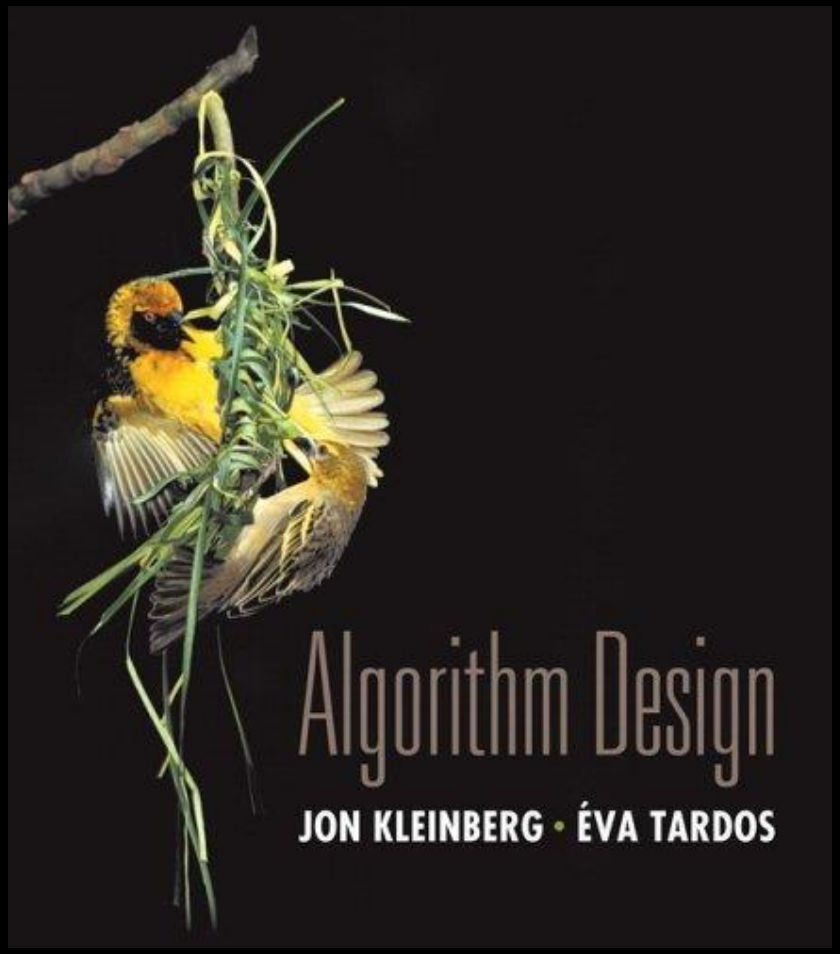



# Κεφάλαιο 11

## Προσεγγιστικοί Αλγόριθμοι

# Προσεγγιστικοί Αλγόριθμοι

Ερώτηση: Έστω ότι θέλουμε να λύσουμε ένα NP-hard πρόβλημα. Τι κάνουμε;
Απάντηση: Η θεωρία λέει ότι είναι μάλλον δύσκολο να βρεθεί ένας τέτοιος πολυωνυμικός αλγόριθμος.

Θα πρέπει να θυσιάσουμε ένα από τα εξής:

- Βέλτιστη επίλυση.
- Επίλυση σε πολυωνυμικό χρόνο.
- Επίλυση οποιουδήποτε στιγμιοτύπου του προβλήματος.

ρ-προσεγγιστικός αλγόριθμος

- Εγγυημένα πολυωνυμική πολυπλοκότητα (χρόνου)
- Εγγυημένα λύνει οποιοδήποτε στιγμιότυπο του προβλήματος
- Εγγυημένα βρίσκει λύσεις με λόγο ρ από τη βέλτιστη λύση

Δυσκολία: Πρέπει να αποδείξουμε ότι λύση είναι κοντά στη βέλτιστη χωρίς όμως να ξέρουμε τη βέλτιστη!

# Μερικές Γενικές Τεχνικές Προσέγγισης

- Τεχνική απληστίας
- Τεχνική τιμολόγησης
- Τεχνική γραμμικού προγραμματισμού και στρογγυλοποίησης
- Τεχνική δυναμικού προγραμματισμού σε στρογγυλοποιημένη εκδοχή της εισόδου

# 11.1 ΕΞΙΣΟΡΡΌΠΗΣΗ ΦΟΡΤΊΟΥ

# Εξισορρόπηση Φορτίου

Είσοδος: m ίδιες μηχανές, n εργασίες, η εργασία j έχει χρόνο επεξεργασίας ίσο με $t_j$.

- Η εργασία j πρέπει να τρέχει συνεχόμενα σε μία μηχανή.
- Μία μηχανή μπορεί να εκτελέσει το πολύ μία εργασία κάθε χρονική στιγμή.

Ορισμός: Έστω J(i) το υποσύνολο των εργασιών που ανατέθηκαν στην μηχανή i. Ο φόρτος της μηχανής i είναι $L_i = \Sigma_{j \in J(i)} t_j$.

Ορισμός: Η διάρκεια εκτέλεσης είναι ο μέγιστος φόρτος ανάμεσα σε όλες τις μηχανές $L = \max_i L_i$.

Εξισορρόπηση Φορτίου: Αντιστοίχησε κάθε εργασία σε μία μηχανή ώστε να ελαχιστοποιείται η διάρκεια εκτέλεσης.

# Εξισορρόπηση Φορτίου: Άπληστος Αλγόριθμος

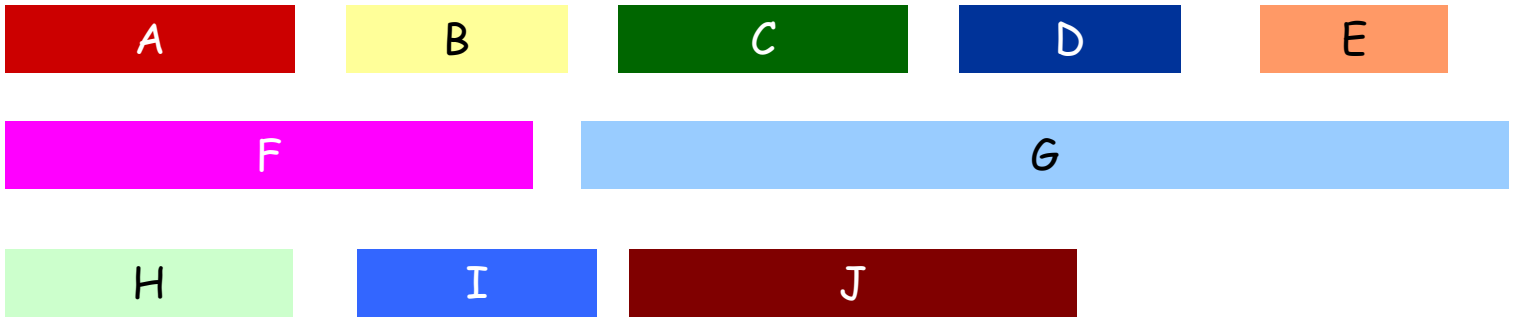
Αλγόριθμος:

- Έστω οι n εργασίες σε κάποια σειρά.
- Αντιστοιχούμε την εργασία j στη μηχανή της οποίας το φορτίο είναι το μικρότερο.

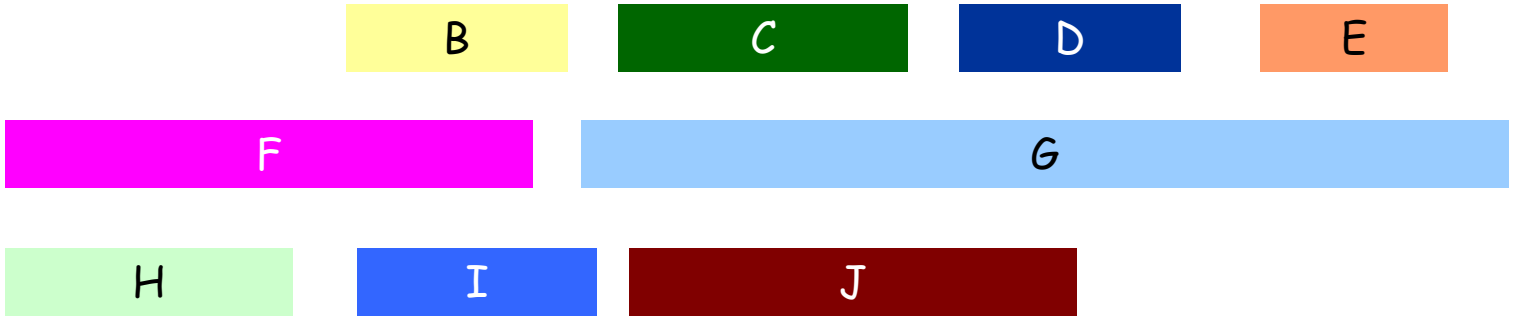
```
Greedy-Scheduling(m, n, t_1,t_2,...,t_n) {
    for i = 1 to m {
        L_i ← 0          ← Φορτίο μηχανής i
        J(i) ← φ         ← Εργασίες της μηχανής i
    }

    for j = 1 to n {
        i = argmin_k L_k ← Η μηχανή i έχει το μικρότερο φορτίο
        J(i) ← J(i) ∪ {j} ← Αντιστοίχηση της j στη μηχανή i
        L_i ← L_i + t_j  ← Ενημέρωση φορτίου της μηχανής i
    }
    return J(1), …, J(m)
}
```

Υλοποίηση: $O(n \log m)$ με μία ουρά προτεραιότητας.

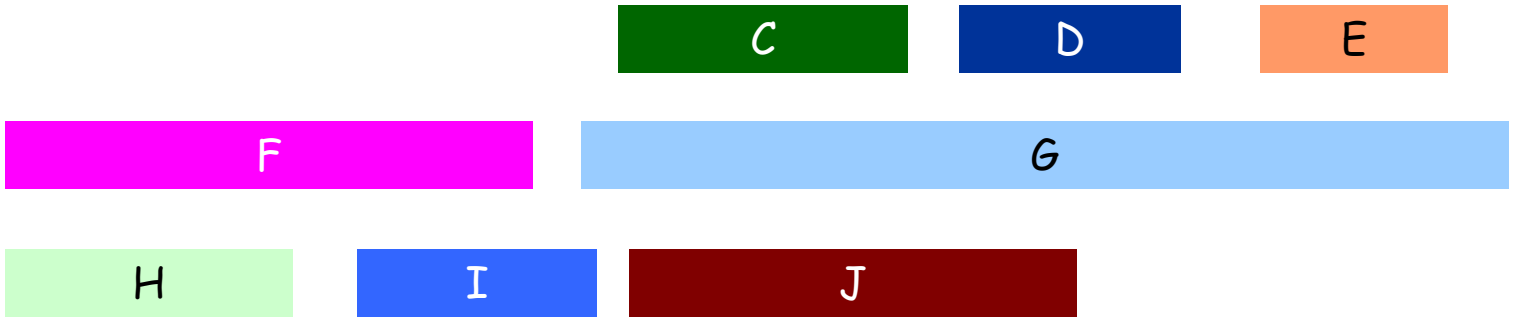
# Εξισορρόπηση Φορτίου: Άπληστος Αλγόριθμος



