ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
UNIVERSITY OF PATRAS

ΑΝΟΙΚΤΑ ακαδημαϊκά μαθήματα ΠΠ

Τεχνολογίες Υλοποίησης Αλγορίθμων

Χρήστος Ζαρολιάγκης

Καθηγητής
Τμήμα Μηχ/κων Η/Υ & Πληροφορικής
Πανεπιστήμιο Πατρών
email: zaro@ceid.upatras.gr

Ενότητα 9

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο

ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ & ΘΡΗΣΚΕΥΜΑΤΩΝ, ΠΟΛΙΤΙΣΜΟΥ & ΑΘΛΗΤΙΣΜΟΥ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ

ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο **«Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Πατρών»** έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Περιεχόμενα

- Ευρετικές Βελτιώσεις του Αλγορίθμου Προροής-Προώθησης
- Πειραματική Αξιολόγηση
- Ροή Δικτύου & Αριθμητική Κινητής Υποδιαστολής

Πολυπλοκότητα Χειρότερης Περίπτωσης Αλγορίθμου Προροής-Προώθησης

Επιλογή ενεργών κορυφών

- Αιθαίρετη: $O(n^2 m)$
- FIFO: $O(n^3)$
- Υψηλότερου Επιπέδου: $O(n^2 \sqrt{m})$

Πολυπλοκότητα Καλύτερης Περίπτωσης Αλγορίθμου Προροής-Προώθησης;

- **Κόστος Ενημέρωσης Ετικετών:** $\Omega(n^2)$, αν $\Omega(n)$ κορυφές πρέπει να ανυψωθούν πάνω από το επίπεδο n

- Πολυπλοκότητα Καλύτερης Περίπτωσης των άλλων τμημάτων: $O(m)$

- *Μπορεί να βελτιωθεί ο χρόνος ενημέρωσης/ανανέωσης ετικετών ;*

Ευρετική Μέθοδος 1 - Ετικέτες Μεγάλων Αποστάσεων

- κορυφή v : $\begin{cases} \text{υψηλή} & \text{αν } d(v) \geq n \\ \text{χαμηλή} & \text{αλλιώς} \end{cases}$

- *Τι διαφοροποιεί τις υψηλές από τις χαμηλές κορυφές ;*
  - Ενεργή **υψηλή** κορυφή v: $\nexists$ v-t διαδρομή κατάλληλων ακμών στο $G(f)$
    $\Downarrow$
    πλεόνασμα των ενεργών υψηλών κορυφών πρέπει να σταλει πίσω στην s
  - Ενεργή **χαμηλή** κορυφή u: κάποιο πλεόνασμα της u μπορεί να προωθηθεί στην t και κάποιο πρέπει να σταλεί πίσω στην s

Ευρετική Μέθοδος 1 - Ετικέτες Μεγάλων Αποστάσεων

- **Πλεόνασμα:** φτάνει σε ενεργές υψηλές κορυφές μέσω ακμών $e \in E$ με $f(e) > 0 \Longrightarrow$ Ροή μπορεί να σταλεί πίσω μέσω αυτών των ακμών
- $E'(f) = \{e' = (y, x) : e = (x, y) \in E \text{ και } f(e) > 0\}$
- Χρήση μόνο ακμών του $E'(f)$ όταν προωθούμε ροή από ενεργές υψηλές κορυφές

Τροποποιημένος Αλγόριθμος Προρροής-Προώθησης

/* αρχικοποίηση */
Θέσε $f(e) = cap(e)$ για όλες τις ακμές με source$(e) = s$;
Θέσε $f(e) = 0$ για όλες τις υπόλοιπες ακμές ;
Θέσε $d(s) = n$ και $d(v) = 0$ για όλες τις κορυφές ;

/* κύριος βρόχος */
while $\exists$ ενεργή κορυφή
{ έστω v μια ενεργή κορυφή;
if $d(v) < n$ και $\exists$ κατάλληλη ακμή $e = (v, w) \in E(f)$ ή
$d(v) \geq n$ και $\exists$ κατάλληλη ακμη $e = (v, w) \in E'(f)$ **then**
{ προώθησε $\delta = \min\{excess(v), r(e)\}$ μονάδες ροής στην e; }
else
{ αύξησε το $d(v)$; }
}

Ευρετική Μέθοδος 1 – Ετικέτες Μεγάλων Αποστάσεων – Τροποποιημένος Κύριος Βρόχος

```
// MF_LH: main loop
for(;;)
{
  node v = U.del();
  if (v == nil) break;
  if (v == t) continue;

  NT  ev = excess[v]; // excess of v
  int dv = dist[v];   // level of v
  edge e;

  if ( dist[v] < n )
  { // MF_BASIC: push across edges out of v }

  if ( ev > 0 )
  { // MF_BASIC: push across edges into v }
  excess[v] = ev;

  if (ev > 0)
  { dist[v]++;
    num_relabels++;
    U.insert(v,dist[v]);
  }
}
```

