Αλγόριθμος DNS (Pekel, Nassimi, Sahni [1981])

για τον πολλαπλασιασμό δύο $n \times n$ πινάκων

Αριθμός επεξεργαστών: $N = n^3 = 2^{3q}$

Δίκτυο ενδοεπικοινωνίας: Υπερκύβος

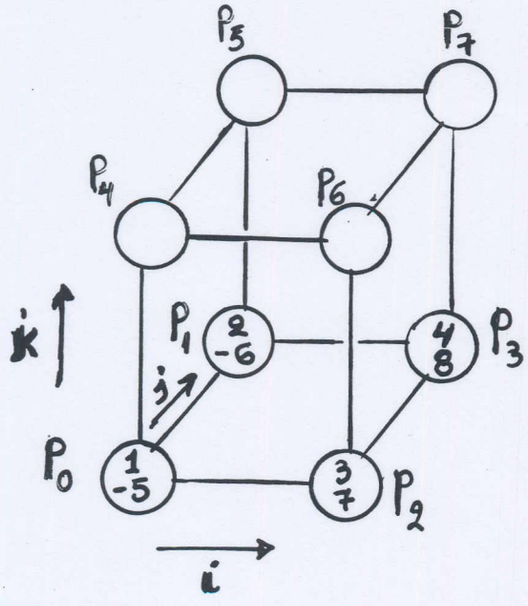
Λογικό 3-D array $\xrightarrow{\text{απεικόνιση}}$ Υπερκύβος 3-D

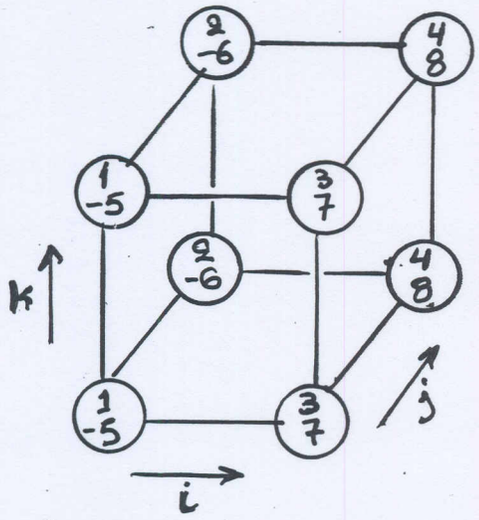
Σχηματική Λειτουργία του αλγορίθμου DNS

Παράδειγμα : $A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$, $B = \begin{bmatrix} -5 & -6 \\ 7 & 8 \end{bmatrix}$ $n=2$, $N=2^3$, $q=1$

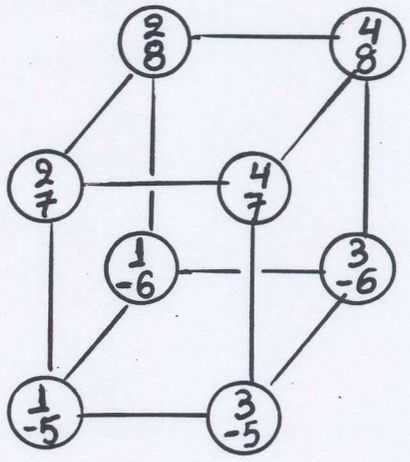
- Αρχική καταχώρηση των A και B

Επεξεργαστής	στοιχείο του A	στοιχείο του B
P_0	$A(0) = a_{11} = 1$	$B(0) = b_{11} = -5$
P_1	$A(1) = a_{12} = 2$	$B(1) = b_{12} = -6$
P_2	$A(2) = a_{21} = 3$	$B(2) = b_{21} = 7$
P_3	$A(3) = a_{22} = 4$	$B(3) = b_{22} = 8$
P_4		
P_5		
P_6		
P_7		



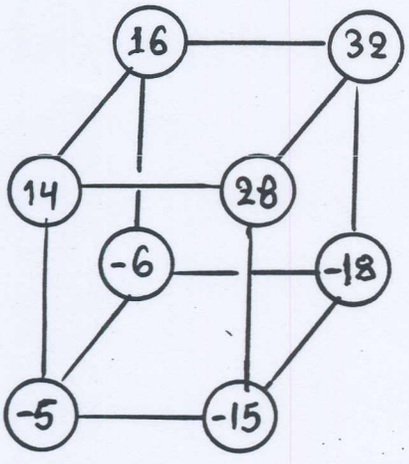


α) Αρχική κατανομή των A και B



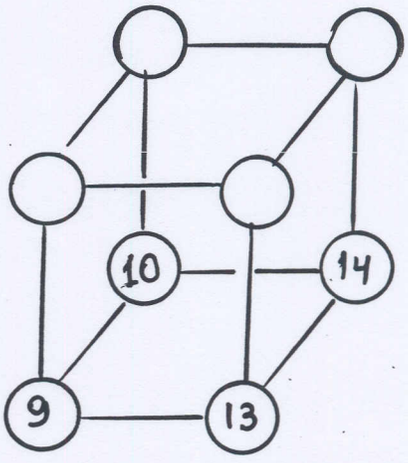
b) Μετακίνηση των A[i,j] από $P_{i,j,0} \rightarrow P_{i,j,j}$