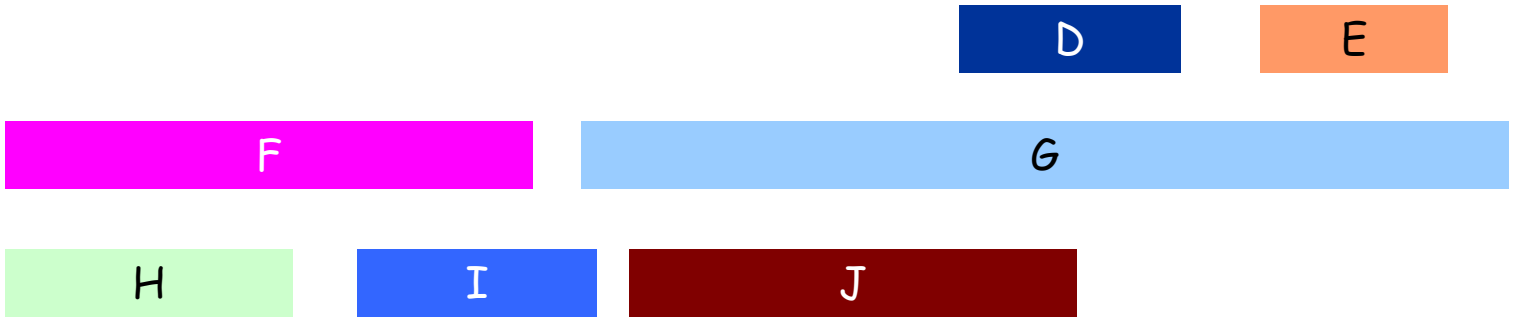

# Εξισορρόπηση Φορτίου: Άπληστος Αλγόριθμος




# Εξισορρόπηση Φορτίου: Άπληστος Αλγόριθμος



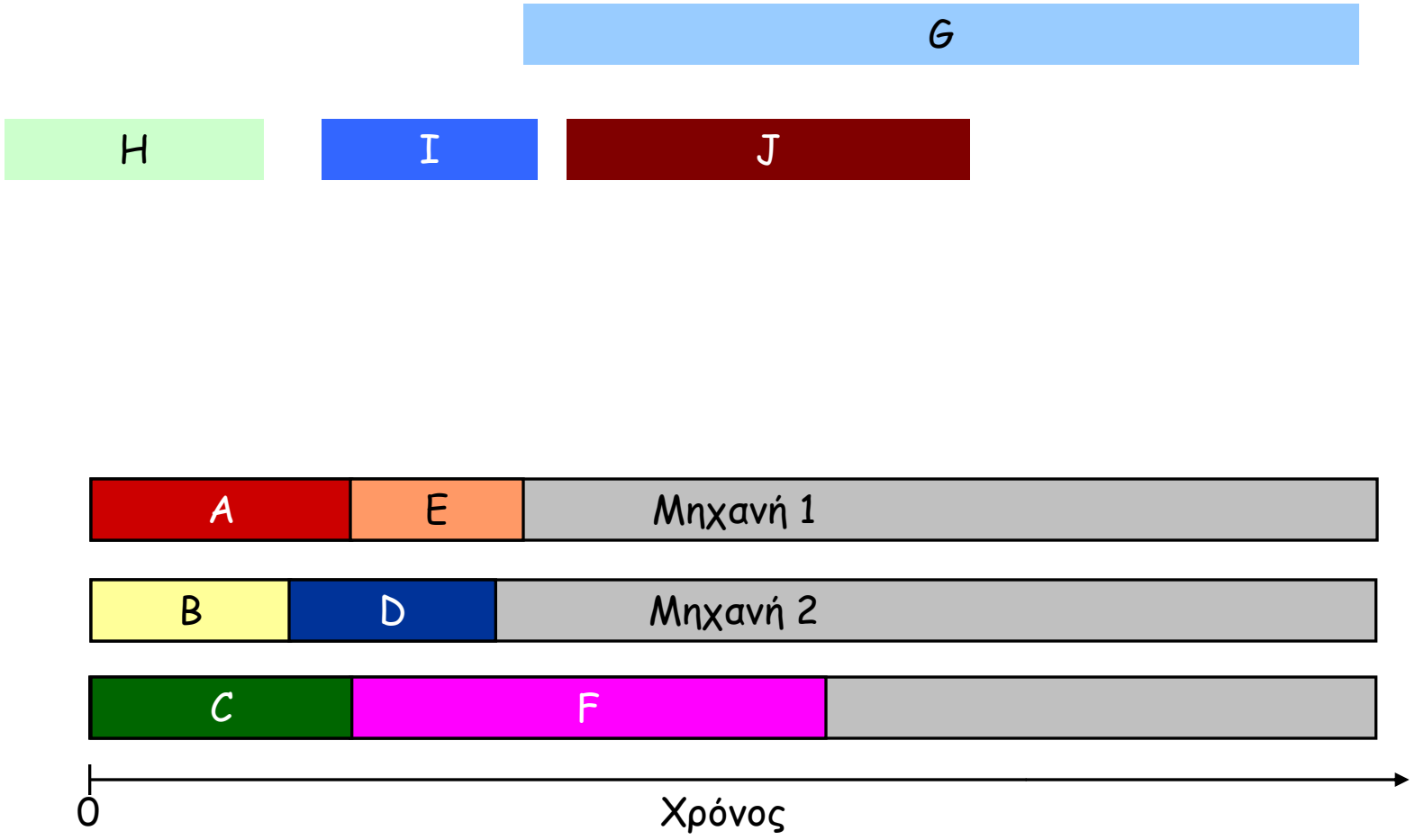

# Εξισορρόπηση Φορτίου: Άπληστος Αλγόριθμος



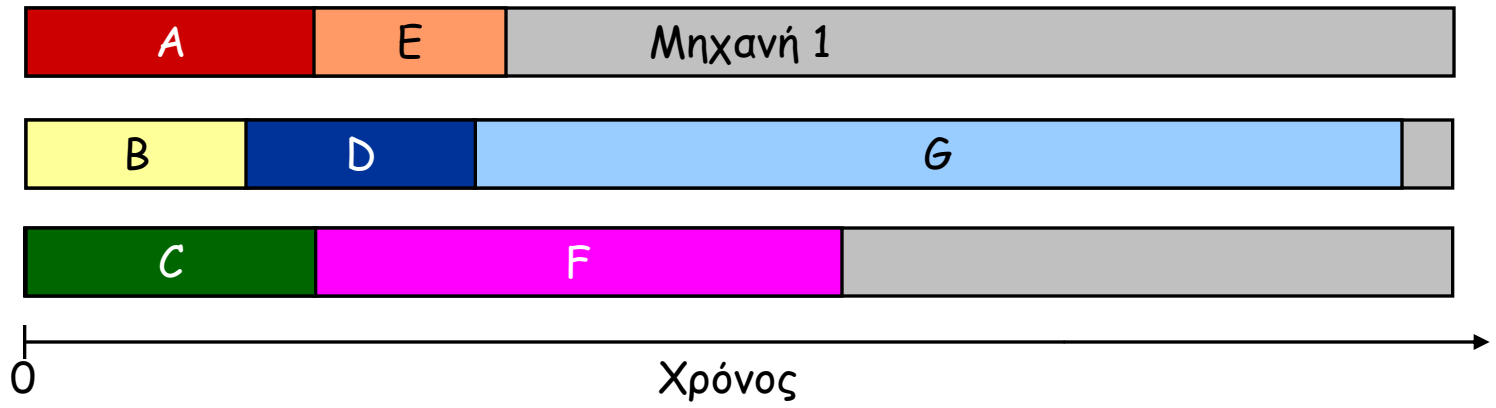

# Εξισορρόπηση Φορτίου: Άπληστος Αλγόριθμος


# Εξισορρόπηση Φορτίου: Άπληστος Αλγόριθμος

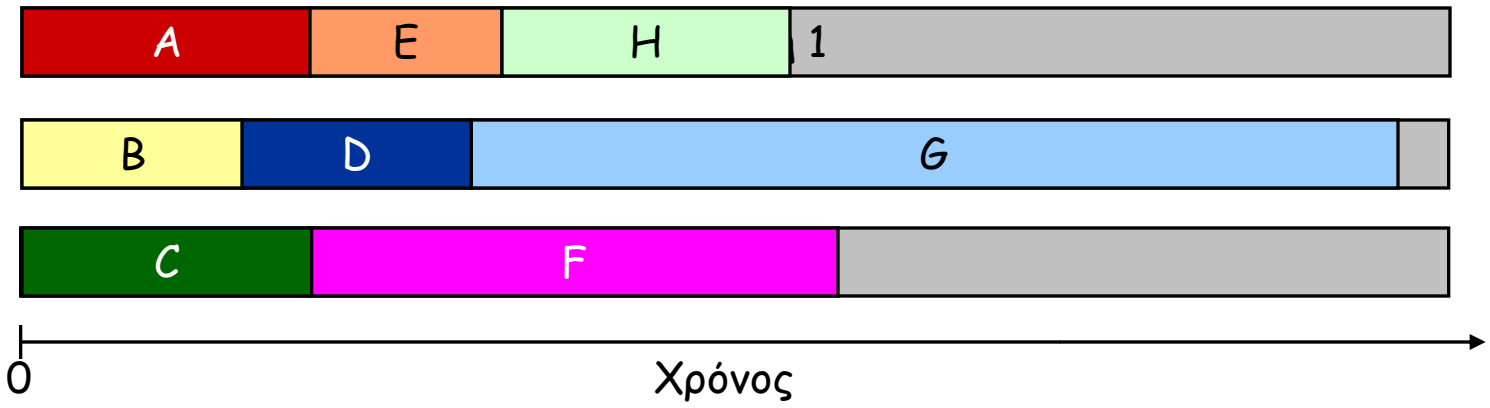

# Εξισορρόπηση Φορτίου: Άπληστος Αλγόριθμος



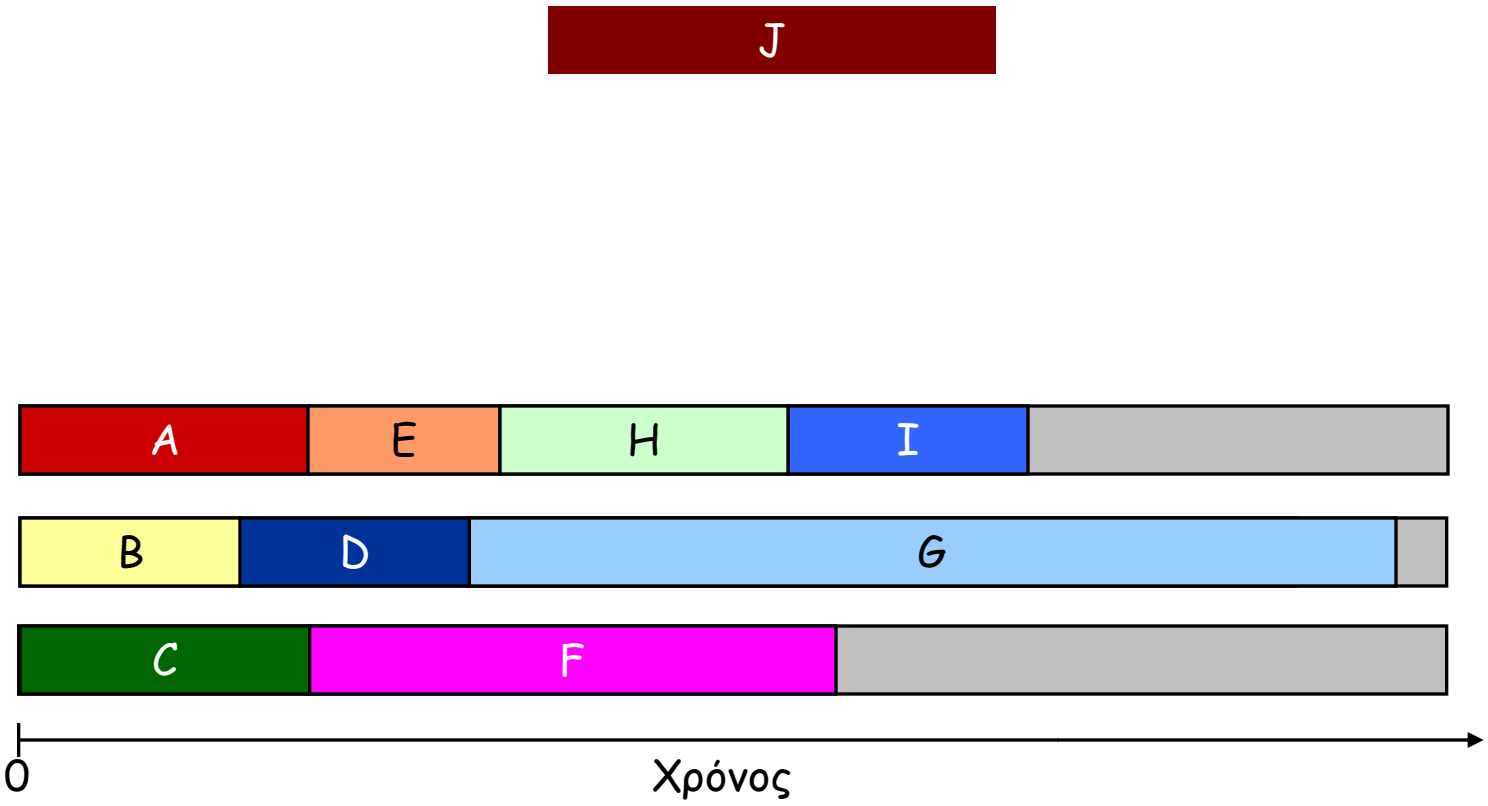
# Εξισορρόπηση Φορτίου: Άπληστος Αλγόριθμος

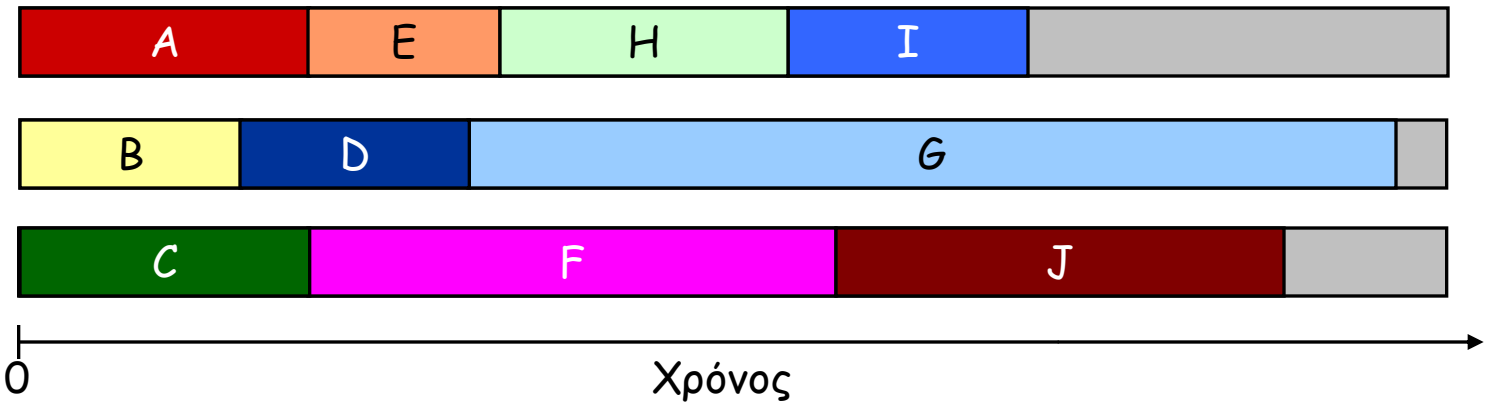
# Εξισορρόπηση Φορτίου: Άπληστος Αλγόριθμος