Ευρετική Μέθοδος 2 – Τοπική Ανανέωση Ετικετών

- Ανανέωση ετικέτας της v (οποτεδήποτε χρειαστεί):
  $d(v) = 1 + \min\{d(w) : (v, w) \in G(f)\}$

Τοπική Ανανέωση Ετικετών

Όταν η ετικέτα της v ανανεώνεται, συνέχισε την ανανέωση έως ότου $\exists$ κατάλληλη εξερχόμενη ακμή από την v

- Διατήρηση μεταβλητής *dmin*, αρχικοποιημένη σε MAXINT
- Αν $e = (v, w) \in G(f)$ είναι κατάλληλη ($d(w) < d(v)$), τότε προώθησε ροή από την v
- Αν $e = (v, w) \in G(f)$ δεν ειναι κατάλληλη ($d(w) \geq d(v)$), τότε $dmin = \min\{dmin, d(w)\}$
- Αν η v είναι ακόμη ενεργή μετά την εξέταση όλων των γειτονικών υπολειπόμενων ακμών της, τότε $d(v) = 1 + dmin$

Ευρετική Μέθοδος 2 – Τοπική Ανανέωση Ετικετών – Τροποποιημένος Κύριος Βρόχος

```
// MF_LRH: main loop
for(;;)
{
  node v = U.del();
  if (v == nil) break;
  if (v == t) continue;

  NT  ev = excess[v]; // excess of v
  int dv = dist[v];   // level of v
  int dmin = MAXINT;  // for local relabeling heuristic
  edge e;

  if ( dist[v] < n )
  { // MF_LRH: push across edges out of v }
  if ( ev > 0 )
  { // MF_LRH: push across edges into v }

  excess[v] = ev;
  if (ev > 0)
  { dist[v]++;
    num_relabels++;
    U.insert(v,dist[v]);
  }
}
```

Ευρετική Μέθοδος 2 – Τοπική Ανανέωση Ετικετών – Τροποποιημένος Κώδικας

```
// MF_LRH: push across edges out of v
for (e = G.first_adj_edge(v); e; e = G.adj_succ(e))
{ num_edge_inspections++;
  NT& fe = flow[e];
  NT  rc = cap[e] - fe;
  if (rc == 0) continue;
  node w = target(e);
  int dw = dist[w];
  if ( dw < dv ) // equivalent to ( dw == dv - 1 )
  { num_pushes++;
    NT& ew = excess[w];
    if (ew == 0) U.insert0(w,dw);
    if (ev <= rc)
    { ew += ev; fe += ev;
      ev = 0;   // stop: excess[v] exhausted
      break;
    }
    else
    { ew += rc; fe += rc;
      ev -= rc;
    }
  }
  else { if ( dw < dmin ) dmin = dw; }
}
```

Ευρετική Μέθοδος 2 – Τοπική Ανανέωση Ετικετών – Τροποποιημένος Κώδικας

```
// MF_LRH: push across edges into v

for (e = G.first_in_edge(v); e; e = G.in_succ(e))
{ num_edge_inspections++;
  NT& fe = flow[e];
  if (fe == 0) continue;
  node w = source(e);
  int dw = dist[w];
  if ( dw < dv ) // equivalent to ( dw == dv - 1 )
  { num_pushes++;
    NT& ew = excess[w];
    if (ew == 0) U.insert0(w,dw);
    if (ev <= fe)
    { fe -= ev; ew += ev;
      ev = 0;    // stop: excess[v] exhausted
      break;
    }
    else
    { ew += fe; ev -= fe;
      fe = 0;
    }
  }
  else { if ( dw < dmin ) dmin = dw; }
}
```

Ευρετική Μέθοδος 3 – Καθολική Ανανέωση Ετικετών

- Ενημερώνει τις τιμές $d(\cdot)$ **όλων** των κορυφών

$$d(v) = \begin{cases} \delta(v,t) & \text{αν } \exists\ v\text{-}t \text{ διαδρομή στο } G(f) \\ n + \delta'(v,s) & \text{αν } \exists\ v\text{-}s \text{ διαδρομή στο } G'(f) \\ & \text{και } \nexists\ v\text{-}t \text{ διαδρομή στο } G(f) \\ 2n-1 & \text{αλλιώς} \end{cases}$$

