Διαμέριση - Partitioning

Διαμέριση

Διαμέριση *είναι η διαμοίραση αντικειμένων σε ομάδες με στόχο την βελτιστοποίηση κάποιας συνάρτησης.*

Στην σύνθεση η **διαμέριση** χρησιμοποιείται ως εξής:

- ✓ Ομαδοποίηση μεταβλητών με κάθε ομάδα να αντιστοιχίζεται σε ένα αποθηκευτικό στοιχείο.
- ✓ Ομαδοποίηση λειτουργιών με κάθε ομάδα να αντιστοιχίζεται σε μία λειτουργική μονάδα.
- ✓ Ομαδοποίηση λειτουργιών σε ομάδες, ώστε κάθε ομάδα να εκτελείται στο ίδιο βήμα ελέγχου.
- ✓ Διαίρεση μίας μεγάλης περιγραφής συμπεριφοράς σε μικρότερες.

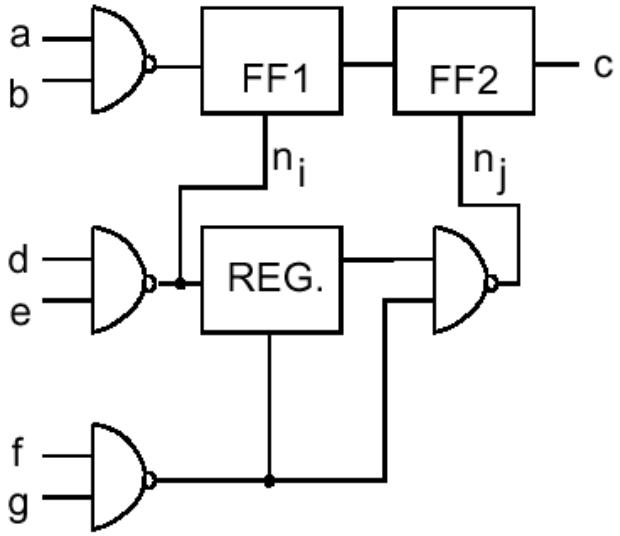
Μέθοδοι Διαμέρισης

Μοντελοποίηση προβλήματος:

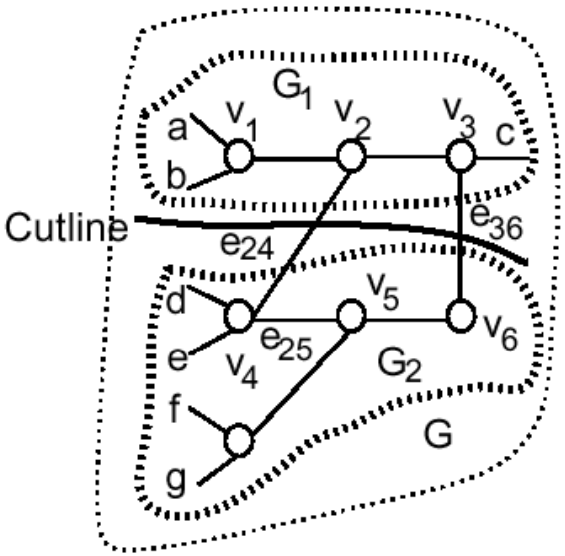
- ✓ Χρησιμοποιείται θεωρία γράφων.
- ✓ Κάθε κόμβος αναπαριστά ένα component (πύλη, flip-flop, καταχωρητή…)
- ✓ Κάθε ακμή αναπαριστά φυσική διασύνδεση δύο components.
- ✓ Γραμμές με πολλές άκρες στο αρχικό κύκλωμα αντικαθίστανται από πολλαπλές γραμμές με δύο άκρες.
- ✓ Για την διαμέριση χρησιμοποιείται γραμμή διαχωρισμού (cutline).

Παράδειγμα: διαμέριση γράφου σε υπογράφους με καθορισμένο μέγιστο μέγεθος (περιορισμός), και στόχο την ελαχιστοποίηση των εξωτερικών διασυνδέσεων μεταξύ τους.