Μετακίνηση των B[i,j] από $P_{i,j,0} \rightarrow P_{i,j,i}$



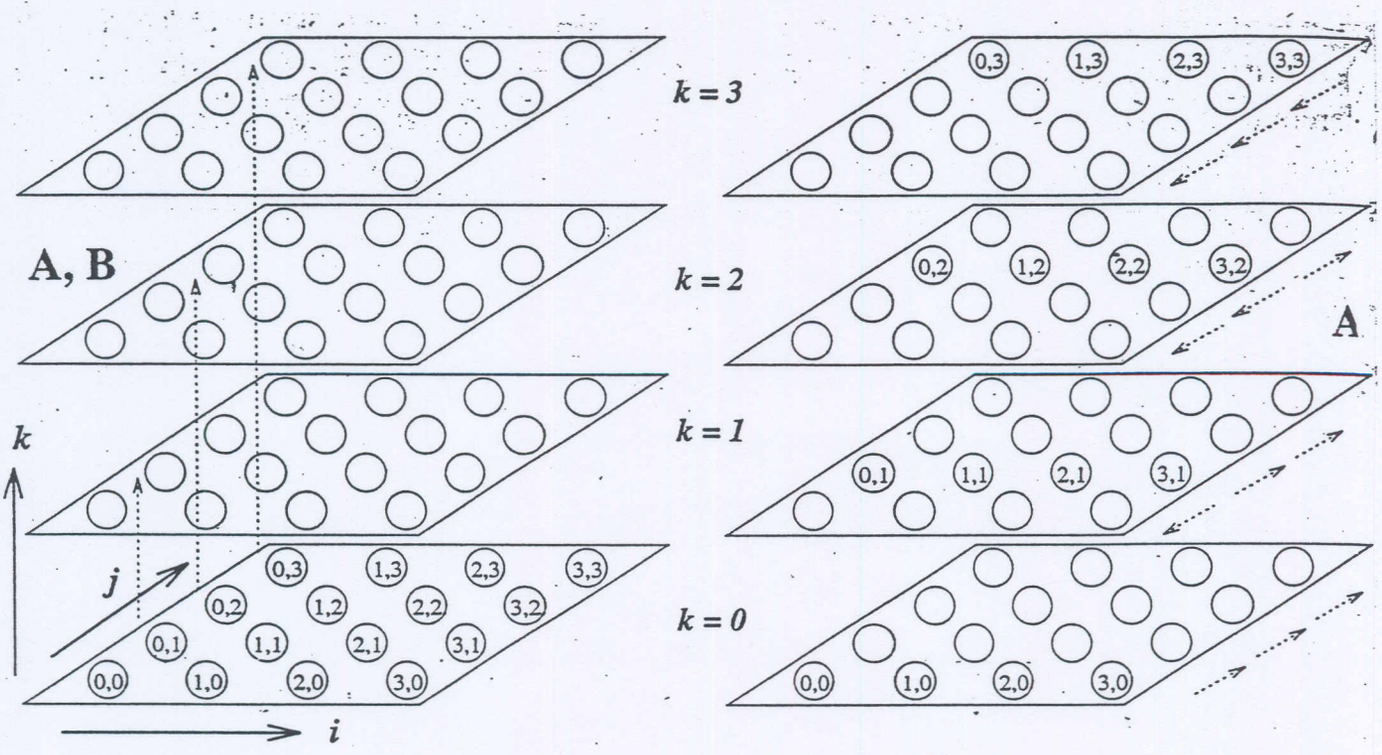
b) Υπολογισμός των γινομένων

$C(i,j,k) = A[i,j,k) * B[i,j,k]$



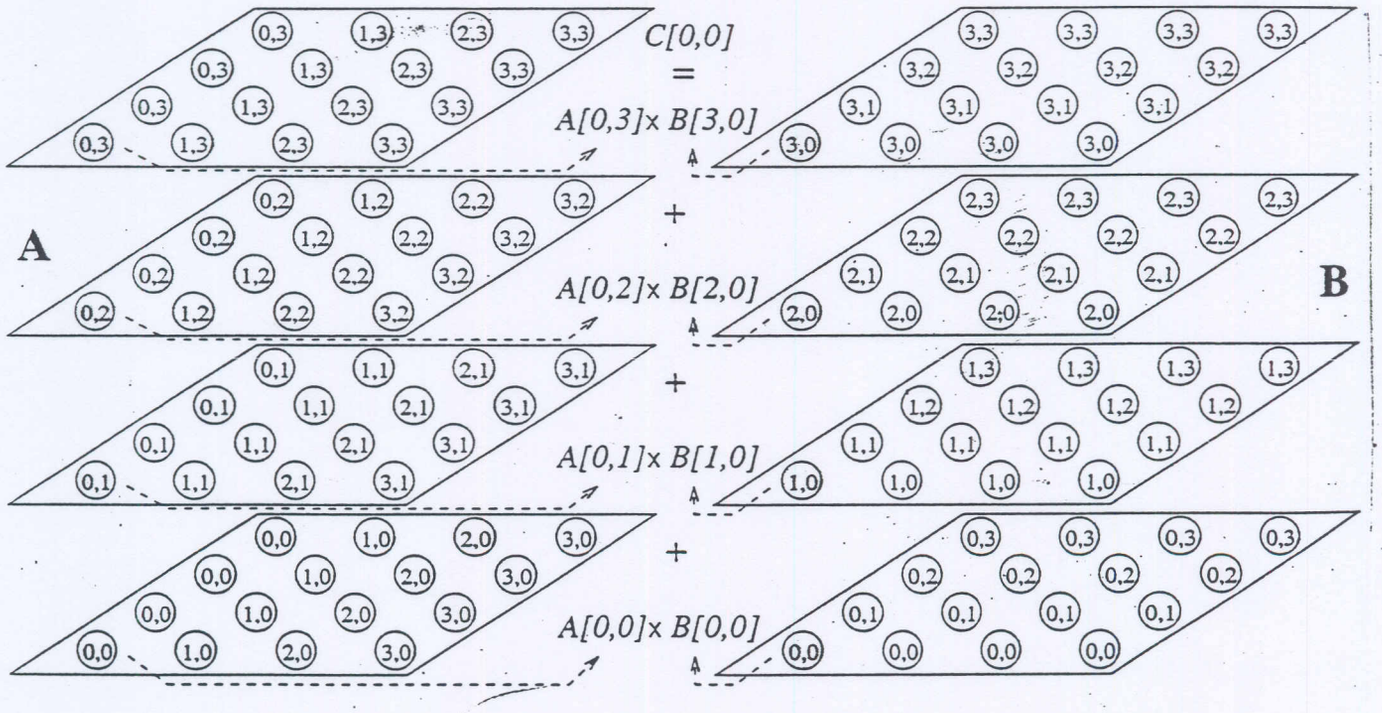
c) Υπολογισμός των αθροισμάτων

$$C(i,j,0) = \sum_{k} C(i,j,k)$$



(α) Αρχική κατανομή των πινάκων A και B

(β) Μετακίνηση των στοιχείων A[i, j] από τον $P_{i,j,0}$ στον $P_{i,j,j}$



(γ) Ολα τα A[i, j] εκπέμπονται κατά μήκος του άξονα j

(δ) Ολα τα B[i, j] εκπέμπονται κατά μήκος του άξονα i

Σχήμα 9 Βήματα Επικοινωνίας στον αλγόριθμο DNS για τον πολ/σμό δύο 4×4 πινάκων A και B με N=64 επεξεργαστές.