- $\delta(v,t) :=$ μήκος (αριθμός ακμών) συντομότερης v-t διαδρομής στο $G(f)$

  $\delta'(v,s) :=$ μήκος (αριθμός ακμών) συντομότερης v-s διαδρομής στο $G'(f)$

Ευρετική Μέθοδος 3 – Καθολική Ανανέωση Ετικετών

- Καθολική Ανανέωση Ετικετών: υλοποίηση με ΑΠΠ $\Longrightarrow O(m)$ χρόνο

  $\Downarrow$

  δεν θα πρέπει να εφαρμόζεται πολύ συχνά

- ΚΑΕ εφαρμόζεται (μετά από) κάθε $h \cdot m$ επιθεωρήσεις ακμών (h κατάλληλη σταθερά)

- Πολυπλοκότητα χειρότερης περίπτωσης αυξάνεται μόνο κατά ένα σταθερό παράγοντα

- Πολυπλοκότητα καλύτερης περίπτωσης μπορει να βελτιωθεί σημαντικά

  - ΚΑΕ εφαρμόζεται μετά από τον κορεσμό της (v_{n-2}, v_{n-1})
  - Τοποθετεί την κορυφή v_i στο επίπεδο $n + i, \forall i, 1 \leq i \leq n-2$
  - Πλεόνασμα κορυφής v_{n-2} θα σταλεί πίσω στην s μετά από μια σειρά n προωθήσεων
  - Δεν χρειάζεται ανανέωση ετικετών $\Longrightarrow$ Μείωση χρόνου από $\Omega(n^2)$ σε $O(n)$

Ευρετική Μέθοδος 3 – Καθολική Ανανέωση Ετικετών

Συναρτήσεις `compute_dist_t` **και** `compute_dist_s` (αποστάσεις προς t & s)

- `compute_dist_t`:
  - $dist[v] \geq n$, $\forall$ κορυφή v πριν από μία κλήση της συνάρτησης
  - «ανάδρομη» ΑΠΠ στο $G(f)$
- `compute_dist_s`:
  - $dist[v] = 2n - 1$, $\forall$ κορυφή v που δεν μπορεί να προσπελάσει την t στο $G(f)$ πρίν από μία κλήση της συνάρτησης
  - $dist[v] < n$, $\forall$ κορυφή v που μπορεί να προσπελάσει την t στο $G(f)$
  - «ανάδρομη» ΑΠΠ στο $G'(f)$
- Όλες οι ενεργές κορυφές u τοποθετούνται στην U με τις καινούριες $dist[u]$
- $U = \emptyset$ πριν την κλήση της `compute_dist_t`, και η U περιέχει όλες τις ενεργές κορυφές που προσπελάζουν την t στο $G(f)$ πριν την κλήση της `compute_dist_s`
- «ανάδρομη» ΑΠΠ $\Longrightarrow$ η (αναγκαία) ουρά Q περνά σαν παράμετρος (η Q είναι άδεια πριν από κάθε κλήση, και μένει άδεια μετά από κάθε κλήση)
- `compute_dist_t`: υπολογισμός $count[d] = |\{v | dist[v] = d, 0 \leq d < n\}|$

Ευρετική Μέθοδος 3 – Καθολική Ανανέωση Ετικετών

```
template<class NT, class SET>
void compute_dist_t(const graph& G, node t, const edge_array<NT>& flow,
                    const edge_array<NT>& cap,
                    const node_array<NT>& excess, node_array<int>& dist,
                    SET& U, b_queue<node>& Q, array<int>& count)
{ int n = G.number_of_nodes();
  Q.append(t); dist[t] = 0; count.init(0); count[0] = 1;

  while ( !Q.empty() )
  { node v = Q.pop();
    int  d = dist[v] + 1;
    edge e;

    for(e = G.first_adj_edge(v); e; e = G.adj_succ(e))
    { if ( flow[e] == 0 ) continue;
      node u = target(e);
      int& du = dist[u];
      if ( du >= n )
      { du = d;
        Q.append(u); count[d]++;
        if ( excess[u] > 0 ) U.insert(u,d);
      }
    }
    for(e = G.first_in_edge(v); e; e = G.in_succ(e))
    { if ( cap[e] == flow[e] ) continue;
      node u = source(e);
      int& du = dist[u];
      if ( du >= n )
      { du = d;
        Q.append(u); count[d]++;
        if (excess[u] > 0) U.insert(u,d);
      }
    }
  }
}
```