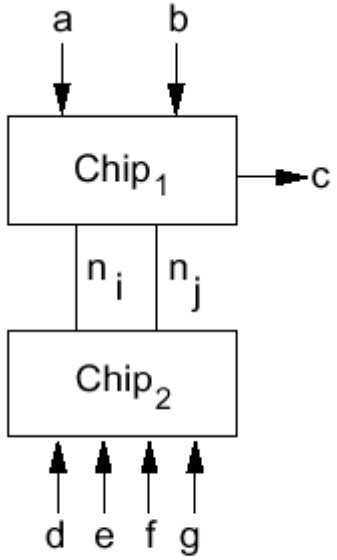
Συνδυαστική Βελτιστοποίηση



Design structure



Graph model



Partitioned Design

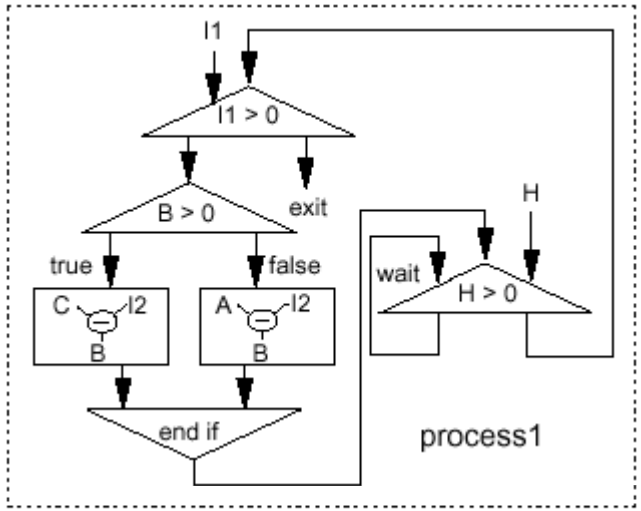
e24, e36 : external connections

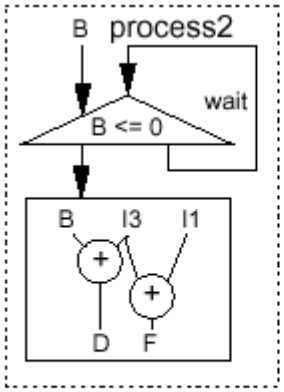
Διαμέριση Συμπεριφοράς

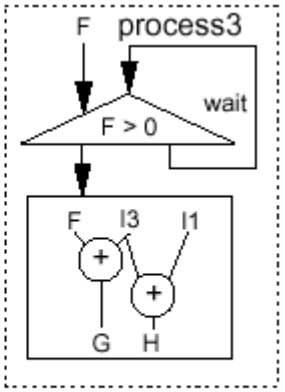
```
entity VHDL EXAMPLE is
      port (I1, I2, I3 : in integer;
            O1 : out integer;)
   signal B, F, H : integer;
end entity;
architecture BEHAVIOR of EXAMPLE is
begin
   process1
      var : A, C, E : integer;
      while (I1 > 0)  loop
            if (B > 0)
               B <= C – I2;
            else
               B <= A – I2;
            end if;
            wait until (H > 0);
      end loop;
   end process1;

   process2
      var : D : integer;
      wait until (B <= 0);
      D := I3 + B;
      F <= I3 + I1;
   end process2;
   process3
      var : G : integer;
      wait until (F > 0);
      O1 <= I3 + G;
      H <= I3 + I1;
   end process3;
end
```

Επικοινωνία Processes

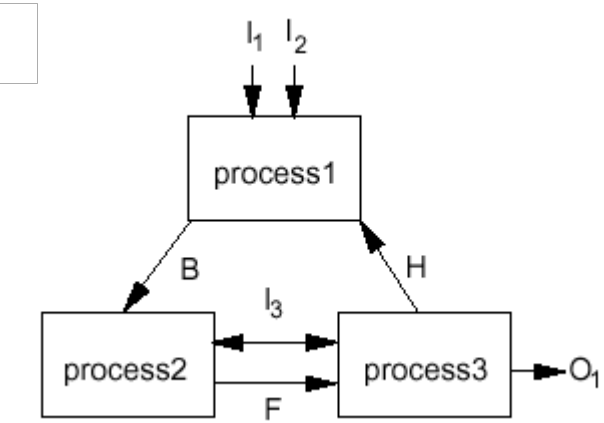






Control/Data Flow Graph

Διαμέριση Συμπεριφοράς



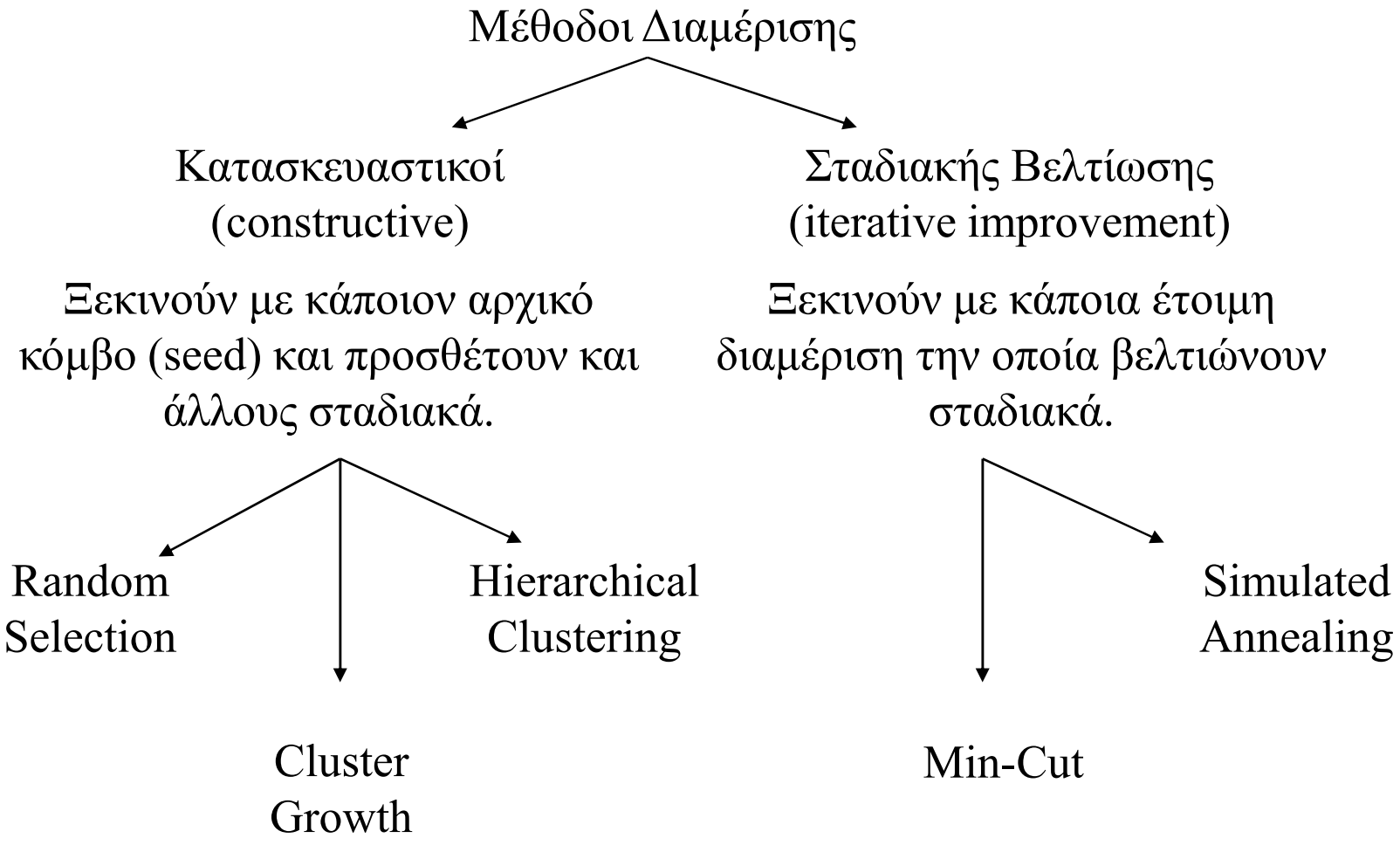
Inter−process communication

- ✓ Κάθε process μπορεί να θεωρηθεί σαν ένα module.
- ✓ Κορυφές του γράφου είναι τα modules και οι ακμές αναπαριστούν γενικές (global) μεταβλητές για επικοινωνία.

