Το Πρόβλημα της Μέγιστης Ροής σε Δίκτυα

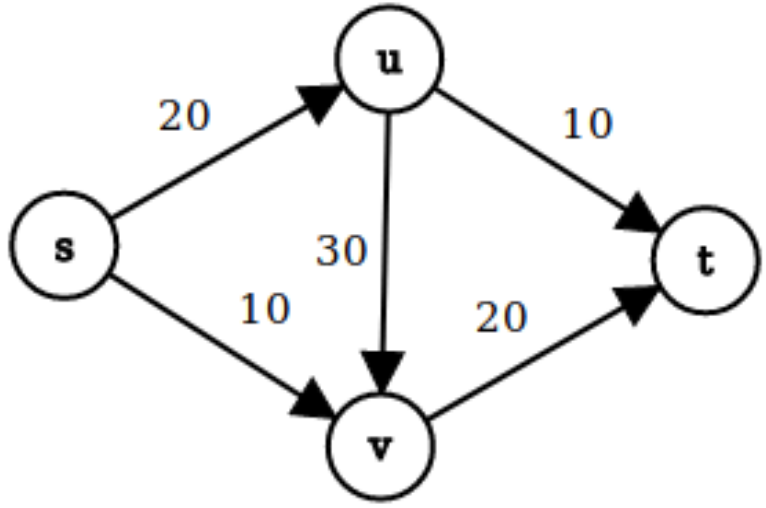
Μερκούρης Παπαμιχαήλ

Τμήμα Πληροφορικής και Τηλεπικοινωνιών
ΕΚΠΑ

2019

Το Πρόβλημα

Προσπαθούμε να μοντελοποιήσουμε το πρόβλημα μεγιστοποίησης της ροής σε ένα ηλεκτρικό κύκλωμα, ένα δίκτυο παροχής νερού ή ένα δίκτυο υπολογιστών.



1. Όλη η ροή παράγεται από την πηγή s και καταναλώνεται στο t.
2. Κάθε αγωγός έχει μια χωρητικότητα, την οποία δεν μπορούμε να ξεπεράσουμε.
3. Οι διασταυρώσεις δεν καταναλώνουν, ούτε παράγουν ροή.

Θέλουμε να μεγιστοποιήσουμε την ροή διατηρώντας τους περιορισμούς.

Ορισμοί

Θα λέμε δίκτυο N την δυάδα $N := < G = (V, E), c >$

1. Το G είναι ένα συνεκτικό κατευθυνόμενο γράφημα, όπου:

$$\exists s, t \in V : d_{in}(s) = 0, d_{out}(t) = 0$$

2. Η c είναι μια αντιστοίχηση της χωρητικότητας στις ακμές του G, $c : E \to \mathbb{N}^*$, δηλαδή:

$$\forall e \in E, c_e \geq 1, c_e \in \mathbb{N}$$

Για ένα δίκτυο $< G, c >$ θεωρούμε την ροή ως μια συνάρτηση $f : E \to \mathbb{N}$ για την οποία ισχύουν:

1. Συνθήκη Χωρητικότητας

$$\forall e \in E, f(e) \leq c_e$$

2. Διατήρηση Ροής

$$\forall v \in V \backslash \{s, t\} \sum_{(w,v) \in E} f(w, v) = \sum_{(v,w) \in E} f(v, w)$$

Το Πρόβλημα Τυπικά

Θεωρούμε σαν τιμή μίας ροής f,

$$val(f) = \sum_{(s,w)\in E} f(s,w)$$

Το πρόβλημα τυπικά είναι το εξής, δεδομένου ενός δικτύου $< G = (V, E), c >$. Θεωρούμε $|E|$ μεταβλητές, τις f_e:

$$max\ val = \sum_{(s,w)\in E} f_{sw}$$

τ. ω.

$$\begin{array}{ll} \forall e \in E & f_e \leq c_e \\ \forall v \in V \backslash \{s,t\} & \sum_{(w,v)\in E} f_{w,v} = \sum_{(v,z)} f_{vz} \\ \forall e \in E & f_e \geq 0, f_e \in \mathbb{N} \end{array}$$

Ένα Ακέραιο Γραμμικό Πρόγραμμα!

