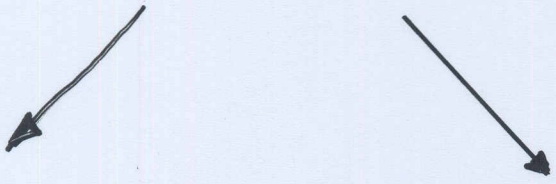
ΠΑΡΑΛΛΗΛΟΙ ΑΛΓΟΡΙΘΜΟΙ

ΓΙΑ ΤΗΝ ΕΠΙΛΥΣΗ ΠΥΚΝΩΝ ΓΡΑΜ. ΣΥΣΤΗΜΑΤΩΝ

ΤΗΣ ΜΟΡΦΗΣ $Ax = b$

A) GAUSSIAN ELIMINATION

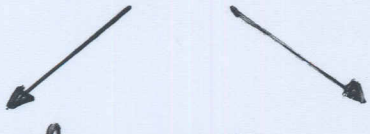


A1. by columns A2. by rows

B) GAUSS – JORDAN

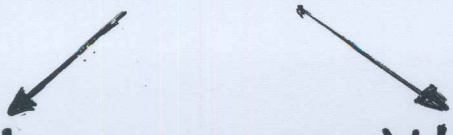
B1. by columns B2. by rows

C) HUARD



C1. columns-rows C2. columns – columns

D) MATRIX FACTORIZATION



D1. LU D2. WZ

Ο αριθμός των αριθμ. πράξεων ανά μέθοδο είναι :

	Διαιρ.	Πολ/σμοί	Προσθέσεις
Gauss	n	$n^3/3 + mn^2 - n/3$	$n^3/3 + (m-1/2)n^2 - (m-1/6)\cdot n$
Jordan	n	$n^3/2 + (m-1/2)n^2$	$n^3/2 + (m-1)n^2 - (m-1/2)n$
Huard	n	$n^3/3 + mn^2 - n/3$	$n^3/3 + (m-1/2)n^2 - (m-1/6)n$

Συμπέρασμα

Υπολ. κόστος Gauss $=$ Υπ. κόστος Huard

Υπολ. κόστος G-Jordan $>$ Υπ. κόστος Gauss.

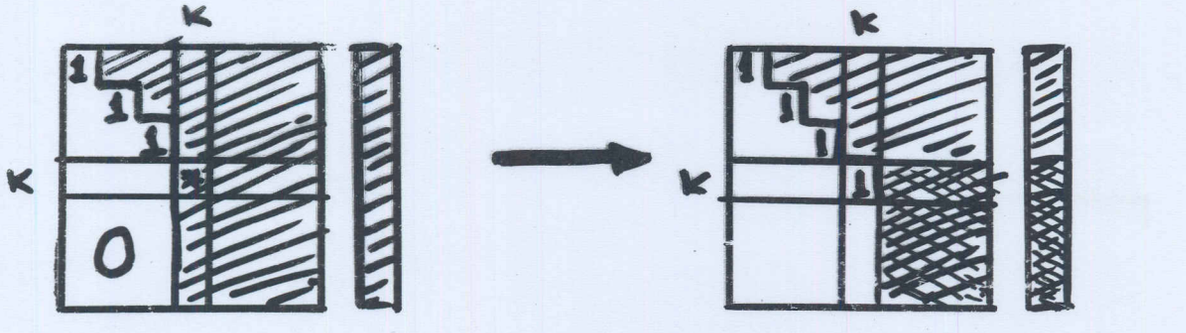
A) Αλγόριθμος GAUSS

```
For k:=1 to n
    c:=1/α_kk
    For j:=k+1 to n+m
        α_kj := α_kj * c
    For i:=k+1 to n
        For j:=k+1 to n+m
            α_ij := α_ij - α_ik * α_kj
```

Τριγωνοποίηση

```
For k:=n-1 to 1 step -1
    For j:=n+1 to n+m
        For i:=k+1 to n
            α_kj := α_kj - α_ki * α_ij
```

Λύση



K βήμα K+1 βήμα

Αν τώρα αλλάξουμε στον προηγούμενο αλγόριθμο τη σειρά των do loops (K, i, j) προκύπτουν οι ακόλουθες παραλλαγές της μεθόδου Gauss.

A1
1. Τύπος KJI (by columns)
2. >> KIJ (by rows)
3. >> JKI (by columns)
4. >> IKJ (by rows)
→ 5. >> IJK (by columns)
6. >> JIK (by rows).