Η διαμέριση μπορεί να έχει στόχο την απόδοση (ταχύτητα) ή επιφάνεια.

Απόδοση	Επιφάνεια
Ομαδοποίηση κόμβων σε **κρίσιμα μονοπάτια** με στόχο την ελαχιστοποίηση των διασυνδέσεων τους	Ομαδοποίηση κόμβων **ίδιου τύπου λειτουργιών** με στόχο την ελαχιστοποίηση των διασυνδέσεων τους

Αλγόριθμοι Διαμέρισης



Αλγόριθμος Random Selection

- ✓ Είναι η απλούστερη κατασκευαστική μέθοδος.
- ✓ Χρησιμοποιείται για την παραγωγή μίας αρχικής διαμέρισης, η οποία παρακάτω περνάει από σταδιακή βελτίωση.
- ✓ Παράγει φτωχό αποτέλεσμα

Αλγόριθμος:

1. Επιλογή ενός τυχαίου κόμβου κάθε φορά.
2. Τοποθέτηση του σε ομάδες με στόχο την διατήρηση του μεγέθους τους σταθερού.

Αλγόριθμος Cluster Growth

- ✓ Εκτελεί τρεις διαδικασίες:

  1. Επιλογή του seed

  2. Επιλογή του κόμβου που θα τοποθετηθεί

  3. Τοποθέτηση του κόμβου

- ✓ Οι κόμβοι seeds (αρχικοί ομάδων) επιλέγονται ως εξής:

  1. Καθορίζονται από τον σχεδιαστή

  2. Επιλέγονται τυχαία.

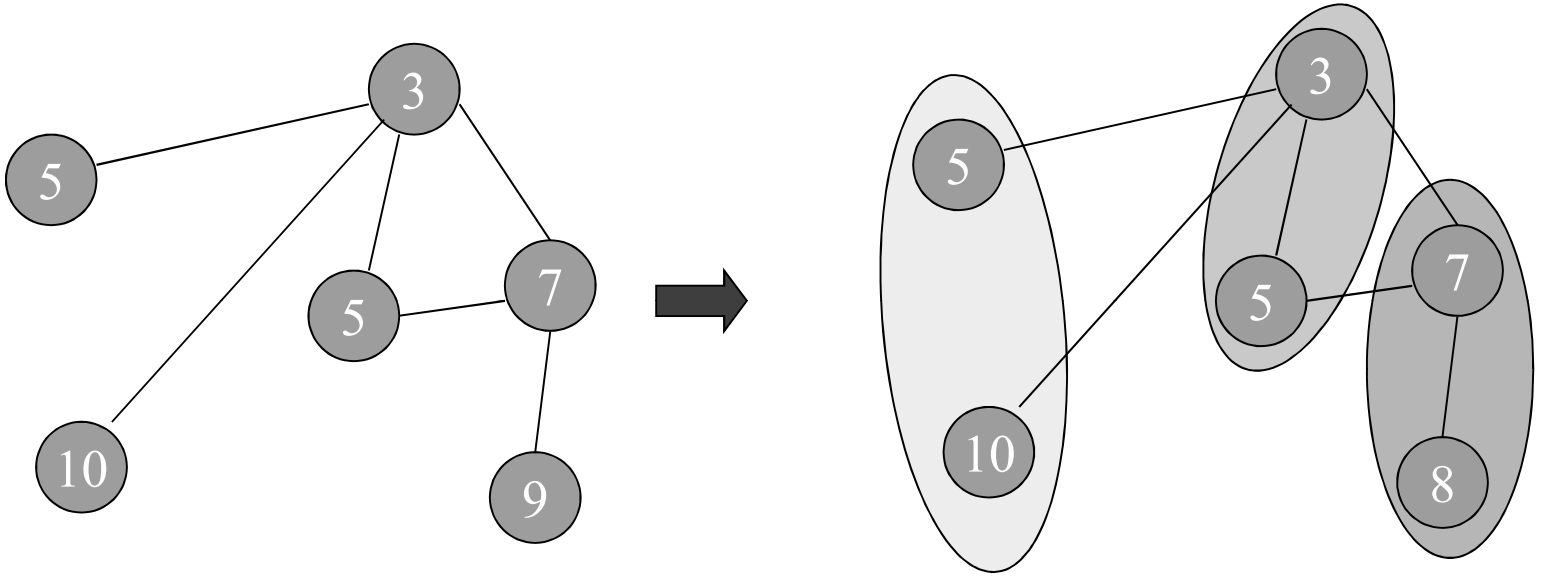
  3. Επιλέγονται με βάση ιδιότητες (μέγεθος, αριθμός ακμών διασύνδεσης,…)

Greedy Approach

- ✓ Τοποθετεί τους κόμβους σε ομάδες με βάση κάποια μέτρηση κοντινότητας (πχ μέγιστο αριθμό διασυνδέσεων μεταξύ μη τοποθετημένων και τοποθετημένων κόμβων).
- ✓ Όταν μία ομάδα γεμίσει προχωράει στην επόμενη.