Ελάχιστος Γραμμικός Προγραμματισμός

Algorithm 1: Κεντρική Ιδέα Αλγορίθμου Simplex

```
1 s := μία αρχική λύση (συνήθως με τιμή 0)
2 while ∃s' ∈ N(s) με val(s') > val(s) do
3 |   s := s'
4 end
```

Με $N(s)$ συμβολίζουμε την γειτονιά της λύσης s.
Κατασκευάσουμε ακολουθία λύσεων:

$$s_1, s_2, ..., s_m, \quad val(s_1) < val(s_2) < ... < val(s_m)$$

αύξουσα ως προς την αξία τους.
Αυτό πρόκειται και να κάνουμε στο Πρόβλημα Μέγιστης Ροής.
Θα αρχίσουμε με

$$\forall e \in E \quad f(e) \leftarrow 0$$

και σταδιακά θα αυξάνουμε την ροή κατά μήκος μονοπατιών του δικτύου.

Υπολειπόμενο Δίκτυο

Για ένα δίκτυο $N =< G = (V, E), c >$, θεωρούμε το υπολειπόμενο δίκτυο για την ροή f,
$N^f :=< G^f := (V^f, E^f), c^f >$, όπου:

1. $V^f = V$
2. Αν $\exists e = (u, v), f(e) < c_e$, τότε:

$$e \in E^f, c_e^f = c_e - f(e)$$

Σε μια ακμή που έχει κορεστεί, μπορούμε μόνο να προσθέσουμε ροή. Ευθύγραμμες Ακμές του N^f.

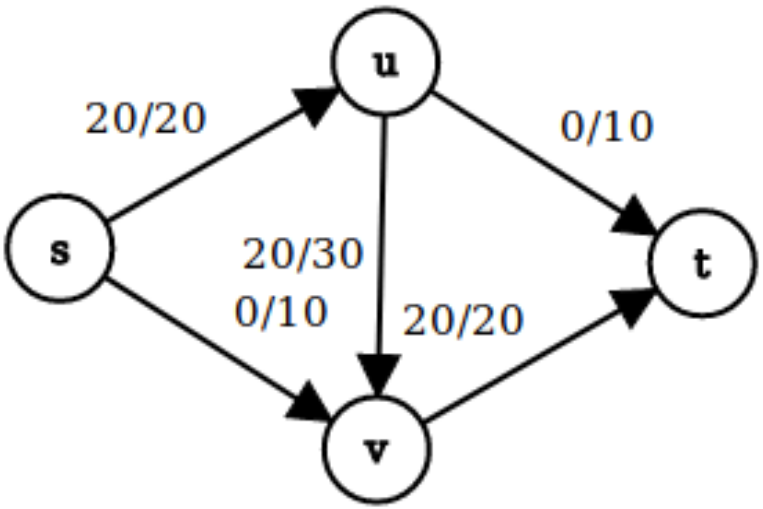
3. Αν $\exists e = (u, v) \in E, f(e) > 0$, τότε:

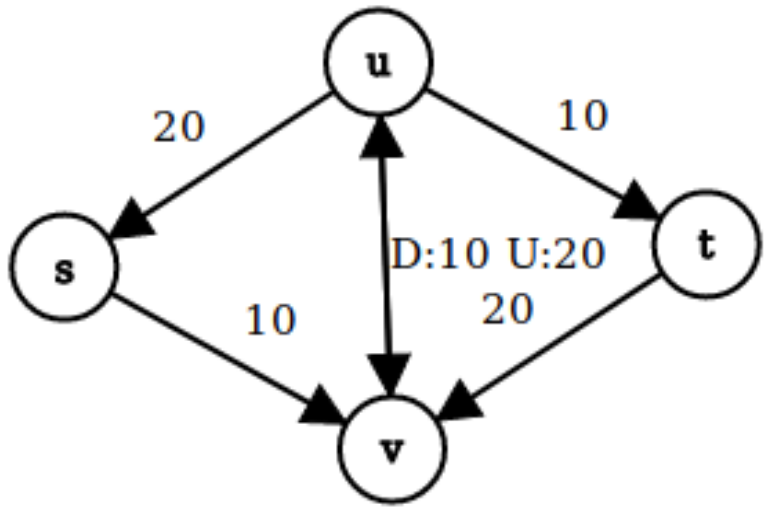
$$\bar{e} = (u, v) \in E, c_{f\,e} = f(e)$$

Σε μια ακμή που έχουμε ροή, μπορούμε να την αφαιρέσουμε. Ανάδρομες Ακμές του N^f.

Παράδειγμα

Αριστερά έχουμε ένα δίκτυο N, στο οποίο έχουμε εφαρμόσει ροή f. Δεξιά έχουμε το υπολειπόμενο δίκτυό του N^f



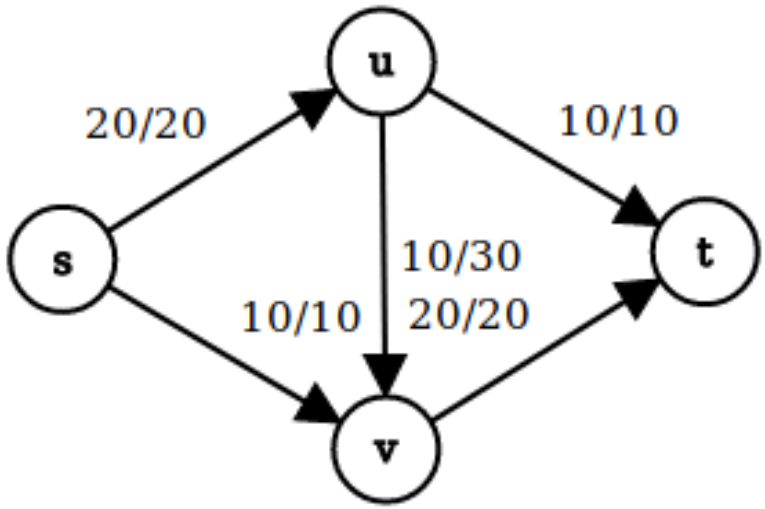


Σχεδιασμός Αλγορίθμου

Στο N^f έχουμε το μονοπάτι s, v, u, t:

- Η ακμές (s, v) και (u, t) είναι ευθύδρομες, με $c_{sv} = 10, c_{ut} = 10$, αυτό σημαίνει ότι μπορούμε να αυξήσουμε την ροή των $(s, v), (u, t)$ σε 10.
- Η ακμή (v, u) είναι ανάδρομη, με $c_{uv} = 20$, αυτό σημαίνει ότι μπορούμε να μειώσουμε την ροή κατά 20.

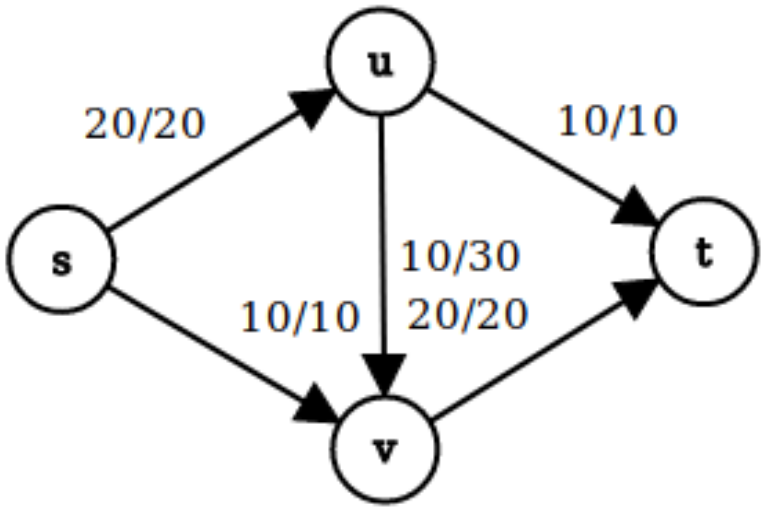
Εφαρμόζοντας τις αλλαγές στο N, έχουμε:

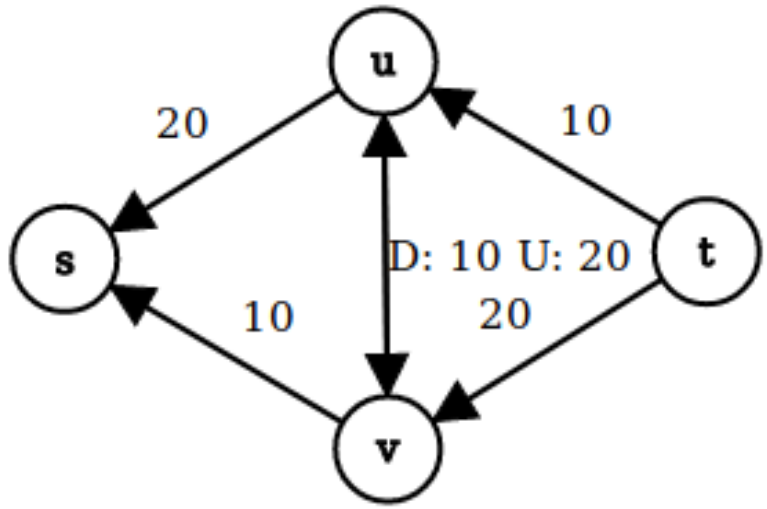


Τώρα έχουμε $val(f') = 30$

Σχεδιασμός Αλγορίθμου (συνέχεια)

Με τις παραπάνω αλλαγές φτάσαμε σε ροή $val(f') = 30$, σε αντίθεση με την ροή $val(f) = 20$ που είχαμε πριν. Ας ξαναφτιάξουμε το υπολειπόμενο γράφημα για την ροή f'.





Παρατηρούμε ότι στο υπολειπόμενο δίκτυο N^f δεν υπάρχει μονοπάτι $s - t$

Ας γράψουμε κάτω την ιδέα μας..

Algorithm 2: Max-Flow (Ford - Fulkerson)

Input : Ένα Δίκτυο $< G, c >$
Output: Η μέγιστη ροή στο $< G, c >$

1 $\forall e \in E, f(e) \leftarrow 0$
2 **while** $\exists$ *μονοπάτι* $s - t$ *στο* $< G^f, c^f >$ **do**
3 Έστω P ένα τέτοιο μονοπάτι
4 f' $\leftarrow$ Αύξησε(P)
5 Συγχώνευσε την f με την f'
6 Υπολόγισε το νέο $< G^f, c^f >$
7 **end**
8 **return** f

- Η μέθοδος Αύξησε(P) στην γραμμή 4 περιγράφει ακριβώς το τι κάναμε προηγουμένως στο παράδειγμα.
- Παρατηρήστε την ομοιότητα του Αλγορίθμος 1 με τον αλγόριθμο Αλγόριθμος 2

Αλγόριθμος Αύξησε

Θεωρούμε την μέθοδο $bottleneck(P)$, όπου:

$$bottleneck(P) := min_{e \in P} \ c_e^f$$

Η ελάχιστη υπολειπόμενη χωρητικότητα καταμήκος του P.