Παράλληλοι Αλγόριθμοι κατανεμημένης μνήμης

Πολλαπλασιασμού Πινάκων

$$\underset{m\times n}{A} \quad , \quad \underset{n\times k}{B} \longrightarrow \underset{m\times k}{C = AB}$$

$$c_{ij} = \sum_{s=1}^{n} a_{is} b_{sj} , \quad 1 \le i \le m, \quad 1 \le j \le k$$

Κόστος κλασικού ακολουθιακού αλγορίθμου

$$O(mnk)$$

ή $O(n^3)$ αν $m \le n$ και $k \le n$

Ακολουθιακός αλγόριθμος πολ/μού πινάκων

```
procedure MATRIX MULTIPLICATION (A, B, C)
    for i=1 to m do
        for j=1 to k do
            (1) c_ij ← 0
            (2) for s=1 to n do
                    c_ij ← c_ij + a_is b_sj
                end for
        end for
    end for
```

Γενικότερα: Κόστος $O(n^x)$, $2 \le x \le 3$

π.χ. Αλγόριθμος του Strassen : $O(n^{2.81})$

Block αλγόριθμος πολ/σμού πινάκων

```
procedure BLOCK_MAT_MULT(A, B, C)
    for i=0 to q-1 do
        for j=0 to q-1 do
            C_ij = 0
            for s=0 to q-1 do
                C_ij = C_ij + A_is * B_sj
            end for
        end for
    end for
```

Στις block μεθόδους, κάθε πίνακας A nxn θεωρείται ότι είναι ένας qxq πίνακας με στοιχεία blocks A_{ij} $(i,j = 0,1,\ldots,q-1)$ όπου κάθε block είναι ένας $\frac{n}{q} \times \frac{n}{q}$ υποπίνακας

Παράδειγμα: $n=4$, $q=2$

$$A = \left[\begin{array}{cc|cc} \alpha_{11} & \alpha_{12} & \alpha_{13} & \alpha_{14} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} & \alpha_{24} \\ \hline \alpha_{31} & \alpha_{32} & \alpha_{33} & \alpha_{34} \\ \alpha_{41} & \alpha_{42} & \alpha_{43} & \alpha_{44} \end{array}\right] = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$$

Για την παράλληλη υλοποίηση του block αλγορίθμου συνήθως επιλέγουμε $q = \sqrt{p} \Leftrightarrow p = q^2$ επεξεργαστές.

1. Ενας απλός Παράλληλος Αλγόριθμος

Θεωρούμε ότι οι δύο $n \times n$ πίνακες A και B είναι διαχωρισμένοι σε p τετραγωνικούς $(n/\sqrt{p}) \times (n/\sqrt{p})$ υποπίνακες $A_{i,j}$ και $B_{i,j}$ $(i,j = 0,1,\cdots,\sqrt{p}-1)$. Οι p αυτοί υποπίνακες αντιστοιχίζονται σε ένα $\sqrt{p} \times \sqrt{p}$ λογικό πλέγμα p επεξεργαστών $P_{0,0},\cdots,P_{\sqrt{p}-1,\sqrt{p}-1}$.

Ο επεξεργαστής $P_{i,j}$ αρχικά αποθηκεύει τους υποπίνακες $A_{i,j}$ και $B_{i,j}$ και μετά υπολογίζει τον υποπίνακα $C_{i,j}$ του πίνακα γινομένου C.

Για τον υπολογισμό του υποπίνακα $C_{i,j}$ ο επεξεργαστής χρειάζεται να αποκτήσει όλους τους υποπίνακες $A_{i,k}$ και $B_{k,j}$ $(k = 0,1,\cdots,\sqrt{p}-1)$. Αυτό επιτυγχάνεται με μια αποστολή όλων προς όλους των υποπινάκων του A στη κάθε γραμμή επεξεργαστών και με μια αποστολή όλων προς όλους των υποπινάκων του B στη κάθε στήλη επεξεργαστών. Αφού ο κάθε επεξεργαστής $P_{i,j}$ αποκτήσει τους υποπίνακες $A_{i,0}, A_{i,1},\cdots,A_{i,\sqrt{p}-1}$ και $B_{0,j}, B_{1,j},\cdots,B_{\sqrt{p}-1,j}$, εκτελεί τον πολλαπλασιασμό και την πρόσθεση των υποπινάκων(βλ. γραμμές 7 και 8 στο σχήμα 1).

Υπερκύβος

Υποθέτουμε ότι το λογικό πλέγμα των επεξεργαστών είναι εμφυτευμένο σε ένα υπερκύβο με p επεξεργαστές. Ο αλγόριθμος απαιτεί δύο βήματα αποστολής μηνυμάτων όλων προς όλους (στο κάθε βήμα εκτελούνται $\sqrt{p}$ ταυτόχρονες αποστολές μηνύματος σε όλες τις γραμμές και στήλες του πλέγματος επεξεργαστών) μεταξύ των ομάδων με $\sqrt{p}$ επεξεργαστές. Τα μηνύματα είναι υποπίνακες με n^2/p στοιχεία. Ο συνολικός χρόνος επικοινωνίας είναι:

$$T_p^{comm} = 2(t_s \log\sqrt{p} + t_w \frac{n^2}{p}(\sqrt{p} - 1))$$

Στη συνέχεια, ο κάθε επεξεργαστής υπολογίζει ένα υποπίνακα C_{ij} ο οποίος απαιτεί $\sqrt{p}$ πολλαπλασιασμούς $(n/\sqrt{p}) \times (n/\sqrt{p})$ υποπινάκων. Αυτό γίνεται σε υπολογιστικό χρόνο

$$T_p^{comp} = \sqrt{p} \times (n/\sqrt{p})^3 = n^3/p$$

Επομένως, ο συνολικός χρόνος της παράλληλης διάτρεξης είναι προσεγγιστικά :

$$T_p^{yper} = \frac{n^3}{p} + t_s \log p + 2t_w \frac{n^2}{\sqrt{p}}.$$

Αρα το κόστος του απλού παράλληλου αλγορίθμου είναι:

$$p \cdot T_p^{yper} = n^3 + t_s p \log p + 2t_w n^2 \sqrt{p}.$$

το οποίο είναι βέλτιστο για $p = O(n^2)$.

2. Αλγόριθμος του Cannon

Ο αλγόριθμος του Cannon είναι μια πιό αποδοτική μορφή του απλού παράλληλου αλγορίθμου για τον πολλαπλασιασμό δύο πινάκων.