Αλγόριθμος Cluster Growth

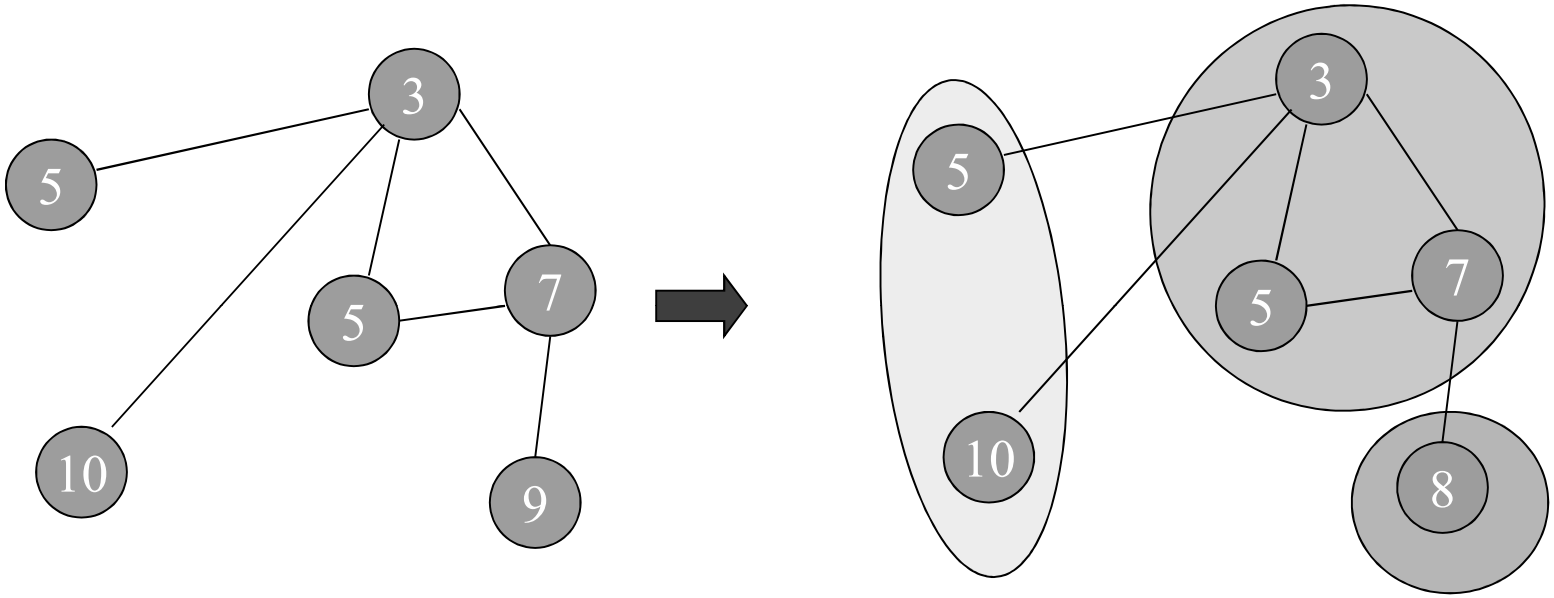
Τυχαία Επιλογή



Περιορισμός: κόστος υπογράφου ≤ 15

Αλγόριθμος Cluster Growth

Επιλογή για ελαχιστοποίηση ακμών διασύνδεσης υπογράφων

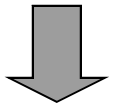


Περιορισμός: κόστος υπογράφου ≤ 15

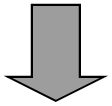
Αλγόριθμος Hierarchical Clustering

Αλγόριθμος:

1. Τοποθετεί τους κόμβους σε ομάδες με βάση την κοντινότητα.
2. Τα δύο πιο “κοντινά” αντικείμενα ομαδοποιούνται και θεωρούνται ένα.
3. Η διαδικασία επαναλαμβάνεται για τα νέα αντικείμενα που προκύπτουν.
4. Σταματά όταν όλοι οι κόμβοι έχουν τοποθετηθεί σε μία ομάδα.