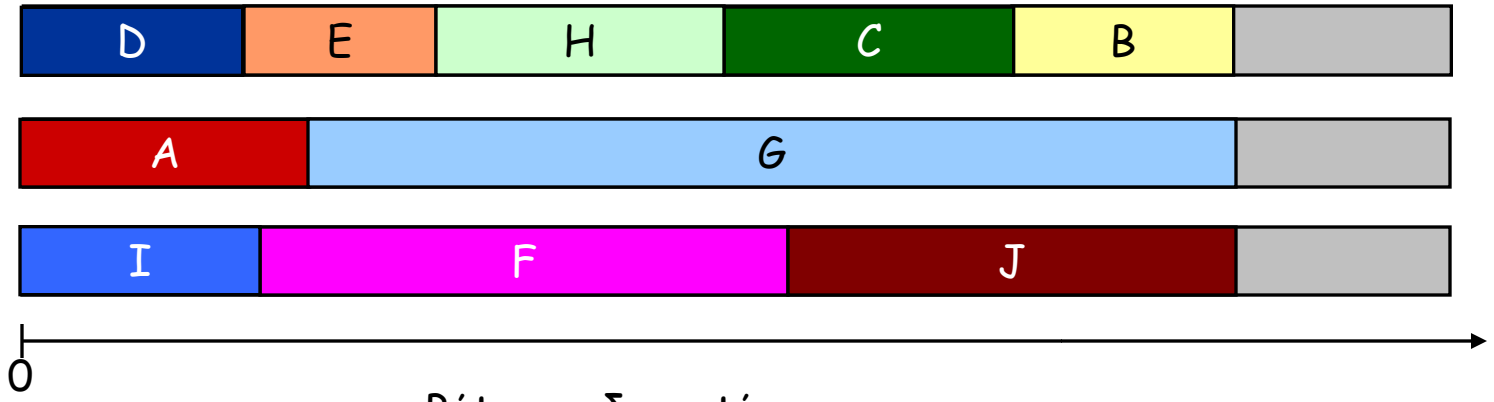
# Εξισορρόπηση Φορτίου: Άπληστος Αλγόριθμος



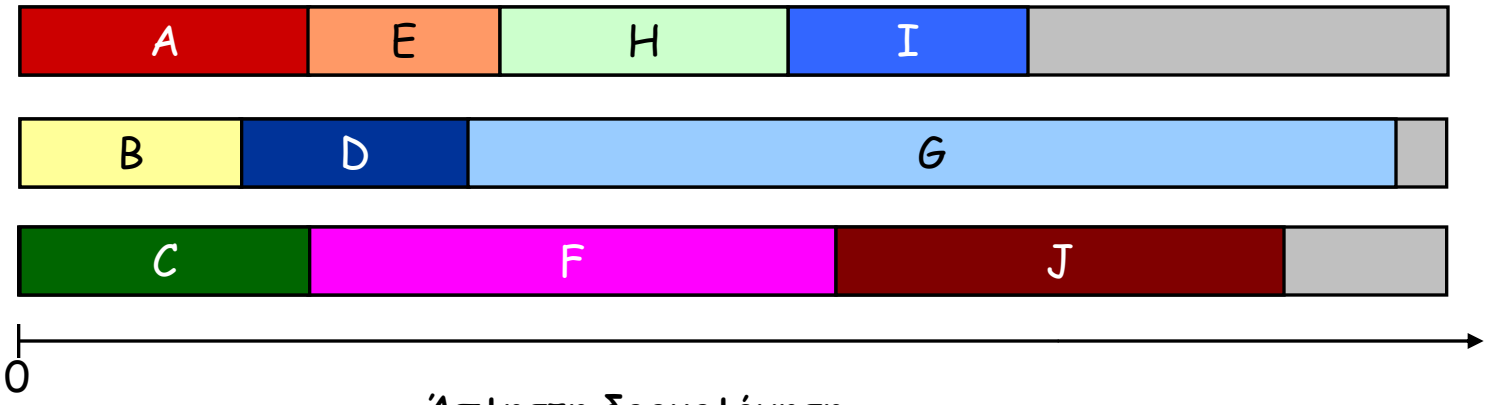
# Εξισορρόπηση Φορτίου: Άπληστος Αλγόριθμος



# Εξισορρόπηση Φορτίου: Άπληστος Αλγόριθμος




Βέλτιστη δρομολόγηση




Άπληστη δρομολόγηση

# Εξισορρόπηση Φορτίου: Ανάλυση Άπληστου Αλγόριθμου

**Θεώρημα:** [Graham, 1966] Ο άπληστος αλγόριθμος είναι μία 2-προσέγγιση.

- Πρώτα η ανάλυση χειρότερης περίπτωσης για τον προσεγγιστικό αλγόριθμο.
- Συγκρίνουμε τη λύση με τη βέλτιστη διάρκεια εκτέλεσης L*.

**Λήμμα 1:** $L^* \geq \frac{1}{m} \sum_j t_j$

**Απόδειξη:**

- Ο συνολικός χρόνος επεξεργασίας είναι $\Sigma_j t_j$ .
- Μία από τις m μηχανές θα πρέπει να κάνει τουλάχιστον 1/m κλάσμα της συνολικής εργασίας.

**Λήμμα 2:** Η βέλτιστη διάρκεια εκτέλεσης $L^* \geq \max_j t_j$.

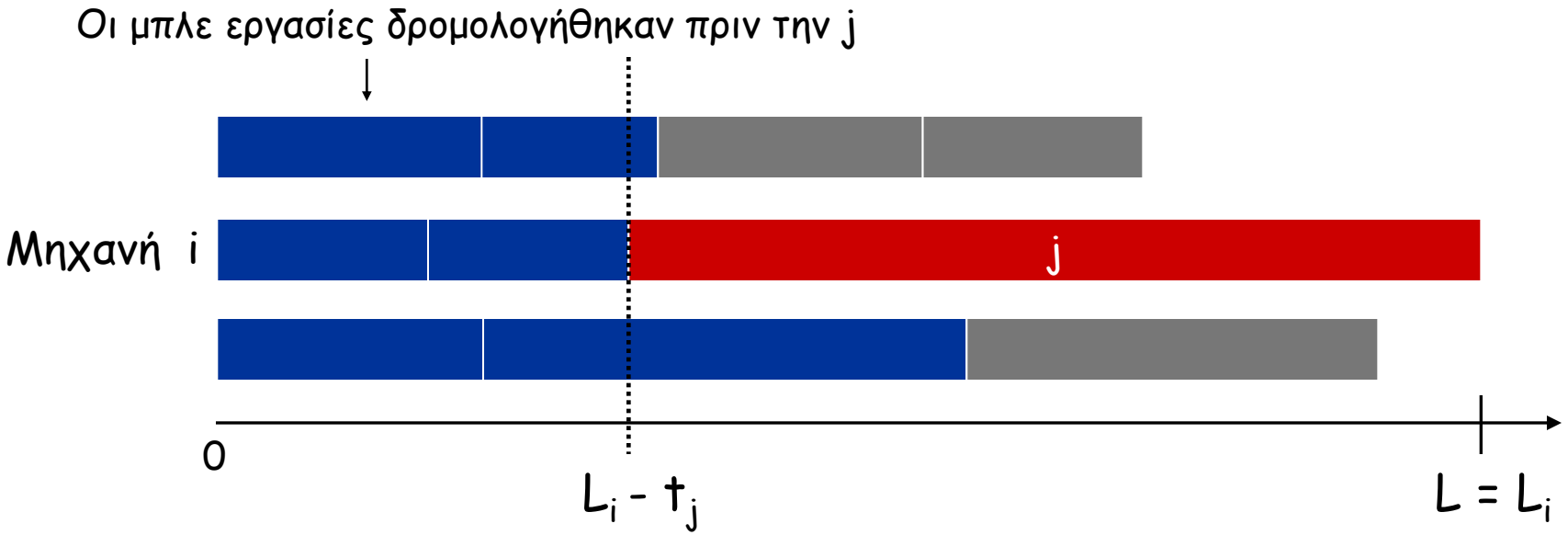
**Απόδειξη:** Κάποια μηχανή θα πρέπει να εξυπηρετήσει την πιο χρονοβόρα εργασία. ▪

# Εξισορρόπηση Φορτίου: Ανάλυση Άπληστου Αλγόριθμου

**Θεώρημα:** Ο άπληστος αλγόριθμος είναι 2-προσέγγιση.

**Απόδειξη:** Έστω το φορτίο $L_i$ στην μηχανή i με μέγιστη διάρκεια εκτέλεσης.

- Έστω j η τελευταία εργασία στην μηχανή i.
- Όταν η εργασία j αντιστοιχήθηκε στην μηχανή i, η i είχε το μικρότερο φόρτο. Το φορτίο της πριν την αντιστοίχηση ήταν $L_i - t_j \Rightarrow L_i - t_j \leq L_k$ για κάθε $1 \leq k \leq m$.



# Εξισορρόπηση Φορτίου: Ανάλυση Άπληστου Αλγόριθμου

**Θεώρημα:** Ο άπληστος αλγόριθμος είναι 2-προσέγγιση.

**Απόδειξη:** Έστω το φορτίο $L_i$ στην μηχανή με μέγιστη διάρκεια εκτέλεσης

- Έστω j η τελευταία εργασία στην μηχανή i.
- Όταν η εργασία j αντιστοιχήθηκε στην μηχανή i, η i είχε το μικρότερο φόρτο. Το φορτίο της πριν την αντιστοίχηση ήταν $L_i - t_j \Rightarrow L_i - t_j \leq L_k$ για κάθε $1 \leq k \leq m$.
- Άθροισμα ανισοτήτων σε όλα τα k και διαίρεση με m:

$$\begin{aligned} L_i - t_j &\leq \tfrac{1}{m}\textstyle\sum_k L_k \\ &= \tfrac{1}{m}\textstyle\sum_k t_k \\ \text{Λήμμα 2} \longrightarrow &\leq L^* \end{aligned}$$

- Άρα $$L_i = \underbrace{(L_i - t_j)}_{\leq L^*} + \underbrace{t_j}_{\leq L^*} \leq 2L^*.$$ ▪

Λήμμα 1

# Εξισορρόπηση Φορτίου: Ανάλυση Άπληστου Αλγόριθμου

Ερώτηση. Είναι η ανάλυση αυστηρή;

Απάντηση. Ουσιαστικά, ναι.

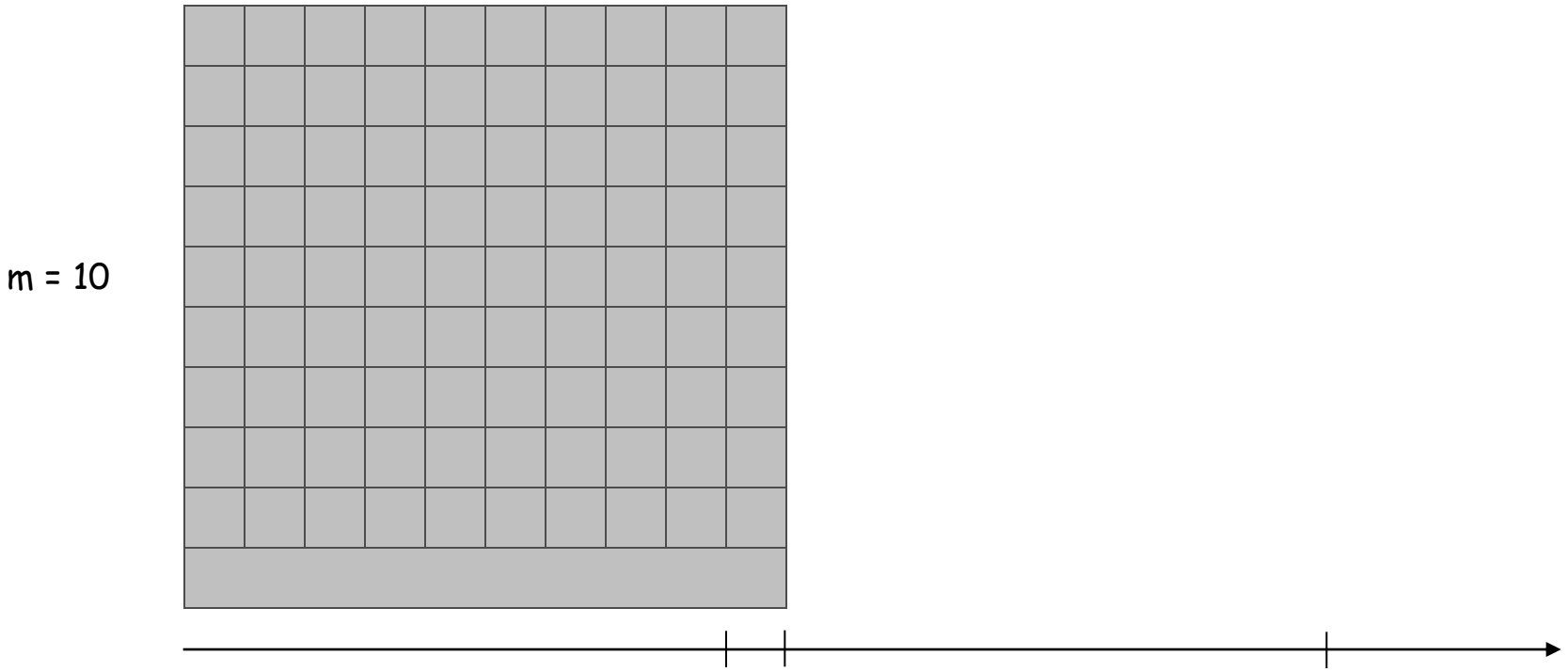
π.χ. m μηχανές, m(m-1) εργασίες διάρκειας 1, μία εργασία διάρκειας m

# Εξισορρόπηση Φορτίου: Ανάλυση Άπληστου Αλγόριθμου

Ερώτηση. Είναι η ανάλυση αυστηρή;

Απάντηση. Ουσιαστικά, ναι (στην πραγματικότητα είναι 2-1/m).