Θα εξετάσουμε αναλυτικώτερα τους αλγορίθμους των versions by columns [1] και [5].

1. Τύπος KJI (μόνο η τριγωνοποίηση).

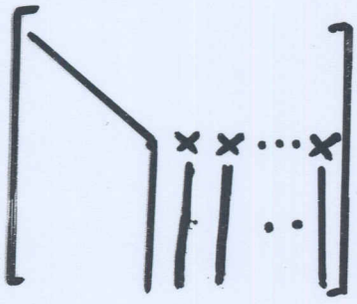
For k:=1 to n-1

T_{kk} : { For i:=k+1 to n
$\alpha_{ik} := \alpha_{ik} / \alpha_{kk}$ } n-k πράξεις

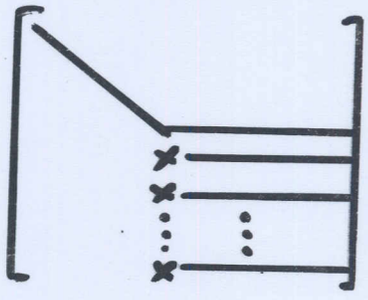
For j:=k+1 to n

T_{kj} : { For i:=k+1 to n
$\alpha_{ij} := \alpha_{ij} - \alpha_{ik} * \alpha_{kj}$ } 2(n-k) πράξεις

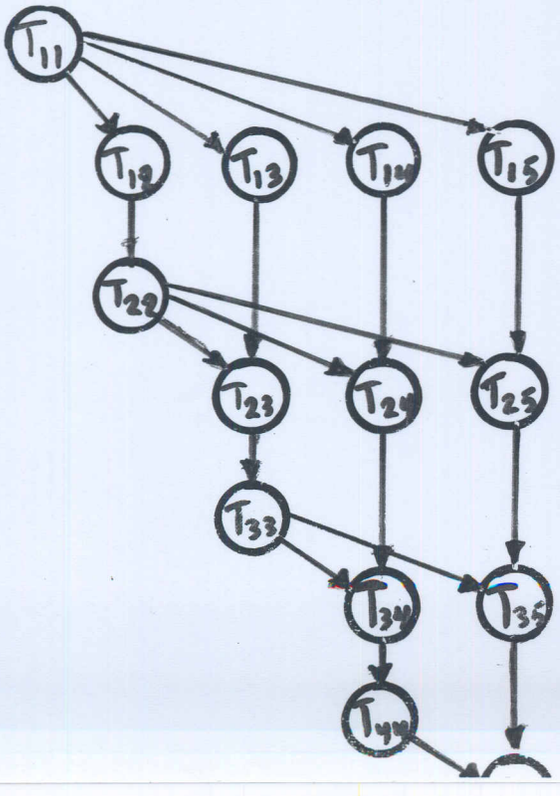
Σχηματικά :



kJI



kIJ

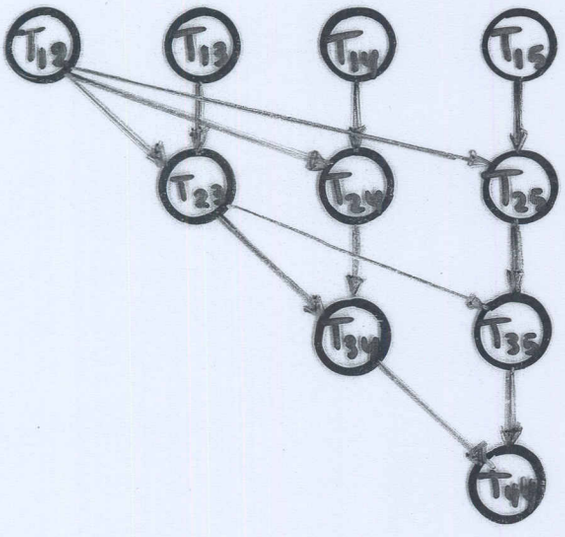


"2-steps" γράφημα

Ο τύπος KJI μπορεί να τροποποιηθεί ως εξής :

1'

```
For k:=1 to n-1
    For j:=k+1 to n
        T_kj : { a_kj := a_kj / a_kk        Σ(n-k)+1 αρ. πράξεις
                 For i:=k+1 to n
                     a_ij := a_ij - a_ik * a_kj }
```