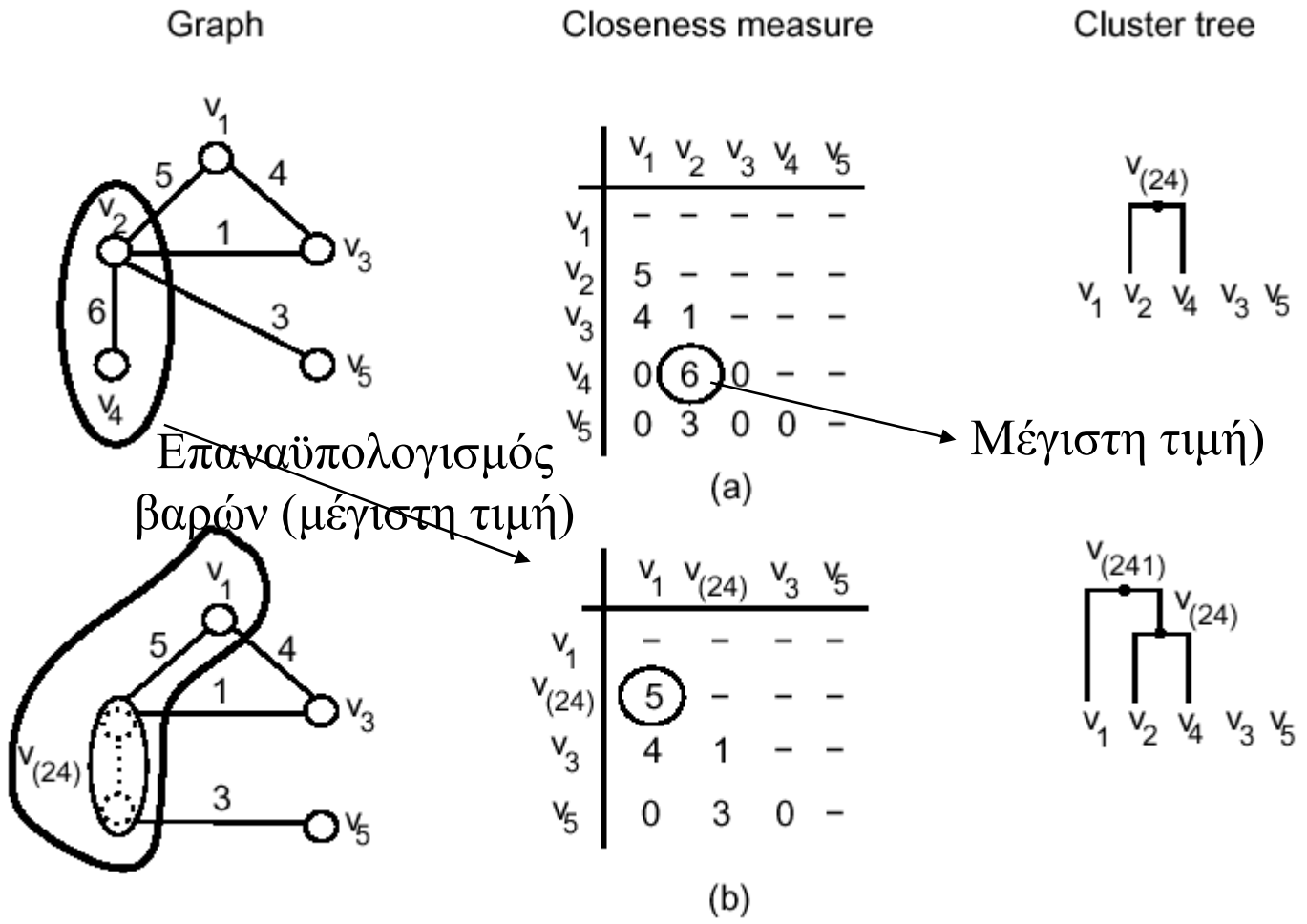


Δημιουργία Ιεραρχικού Δέντρου Ομάδων

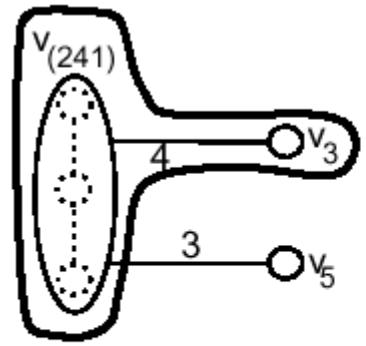


Κάθε διαχωριστική γραμμή στο δέντρο
υποδεικνύει ομάδες (clusters).

Αλγόριθμος Hierarchical Clustering

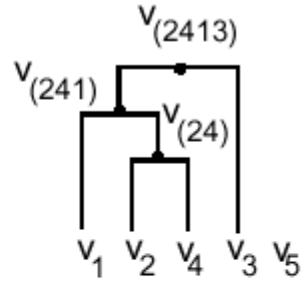


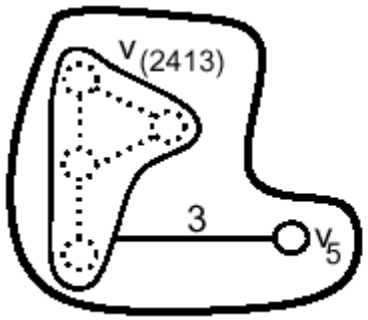
Αλγόριθμος Hierarchical Clustering



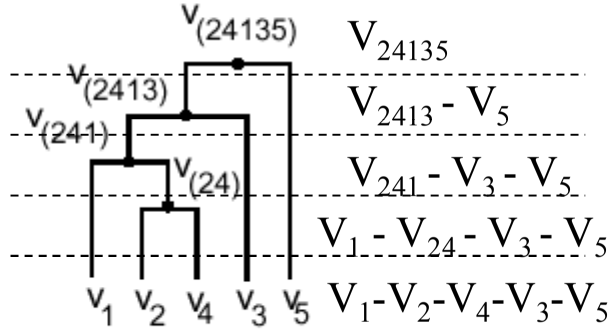
	v(241)	v3	v5
v(241)	–	–	–
v3	4	–	–
v5	3	0	–

(c)





	v(2413)	v5
v(2413)	–	–
v5	3	–



- ✓ Μία cutline κοντά στα φύλλα οδηγεί σε διαμέριση με πολλές και μικρές ομάδες από κόμβους κοντινούς.
- ✓ Μία cutline μακριά από τα φύλλα οδηγεί σε διαμέριση με λίγες και μεγάλες ομάδες από πιο μακρινούς κόμβους.

Αλγόριθμος Hierarchical Clustering

Παραλλαγή Ιεραρχικής Μεθόδου: Μέθοδος Πολλαπλών Βαθμίδων

- ✓ Το Clustering γίνεται σε πολλαπλές βαθμίδες (stages).
- ✓ Κάθε βαθμίδα χρησιμοποιεί διαφορετικό κριτήριο κοντινότητας.
- ✓ Πρώτα λαμβάνεται υπόψη το κριτήριο υψηλότερης προτεραιότητας, μετά το επόμενο κ.ο.κ.
- ✓ Λαμβάνει υπόψη πολλαπλά κριτήρια χωρίς να χρειάζεται καθορισμός βαρών (weights) που είναι πειραματικός.