Algorithm 3: Αύξησε

Input : Ένα μονοπάτι P, ενός υπολειπόμενου δικτύου $< G^f, c^f >$

Output: Μία ροή f, πάνω στις ακμές του P

```
1 b ← bottleneck(P)
2 for e ∈ E(P) do
3 |   if e ευθύδρομη then
4 |   |   f(e) ← f(e) + b
5 |   else if e ανάδρομη then
6 |   |   f(e) ← f(e) − b
7 |   end
8 end
9 return f
```

Απόδειξη Ορθότητας (1)

Ισχυρισμός 1

Η f που επιστρέφει ο Αύξησε() είναι ροή, τηρεί τους περιορισμούς χωρητικότητας και διατήρησης.

Ισχυσρισμός 2

Σε κάθε ενδιάμεσο στάδιο του Αλγορίθμου Max-Flow οι τιμές της ροής και οι υπολειπόμενες χωρητικότητες στο $< G^f, c^f >$ είναι ακέραιοι αριθμοί.

Ισχυσρισμός 3

Έστω ότι f είναι μια ροή στο N και έστω P είναι ένα μονοπάτι στο N^f. Τότε:

$$val(f') = val(f) + bottleneck(P)$$

αφού $bottleneck(P) > 0$ έχουμε $val(f') > val(f)$

Απόδειξη Ορθότητας (2)

Θεώρημα 1
Ο Αλγόριθμος Max - Flow τερματίζει

Θεώρημα 2
Ο Αλγόριθμος Max - Flow τρέχει σε χρόνο $O(mC)$

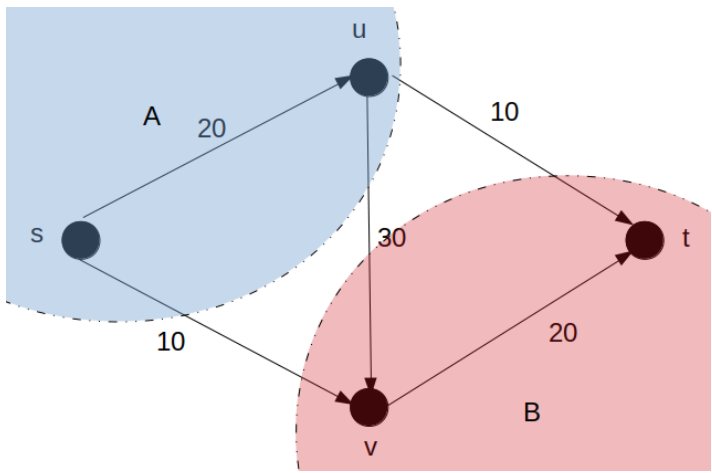
Τα άσχημα νέα..
Για το Θεώρημα 2 θα πρέπει να παρατηρήσουμε ότι ο Αλγόριθμος Max - Flow των Ford - Fulkerson είναι ψευδοπολυώνυμος, δηλαδή εκθετικός, αφού εξαρτάται από την τιμή του C και ότι από το μέγεθος της εισόδου της κωδικοποίησς του.

Βήματα προς την βελτιστότητα..

Έχουμε δείξει ότι ο Αλγόριθμος Max - Flow:

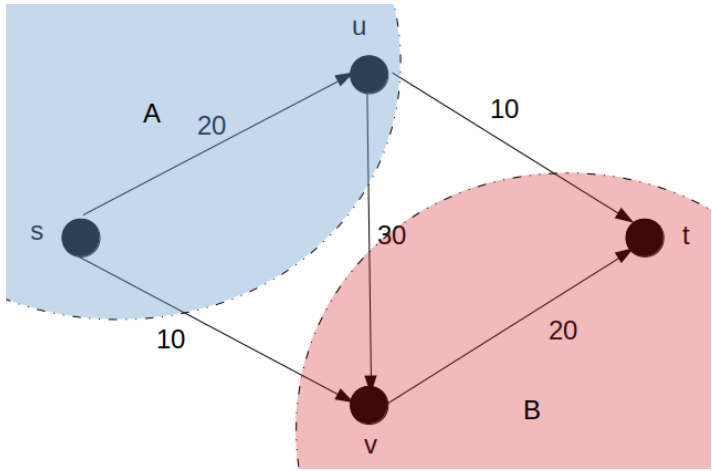
1. τερματίζει
2. επιστρέφει μία ορθή ροή για το δίκτυο $< G, c >$