Ευρετική Μέθοδος 3 – Καθολική Ανανέωση Ετικετών

```
template<class NT, class SET>
void compute_dist_s(const graph& G, node s, const edge_array<NT>& flow,
                    const node_array<NT>& excess, node_array<int>& dist,
                    SET& U, b_queue<node>& Q)
{
  int n = G.number_of_nodes();
  int max_level = 2*n - 1;

  Q.append(s);
  dist[s] = n;

  while ( !Q.empty() )
  { node v = Q.pop();
    int  d = dist[v] + 1;
    edge e;
    for(e = G.first_adj_edge(v); e; e = G.adj_succ(e))
    { if ( flow[e] == 0 ) continue;
      node u = target(e);
      int& du = dist[u];
      if ( du == max_level )
      { du  = d;
        if (excess[u] > 0) U.insert(u,d);
        Q.append(u);
       }
     }
   }
}
```

Ευρετική Μέθοδος 4 – Καθολική Ανανέωση Ετικετών: μέθοδος δύο φάσεων

- **Φάση 1**
  - προώθησε ροή μόνο από τις κορυφές v με $d(v) < n$
  - τερμάτισε όταν $\nexists$ ενεργή κορυφή v με $d(v) < n$
    $\Longrightarrow$ μέγιστη προροή, αφού $\nexists$ v-t διαδρομή στο $G(f)$
- **Φάση 2**
  - προώθησε ροή μόνο από τις κορυφές v με $d(v) \geq n$
  - τερμάτισε όταν $\nexists$ καμία ενεργή κορυφή

Ευρετική Μέθοδος 4 – Καθολική Ανανέωση Ετικετών: μέθοδος δύο φάσεων

```
// MF_GRH: initialization

// initialize flow and excess and saturate edges out of s, as in MF_BASIC

// MF_GRH: additional data structures
b_queue<node> Q(n);
int phase_number = 1;
array<int> count(n);
list<node> S;
int heuristic  = (int) (h*m);
int limit_heur = heuristic;

// MF_GRH: initialize dist and U for first phase
node_array<int> dist(G);
dist.init(G,n);
compute_dist_t(G,t,flow,cap,excess,dist,U,Q,count);

// MF_GRH: initialize counters
num_relabels = num_pushes = num_edge_inspections = 0;
num_global_relabels  = 0;

// MF_GRH: main loop
```

Ευρετική Μέθοδος 4 – Καθολική Ανανέωση Ετικετών: μέθοδος δύο φάσεων

```
// MF_GRH: main loop

for(;;)
{
  // MF_GRH: extract v from queue

  NT  ev   = excess[v]; // excess of v
  int dv   = dist[v];   // level of v
  int dmin = MAXINT;
  edge e;

  if ( dist[v] < n )
  { // push across edges out of v }
  if ( ev > 0 )
  { // push across edges into v }

  excess[v] = ev;

  if (ev > 0)
  { // MF_GRH: update distance label(s) }
}
```

Ευρετική Μέθοδος 4 – Καθολική Ανανέωση Ετικετών: μέθοδος δύο φάσεων

Πώς επιλέγουμε κορυφές από την U;

```
// MF_GRH: extract v from queue

node v = U.del();
if (v == nil)
{
  if ( phase_number == 2 ) break; // done

  dist.init(G,n);
  compute_dist_t(G,t,flow,cap,excess,dist,U,Q,count);

  node u;
  forall_nodes(u,G)
  { if (dist[u] == n)
    { S.append(u);          // S collects all nodes that cannot reach t
      dist[u] = max_level;
    }
  }
  phase_number = 2;
  compute_dist_s(G,s,flow,excess,dist,U,Q);
  continue;
}
if (v == t) continue;
```

Ευρετική Μέθοδος 4 – Καθολική Ανανέωση Ετικετών: μέθοδος δύο φάσεων

Πώς ενημερώνουμε τις ετικέτες αποστάσεων ;

- Ακέραιες μεταβλητές `limit_heur` και `heuristic`
- Αρχικοποίησε την `heuristic` σε $h \cdot m$, αύξησε την `limit_heur` κατά `heuristic` οποτεδήποτε εφαρμόζεται η ΚΑΕ, και εφάρμοσε την ΚΑΕ όποτε ο αριθμών επιθεωρήσεων των ακμών υπερβαίνει το `limit_heur`

```
// MF_GRH: update distance label(s)

if (num_edge_inspections <= limit_heur)
  { // MF_GRH: update the distance label of v }
else
  { limit_heur += heuristic;
    num_global_relabels++;
    // MF_GRH: global relabel
  }
```

Ευρετική Μέθοδος 4 – Καθολική Ανανέωση Ετικετών: μέθοδος δύο φάσεων

- Αν $phase = 1$ και $dmin \geq n$, τότε $dist[v] = n$ και δεν εισάγεται η v στην U, αφού η v δεν μπορεί να προσπελάσει πλέον την t στο $G(f)$

```
// MF_GRH: update the distance label of v

dmin++; num_relabels++;
if ( phase_number == 1 && dmin >= n) dist[v] = n;
else { dist[v] = dmin;
       U.insert(v,dmin);
     }
```