Αλγόριθμοι Σταδιακής Βελτίωσης

Αλγόριθμος Min-CUT

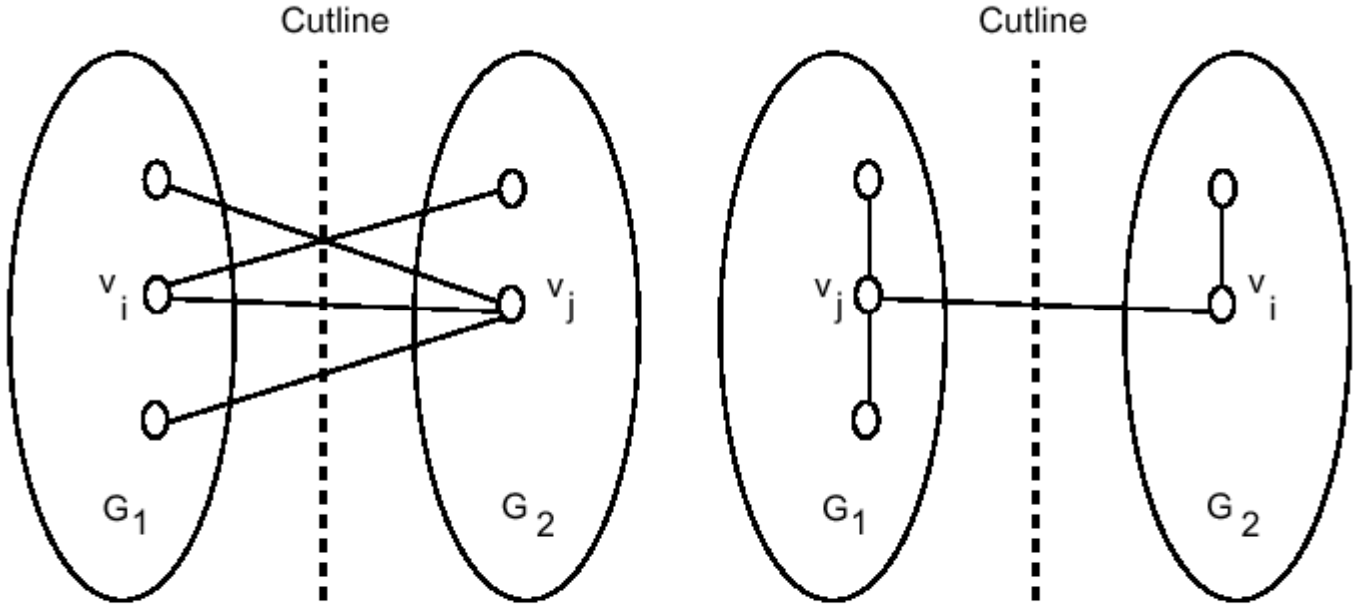
Ιδέα: Εναλλάσσει δύο ομάδες κορυφών μεταξύ δύο διαμερίσεων με στόχο την μέγιστη βελτίωση.

Στόχος: Διαίρεση ενός γράφου σε δύο διαμερίσεις ίσου μεγέθους, με ελαχιστοποίηση των διασυνδέσεων μεταξύ τους.

1. Ξεκινάει με μία αρχική διαμέριση σε δύο ομάδες.
2. Σε κάθε επανάληψη εναλλάσσει k ζεύγη κορυφών μεταξύ των δύο διαμερίσεων.
3. Σταματάει όταν δεν υπάρχει παραπέρα βελτίωση.
4. Η διαδικασία επαναλαμβάνεται ξεκινώντας από πολλές διαφορετικές αρχικές διαμερίσεις.
5. Τελικά επιλέγεται η καλύτερη λύση.

Αλγόριθμος Min Cut

Ελαχιστοποίηση Διασυνδέσεων



Before interchange of Vi and Vj

After interchange of Vi and Vj

Αλγόριθμος Min Cut

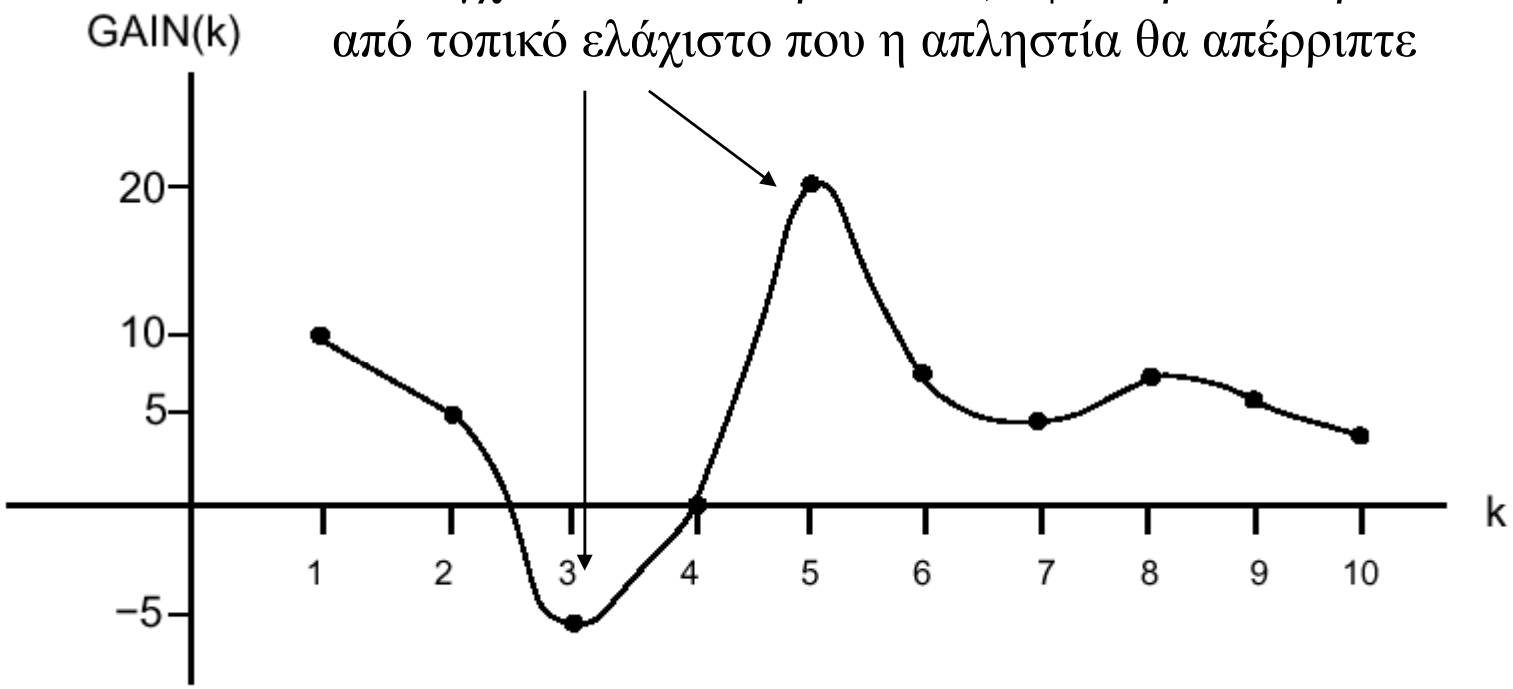
- ✓ Ο αλγόριθμος MinCut είναι Greedy (σταματά όταν δεν υπάρχει βελτίωση).
- ✓ Η λύση που επιτυγχάνεται αντιστοιχεί σε τοπικό ελάχιστο.