π.χ.: m μηχανές, m(m-1) εργασίες διάρκειας 1, μία εργασία διάρκειας m



# Εξισορρόπηση Φορτίου: Κανόνας Μεγαλύτερης Διάρκειας Εργασία Πρώτη

Μεγαλύτερης Διάρκειας Εργασία Πρώτη. Ταξινόμηση n εργασιών σε φθίνουσα σειρά διάρκειας εκτέλεσης, και έπειτα εκτέλεση του άπληστου αλγόριθμου.

```
LPT-List-Scheduling(m, n, t_1,t_2,...,t_n) {
    Sort jobs so that t_1 ≥ t_2 ≥ ... ≥ t_n

    for i = 1 to m {
        L_i ← 0          ← Φορτίο μηχανής i
        J(i) ← φ         ← Εργασίες της μηχανής i
    }

    for j = 1 to n {
        i = argmin_k L_k ← Η μηχανή i έχει το μικρότερο φορτίο
        J(i) ← J(i) ∪ {j} ← Αντιστοίχηση της j στη μηχανή i
        L_i ← L_i + t_j  ← Ενημέρωση φορτίου της μηχανής i
    }
    return J(1), …, J(m)
}
```

# Εξισορρόπηση Φορτίου: Κανόνας Μεγαλύτερης Διάρκειας Εργασία Πρώτη

Παρατήρηση: Αν έχουμε m το πολύ εργασίες, τότε ο άπληστος είναι βέλτιστος.
Απόδειξη: Βάλε την κάθε εργασία σε δικιά της μηχανή. ▪

Λήμμα 3. Αν υπάρχουν περισσότερες από m εργασίες, $L^* \geq 2\, t_{m+1}$.
Απόδειξη:

- Έστω οι πρώτες m+1 εργασίες $t_1, \ldots, t_{m+1}$.
- Αφού τα $t_i$ είναι σε φθίνουσα σειρά, κάθε μία απαιτεί τουλάχιστον $t_{m+1}$ χρόνο.
- Υπάρχουν m+1 εργασίες και m μηχανές, και άρα (αρχή περιστερώνα) τουλάχιστον μία μηχανή έχει δύο εργασίες ▪

Θεώρημα: Ο αλγόριθμος είναι 3/2-προσέγγιση.
Απόδειξη: Όπως και πριν:

$$L_i = \left(L_i - t_j\right) + t_j \leq \frac{3}{2} L^*$$

Λήμμα 3
(υπόθεση ότι πλήθος εργασιών >m)

# Εξισορρόπηση Φορτίου: Κανόνας Μεγαλύτερης Διάρκειας Εργασία Πρώτη

Ερώτηση: Είναι η 3/2-προσέγγιση αυστηρή;
Απάντηση: Όχι.

Θεώρημα. [Graham, 1969] Ο αλγόριθμος είναι 4/3-προσέγγισση.
Απόδειξη: άλλη φορά αν και όχι τόσο δύσκολη…

Ερώτηση: Είναι η 4/3-προσέγγιση αυστηρή;
Απάντηση: Ουσιαστικά ναι.

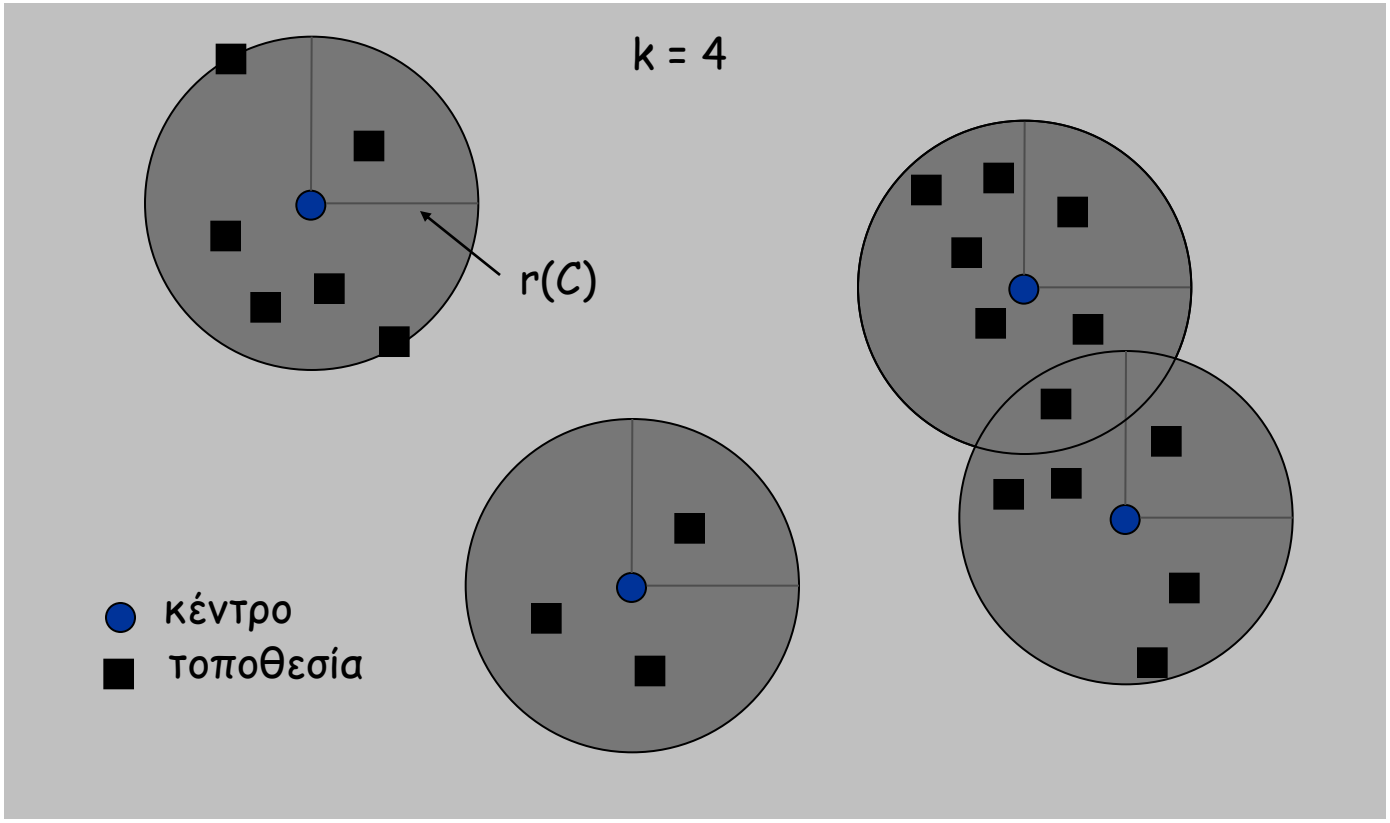
π.χ. m μηχανές, n = 2m+1 εργασίες, 2 εργασίες διάρκειας m+1, m+2, …, 2m-1 και μία εργασία διάρκειας m. (κάντε το μόνοι σας…)

# 11.2 Επιλογή Κέντρων

# Πρόβλημα Επιλογής Κέντρων

Είσοδος: Σύνολο από n τοποθεσίες $s_1, \ldots, s_n$ και ένα ακέραιο $k > 0$.

Πρόβλημα Επιλογής Κέντρων: Επέλεξε k κέντρα C έτσι ώστε να ελαχιστοποιείται η μέγιστη απόσταση από μία τοποθεσία προς το κοντινότερο κέντρο.



# Πρόβλημα Επιλογής Κέντρων

Σημειογραφία:

- dist(x, y) = απόσταση μεταξύ x και y.
- $dist(s_i, C) = \min_{c \in C} dist(s_i, c)$ = απόσταση από $s_i$ στο κοντινότερο κέντρο.
- $r(C) = \max_i dist(s_i, C)$ = ελάχιστη ακτίνα κάλυψης.

Στόχος: Εύρεση συνόλου κέντρων $C^*$ που ελαχιστοποιεί την $r(C)$, με τον περιορισμό ότι $|C^*| = k$.

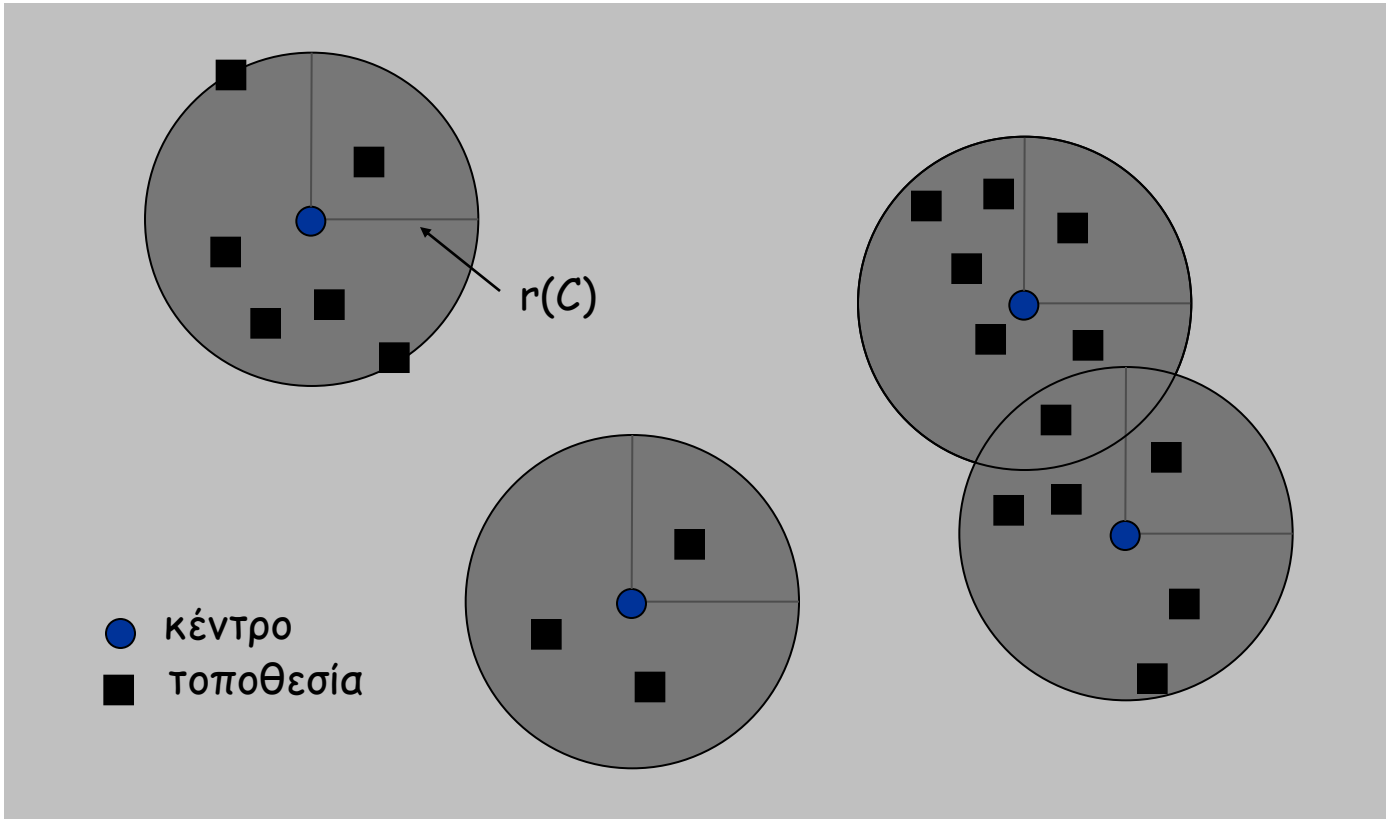
Ιδιότητες συνάρτησης απόστασης:

- dist(x, x) = 0
- dist(x, y) = dist(y, x)
- $dist(x, y) \leq dist(x, z) + dist(z, y)$ (τριγωνική ανισότητα)

# Παράδειγμα Επιλογής Κέντρων

Κάθε τοποθεσία είναι ένα σημείο στο επίπεδο, ένα κέντρο μπορεί να είναι οποιοδήποτε σημείο στο επίπεδο, dist(x, y) = ευκλείδια απόσταση.