Ευρετική Μέθοδος 4 – Καθολική Ανανέωση Ετικετών: μέθοδος δύο φάσεων

Καθολική Ανανέωση Ετικετών

- Φάση 1: $\forall v$, υπολογισμός της v-t απόστασης στο $G(f)$·
  κορυφές που δεν μπορούν να προσπελάσουν την t λαμβάνουν απόσταση n

- Αν $\nexists$ ενεργή κορυφή που να προσπελάζει την t, η Φάση 1 τελειώνει·
  το S περιέχει όλες τις κορυφές που δεν μπορούν να προσπελάσουν την t

- Φάση 2: (επανα)υπολόγισε την απόσταση v-s, $\forall v \in S$·
  οι κορυφές στο $V - S$ προσπελαύνουν την t στο $G(f)$ και άρα είναι άνευ σημασίας για τη Φάση 2

Ευρετική Μέθοδος 4 – Καθολική Ανανέωση Ετικετών: μέθοδος δύο φάσεων

```
// MF_GRH: global relabel
U.clear();

if (phase_number == 1)
{ dist.init(G,n);
  compute_dist_t(G,t,flow,cap,excess,dist,U,Q,count);
  if ( U.empty() )
  { node u;
    forall_nodes(u,G)
    { if (dist[u] == n)
      { S.append(u);
        dist[u] = max_level;
      }
    }
    phase_number = 2;
    compute_dist_s(G,s,flow,excess,dist,U,Q);
  }
}
else
{ node u;
  forall(u,S) dist[u] = max_level;
  compute_dist_s(G,s,flow,excess,dist,U,Q);
}
```

Ευρετική Μέθοδος 4 – Καθολική Ανανέωση Ετικετών: μέθοδος δύο φάσεων

```
template<class NT, class SET>
NT MAX_FLOW_GRH_T(const graph& G, node s, node t,
                  const edge_array<NT>& cap, edge_array<NT>& flow,
                  SET& U, int& num_pushes, int& num_edge_inspections,
                  int& num_relabels, int& num_global_relabels, float h)
{ if (s == t) error_handler(1,"MAXFLOW: source == sink");

  // MF_GRH: initialization

  // MF_GRH: main loop

#ifdef TEST
  assert(CHECK_MAX_FLOW_T(G,s,t,cap,flow));
#endif

  return excess[t];
}
```

Ευρετική Μέθοδος 5 - GAP (Χάσμα)

- Αν η ανανέωση ετικετών της v (φάση 1) αφήνει το επίπεδο κορυφών $d(v)$ κενό, τότε η v (και όλες οι κορυφές που μπορεί να προσπελάσει η v) δεν μπορούν να προσπελάσουν την t στο $G(f)$

- **Ευρετική Μέθοδος Gap:** μετακίνησε την v και όλες τις προσπελάσιμες από την v κορυφές στο επίπεδο n, οποτεδήποτε το επίπεδο της v μείνει κενό κορυφών μετά από μια ανανέωση ετικέτας της v

Ευρετική Μέθοδος 5 - GAP (Χάσμα)

- Διατήρηση πίνακα $count[d], 0 \leq d < n$
- Επαναυπολογισμός *count* στην `compute_dist_t` και ενημέρωσή του όταν ανανεώνεται η ετικέτα μιας κορυφής
- Όταν μία κορυφή v μετακινείται από ένα επίπεδο $d(v)$ στο επίπεδο *dmin*, μειώνεται το $count[d(v)]$ και αυξάνεται το $count[dmin]$ (αν $d(v)$ ή *dmin* είναι μικρότερες του n)
- Όταν $count[d(v)] = 0$, μετακίνηση της v και όλων των προσπελάσιμων κορυφών της στο $G(f)$ στο επίπεδο n
- Οι προσπελάσιμες από την v κορυφές στο $G(f)$ υπολογίζονται με χρήση ΑΠΠ και ανακύκλωση της ουράς Q

Ευρετική Μέθοδος 5 - GAP (Χάσμα)

```
// MF_GAP: update the distance label of v

num_relabels++;
if (phase_number == 1)
{ if ( --count[dv] == 0 || dmin >= n - 1)
  { // v cannot reach t anymore
    // move all vertices reachable from v to level n
  }
  else
  { dist[v] = ++dmin; count[dmin]++;
    U.insert(v,dmin);
  }
}
else // phase_number == 2
{ dist[v] = ++dmin;
  U.insert(v,dmin);
}
```