Θεωρούμε ότι οι δύο $n \times n$ πίνακες A και B είναι διαχωρισμένοι σε p τετραγωνικούς $(n/\sqrt{p}) \times (n/\sqrt{p})$ υποπίνακες $A_{i,j}$ και $B_{i,j}$ $(i, j = 0, 1, \cdots, \sqrt{p}-1)$. Οι p αυτοί υποπίνακες αντιστοιχίζονται σε ένα $\sqrt{p} \times \sqrt{p}$ λογικό πλέγμα p επεξεργαστών $P_{0,0}, \cdots, P_{\sqrt{p}-1,\sqrt{p}-1}$.

Ο επεξεργαστής $P_{i,j}$ αρχικά αποθηκεύει τους υποπίνακες $A_{i,j}$ και $B_{i,j}$. Αν και κάθε επεξεργαστής στην i γραμμή χρειάζεται όλους τους $\sqrt{p}$ υποπίνακες $A_{i,k}$, $k = 0, 1, \cdots, \sqrt{p} - 1$, είναι δυνατόν να δρομολογηθούν οι υπολογισμοί των $\sqrt{p}$ επεξεργαστών στην i γραμμή έτσι ώστε, στη κάθε χρονική στιγμή, κάθε επεξεργαστής να χρησιμοποιεί ένα διαφορετικό υποπίνακα $A_{i,k}$.

Οι υποπίνακες αυτοί μπορεί να περιστρέφονται συστηματικά μεταξύ των επεξεργαστών μετά από κάθε πολλαπλασιασμό υποπινάκων, έτσι ώστε μετά από κάθε περιστροφή ο κάθε επεξεργαστής να αποκτά ένα νέο υποπίνακα $A_{i,k}$. Αν η ίδια δρομολόγηση εφαρμοσθεί και στις στήλες τότε στη κάθε χρονική στιγμή κανένας επεξεργαστής δεν κατέχει περισσότερα από ένα υποπίνακα από τον κάθε πίνακα.

Πλέγμα

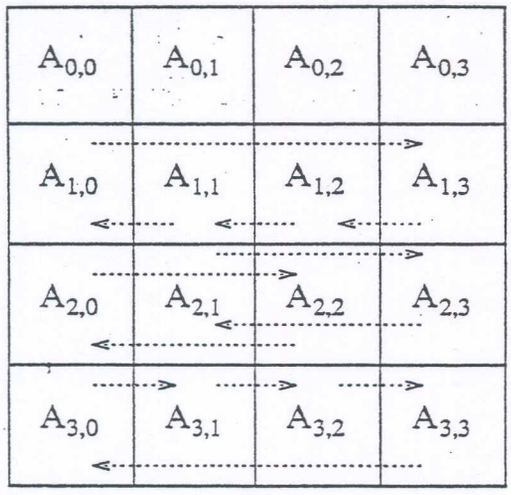
Αν χρησιμοποιήσουμε ένα αναδιπλούμενο $\sqrt{p} \times \sqrt{p}$ πλέγμα επεξεργαστών αντί του υπερκύβου(η αποστολή μηνυμάτων γίνεται με αποθήκευση και προώθηση) τότε ο χρόνος της παράλληλης διάτρεξης επηρεάζεται μόνο στον συντελεστή του t_s και είναι:

$$T_p^{mesh} = \frac{n^3}{p} + 2t_s\sqrt{p} + 2t_w\frac{n^2}{\sqrt{p}}.$$

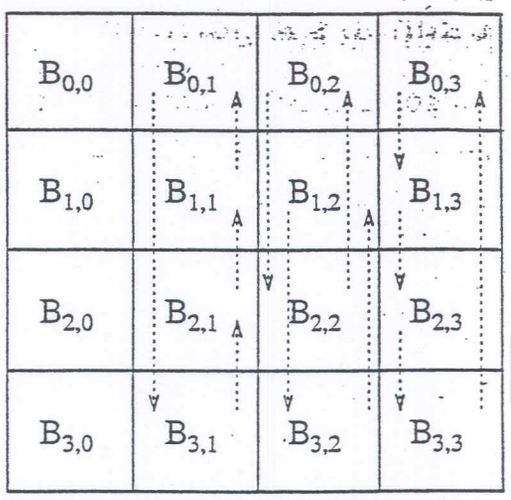
Αν τα μηνύματα είναι μεγάλα, τότε ο χρόνος της παράλληλης εκτέλεσης του απλού αλγορίθμου είναι ο ίδιος και σε μη αναδιπλούμενο πλέγμα που υποστηρίζει αποστολή μηνυμάτων με διαχωρισμό και απόδοση και επικοινωνία δύο κατευθύνσεων.

Ετσι λοιπόν, η συνολικά απαιτούμενη μνήμη του αλγορίθμου για όλους τους επεξεργαστές είναι $\Theta(n^2)$. Ο αλγόριθμος του Cannon στηρίζεται πάνω σε αυτή την ιδέα.

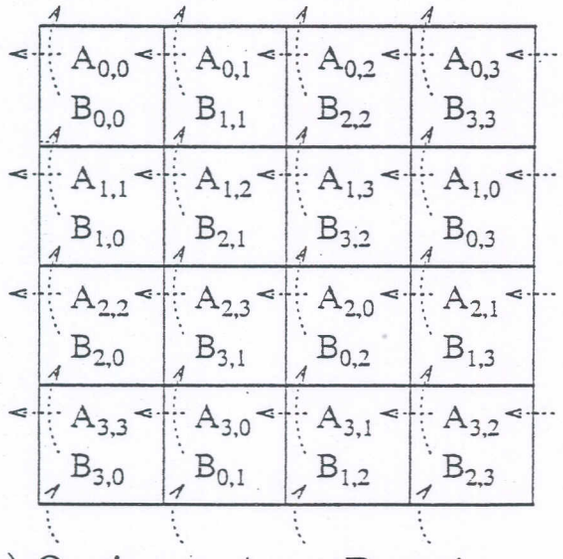
Η δρομολόγηση για τον πολλαπλασιασμό των υποπινάκων με $p = 16$ επεξεργαστές στον αλγόριθμο του Cannon παρουσιάζεται στο παρακάτω σχήμα 2.