<u>Εναλλακτική Λύση</u>

1. Κάνουμε εναλλαγές όπως πριν, αλλά δεν σταματάμε όταν μία εναλλαγή δεν δίνει βελτίωση.
2. Επιλέγουμε τις n πρώτες που μεγιστοποιούν το συνολικό κέρδος

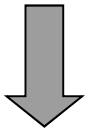
Αλγόριθμος Min Cut

Επιτυγχάνει το ολικό βέλτιστο, αφού πρώτα περάσει από τοπικό ελάχιστο που η απληστία θα απέρριπτε



Αλγόριθμος Simulated Annealing

Αποδέχεται ακολουθίες εναλλαγών όπου μερικές έχουν αρνητικό κέρδος, εφόσον όμως το συνολικό κέρδος είναι θετικό.



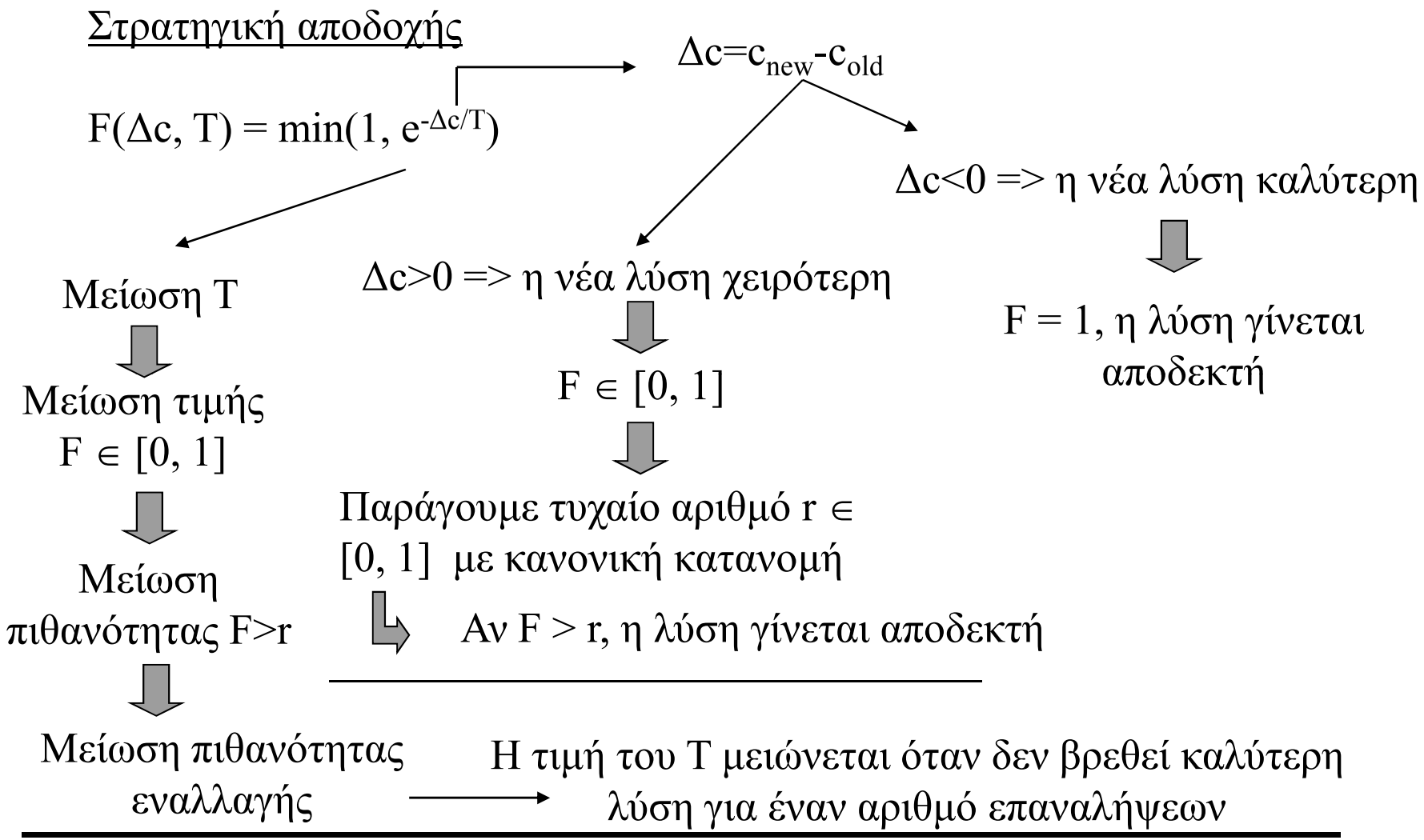
Αποφεύγει τοπικά ελάχιστες λύσεις

Μοιάζει με την φυσική διαδικασία κατά την οποία για να βρεθεί ή ελάχιστη ενεργειακή κατάσταση ενός υλικού πρώτα το λιώνουμε και μετά το ψύχουμε σταδιακά.