Ευρετική Μέθοδος 5 - GAP (Χάσμα)

```
// move all vertices reachable from v to level n

dist[v] = n;
if ( dmin < n )
{ Q.append(v);
  node w,z;
  while ( !Q.empty() )
  { edge e;
    w = Q.pop(); num_gaps++;
    forall_out_edges(e,w)
    { if ( flow[e] < cap[e] && dist[z = G.target(e)] < n)
      { Q.append(z);
        count[dist[z]]--; dist[z] = n;
      }
    }
    forall_in_edges(e,w)
    { if ( flow[e] > 0 && dist[z = G.source(e)] < n)
      { Q.append(z);
        count[dist[z]]--; dist[z] = n;
      }
    }
  }
}
```

Ευρετική Μέθοδος 5 - GAP (Χάσμα)

- Κύριος βρόχος: απαιτείται μόνο μία αλλαγή
- Όταν η ευρετική μέθοδος GAP μετακινεί μία κορυφή στο επίπεδο n, δεν την αφαιρεί από το σύνολο των ενεργών κορυφών (παρόλο που θα έπρεπε για την Φάση 1)
- Λύση: όταν μια κορυφή στο επίπεδο n αφαιρείται από το σύνολο των ενεργών κορυφών στη Φάση 1, αγνοούμε την κορυφή και συνεχίζουμε στην επόμενη επανάληψη

Ευρετική Μέθοδος 5 - GAP (Χάσμα)

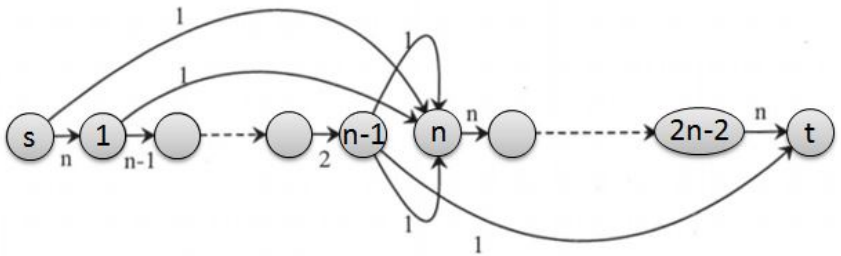
```
// MF_GAP: main loop
for(;;)
{
  // MF_GRH: extract v from queue
  if (dist[v] == n && phase_number == 1) continue;
  NT  ev   = excess[v]; // excess of v
  int dv   = dist[v];   // level of v
  int dmin = MAXINT;
  edge e;

  if ( dist[v] < n ) { // push across edges out of v }
  if ( ev > 0 ) { // push across edges into v }
  excess[v] = ev;

  if (ev > 0) // MF_GAP: update distance label(s)
  {
    if (num_edge_inspections <= limit_heur)
    { // MF_GAP: update the distance label of v }
    else
    { limit_heur += heuristic;
      num_global_relabels++;
      // MF_GRH: global relabel
    }
  }
}
```

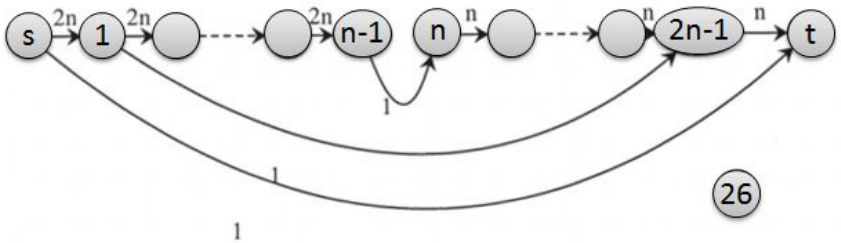
Ευρετική Μέθοδος 5 - GAP (Χάσμα)

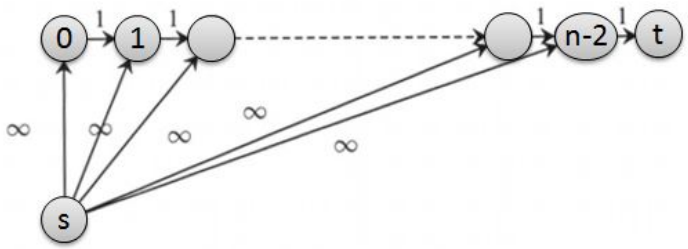
```
template<class NT, class SET>
NT MAX_FLOW_GAP_T(const graph& G, node s, node t,
                  const edge_array<NT>& cap, edge_array<NT>& flow,
                  SET& U, int& num_pushes, int& num_edge_inspections,
                  int& num_relabels, int& num_global_relabels,
  int& num_gaps, float h)
{ if (s == t) error_handler(1,"MAXFLOW: source == sink");

  // MF_GRH: initialization
  num_gaps = 0; // number of nodes moved by the GAP heuristic

  // MF_GAP: main loop

#ifdef TEST
  assert(CHECK_MAX_FLOW_T(G,s,t,cap,flow));
#endif

  return excess[t];
}
```