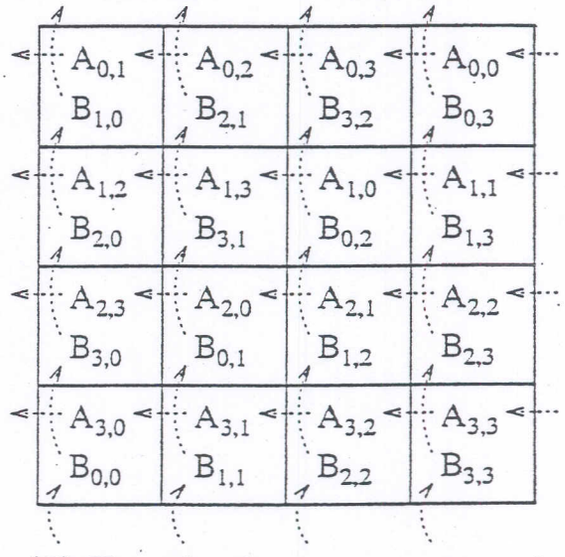
(α) Αρχική διευθέτηση του A



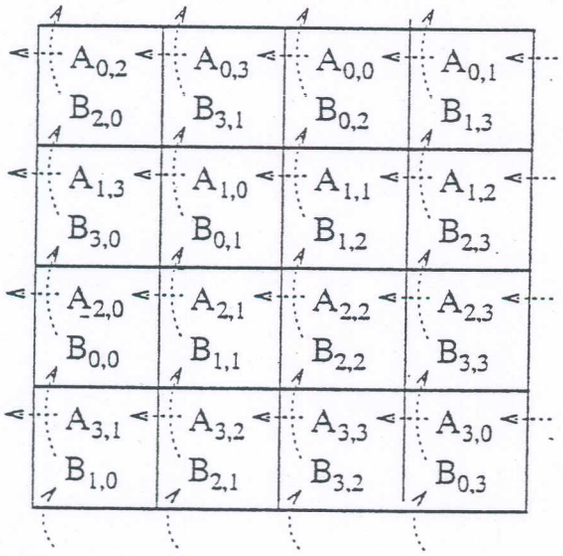
(β) Αρχική διευθέτηση του B



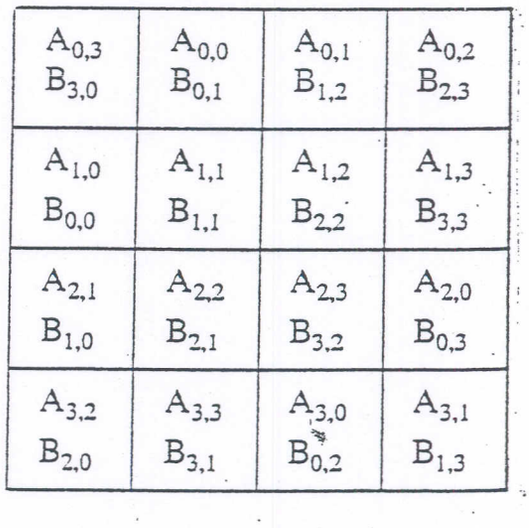
(γ) Οι πίνακες A και B μετά την αρχική διευθέτηση



(δ) Τοποθετήσεις των υποπινάκων μετά την πρώτη ολίσθηση



(ε) Τοποθετήσεις των υποπινάκων μετά την δεύτερη ολίσθηση



(στ) Τοποθετήσεις των υποπινάκων μετά την τρίτη ολίσθηση

Σχήμα 2 Βήματα Επικοινωνίας στον αλγόριθμο του Cannon (p = 16).

Στο πρώτο βήμα επικοινωνίας του αλγορίθμου διευθετούνται οι υποπίνακες των A και B έτσι ώστε ο κάθε επεξεργαστής να πολλαπλασιάζει τους τοπικούς του υποπίνακες.

Οπως φαίνεται στο σχήμα 2(α), η διευθέτηση αυτή επιτυγχάνεται για τον πίνακα A με ολίσθηση όλων των υποπινάκων $A_{i,j}$ προς τα αριστερά (με αναδίπλωση) κατά i θέσεις.

Παρόμοια, όπως φαίνεται στο σχήμα 2(β), όλοι οι υποπίνακες $B_{i,j}$ ολισθαίνουν προς τα πάνω(με αναδίπλωση) κατά j θέσεις.

Οι δύο ανωτέρω λειτουργίες κυκλικής ολίσθησης σε κάθε γραμμή και στήλη των επεξεργαστών τροφοδοτούν τον κάθε επεξεργαστή $P_{i,j}$ με τους υποπίνακες $A_{i,(j+i)\mathrm{mod}\sqrt{p}}$ και $B_{(i+j)\mathrm{mod}\sqrt{p},j}$.

Στο σχήμα 2(γ) φαίνονται οι υποπίνακες των A και B μετά από την πρώτη διευθέτηση, όπου ο κάθε επεξεργαστής είναι έτοιμος να εκτελέσει τον πρώτο πολλαπλασιασμό υποπινάκων.

Μετά από το βήμα του πολλαπλασιασμού υποπινάκων, κάθε υποπίνακας του A μετακινείται μία θέση αριστερά και κάθε υποπίνακας του B μετακινείται μία θέση πάνω(με αναδίπλωση), όπως φαίνεται στο σχήμα 2(δ).

Μία διαδοχική σειρά από $\sqrt{p}$ τέτοιους πολλαπλασιασμούς υποπινάκων και ζευγών ολισθήσεων απλού βήματος για κάθε $A_{i,k}$ και B_{kj} $(k = 0, 1, \cdots, \sqrt{p} - 1)$ στον κάθε επεξεργαστή $P_{i,j}$ υπολογίζει τον υποπίνακα $C_{i,j}$ του τελικού πίνακα γινομένου C.

Υπερκύβος

Η αρχική διευθέτηση των δύο πινάκων(βλ. σχήμα 2(α),(β)) περιλαμβάνει μια κατά γραμμές και μία κατά στήλες κυκλική ολίσθηση. Στις ολισθήσεις αυτές η μέγιστη απόσταση ολίσθησης ενός υποπίνακα είναι $\sqrt{p} - 1$.