Δεν έχουμε δείξει όμως ότι ο αλγόριθμος επιστρέφει μία βέλτιστη ροή.



Κάποια στιγμή η ροή θα πρέπει να περάσει από το A στο B, συνεπώς μια οποιαδήποτε διαμέριση του δικτύου θέτει ένα άνω όριο στη μέγιστη ροή

Αποκοπές: Ορισμοί



Ορισμός Αποκοπή

Έστω ένα δίκτυο $< G = (V, E), c >$ θα λέμε ότι μια διαμέριση του V σε A, B είναι μια αποκοπή στο δίκτυο, αν $s \in A, t \in B$ και την συμβολίζουμε με (A, B)

Ορισμός Αποκοπή

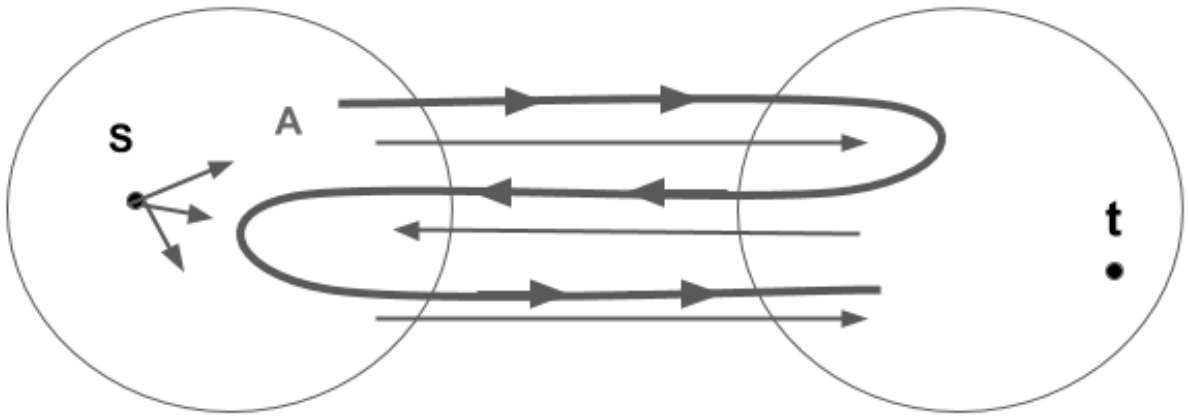
Όρίζουμε την χωρητικότητα σε μία αποκοπή (A, B):

$$c(A, B) := \sum_{\substack{e=(a,b)\in E,\\ a\in A, b\in B}} c_e$$

Αποκοπές 1

Ισχυρισμος 4

Έστω ένα δίκτυο $< G, c >$, μια ροή f και (A, B) κάποια αποκοπή του δικτύου. Τότε $val(f)) = f^{out}(A) - f^{in}(A)$



Το S παράγει κάποια ροή. Όλη από αυτή την ροή θα περάσει από το A στο B. Κάποια ροή όμως θα περάσει δύο φορές: Θα περάσει από το A στο B και μετά στο A (και μετά πάλι στο B). Οπότε απλα θα αφαιρέσουμε την εισερχόμενη ροή στο A (για να μην την μετρήσουμε δύο φορές).

Αποκοπές 2

Ισχυρισμος 5

Έστω ένα δίκτυο $< G, c >$ και μία ροή f και (A, B) κάποια αποκοπή του δικτύου. Τότε $val(f) = f^{in}(B) - f^{out}(B)$.