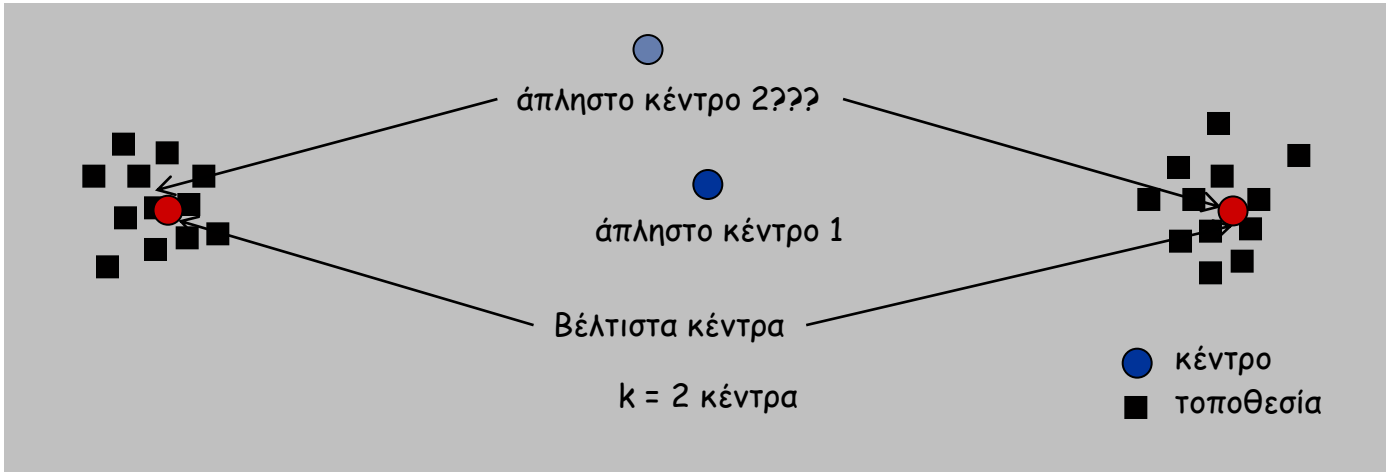
Παρατήρηση: ο χώρος αναζήτησης μπορεί να είναι άπειρος!



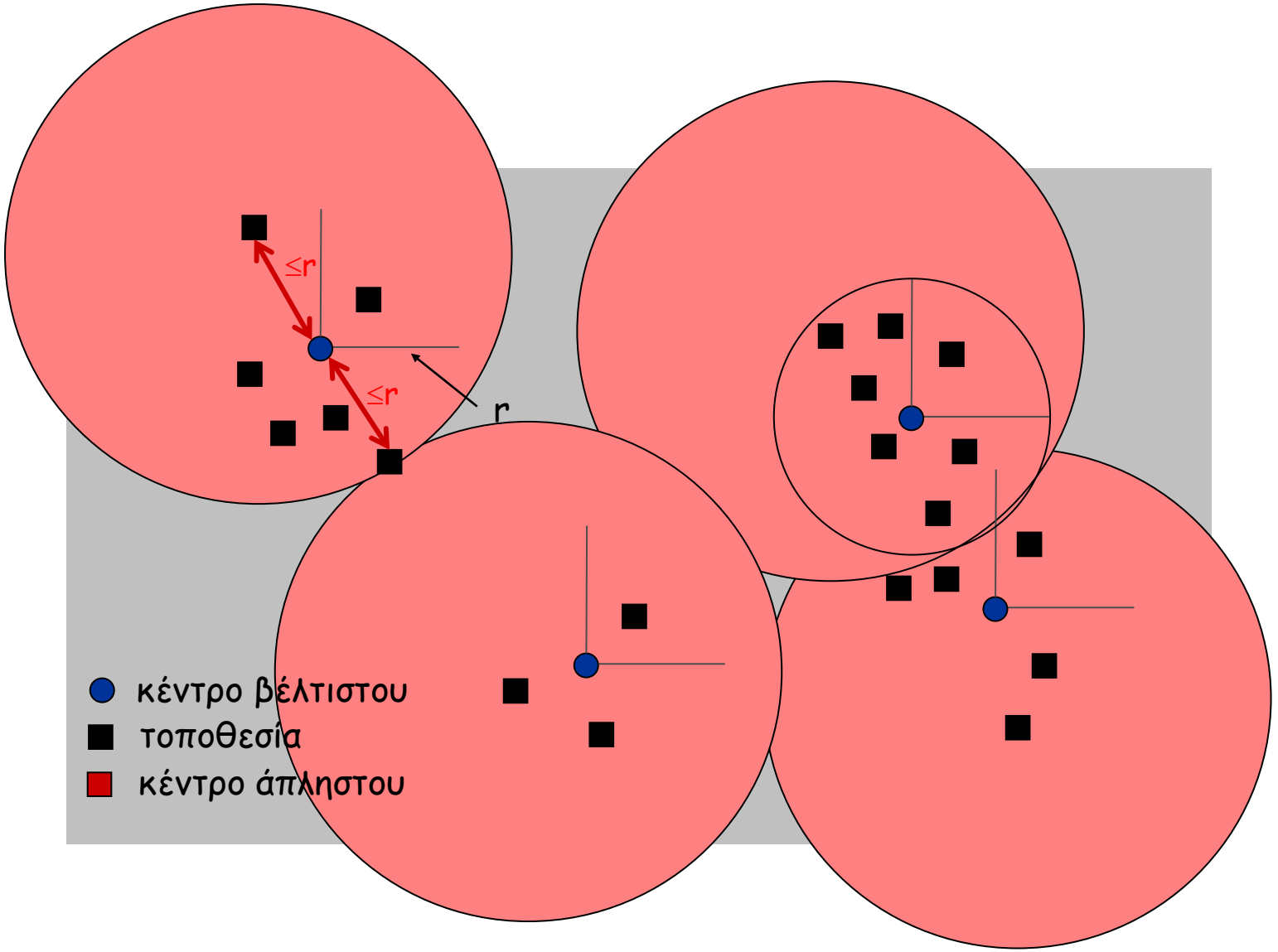
# Άπληστος Αλγόριθμος: Κακή Περίπτωση

Άπληστος Αλγόριθμος: Τοποθέτησε το πρώτο κέντρο στην καλύτερη δυνατή θέση αν είχαμε μόνο ένα κέντρο, και έπειτα θα προσθέτουμε τα κέντρα με στόχο να μειώσουμε την ακτίνα κάλυψης όσο το δυνατό περισσότερο.

Παρατήρηση: πολύ άσχημο!



# Αν Γνωρίζουμε την Ακτίνα r;



# Αν δεν Γνωρίζουμε την Ακτίνα r;

Μία γενική τεχνική σε προβλήματα βελτιστοποίησης:

- Θεώρησε ότι γνωρίζουμε την τιμή όπου επιτυγχάνεται μία βέλτιστη λύση.

Πώς όμως ακυρώνουμε την υπόθεση;

1. (Γενική) Δυαδική Αναζήτηση στο χώρο των τιμών της βέλτιστης λύσης.
2. Λύση που εξαρτάται από το πρόβλημα.

# Επιλογή Κέντρων: Άπληστος Αλγόριθμος

Άπληστος Αλγόριθμος: Επαναλαμβανόμενα επέλεξε το επόμενο κέντρο ως η τοποθεσία που είναι μακρύτερα από κάθε υπάρχον κέντρο.

```
Greedy-Center-Selection(k, n, s1,s2,…,sn) {

   C = ϕ
   repeat k times {
      Select a site si with maximum dist(si, C)
      Add si to C
   }
   return C
}
```

Τοποθεσία μακρύτερα από κάθε κέντρο

Παρατήρηση: Στον τερματισμό όλα τα κέντρα του $C$ είναι ανά δύο τουλάχιστον $r(C)$ μακρυά.

Απόδειξη: Κατασκευή του αλγόριθμου.

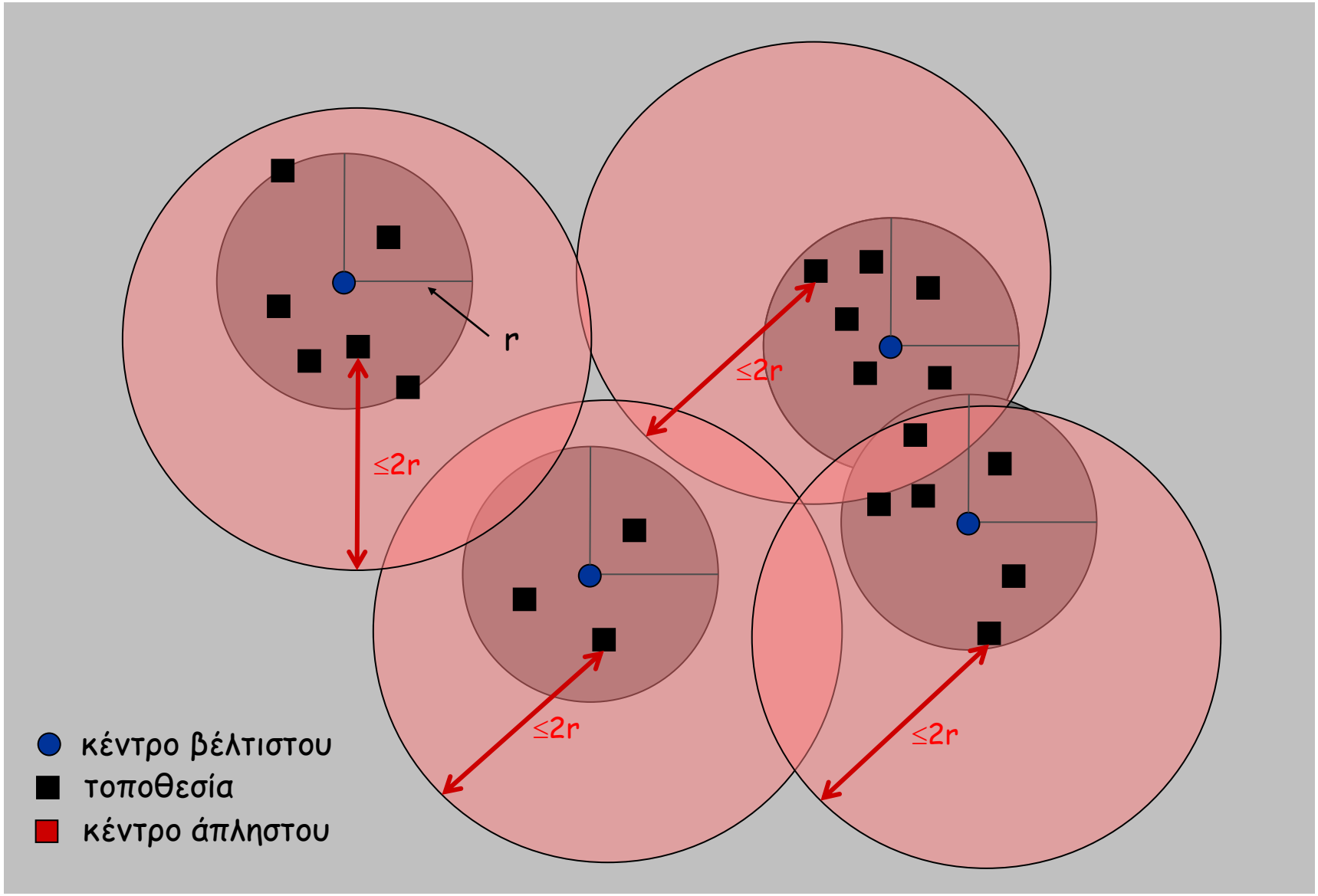
# Επιλογή Κέντρου: Ανάλυση Άπληστου Αλγόριθμου

**Θεώρημα:** Έστω $C^*$ το βέλτιστο σύνολο κέντρων. Τότε: $r(C) \leq 2r(C^*)$.

**Απόδειξη:** (αντίφαση) Έστω $r(C) > 2\, r(C^*)$.

- Έστω $s$ μία τοποθεσία που $dist(s, C) > 2r(C^*)$.
- Κατά τη διάρκεια εκτέλεσης του αλγόριθμου έστω $c'$ το τρέχον κέντρο που επιλέγει ενώ το σύνολο κέντρων μέχρι τώρα είναι $C'$.
- $dist(c', C') \geq dist(s, C') \geq dist(s, C) > 2r(C^*)$
- Άρα ο άπληστος αλγόριθμος είναι μία σωστή υλοποίηση του αλγόριθμου που γνώριζε την ακτίνα $r(C^*)$.
- >k κέντρα για την κάλυψη του συνόλου σημείων με αποτέλεσμα ο προηγούμενος αλγόριθμος να συμπεράνει ότι δεν υπάρχει κάλυψη με k κέντρα ακτίνας r. Άτοπο.▪

# Απόδειξη



# Επιλογή Κέντρων

**Θεώρημα:** Έστω $C^*$ η βέλτιστη λύση των κέντρων. Τότε $r(C) \leq 2r(C^*)$.

**Θεώρημα:** Ο άπληστος αλγόριθμος είναι 2-προσέγγιση του προβλήματος επιλογής κέντρου.

**Παρατήρηση:** Ο άπληστος αλγόριθμος επιλέγει ως κέντρα τις τοποθεσίες αλλά ακόμα και έτσι είναι το πολύ κατά ένα παράγοντα 2 μακριά από το βέλτιστο αλγόριθμο που μπορεί να τοποθετήσει τα κέντρα παντού.