Σε ένα υπερκύβο (με αποστολή μηνυμάτων με διαχωρισμό και απόδοση) οι δύο λειτουργίες ολίσθησης απαιτούν συνολικό χρόνο

$$2(t_s + t_w \frac{n^2}{p} + t_h \log\sqrt{p})$$

Κάθε μιά από τις $\sqrt{p}$ ολισθήσεις απλού βήματος στην φάση υπολογισμός-και-ολίσθηση του αλγορίθμου απαιτεί χρόνο $t_s + t_w n^2/p$.

Επομένως, ο συνολικός χρόνος επικοινωνίας(για τους δύο πίνακες) σε αυτή τη φάση του αλγορίθμου είναι:

$$T_p^{comm} = 2(t_s + t_w \frac{n^2}{p})\sqrt{p}.$$

Αν ο p είναι αρκετά μεγάλος, τότε ο χρόνος επικοινωνίας για την αρχική διευθέτηση μπορεί να είναι αμελητέος σε σύγκριση με τον χρόνο που απαιτείται για την φάση υπολογισμού-και-ολίσθησης.

Στον αλγόριθμο αυτόν, ο κάθε επεξεργαστής εκτελεί $\sqrt{p}$ πολλαπλασιασμούς $(n/\sqrt{p}) \times (n/\sqrt{p})$ υποπινάκων. Αυτό γίνεται σε υπολογιστικό χρόνο

$$T_p^{comp} = \sqrt{p} \times (n/\sqrt{p})^3 = n^3/p$$

Επομένως, ο συνολικός χρόνος της παράλληλης διάτρεξης είναι:

$$T_p^{yper} = \frac{n^3}{p} + 2\sqrt{p}t_s + 2t_w\frac{n^2}{\sqrt{p}}.$$

Αρα το κόστος του παράλληλου αλγορίθμου του Cannon είναι το ίδιο με εκείνο του απλού παράλληλου αλγορίθμου.

Πλέγμα

Αν για την υλοποίηση του αλγορίθμου του Cannon χρησιμοποιήσουμε ένα αναδιπλούμενο $\sqrt{p} \times \sqrt{p}$ πλέγμα, τότε ο χρόνος που απαιτείται για την αρχική διευθέτηση των δύο πινάκων είναι

$$2(t_s + t_w\frac{n^2}{p})\sqrt{p}.$$

Ο ίδιος επίσης χρόνος απαιτείται και στην φάση υπολογισμ και-ολίσθηση του αλγορίθμου.

Επομένως, ο συνολικός χρόνος παράλληλης εκτέλεσης του αλγορίθμου του Cannon σε τετραγωνικό πλέγμα είναι:

$$T_p^{mesh} = \frac{n^3}{p} + 4\sqrt{p}t_s + 4t_w\frac{n^2}{\sqrt{p}}.$$

Πολλαπλασιασμός πίνακα με διάνυσμα
(με διαφορετικούς διαχωρισμούς)

$$\underset{n\times n}{A} \cdot \underset{n\times 1}{u} = \underset{n\times 1}{V}$$

Ακολουθιακός αλγόριθμος

```
Procedure MAT_VECT (A, u, V)
    For i:=0 to n-1 do
        V_i := 0
        for j:=0 to n-1 do
            V_i := V_i + a_ij u_j
```

Χρόνος ακολουθιακής εκτέλεσης: $W = n^2$

(1 πολ/σμός + 1 πρόσθεση = 1 χρονική μονάδα)

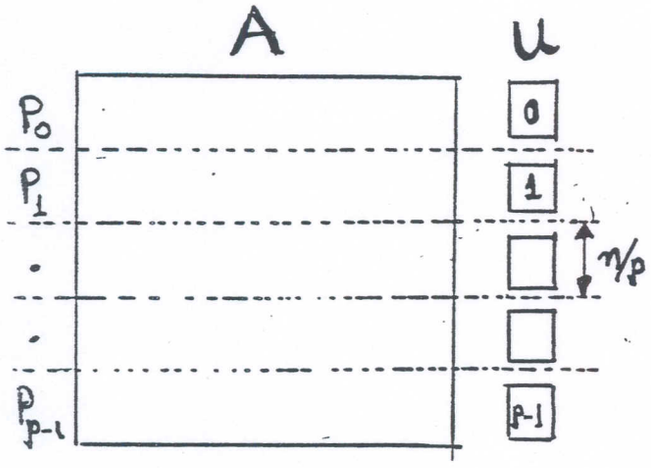
Παράλληλος αλγόριθμος

- Οριζόντιες Λωρίδες
- Κατακόρυφες Λωρίδες
- Τετράγωνα Τεμάχια επιφανείας

1. Διαχωρισμός πίνακα κατά γραμμές

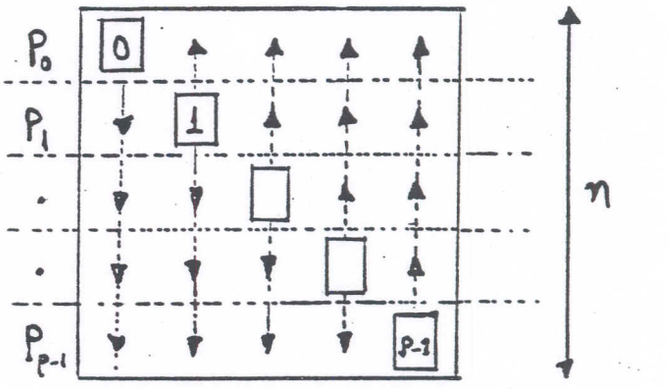
Αριθμός επεξεργαστών $p = 2^n$ (συνήθως)

1 γραμμή A } → 1 επεξεργαστής
1 στοιχείο u }



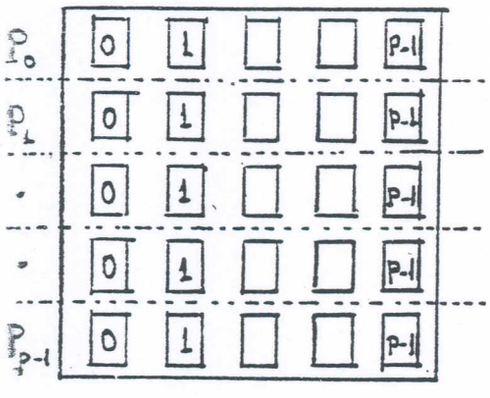
(α)

Αρχικός διαχωρισμός του A και του αρχικού διανύσματος u



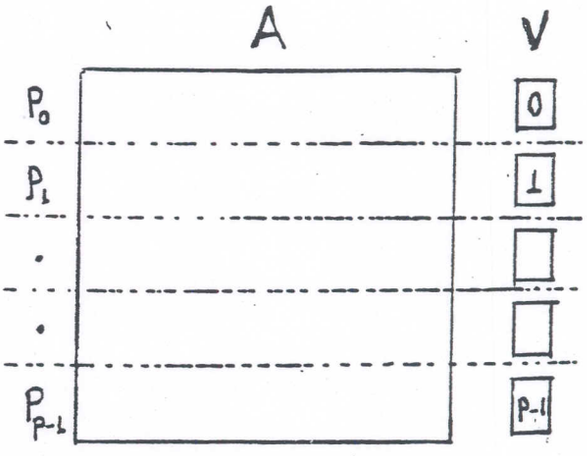
(b)

Κατανομή του διανύσματος u σε όλους τους επεξεργαστές με εκπομπή από όλους σε όλους.