Αλγόριθμος Simulated Annealing

- ✓ Ξεκινάει με μία τυχαία αρχική διαμέριση.
- ✓ Οι μετακινήσεις (από ένα αντικείμενο σε ένα άλλο) επιλέγονται τυχαία.
- ✓ Το κόστος κάθε μετακίνησης ελέγχεται εάν ικανοποιεί κάποιο κριτήριο αποδοχής το οποίο εξαρτάται από την τρέχουσα θερμοκρασία T.
- ✓ Η πιθανότητα αποδοχής είναι ανάλογη με την τρέχουσα θερμοκρασία.
- ✓ Όσο η θερμοκρασία μειώνεται, μειώνεται και η πιθανότητα να γίνει η εναλλαγή.

Αλγόριθμος Simulated Annealing



Εφαρμογές Διαμέρισης

- ✓ **Scheduling**: διαμέριση των λειτουργιών σε ομάδες. Κάθε ομάδα ανατίθεται σε κάποιο βήμα ελέγχου.
- ✓ **Unit Selection**: διαμέριση λειτουργικών τελεστών σε ομάδες. Κάθε ομάδα αντιστοιχεί σε ένα **είδος** λειτουργικής μονάδας (πχ. η πρόσθεση μπορεί να ανατεθεί σε αθροιστές, αθροιστές/αφαιρέτες, ALUs, κλπ).
- ✓ **Allocation**: διαμέριση λειτουργιών σε ομάδες. Κάθε ομάδα ανατίθεται σε ένα resource.
- ✓ **Behavioral Decomposing**: διαμέριση μίας περιγραφής συμπεριφοράς σε processes. Οι processes αυτές αντιστοιχούνται σε ισάριθμα FSMDs.
- ✓ **Chip partition**: Διαμέριση του chip σε modules.

Unit Selection

- ✓ Διαμέριση των λειτουργιών σε ένα CDFG σε ομάδες με βάση την ομοιότητα τους και την συνταύτιση τους σε χρόνο (concurrency).
- ✓ Ομοιότητα: αριθμός κοινών ιδιοτήτων. ⟹ $|A_1 \cap A_2| / |A_1 \cup A_2|$

Έστω $fcost(o_1, o_2, o_3, \ldots, o_n)$ το κόστος λειτουργικής μονάδας που εκτελεί τις λειτουργίες $o_1, o_2, o_3, \ldots, o_n$

Unit Selection

$$\text{Functional Proximity: } fp(o_i, o_j) = \frac{fcost(o_i) + fcost(o_j) - fcost(o_i,o_j)}{fcost(o_i,o_j)}$$

Φυσική Σημασία:

Όσο μικρότερο είναι το κόστος υλοποίησης των λειτουργιών-τελεστών από κοινή μονάδα ($fcost(o_i, o_j)$) τόσο μεγαλύτερη η εγγύτητα (proximity)

Μεγάλη εγγύτητα ⇒ *Υλοποίηση από ίδια μονάδα* ⇒ *Μείωση παραλληλισμού*

<u>Παράδειγμα</u>

Κόστος αθροιστή = Κόστος αφαιρέτη = 6 πύλες

Κόστος αθροιστή/αφαιρέτη = 8 πύλες

} $fp(+,-) = (6+6-8)/8 = 0,5$

Unit Selection

Communication Proximity: $cp(o_i, o_j) = \frac{cconn(o_i,o_j)}{tconn(o_i,o_j)}$

$cconn(o_i,o_j)$ ⟶ Κοινές διασυνδέσεις

$tconn(o_i,o_j)$ ⟶ Συνολικές διασυνδέσεις

Φυσική Σημασία:

Μεγάλη τιμή της cp υποδεικνύει πολλές κοινές διασυνδέσεις για την υλοποίηση των λειτουργιών (δηλ. συμφέρει η χρήση κοινής μονάδας – μη παραλληλισμός)

<u>Παράδειγμα:</u> ένας τελεστής + και ένας – έχουν μία κοινή είσοδο και συνολικά 6 εισόδους. Τότε cp(+, -)=1/6

Potential Parallelism: $ppar(o_i, o_j) = \begin{cases} 1: \text{οι } o_i, o_j \text{ μπορούν να εκτελεστούν παράλληλα} \\ 0: \text{αλλιώς} \end{cases}$

Unit Selection

Μετρική Κοντινότητας

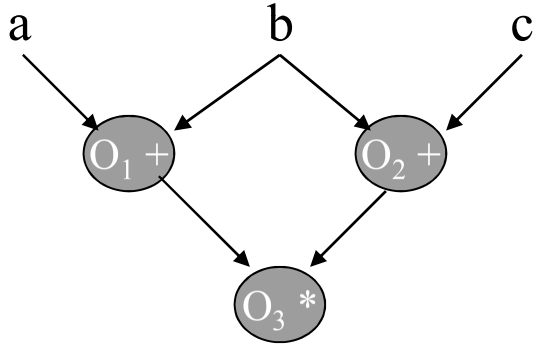
- ✓ Καθορίζει την απόσταση μεταξύ κάθε ζεύγους λειτουργιών.
- ✓ Μικρή απόσταση ⇒ μεγάλο πλεονέκτημα εάν μοιραστούν την ίδια μονάδα
- ✓ Ομαδοποίηση λειτουργιών σε μία μονάδα μειώνει την δυνατότητα παραλληλισμού τους

Distance: $dist(o_i, o_j) = -(a_1 \times fp(o_i, o_j)) - (a_2 \times cp(o_i, o_j)) + (a_3 \times ppar(o_i, o_j))$

Βάρη σημαντικότητας μετρικών

Unit Selection

- Όσο μεγαλύτερο βάρος δίνουμε σε μία μετρική, τόσο περισσότερο την λαμβάνουμε υπόψη στις αποφάσεις.
- Με χρήση της μετρικής απόστασης εφαρμόζεται η ιεραρχική μέθοδος διαμέρισης.



$fcost(o_1) = fcost(o_2) = fcost(o_1,o_2)$ άρα $fp(o_1, o_2)=1$

$cp(o_1, o_2) = 1/6$

$ppar(o_1, o_2) = 1$

$\Rightarrow$ $dist(o_1, o_2) = a_3-a_1-(1/6)a_2$