π.χ., σημεία στο επίπεδο

**Ερώτηση:** Υπάρχει ελπίδα για 3/2-προσέγγιση; 4/3;

**Θεώρημα:** Εκτός και P = NP, δεν υπάρχει $\rho$-προσέγγιση για το πρόβλημα επιλογής κέντρων για κάθε $\rho < 2$.

# Επιλογή Κέντρων: Δυσκολία Προσέγγισης

Θεώρημα. Εκτός και αν P = NP, δεν υπάρχει $\rho$-προσέγγιστικός αλγόριθμος για το πρόβλημα επιλογής k κέντρων με μετρική απόσταση για κάθε $\rho < 2$.

Απόδειξη. Θα δείξουμε πως χρησιμοποιείται ένας $(2 - \varepsilon)$-προσεγγιστικός αλγόριθμος για το πρόβλημα της επιλογής κέντρων για λύση του προβλήματος Κυρίαρχου Συνόλου (ΚΣ) (dominating set).

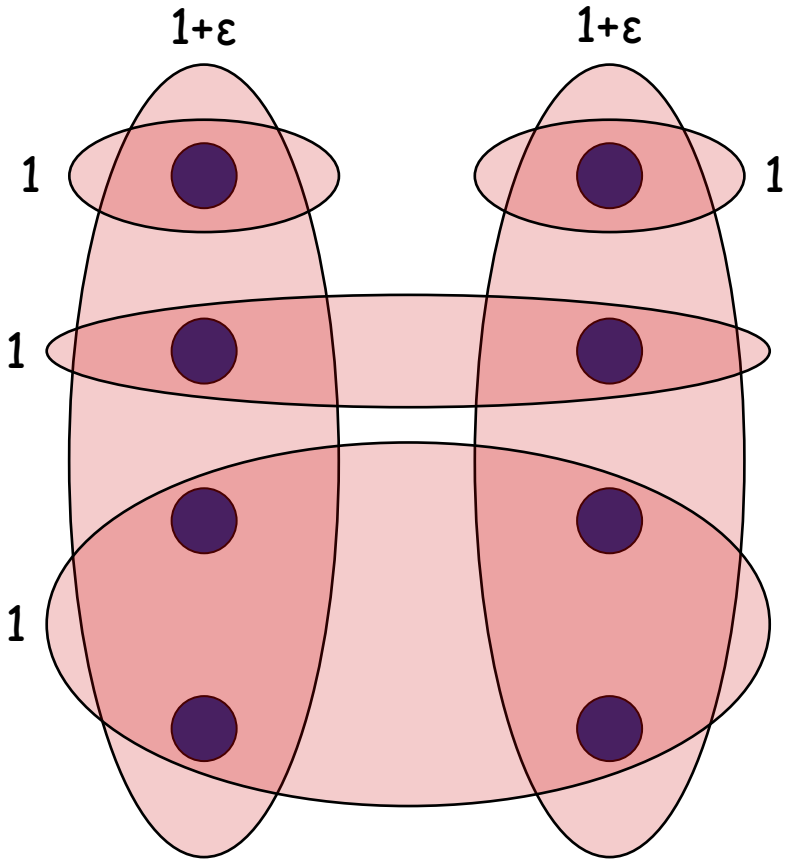
- Έστω $G = (V, E)$, k ένα στιγμιότυπο του ΚΣ.
- Κατασκευάζουμε στιγμιότυπο $G'$ για k κέντρα με τοποθεσίες το V και αποστάσεις:
  - $d(u, v) = 1$ αν $(u, v) \in E$
  - $d(u, v) = 2$ αν $(u, v) \notin E$
- Το $G'$ ικανοποιεί την τριγωνική ανισότητα.
- **Ισχυρισμός:** Το $G$ έχει κυρίαρχο σύνολο μεγέθους k αν και μόνο αν υπάρχουν k κέντρα $C^*$ ώστε $r(C^*) = 1$.
- Επομένως, αν το $G$ έχει κυρίαρχο σύνολο μεγέθους k, ένας $(2 - \varepsilon)$-προσεγγιστικός αλγόριθμος στο $G'$ θα βρει μία λύση $C^*$ με $r(C^*) = 1$ αφού δεν μπορεί να χρησιμοποιήσει ακμή απόστασης 2.

# 11.3 ΚΆΛΥΨΗ ΣΥΝΌΛΟΥ

# Σταθμισμένο Πρόβλημα Κάλυψης Συνόλου

Κάλυψη Συνόλου: Δοθέντος ενός συνόλου U και μίας λίστας υποσυνόλων $S_1, S_2, \ldots, S_m$ του U, ένα κάλυμμα συνόλου του U είναι μία συλλογή αυτών των υποσυνόλων ώστε η ένωσή τους να δίνει U.

Σταθμισμένη Κάλυψη Συνόλου: Κάθε σύνολο $S_i$ έχει μία βαρύτητα $w_i$ και στόχος μας είναι να βρούμε το κάλυμμα συνόλου με το ελάχιστο συνολικό βάρος.



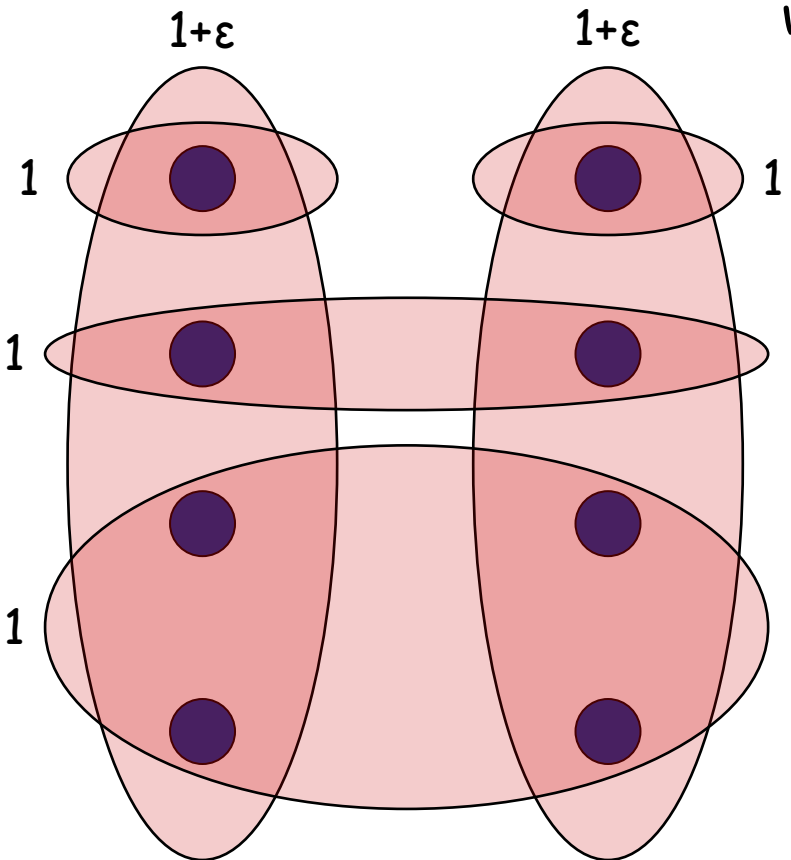
# Άπληστος Αλγόριθμος

Δύο αντικρουόμενα κριτήρια:

1. Μεγάλο πλήθος στοιχείων υποσυνόλου $|S_i|$
2. Μικρό βάρος $w_i$ του υποσυνόλου $S_i$

Έστω R το σύνολο των ακάλυπτων στοιχείων μέχρι κάποια στιγμή. Βάρος ανά καλυπτόμενο στοιχείο:

$w_i / |S_i \cap R|$




Βέλτιστο: 2+2ε

Άπληστος: 4

```
Greedy-Set-Cover:
Start with R = U and no sets selected
While R ≠ ∅
  Select set S_i that minimizes w_i/|S_i ∩ R|
  Delete set S_i from R
EndWhile
Return the selected sets
```

# Ανάλυση

Προσθέτουμε αμέσως μετά την επιλογή του $S_i$ την εξής γραμμή:

$c_s = w_i / |S_i \cap R|$ για όλα τα $s \in S_i \cap R$

Το $c_s$ είναι το κόστος που πληρώθηκε (τιμή) για την κάλυψη κάθε νέου στοιχείου. Έστω *C* η κάλυψη συνόλου του άπληστου αλγόριθμου.

Προφάνως:

$$\sum_{S_i \in C} w_i = \sum_{s \in U} c_s$$

# Ανάλυση

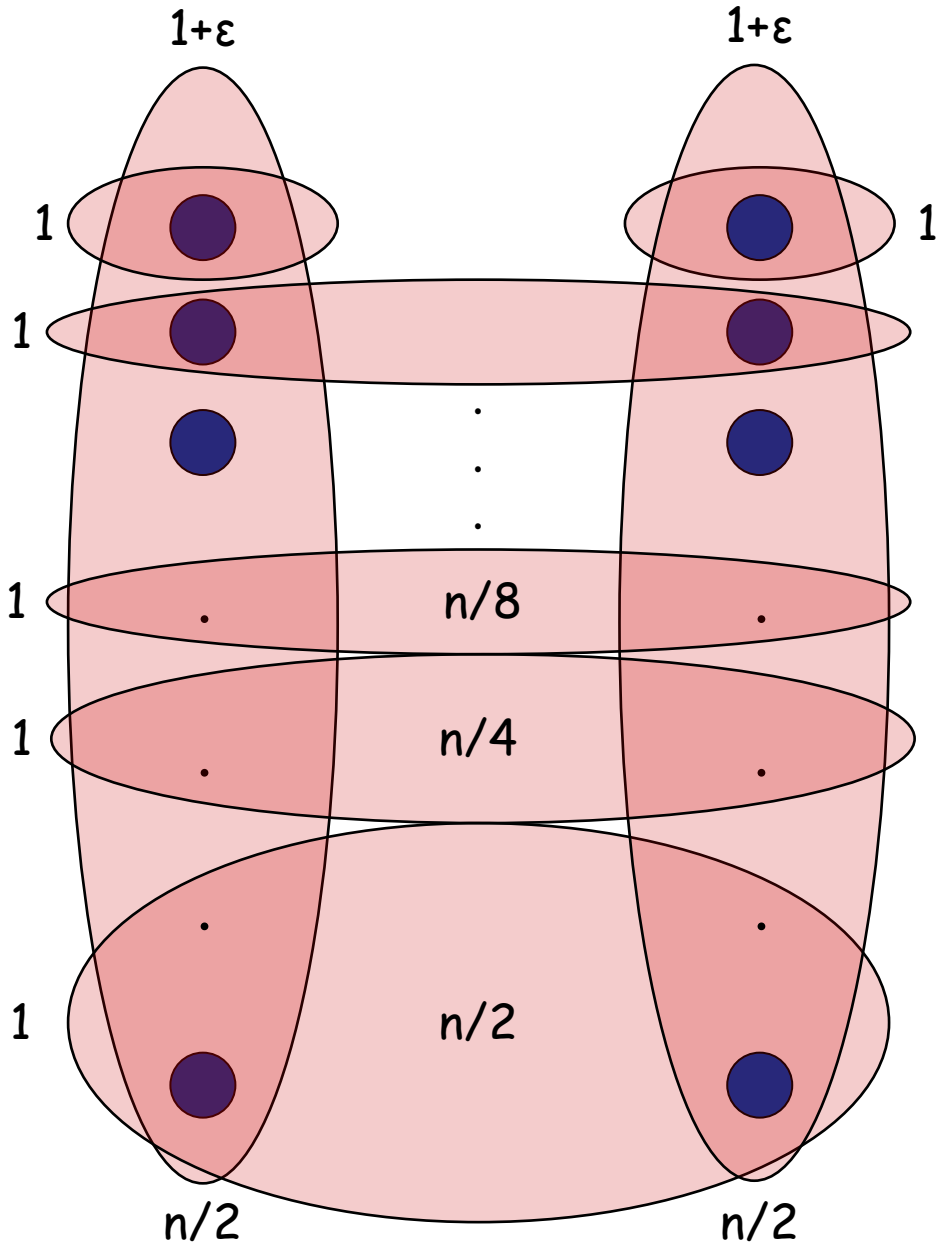
Για κάθε σύνολο $S_k$: $\sum_{s \in S_k} c_s \leq H(|S_k|) \cdot w_k$

Έστω $d^* = \max_i |S_i|$

Η κάλυψη συνόλου C που επιλέγεται από τον άπληστο αλγόριθμο έχει βαρύτητα το πολύ $H(d^*)$ επί το βέλτιστο βάρος $w^*$.

Αν υπήρχε καλύτερη προσέγγιση από αυτή τότε θα ίσχυε P = NP.