Ευρετικές Μέθοδοι – Πειραματική Μελέτη

- Σύνολα δεδομένων: 1 τυχαίο (rand) και 3 συνθετικά (CG1, CG2, AMO) δίκτυα







Ευρετικές Μέθοδοι – Πειραματική Μελέτη

- Διαφορετικές ευρετικές μέθοδοι vs 3 κανόνες επιλογής: $n \in \{1000, 2000\}$

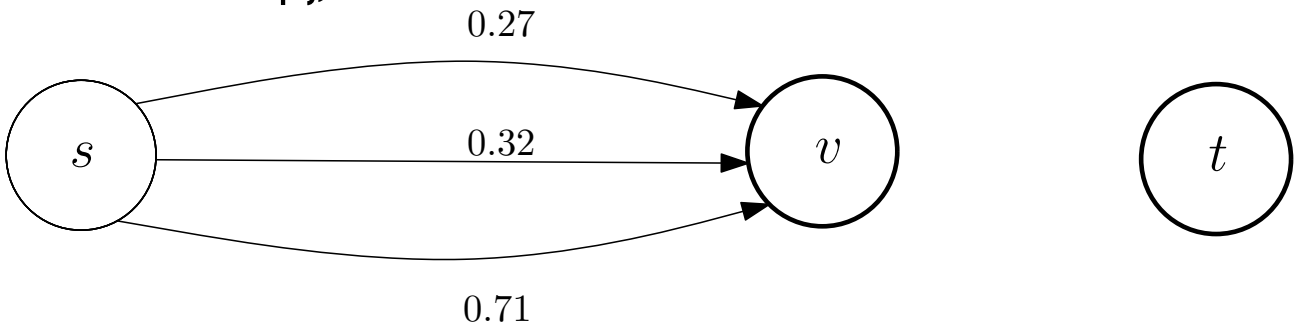
Gen	Rule	BASIC	HL	LRH	GRH	GAP	LEDA
rand	FF	5.84	6.02	4.75	0.07	0.07	—
		33.32	33.88	26.63	0.16	0.17	—
	HL	6.12	6.3	4.97	0.41	0.11	0.07
		27.03	27.61	22.22	1.14	0.22	0.16
	MF	5.36	5.51	4.57	0.06	0.07	—
		26.35	27.16	23.65	0.19	0.16	—
CG1	FF	3.46	3.62	2.87	0.9	1.01	—
		15.44	16.08	12.63	3.64	4.07	—
	HL	20.43	20.61	20.51	1.19	1.33	0.8
		192.8	191.5	193.7	4.87	5.34	3.28
	MF	3.01	3.16	2.3	0.89	1.01	—
		12.22	12.91	9.52	3.65	4.12	—
CG2	FF	50.06	47.12	37.58	1.76	1.96	—
		239	222.4	177.1	7.18	8	—
	HL	42.95	41.5	30.1	0.17	0.14	0.08002
		173.9	167.9	120.5	0.3599	0.28	0.1802
	MF	45.34	42.73	37.6	0.94	1.07	—
		198.2	186.8	165.7	4.11	4.55	—
AMO	FF	12.61	13.25	1.17	0.06	0.06	—
		55.74	58.31	5.01	0.1399	0.1301	—
	HL	15.14	15.8	1.49	0.13	0.13	0.07001
		62.15	65.3	6.99	0.26	0.26	0.1399
	MF	10.97	11.65	0.04999	0.06	0.06	—
		46.74	49.48	0.1099	0.1301	0.1399	—

- KAE (GRH) και GAP vs 3 κανόνες επιλογής: $n = 5000 \cdot 2^i$, $i = 0, 1, 2$

Gen	Rule	GRH			GAP			LEDA		
rand	FF	0.16	0.41	1.16	0.15	0.42	1.05	—	—	—
	HL	1.47	4.67	18.81	0.23	0.57	1.38	0.16	0.45	1.09
	MF	0.17	0.36	1.06	0.14	0.37	0.92	—	—	—
CG1	FF	3.6	16.06	69.3	3.62	16.97	71.29	—	—	—
	HL	4.27	20.4	77.5	4.6	20.54	80.99	2.64	12.13	48.52
	MF	3.55	15.97	68.45	3.66	16.5	70.23	—	—	—
CG2	FF	6.8	29.12	125.3	7.04	29.5	127.6	—	—	—
	HL	0.33	0.65	1.36	0.26	0.52	1.05	0.15	0.3	0.63
	MF	3.86	15.96	68.42	3.9	16.14	70.07	—	—	—
AMO	FF	0.12	0.22	0.48	0.11	0.24	0.49	—	—	—
	HL	0.25	0.48	0.99	0.24	0.48	0.99	0.12	0.24	0.52
	MF	0.11	0.24	0.5	0.11	0.24	0.48	—	—	—