Θεώρημα

Έστω ένα δίκτυο $< G, c >$ και μία ροή f και μια αποκοπή (A, B). Τότε τα $val(f) \leqslant c(A, B)$.

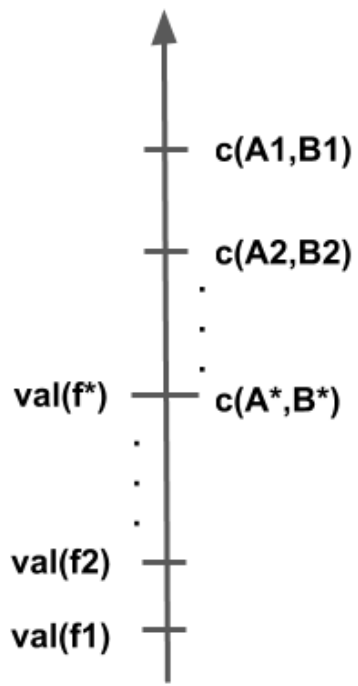
Απόδειξη:

$val(f) = f^{out}(A) - f^{in}(A) \leqslant f^{out}(A)$

$f^{out}(A) = \sum_{e=(a,b)\in E, a\in A} f(e) \leqslant \sum_{e=(a,b)\in E, a\in A} c(e) = c(A, B)$

Ενα επιχείρημα για την βελτιστοποίηση

Το παραπάνω θεώρημα μας λεεί ότι για ένα δίκτυο
$< G, c >= N$
$\forall f$ ροή του N, $\forall (A, B)$ αποκοπή του N $val(f) \leqslant c(A, B)$



Με άλλα λόγια θα ξέρουμε ότι έχουμε βρεί μια μέγιστη ροή $f^\star$ ανν βρούμε μια αποκοπή $(A^\star, B^\star)$ τ.ω. $val(f^\star) = c(A^\star, B^\star)$

Ομοίως για να βρούμε μια ελάχιστη αποκοπή $(A^\star, B^\star)$, αρκεί να βρούμε μια ροή τ.ω. $c(A^\star, B^\star) = val(f^\star)$

Απόδειξη Βελτιστοποίσης

Θεώρημα

Έστω ένα δίκτυο $N =< G, c >$ και μια ροή στο N, f^*. Έστω το υπολοιπόμενο δίκτυο N^{f^*}. Τότε:

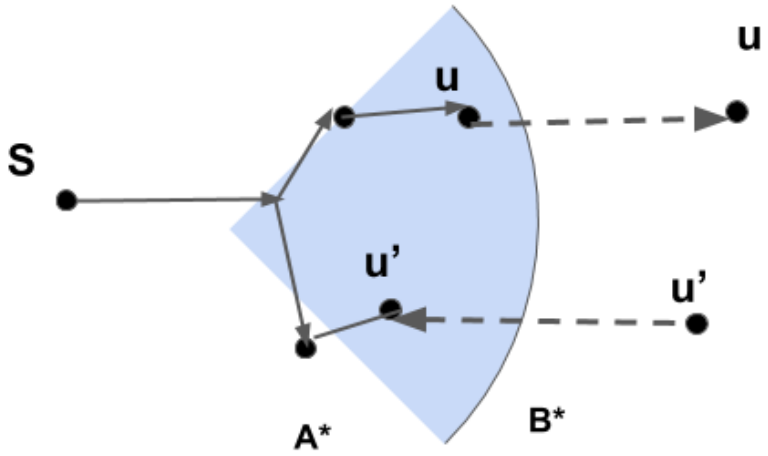
$\nexists$ μονοπάτι στο $s - t$ στο N^{f^*} $=>$
$\exists$ αποκοπή (A^*, B^*) του N τ.ω. $val(f^*) = c(A^*, B^*)$
Άρα η f^* ειναι η μέγιστη ροή και (A^*, B^*) είναι η ελάχιστη τομή