# Είναι Αυστηρό Όριο;

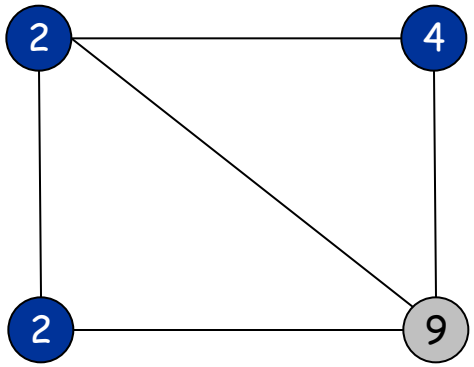


# 11.4 Η ΜΈΘΟΔΟΣ ΤΗΣ ΤΙΜΟΛΌΓΗΣΗΣ: ΚΆΛΥΨΗ ΚΟΡΥΦΏΝ

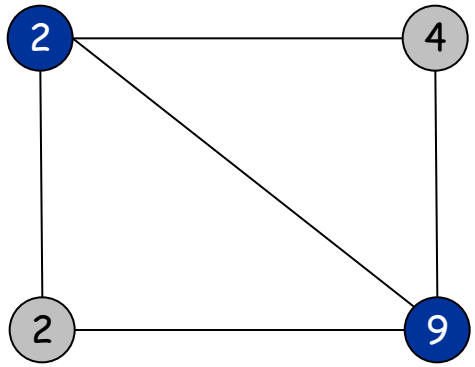
# Σταθμισμένο Πρόβλημα Κάλυψης Κορυφών

Κάλυψη Κορυφών: Δοθέντος ενός γραφήματος $G = (V, E)$, ένα κάλυμμα κορυφών είναι ένα σύνολο $S \subseteq V$ έτσι ώστε κάθε ακμή στο $E$ έχει τουλάχιστον ένα άκρο στο $S$.

Σταθμισμένη Κάλυψη Κορυφών: Δοθέντος $G$ με βάρη στις κορυφές, βρες ένα κάλυμμα κορυφών ελάχιστου συνολικού βάρους.




βάρος = 2 + 2 + 4




βάρος = 11

# Αναγωγή σε Κάλυψη Συνόλου

Μπορούμε να χρησιμοποιήσουμε τον προσεγγιστικό αλγόριθμο του Προβλήματος Κάλυψης Συνόλου για να πάρουμε έναν προσεγγιστικό αλγόριθμο με συντελεστή προσέγγισης $H(d)$ για το σταθμισμένο πρόβλημα της Κάλυψης Κορυφών, όπου $d$ είναι ο μέγιστος βαθμός του γραφήματος.

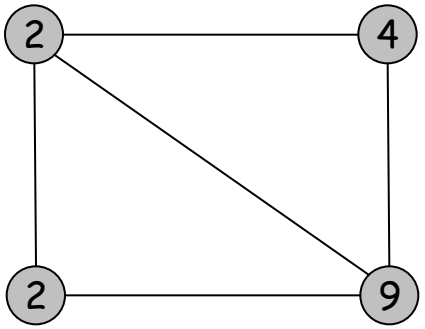
Απόδειξη: Χρήση της αναγωγής για NP-πληρότητα.

# Μέθοδος Τιμολόγησης

Κάθε ακμή πρέπει να καλυφθεί από μία κορυφή.
Η ακμή e = (i, j) πληρώνει ένα κόστος $p_e \geq 0$ για τη χρήση των κορυφών i και j .

Δικαιοσύνη: Οι προσκείμενες ακμές στην κορυφή i πρέπει να πληρώσουν $\leq w_i$ συνολικά.

$$\text{για κάθε κορυφή } i : \sum_{e=(i,j)} p_e \leq w_i$$




Λήμμα: Για κάθε κάλυμμα κορυφής S και κάθε δίκαιη τιμή $p_e$: $\sum_e p_e \leq w(S)$.

Απόδειξη:
$$\sum_{e \in E} p_e \leq \sum_{i \in S} \sum_{e=(i,j)} p_e \leq \sum_{i \in S} w_i = w(S).$$ ▪

κάθε ακμή e που καλύπτεται από τουλάχιστον ένα κόμβο στο S

αθροίζει τα δίκαια κόστη για κάθε κόμβο στο S

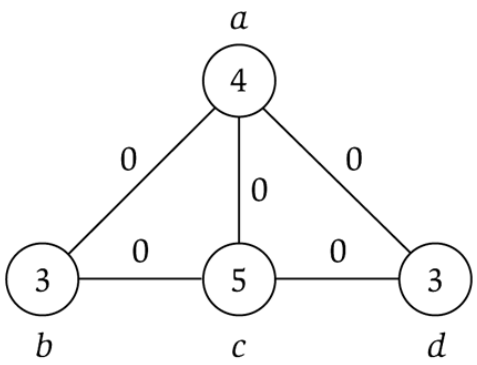
# Μέθοδος Τιμολόγησης

Θέτουμε τα κόστη στις ακμές και ταυτόχρονα βρίσκουμε το κάλυμμα κορυφών.

```
Weighted-Vertex-Cover-Approx(G, w) {
   foreach e in E
      p_e = 0

   while (∃ edge i-j such that neither i nor j are tight)
      select such an edge e
      increase p_e as much as possible until i or j tight
   }

   S ← set of all tight nodes
   return S
}
```

$$\sum_{e=(i,j)} p_e = w_i$$

# Μέθοδος Τιμολόγησης




**(a)**

# Μέθοδος Τιμολόγησης: Ανάλυση

Η μέθοδος τιμολόγησης είναι μία 2-προσέγγιση.

Απόδειξη:

- Ο αλγόριθμος τερματίζει αφού μετά από κάθε επανάληψη μία τουλάχιστον κορυφή γίνεται ερμητική.

- Έστω S = σύνολο όλων των ερμητικών κόμβων μετά τον τερματισμό του αλγορίθμου. Το S είναι ένα κάλυμμα κορυφών: αν μία ακμή (i,j) δεν είναι καλυμμένη τότε ούτε ο i ούτε ο j δεν είναι ερμητικοί. Τότε όμως η επανάληψη δεν θα τερματιζόταν.

- Έστω S* το βέλτιστο κάλυμμα κορυφών. Θ.δ.ο. $w(S) \leq 2w(S^*)$.

$$w(S) = \sum_{i \in S} w_i = \sum_{i \in S} \sum_{e=(i,j)} p_e \leq \sum_{i \in V} \sum_{e=(i,j)} p_e = 2 \sum_{e \in E} p_e \leq 2w(S^*). \quad \blacksquare$$

- $\sum_{i \in S} w_i$: όλοι οι κόμβοι του S είναι ερμητικοί
- $\leq$: $S \subseteq V$, κόστη $\geq 0$
- $= 2 \sum_{e \in E} p_e$: κάθε ακμή μετριέται δύο φορές
- $\leq 2w(S^*)$: λήμμα δικαιοσύνης

# 11.6 Γραμμικός Προγραμματισμός και Στρογγυλοποίηση: Κάλυψη Κορυφών

- Ακέραιος Προγραμματισμός
- Γραμμικός Προγραμματισμός

# Γραμμικός Προγραμματισμός

Ελαχιστοποίηση/Μεγιστοποίηση γραμμικής αντικειμενικής συνάρτησης υποκείμενη σε γραμμικές ανισότητες.

- Είσοδος: ακέραιοι $c_j$, $b_i$, $a_{ij}$ .
- Έξοδος: πραγματικοί αριθμοί $x_j$.

$$\begin{aligned} (\mathrm{P}) \quad \max \quad & c^t x & & \\ & Ax & \geq & b \\ & x & \geq & 0 \end{aligned}$$

$$\begin{aligned} (\mathrm{P}) \quad \max \quad & \sum_{j=1}^{n} c_j x_j & & & \\ & \sum_{j=1}^{n} a_{ij} x_j & \geq & b_i & 1 \leq i \leq m \\ & x_j & \geq & 0 & 1 \leq j \leq n \end{aligned}$$

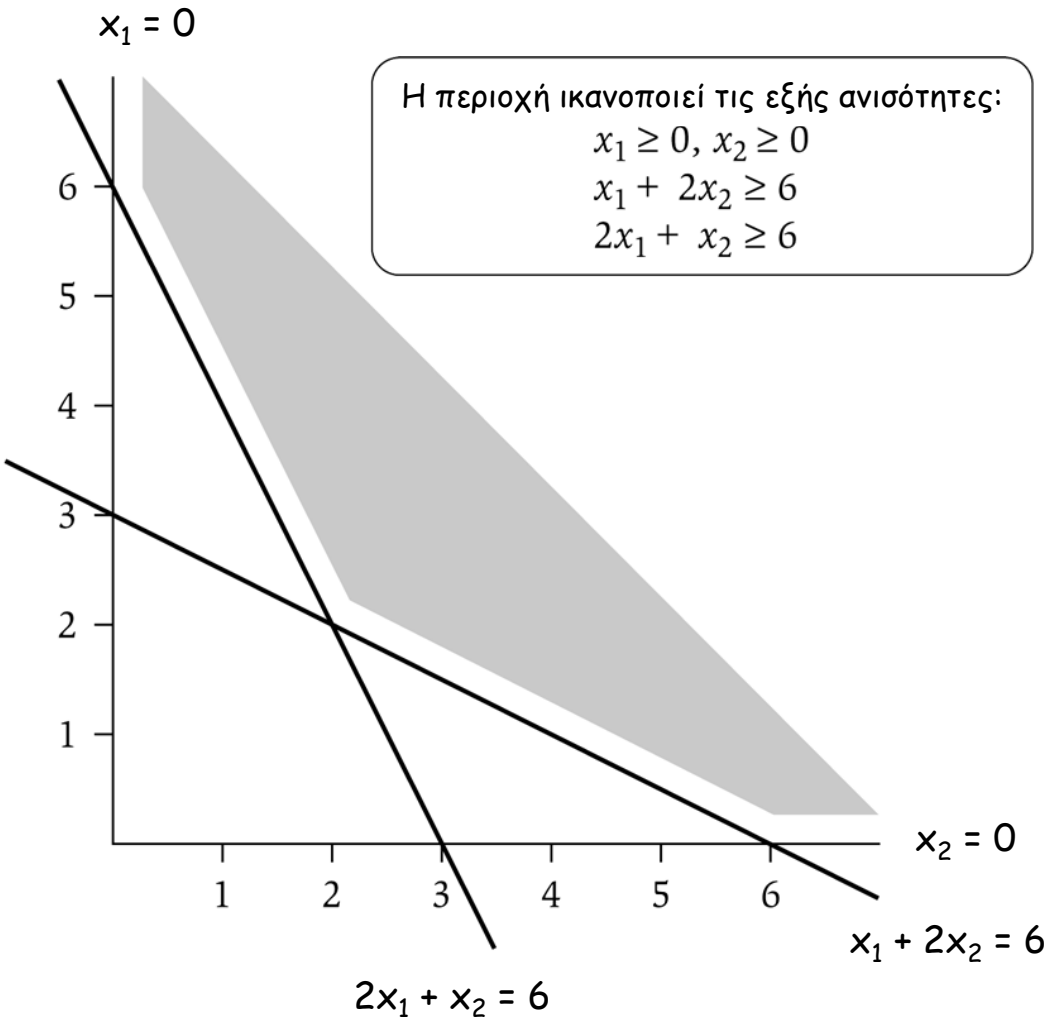
Γραμμικό. Όχι $x^2$, $xy$, $\arccos(x)$, $x(1-x)$, κτλ.

Αλγόριθμος Simplex. [Dantzig 1947] Λύνει το LP στην πράξη.

Αλγόριθμος Ελλειψοειδούς. [Khachian 1979] Λύνει το LP σε πολυωνυμικό χρόνο.

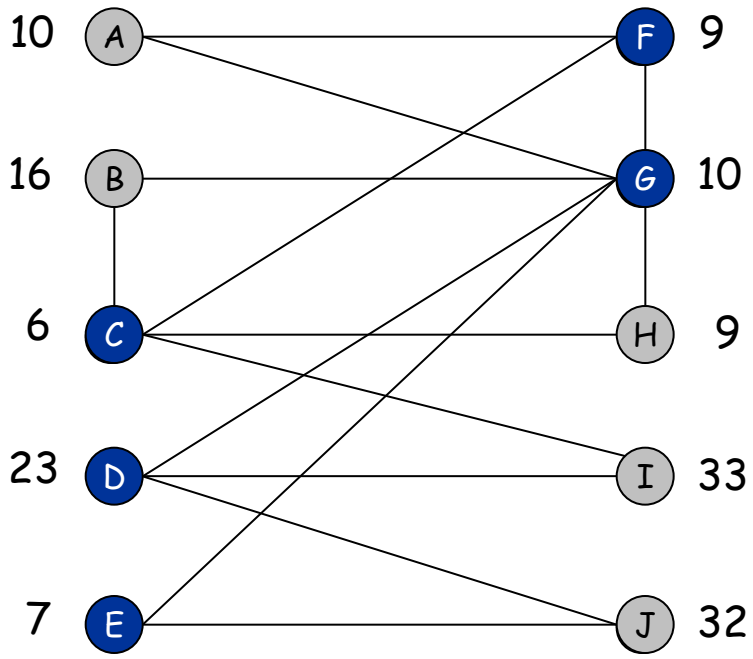
# Εφικτή Περιοχή ενός Γραμμικού Προγράμματος

Η γεωμετρία του LP σε 2-διαστάσεις.



# Σταθμισμένο Πρόβλημα Κάλυψης Κορυφών

Δοθέντος ενός μη κατευθυνόμενου γραφήματος $G = (V, E)$ με βαρύτητα κορυφών $w_i \geq 0$, βρείτε ένα ελάχιστης βαρύτητας υποσύνολο των κορυφών $S$ έτσι ώστε κάθε ακμή να πρόσκειται σε τουλάχιστον μία κορυφή του $S$.