Ροή Δικτύου & Αριθμητική Κινητής Υποδιαστολής

- *Αριθμητικά σφάλματα στρογγυλοποίησης* (π.χ., αριθμητική κινητής υποδιαστολής) ;



- *Υπόθεση:* αριθμητικό σύστημα κινητής υποδιαστολής με 2-ψήφιο δεκαδικό μέρος και στρογγυλοποίηση με αποκοπή
- Μετά την **αρχικοποίηση:** $excess(v) = 0.27 \oplus 0.32 \oplus 0.71 = 1.3$ **σωστό**· $\nexists$ απαλοιφή ψηφίου στην πράξη $\oplus$
- Αποστολή **ροής πίσω στην** s:

  άνω (v, s): $1.3 \ominus 0.27 = 1.1 \neq 1.03 = 1.3 - 0.27$
  μεσαία (v, s): $1.1 \ominus 0.32 = 0.8$
  κάτω (v, s): $0.8 \ominus 0.71 = 0.1$

  το τελευταιο ψηφίο απαλοίφεται όταν τα αθροίσματα ευθυγραμμίζονται για την πράξη $\ominus$
- **Πρόβλημα:** $excess(v) = 0.1$, $\nexists$ ακμή εξερχόμενη της $v \Longrightarrow$ μπορεί να θέσει το πρόγραμμα σε ατέρμονα βρόχο

Ροή Δικτύου & Αριθμητική Κινητής Υποδιαστολής

- $D = \sum_{(s,v)\in E} cap(s,v)$ · χρησιμοποιούμενος αριθμητικός τύπος: *double*
- Αν όλες οι χωρητικότητες των ακμών είναι ακέραιες $\Longrightarrow \nexists$ υπερχείληση όσο $D < 2^{53}$
- Αν οι χωρητικότητες των ακμών δεν είναι ακέραιες, τότε

$$cap1[e] = sign(cap[e])\lfloor |cap[e]| \cdot S \rfloor / S,$$

  S: μεγαλύτερη δύναμη του δύο έτσι ώστε $S < 2^{53}/D$

- $\nexists$ σφάλμα στρογγυλοποίησης σε σχέση με τις $cap1$
  απόλυτο σφάλμα στην τιμή της μέγιστης ροής $\leq m \cdot D \cdot 2^{-52}$
- Τι συμβαίνει με το **σχετικό σφάλμα ;**

Ροή Δικτύου & Αριθμητική Κινητής Υποδιαστολής

Ένας πιο εκλεπτισμένος τρόπος για να φράξουμε το σχετικό σφάλμα

- Άρχισε με τις $cap1$ και υπολόγισε μια μέγιστη ροή f έστω f_{opt} η μέγιστη ροή σε σχέση με τις cap
- $\left||f_{opt}| - |f|\right| \leq m \cdot D \cdot 2^{-52}$
- Αν $m \cdot D \cdot 2^{-52} \ll |f|$, τότε το σχετικό σφάλμα $\frac{\left||f_{opt}|-|f|\right|}{|f|}$ είναι μικρό
- Αλλιώς, έστω $B = |f| + m \cdot D \cdot 2^{-52} \Longrightarrow |f_{opt}| \leq B$
- Οποιοδήποτε $cap[e] > B$ μπορεί να μειωθεί σε B χωρίς να αλλάξει η μέγιστη ροή
- Επαναυπολογισμός των D και S, και επανάληψη της διαδικασίας μέχρι το σχετικό σφάλμα να γίνει μικρό

Τέλος Ενότητας

Σημείωμα Ιστορικού Εκδόσεων Έργου

Το παρόν έργο αποτελεί την έκδοση **1.0**.

Σημείωμα Ιστορικού Εκδόσεων Έργου

Σημείωμα Ιστορικού Εκδόσεων Έργου

Διατήρηση Σημειωμάτων

Οποιαδήποτε αναπαραγωγή ή διασκευή του υλικού θα πρέπει να συμπεριλαμβάνει :

- το Σημείωμα Αναφοράς
- το Σημείωμα Αδειοδότησης
- τη δήλωση Διατήρησης Σημειωμάτων
- το Σημείωμα Χρήσης Έργων Τρίτων (εφόσον υπάρχει) μαζί με τους συνοδευόμενους υπερσυνδέσμους