό πλάνο ροής είναι ακέραιος, τότε κάθε u_{ij}^{*}και θ που συναντάται θα είναι ακέραιος. Αλλά τότε, εάν η μέγιστη ροή u δεν είναι +∞, την προσεγγίζουμε σε ακέραια βήματα και πρέπει τελικά να τη φτάσουμε. Παρομοίως, εάν τα αρχικά δεδομένα και η ροή είναι κλασματικά, το u αυξάνει σε κάθε βήμα τουλάχιστον κατά ένα κλασματικό ποσό και η δομή τερματίζει σε έναν πεπερασμένο αριθμό βημάτων. Εάν τα δεδομένα είναι άρρητος, τότε μπορεί να αποδειχθεί ότι βρίσκει τη μέγιστη ροή σε έναν πεπερασμένο αριθμό βημάτων αρκεί οι κόμβοι κατά την αναζήτηση διαδρομής

να μελετώνται σε μία βάση πρώτος-αναγνωρισμένος-πρώτος-σαρωμένος.

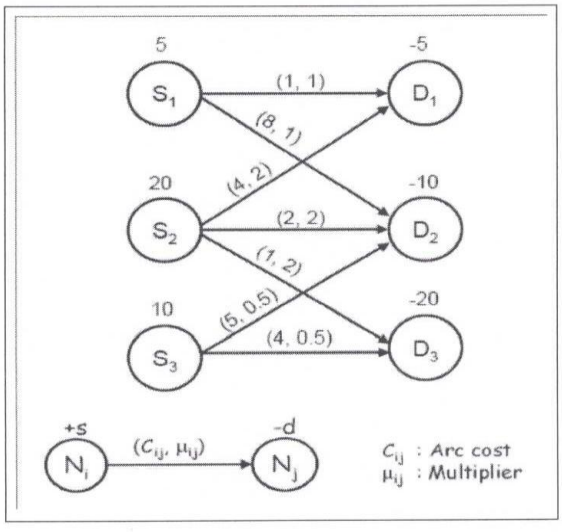
Σημειώστε ότι η εκκίνηση με ένα πλάνο ακέραιας ροής x_{ij}όταν όλα τα u_{ij}είναι ακέραιοι οδηγεί σε ακέραιο u_{ij}^*. Κατά συνέπεια, το τελικό (και βέλτιστο) πλάνο ροής πρέπει να είναι ακέραιο. Αυτό είναι ένα ιδιαίτερο παράδειγμα της ιδιότητας ακεραιότητας δικτύου που παρουσιάζεται στο Κεφάλαιο 8.

Παράδειγμα 5.11 : Γενικευμένα Δίκτυα: Χρησιμοποιώντας την ΥΠΕΡΒΟΛΗ= Επιλογή

Για γενικευμένα δίκτυα μπορείτε να προσδιορίσετε είτε ΥΠΕΡΒΟΛΗ=ΠΡΟΣΦΟΡΑ ή ΥΠΕΡΒΟΛΗ=ΖΗΤΗΣΗ για να δηλώσετε ποιοι κομβικοί περιορισμοί ροής σχετίζονται με μειωμένες μεταβλητές. Η προεπιλογή είναι ΥΠΕΡΒΟΛΗ=ΚΑΜΙΑ.

Χρησιμοποιώντας την ΥΠΕΡΒΟΛΗ=ΠΡΟΣΦΟΡΑ Επιλογή

Αναλογιστείτε το απλό δίκτυο που φαίνεται στην Εικόνα 5.48. Όπως μπορείτε να δείτε, το άθροισμα των θετικών supdem τιμών (35) ισούται με το απόλυτο άθροισμα των αρνητικών. Ωστόσο, τα τόξα που συνδέουν του κόμβους παροχής και ζήτησης έχουν ποικίλους πολλαπλασιαστές τόξων. Τώρα ας επιλύσουμε το πρόβλημα χρησιμοποιώντας την ΥΠΕΡΒΟΛΗ=ΠΡΟΣΦΟΡΑ επιλογή.



Εικόνα 5.48 :
Γενικευμένο Δίκτυο: Προσφορά=Ζήτηση

Μπορείτε να χρησιμοποιήσετε τον παρακάτω SAP κώδικα για να δημιουργήσετε σετ δεδομένων εισόδου:

```
data garcs;
   input _from_ $ _to_ $ _cost_ _mult_;
datalines;
s1 d1 1 .
s1 d2 8 .
s2 d1 4 2
s2 d2 2 2
s2 d3 1 2
s3 d2 5 0.5
s3 d3 4 0.5
;

data gnodes;
   input _node_ $ _sd_ ;
datalines;
s1  5
s2  20
s3  10
```

```
d1  -5
d2  -10
d3  -20
;
```

Για την επίλυση του προβλήματος, καλέστε την PROC NETFLOW:

```
title1 'The NETFLOW Procedure';
proc netflow
   arcdata  = garcs
   nodedata = gnodes
   excess   = supply
   conout   = gnetout;
run;
```

Η βέλτιστη λύση απεικονίζεται στην Έξοδο 5.11.1.