"greedy" γράφημα

5 Τύπος IJK

For $\underline{i}:=2$ to n

For $\underline{j}:=2$ to i

$T_{ij}:\{\alpha_{i,j-1}:=\alpha_{i,j-1}/\alpha_{j-1,j-1}$

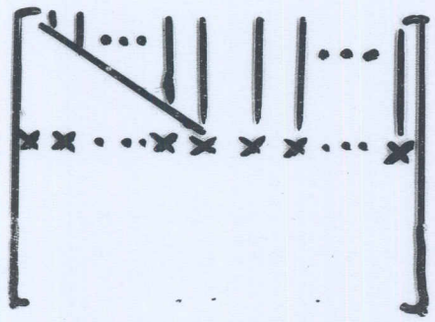
For $\underline{k}:=1$ to $j-1$

$\alpha_{ij}:=\alpha_{ij}-\alpha_{ik}*\alpha_{kj}\}$

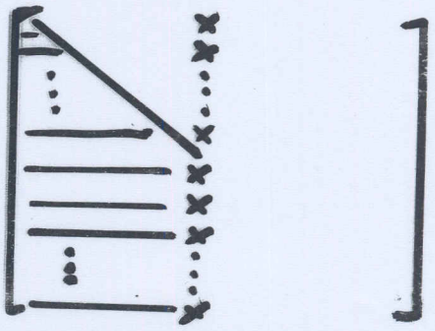
For $\underline{j}:=i+1$ to n

$U_{ij}:\{$For $\underline{k}:=1$ to $i-1$

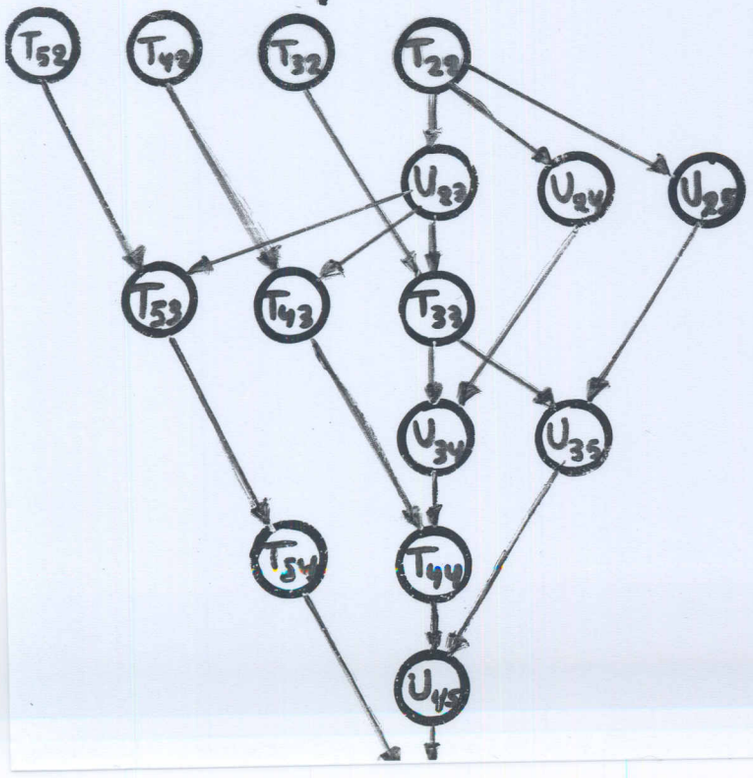
$\alpha_{ij}:=\alpha_{ij}-\alpha_{ik}*\alpha_{kj}\}$



IJK



JIK



"double greedy"

γράφημα

Παρατήρηση

Οι διάφορες παραλλαγές του αλγορίθμου της απαλοιφής του Gauss οδηγούν σε τρείς διαφορετικούς παράλληλους αλγορίθμους (3 είδη γραφημάτων tasks).

Συμπέρασμα

(i) $T_{(IJK)} \geq T_{(KJI)} \geq T_{(KJI')}$

(ii) $T_{(KJI)_{opt}} \geq T_{(KJI')_{opt}}$

(iii) $T_{(IJK)_{opt}} \geq T_{(KJI')_{opt}}$ για $\alpha \leq 1/3$

όπου

$T(x)$: ο ασυμπτωτικός χρόνος εκτέλεσης του παράλληλου αλγορίθμου για την version x

κ. $T(x)_{opt}$: ο χρόνος ενός optimal αλγορίθμου

α : $\alpha \cdot n = p$ (ο αριθμός των processors)

Αρα μεταξύ των τριών διαφορετικών παραλλήλων αλγορίθμων Gauss ο τύπος KJI' έχει την καλύτερη απόδοση για οποιοδήποτε αριθμό processors.
Η λιγώτερο παραλληλίσιμη version είναι η IJK.