Απόδειξη:

Θα βρούμε την αποκοπή (A^*, B^*) που περιγράφεται στην διατύπωση του θεωρήματος
$A^* := \{u \in (G) \quad | \quad \exists$ μονοπάτι $s - t \quad \sigma\tau o \quad G^*\}$
$B^* := V \setminus A^*$

Απόδειξη Βελτιστοποίσης 2

Θεώρημα



Υποθέτουμε ότι $e = (u, v) \in E(G)$, οπου u $\in A^*$, u $\in B^*$. Ισχυριζομαστε $f(e) = c_e$. Αν δεν συνέβαινε αυτό η e Θα ήταν ευθύδρομη ακμή στο υπολοιπόμενο γρλαφημα G^f, και αφου $u \in A^* => \exists$ διαδρομή s - u στο G^f, μαζι με την e θα είχαμε ένα s - u μονοπάτι στο G^f. **Άτοπο** αφού u^*.

Υποθέτουμε ότι $e^{'} = (u^{'}, v^{'})$ είναι μια ακμή του G, οπου $u^{'} \in B^*$ και $u^{'} \in A^*$. Ισχυριζόμαστε ότι $f(e') = 0$. Αν δεν συνέβαινε αυτό, η e' θα προκαλούσε μια ανάδρομη ακμή e'' $= (u;, u;)$ στο υπολοιπόμενο γραφημα G^f και αφού u'$\in A^*$ θα υπάρχει μια διαδρομή s - u στο G^f, με την ακμή e'' θα παίρναμε μια διαδρομή s - u στο G^f. **Άτοπο** αφού $u' \in B^*$.

Απόδειξη Βελτιστοποήσης (3)

Όλες οι ακμές που εξέρχονται από το A* είναι πλήρως κορεσμένες, ενώ όσες εισέρχονται στο A* είναι τελείως αχρησιμοποίητες. Δηλαδή:

$$val(f) = f^{out}(A*) - f^{in}(A*)$$

$$= \sum_{\substack{e=(a*,b*)\in E \\ a*\in A*}} f(e)$$

$$= c(A^*, B^*)$$

Οπότε τώρα παίρνουμε και το εξής θεώρημα:

Θεώρημα

Η ροή που επιστρέφει ο αλγόριθμος Max-Flow των Ford - Fulkerson είναι μέγιστη.

Επίλογος

- Λύνοντας το Πρόβλημα Μέγιστης Ροής, έχουμε λύσει το Πρόβλημα Ελάχιστης Αποκοπής! Η λύση του ενός προβλήματος είναι τεκμήριο για την ποιότητα της λύσης του άλλου.
- Αυτό δεν είναι κάτι τυχαίο: στον Γραμμικό Προγραμματισμό τα δύο προβλήματα λέγονται Δυϊκά.
- Εδώ μπορούμε να δούμε το Max-Flow σαν κύριο ΓΠ και το Min-Cut σαν το δυϊκό του.

Πηγές, Ευχαριστίες & Εργαλία

Πηγές

1. Σχεδιασμός Αλγορίθμων, Jon Kleinberg, Eva Tardos, εκδόσεις Κλειδάριθμος 2008
2. Αλγόριθμοι, Sanjoy Dasgupta, Christos Papadimitriou, Umesh Vaziriani, εκδόσεις Κλειδάριθμος 2009

Ευχαριστώ,

την Κατερίνα Ρουσσάκη, φοιτήτρια του Τμήματος Πληροφορικής και Τηλεπικοινωνιών ΕΚΠΑ, που με βοήθησε στη δακτυλογράφηση των διαφανειών σε LaTeX, καθώς και σε πολλά από τα σχήματα

Εργαλεία

Οι διαφάνειες γράφτηκαν σε LaTeX, στο Overleaf, ενώ για τα γραφήματα αυτή εδώ η εφαρμογή.