Συνολική βαρύτητα = 55

# Σταθμισμένο Πρόβλημα Κάλυψης Κορυφών: Ακέραιο Πρόγραμμα (IP)

Μοντελοποίηση με Ακέραιο Προγραμματισμό:

- Μοντελοποιούμε τη χρήση κάθε κορυφής i με μία 0/1 μεταβλητή $x_i$.

$$x_i = \begin{cases} 0 & i \notin S \\ 1 & i \in S \end{cases}$$

$S \rightarrow$ Σύνολο κάλυψης κορυφών

Το S είναι σε 1-1 αντιστοιχία με τις 0/1 αναθέσεις:

$S = \{i \in V : x_i = 1\}$

- Αντικειμενική συνάρτηση: ελαχιστοποίηση $\Sigma_{i \in V} w_i x_i$

- Θα πρέπει να επιλέγουμε το άκρο κάθε ακμής i ή j: $x_i + x_j \geq 1$

# Ακέραιος Προγραμματισμός

Δοθέντων ακεραίων $a_{ij}$ και $b_i$, βρείτε ακέραιους $x_j$ έτσι ώστε:

$$\begin{array}{lll} \max & w^t x & \\ & Ax \geq & b \\ & x & \text{ακέραιοι} \end{array}$$

$$\begin{array}{lll} \max & \sum_{i=1}^{n} w_i x_i & \\ \sum_{j=1}^{n} a_{ij} x_j & \geq & b_i \qquad 1 \leq i \leq m \\ x_j & \geq & 0 \qquad 1 \leq j \leq n \\ x_j & & \text{ακέραιος} \quad 1 \leq j \leq n \end{array}$$

Παρατήρηση. Η μοντελοποίηση του καλύμματος κορυφών αποδεικνύει ότι ο Ακέραιος Προγραμματισμός (Integer Programming – IP) είναι NP-δύσκολος.

Ακόμα και όταν όλοι οι συντελεστές είναι 0/1 και το πολύ δύο μεταβλητές ανά ανισότητα.

# Σταθμισμένο Πρόβλημα Κάλυψης Κορυφών: Ακέραιο Πρόγραμμα (IP)

$$
\begin{aligned}
(ILP)\ \min \quad & \sum_{i \in V} w_i x_i & & \\
& x_i + x_j \quad \geq \quad 1 & & (i,j) \in E \\
& x_i \quad \in \quad \{0,1\} & & i \in V
\end{aligned}
$$

Παρατήρηση: Αν x* είναι μία βέλτιστη λύση στο (ILP), τότε το S = {i ∈ V : $x^*_i$ = 1} είναι μίας ελάχιστης βαρύτητας κάλυμμα κορυφών.

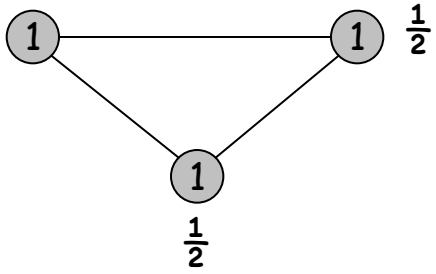
# LP Χαλάρωση

Μοντελοποίηση με Γραμμικό Προγραμματισμό.

$$
\begin{aligned}
(LP)\ \min \quad & \sum_{i \in V} w_i x_i & & \\
& x_i + x_j \quad \geq \quad 1 \quad & (i,j) \in E \\
& x_i \quad \geq \quad 0 \quad & i \in V
\end{aligned}
$$

Παρατήρηση. Η βέλτιστη λύση του (LP) είναι ≤ βέλτιστη λύση (ILP).
Απόδειξη: Το LP έχει λιγότερους περιορισμούς.

Σημείωση: Το LP δεν είναι ισοδύναμο του καλύμματος κορυφών.




Πως η λύση του LP βοηθά στην εύρεση μικρού Κ.Κ.;
Λύσε το LP και στρογγυλοποίησε τις ρητές τιμές.

# Σταθμισμένο Πρόβλημα Κάλυψης Κορυφών

**Θεώρημα:** Αν $x^*$ είναι η βέλτιστη λύση του (LP), τότε $S = \{i \in V : x^*_i \geq \frac{1}{2}\}$ είναι ένα κάλυμμα κορυφών με βαρύτητα το πολύ διπλάσια από την ελάχιστη βαρύτητα.

**(1) Απόδειξη:** [το S είναι ένα κάλυμμα κορυφών]

- Έστω μία ακμή $(i, j) \in E$.
- Αφού $x^*_i + x^*_j \geq 1$, είτε $x^*_i \geq \frac{1}{2}$ ή $x^*_j \geq \frac{1}{2}$ $\Rightarrow$ $(i, j)$ καλύπτεται.

**(2) Απόδειξη:** [Το S έχει την επιθυμητή βαρύτητα]

- Έστω $S^*$ το βέλτιστο κάλυμμα κορυφών. Τότε:

$$\sum_{i \in S^*} w_i \geq \sum_{i \in S} w_i x_i^* \geq \frac{1}{2} \sum_{i \in S} w_i$$

Το LP είναι χαλάρωση $\qquad$ $x^*_i \geq \frac{1}{2}$

# Σταθμισμένο Πρόβλημα Κάλυψης Κορυφών

Θεώρημα. Ένας 2-προσεγγιστικός αλγόριθμος για το σταθμισμένο πρόβλημα κάλυψης κορυφών.

Θεώρημα. [Dinur-Safra 2001] Αν P ≠ NP, τότε δεν υπάρχει ρ-προσέγγιση για ρ < 1.3607, ακόμα και με μοναδιαία βάρη.

$10\sqrt{5} - 21$

Ανοικτό Πρόβλημα Έρευνας: Κλείσιμο του χάσματος.

# 11.8 Το Πρόβλημα του Σακιδίου

# Σχήμα Προσέγγισης Πολυωνυμικού Χρόνου

PTAS (Polynomial Time Approximation Scheme):

$(1 + \varepsilon)$-προσεγγιστικός αλγόριθμος για κάθε σταθερά $\varepsilon > 0$.

- Εξισορρόπηση Φορτίου. [Hochbaum-Shmoys 1987]
- Ευκλείδιο TSP. [Arora 1996]

Συνέπεια: Οι PTAS παράγουν αυθαίρετα υψηλής ποιότητας λύσεις, δίνοντας ένα trade-off μεταξύ ακρίβειας και χρόνου.

Τι θα κάνουμε: PTAS για το πρόβλημα του σακιδίου μέσω στρογγυλοποίησης και κλιμάκωσης.

# Το Πρόβλημα του Σακιδίου

- Δοθέντων n αντικειμένων και ενός σακιδίου.
- Το αντικείμενο i έχει τιμή $v_i > 0$ και βάρος $w_i > 0$.
- Το σακίδιο μεταφέρει βάρος μέχρι W (έστω $w_i \leq W$).
- Στόχος: γέμισμα σακιδίου για μεγιστοποίηση συνολικής τιμής.

Παράδειγμα: Το {3, 4} έχει τιμή 40.

W = 11

| Αντικείμενο | Τιμή | Βάρος |
|---|---|---|
| 1 | 1 | 1 |
| 2 | 6 | 2 |
| 3 | 18 | 5 |
| 4 | 22 | 6 |
| 5 | 28 | 7 |

# Πρόβλημα Σακιδίου: Δυναμικός Προγραμματισμός 1

OPT(i, w) = μέγιστης τιμής υποσύνολο των αντικειμένων 1,..., i με όριο βάρους w.

- Περίπτωση 1: ο OPT δεν επιλέγει το αντικείμενο i.
  - Ο OPT επιλέγει το καλύτερο από 1, ..., i–1 με όριο βάρους w
- Περίπτωση 2: ο OPT επιλέγει το αντικείμενο i.
  - Νέο όριο βάρους = $w - w_i$
  - Ο OPT επιλέγει το καλύτερο από 1, ..., i–1 με όριο βάρους $w - w_i$

$$OPT(i,w) = \begin{cases} 0 & i = 0 \\ OPT(i-1,w) & w_i > w \\ \max\{OPT(i-1,w), v_i + OPT(i-1,w-w_i)\} & \text{διαφορετικά} \end{cases}$$

Χρόνος Εκτέλεσης. O(n W).

- W = βάρος σακιδίου.
- Δεν είναι πολυωνυμικό ως προς το μέγεθος εισόδου!

# Πρόβλημα Σακιδίου: Δυναμικός Προγραμματισμός 2

OPT(i, v) = ελάχιστου βάρους υποσύνολο αντικειμένων 1, …, i με τιμή ίση με v.

- Περίπτωση 1: Ο OPT δεν επιλέγει το αντικείμενο i.
  - Ο OPT επιλέγει την καλύτερη από 1, …, i-1 με τιμή ακριβώς v
- Περίπτωση 2: Ο OPT επιλέγει το αντικείμενο i.
  - Χρησιμοποιείται βάρος $w_i$, νέα τιμή ίση με $v - v_i$
  - Ο OPT επιλέγει την καλύτερη από 1, …, i-1 με τιμή ακριβώς $v - v_i$

$$OPT(i,v) = \begin{cases} 0 & v = 0 \\ \infty & i = 0,\ v > 0 \\ OPT(i-1,v) & v_i > v \\ \min\{OPT(i-1,v), w_i + OPT(i-1, v-v_i)\} & \text{διαφορετικά} \end{cases}$$

$V^* \le n\, v_{max}$

Χρόνος Εκτέλεσης. $O(n\, V^*) = O(n^2 v_{max})$.

- $V^*$ = βέλτιστη τιμή = μέγιστο v έτσι ώστε $OPT(n, v) \le W$.
- Δεν είναι πολυωνυμικό στο μέγεθος της εισόδου!

# PTAS

Διαίσθηση για τον προσεγγιστικό αλγόριθμο:

- Στρογγυλοποίησε τις τιμές ώστε να ανήκουν σε μικρότερο διάστημα.
- Δυναμικός προγραμματισμός σε στρογγυλοποιημένο στιγμιότυπο.
- Επέστρεψε βέλτιστα αντικείμενα στο στρογγυλοποιημένο στιγμιότυπο.

| Item | Value | Weight |
|---|---|---|
| 1 | 934,221 | 1 |
| 2 | 5,956,342 | 2 |
| 3 | 17,810,013 | 5 |
| 4 | 21,217,800 | 6 |
| 5 | 27,343,199 | 7 |

W = 11

Αρχικό στιγμιότυπο

| Item | Value | Weight |
|---|---|---|
| 1 | 1 | 1 |
| 2 | 6 | 2 |
| 3 | 18 | 5 |
| 4 | 22 | 6 |
| 5 | 28 | 7 |

W = 11

Στρογγυλοποιημένο στιγμιότυπο.

# PTAS

Στρογγυλοποίηση τιμών $\bar{v}_i = \left\lceil \frac{v_i}{\theta} \right\rceil \theta, \hat{v}_i = \left\lceil \frac{v_i}{\theta} \right\rceil$

- $v_{max}$ = μέγιστη τιμή στο αρχικό στιγμιότυπο
- $\varepsilon$ = παράμετρος ακρίβειας
- $\theta$ = παράγοντας κλιμάκωσης = $\varepsilon\, v_{max} / n$

Παρατήρηση: Οι βέλτιστες λύσεις με $\bar{V}$ ή $\hat{V}$ είναι ισοδύναμες.

Διαίσθηση. Το $\bar{V}$ είναι κοντά στο v και άρα η βέλτιστη λύση που χρησιμοποιεί το $\hat{V}$ είναι σχεδόν βέλτιστη.

Μικρές τιμές και ακέραιες ώστε ο δυναμικός προγραμματισμός να είναι γρήγορος.

Χρόνος εκτέλεσης. $O(n^3 / \varepsilon)$.

- Για το δυναμικό πρόγραμμα ΙΙ ο χρόνος εκτέλεσης είναι $O(n^2 \hat{v}_{max})$, όπου $\hat{v}_{max} = \left\lceil \frac{v_{max}}{\theta} \right\rceil = \left\lceil \frac{n}{\varepsilon} \right\rceil$

# PTAS

Στρογγυλοποίηση των τιμών σε $\bar{v}_i = \left\lceil \frac{v_i}{\theta} \right\rceil \theta$

**Θεώρημα.** Αν το S είναι η λύση του αλγόριθμού μας και S* είναι μία αυθαίρετη και έγκυρη (ως προς το βάρος) λύση τότε: $(1+\varepsilon)\sum_{i \in S} v_i \geq \sum_{i \in S^*} v_i$

**Απόδειξη.** Έστω S* μία αυθαίρετη λύση που ικανοποιεί τα όρια βάρους.

$$\sum_{i \in S^*} v_i \leq \sum_{i \in S^*} \bar{v}_i$$ στρογγυλοποίηση

$$\leq \sum_{i \in S} \bar{v}_i$$ Επίλυση νέου στιγμιοτύπου βέλτιστα

$$\leq \sum_{i \in S} (v_i + \theta)$$ Δεν στρογγυλοποιούμε παραπάνω από θ

$$\leq \sum_{i \in S} v_i + n\theta$$ $|S| \leq n$

$$\leq (1+\varepsilon) \sum_{i \in S} v_i$$ $n\theta = \varepsilon v_{max}$, $v_{max} \leq \Sigma_{i \in S} v_i$