(c)

Μετά την εκπομπή ολόκληρο το διάνυσμα u κατανέμεται στον κάθε επεξεργαστή



(d)

Τελική κατανομή του πίνακα και του διανύσματος αποτελέσματος V

Σχήμα 1.

Ο επεξεργαστής P_i αρχικά αποθηκεύει το u_i και τα στοιχεία $a_{i0}, a_{i1}, \ldots, a_{i,n-1}$ του A και αναλαμβάνει τον υπολογισμό του V_i.

Σύμφωνα με τον ακολουθιακό αλγόριθμο το διάνυσμα u πολλαπλασιάζεται με κάθε γραμμή του A. Επομένως κάθε επεξεργαστής χρειάζεται ολόκληρο το διάνυσμα u.

Επειδή ο κάθε επεξεργαστής ξεκινά κατέχοντας μόνο ένα στοιχείο του u, απαιτείται μια εκπομπή (αποστολή) όλων των στοιχείων του u από όλους τους επεξεργαστές προς όλους. (βλ. (b)).

Αφού το διάνυσμα u κατανεμηθεί σε όλους τους επεξεργαστές (βλ. (c)), τότε ο επεξεργαστής P_i υπολογίζει τη συνιστώσα

$$V_i = \sum_{j=0}^{n-1} a_{ij} u_j$$

Όπως φαίνεται στο σχήμα (d), το διάνυσμα αποτέλεσμα V αποθηκεύεται ακριβώς με τον ίδιο τρόπο όπως το αρχικό διάνυσμα u.

Χρόνος παράλληλης εκτέλεσης με $p=n$ επεξεργαστές	$O(n)$	βέλτιστος

Αποστολή διανύσματος όλων προς όλους : $O(n)$

Πολ/σμός μιας γραμμής του A με το u : $O(n)$

Αριθμός επεξεργαστών $\underline{p < n}$

$\frac{n}{p}$ γραμμές του A
$\frac{n}{p}$ στοιχεία του u
} → 1 επεξεργαστής

Αποστολή όλων προς όλους μηνυμάτων μεγέθους $\frac{n}{p}$ (μεταξύ p επεξεργαστών) } : $t_s \log p + t_w \frac{n}{p}(p-1)$

Πολ/σμός $\frac{n}{p}$ γραμμών του A με n στοιχεία του u στο κάθε επεξεργαστή } : $\frac{n^2}{p}$

Χρόνοι παράλληλης εκτέλεσης

p κεντρικό $\frac{n}{p}(p-1)$

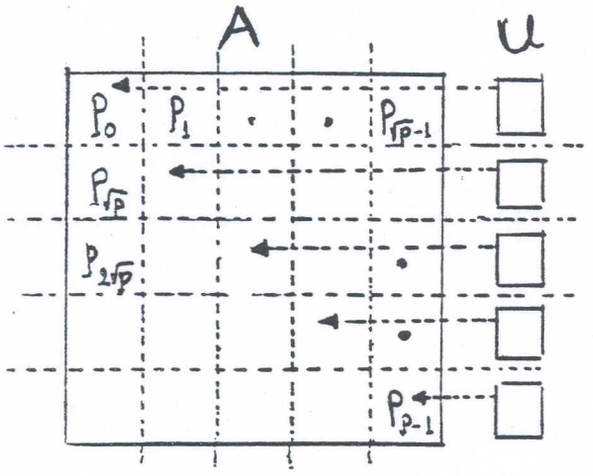
Υπερκύβος : $T_p = \frac{n^2}{p} + t_s \log p + t_w n$ βέλτιστος αν $p = O(n)$

Διδιάστατο δίκτυο (με αναδίπλωση) wrap-around : $T_p = \frac{n^2}{p} + 2t_s(\sqrt{p}-1) + t_w n$

2: Διαχωρισμός σε τετράγωνα τεμάχια.

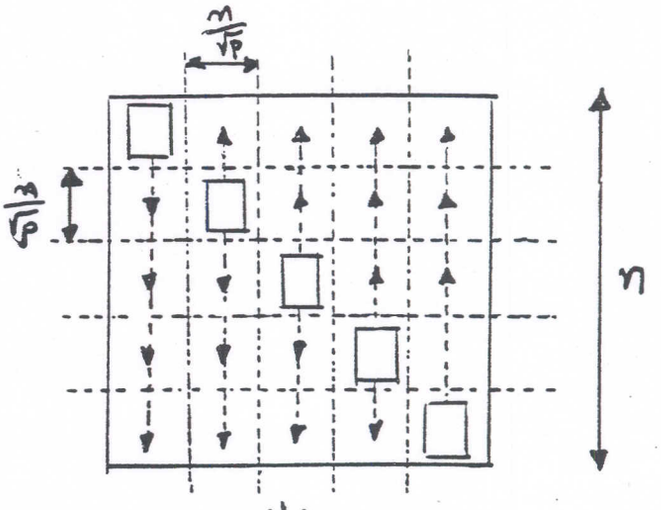
Αριθμός επεξεργαστών $\underline{P = n^2}$

1 στοιχείο του A } → 1 επεξεργαστής
1 στοιχείο του u }



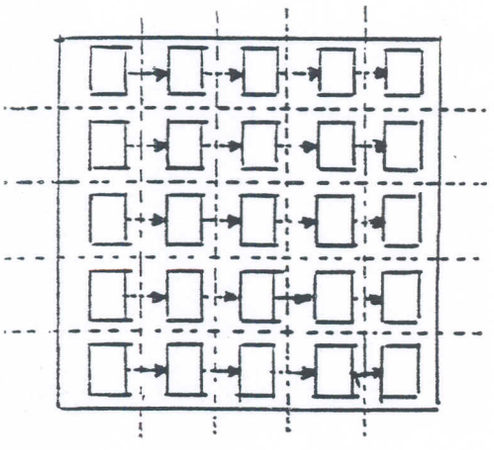
(α)