Έξοδος 5.11.1 :

Βέλτιστη Λύση που Λαμβάνεται με τη Χρήση της ΥΠΕΡΒΟΛΗ=ΠΡΟΣΦΟΡΑ Επιλογής

The NETFLOW Procedure

Obs	_from_	_to_	_cost_	_CAPAC_	_LO_	_mult_	_SUPPLY_	_DEMAND_	_FLOW_	_FCOST_
1	s1	d1	1	99999999	0	1.0	5	5	5	5
2	s2	d1	4	99999999	0	2.0	20	5	0	0
3	s1	d2	8	99999999	0	1.0	5	10	0	0
4	s2	d2	2	99999999	0	2.0	20	10	5	10
5	s3	d2	5	99999999	0	0.5	10	10	0	0
6	s2	d3	1	99999999	0	2.0	20	20	10	10
7	s3	d3	4	99999999	0	0.5	10	20	0	0

Σημείωση:

Εάν δεν προσδιορίσετε την ΥΠΕΡΒΟΛΗ = Επιλογή, ή εάν προσδιορίσετε την ΥΠΕΡΒΟΛΗ = ΖΗΤΗΣΗ Επιλογή, η διαδικασία δηλώνει ότι το πρόβλημα είναι ανέφικτο. Συνεπώς, στην περίπτωση προβλημάτων της πραγματικής ζωής, θα χρειαζόταν να έχετε λίγες περισσότερες λεπτομέρειες σχετικά με τον τρόπο που οι πολλαπλασιαστές τόξων καταλήγουν να επιδρούν στο δίκτυο --- είτε τείνουν να δημιουργούν υπερβολική ζήτηση είτε υπερβολική προσφορά.

Χρησιμοποιώντας την ΥΠΕΡΒΟΛΗ=ΖΗΤΗΣΗ Επιλογή

Αναλογιστείτε το προηγούμενο παράδειγμα αλλά με μία μικρή τροποποίηση: τα τόξα έξω από τον κόμβο S_1έχουν πολλαπλασιαστές 0.5 και τα τόξα έξω από τον κόμβο S_2έχουν πολλαπλασιαστές 1. Μπορείτε να χρησιμοποιήσετε τον παρακάτω SAP κώδικα για να δημιουργήσετε σετ δεδομένων εισόδου τόξου:

```
data garcs1;
   input _from_ $ _to_ $ _cost_ _mult_;
datalines;
s1 d1 1 0.5
s1 d2 8 0.5
s2 d1 4 .
s2 d2 2 .
s2 d3 1 .
s3 d2 5 0.5
s3 d3 4 0.5
;
```

Σημειώστε ότι το σετ δεδομένων κόμβου παραμένει αναλλοίωτο. Μπορείτε να καλέσετε την PROC NETFLOW για την επίλυση του προβλήματος:

```
title1 'The NETFLOW Procedure';

proc netflow
   arcdata  = garcs1
   nodedata = gnodes
   excess = demand
   conout   = gnetout1;
run;
```

Η βέλτιστη λύση απεικονίζεται στην Έξοδο 5.11.2.

Έξοδος 5.11.2:

Βέλτιστη Λύση που Λαμβάνεται με τη Χρήση της ΥΠΕΡΒΟΛΗ=ΖΗΤΗΣΗ Εντολής

The NETFLOW Procedure

Obs	_from_	_to_	_cost_	_CAPAC_	_LO_	_mult_	_SUPPLY_	_DEMAND_	_FLOW_	_FCOST_
1	s1	d1	1	99999999	0	0.5	5	5	5.0000	5.0000
2	s2	d1	4	99999999	0	1.0	20	5	0.0000	0.0000
3	s1	d2	8	99999999	0	0.5	5	10	0.0000	0.0000
4	s2	d2	2	99999999	0	1.0	20	10	5.0000	10.0000
5	s3	d2	5	99999999	0	0.5	10	10	0.0000	0.0000
6	s2	d3	1	99999999	0	1.0	20	20	15.0000	15.0000
7	s3	d3	4	99999999	0	0.5	10	20	10.0000	40.0000

ΠΕΡΙΕΧΟΜΕΝΑ