Κατανομή αρχικών δεδομένων και βήματα επικοινωνίας για την ενδυγράμμιση του u κατά μήκος της διαγωνίου



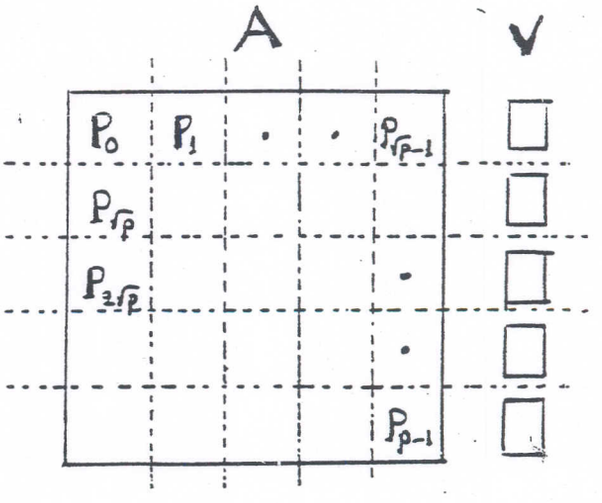
(b)

Εκπομπή ενός προς όλους των στοιχείων του u κατά μήκος των στηλών.



(c)

Συγκέντρωση από κόμβου των μερικών αποτελεσμάτων (ένός προς άλλους)



(d)

Τελική κατανομή του διανύσματος αποτελέσματος V

Σχήμα 2α

Ο πίνακας A nxn διαμοιράζεται στους n^2 επεξεργαστές έτσι ώστε κάθε επεξεργαστής καταχωρεί ένα στοιχείο.

Το nx1 διάνυσμα u καταλήγει στην τελευταία στήλη των n επεξεργαστών.

Επειδή ο αλγόριθμος πολλαπλασιάζει τα στοιχεία του u με τα αντίστοιχα στοιχεία της κάθε γραμμής του A, πρέπει το διάνυσμα u να καταλήξει έτσι ώστε το στοιχείο u_i να είναι διαθέσιμο από το i στοιχείο κάθε γραμμής του A. Τα βήματα επικοινωνίας που απαιτούνται γι αυτό φαίνονται στο Σχήμα 2 ((α) και (b)).

Πριν τον πολλαπλασιασμό, τα στοιχεία του A και του u πρέπει να βρίσκονται στις ίδιες σχετικές θέσεις όπως φαίνεται στο Σχήμα 2 (c)

Ο αλγόριθμος αυτός εκτελεί 3 βασικές λειτουργίες επικοινωνίας που είναι:

1. <u>Ένας προς ένα επικοινωνία</u> για την διευθέτηση του διανύσματος u στην κυρία διαγώνιο.
2. <u>Ένας προς όλους εκπομπή</u> του κάθε στοιχείου του u στους n επεξεργαστές της κάθε στήλης
3. <u>Συγκέντρωση από κόμβου</u> σε κάθε γραμμή..

Οι παραπάνω λειτουργίες απαιτούν χρόνο

$O(n)$ στο <u>διδιάστατο δίκτυο</u>

$O(\log n)$ στον <u>υπερκύβο</u>

Χρόνος παράλληλης εκτέλεσης

με $p = n^2$ επεξεργαστές

	Χρόνος (T_p)	Κόστος ($p \cdot T_p$)
Διδιάστατο δίκτυο	$O(n)$	$O(n^3)$
Υπερκύβος	$O(\log n)$	$O(n^2 \log n)$

Ο αλγόριθμος δεν είναι βέλτιστος.

Αριθμός επεξεργαστών $\underline{p < n^2}$

$\frac{n}{\sqrt{p}} \times \frac{n}{\sqrt{p}}$ υποπίνακας του A $\Big\}$ $\longrightarrow$ 1 επεξεργαστής

$\frac{n}{\sqrt{p}}$ στοιχεία του u

Χρόνος παράλληλης εκτέλεσης

1) Διδιάστατο δίκτυο με διοχέτευση και απόδοση

$$T_p = T_{comp} + T_1 + T_2 + T_3$$

όπου:

$$T_{comp} = \frac{n^2}{p} \quad \leftarrow \text{υπολογιστικός χρόνος}$$

$$T_1 = t_s + t_w \frac{n}{\sqrt{p}} + t_h \sqrt{p} \quad \leftarrow \text{χρόνος διανέμησης του } u \text{ στην κύρια διαγώνιο}$$

$$T_2 = \left(t_s + t_w \frac{n}{\sqrt{p}}\right) \log \sqrt{p} + t_h \sqrt{p} \quad \leftarrow \text{χρόνος αποστολής του } u \text{ κατά στήλες}$$

$$T_3 = \left(t_s + t_w \frac{n}{\sqrt{p}}\right) \log \sqrt{p} + t_h \sqrt{p} \quad \leftarrow \text{χρόνος συγκέντρωσης}$$

Άρα $$T_p \simeq \frac{n^2}{p} + t_s \log p + t_w \frac{n}{\sqrt{p}} \log p + 3 t_h \sqrt{p}$$

Β) Διδιάστατο δίκτυο με αποθήκευση και προώθηση

Χρόνος παράλληλης εκτέλεσης

$$T_p \simeq \frac{n^2}{p} + 2t_s\sqrt{p} + 3t_w \cdot n$$

Σύγκριση των διαχωρισμών 1) κατά οριζόντιες ζωνίδες και
2) κατά τετράγωνα τεμάχια.

1) $$T_p = \frac{n^2}{p} + 2t_s(\sqrt{p}-1) + t_w n$$

2) $$T_p = \frac{n^2}{p} + t_s \log p + t_w \frac{n}{\sqrt{p}} \log p + 3t_h\sqrt{p}$$

Ο πολλαπλασιασμός πίνακα με διάνυσμα είναι ταχύτερα στο διαχωρισμό σε block τετράγωνικές περιοχές

Το συμπέρασμα είναι το ίδιο και στον υπερκύβο.