Οταν χρησιμοποιούμε optimal αλγορίθμους η μέθοδος KJI' είναι η πιό αποδοτική για κάποιο α.
$(\alpha \leq 1/3)$.

Παράδειγμα αλγορίθμου τριγωνοποίησης με οδήγηση (pivoting).

Αλγόριθμος ΚΙΤ

For $k:=1$ to $n-1$

T_{kk}:
- προσδιορισμός ℓ :
- $|\alpha_{\ell k}| = \max(|\alpha_{kk}|, \ldots, |\alpha_{nk}|)$
- $piv_k := \ell$
- $\alpha_{piv_k,k} \leftrightarrow \alpha_{kk}$
- $c := 1/\alpha_{kk}$
- For $i:=k+1$ to n
  - $\alpha_{ik} := \alpha_{ik} * c$

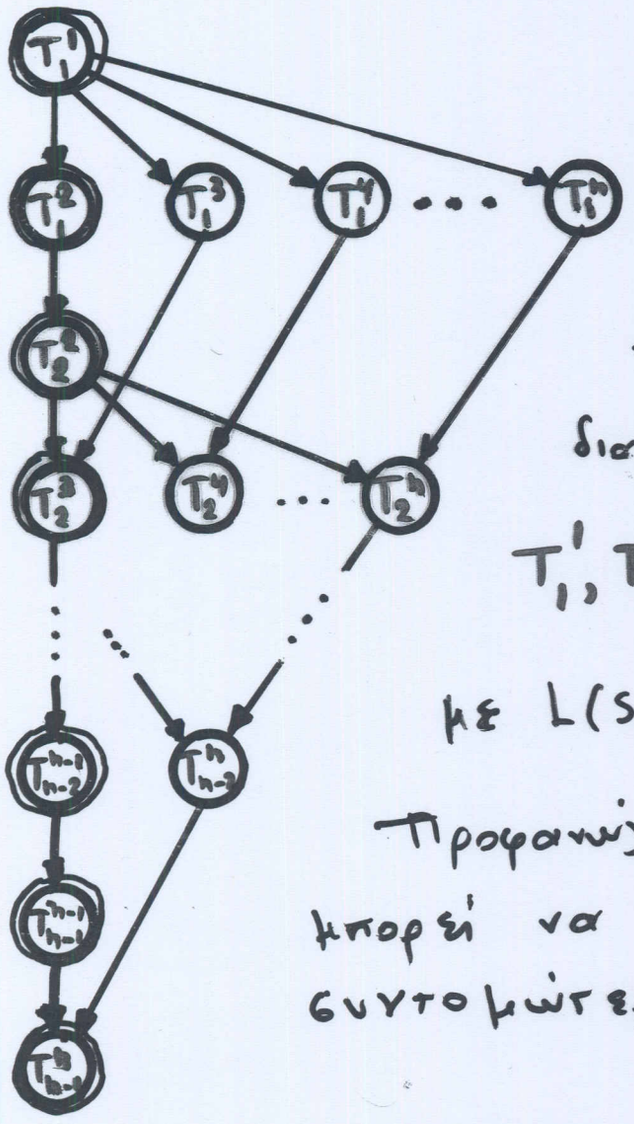
For $j:=k+1$ to n

T_{kj} :
- $\alpha_{piv_k,j} \leftrightarrow \alpha_{kj}$
- for $i:=k+1$ to n
  - $\alpha_{ij} := \alpha_{ij} - \alpha_{ik} * \alpha_{kj}$

Οι χρόνοι εκτέλεσης είναι:

$$W(T_{kj}) = \begin{cases} n+1-k & \text{αν } k=j \\ n-k & \text{αν } k \neq j \end{cases}$$

Το αντίστοιχο μέγιστα παράλληλο γράφημα του αλγορίθμου KJI είναι:



Το μακρύτερο μονοπάτι S_1 διασχίζει τους κόμβους

$$T_1^1, T_1^2, T_2^2, T_2^3, T_3^3, \ldots, T_{n-1}^{n-1}, T_{n-1}^n$$

με $L(S_1) = n^2 - 1$.

Προφανώς ο αλγόριθμος δεν μπορεί να εκτελεσθεί σε χρόνο συντομώτερο από $L(S_1)$.

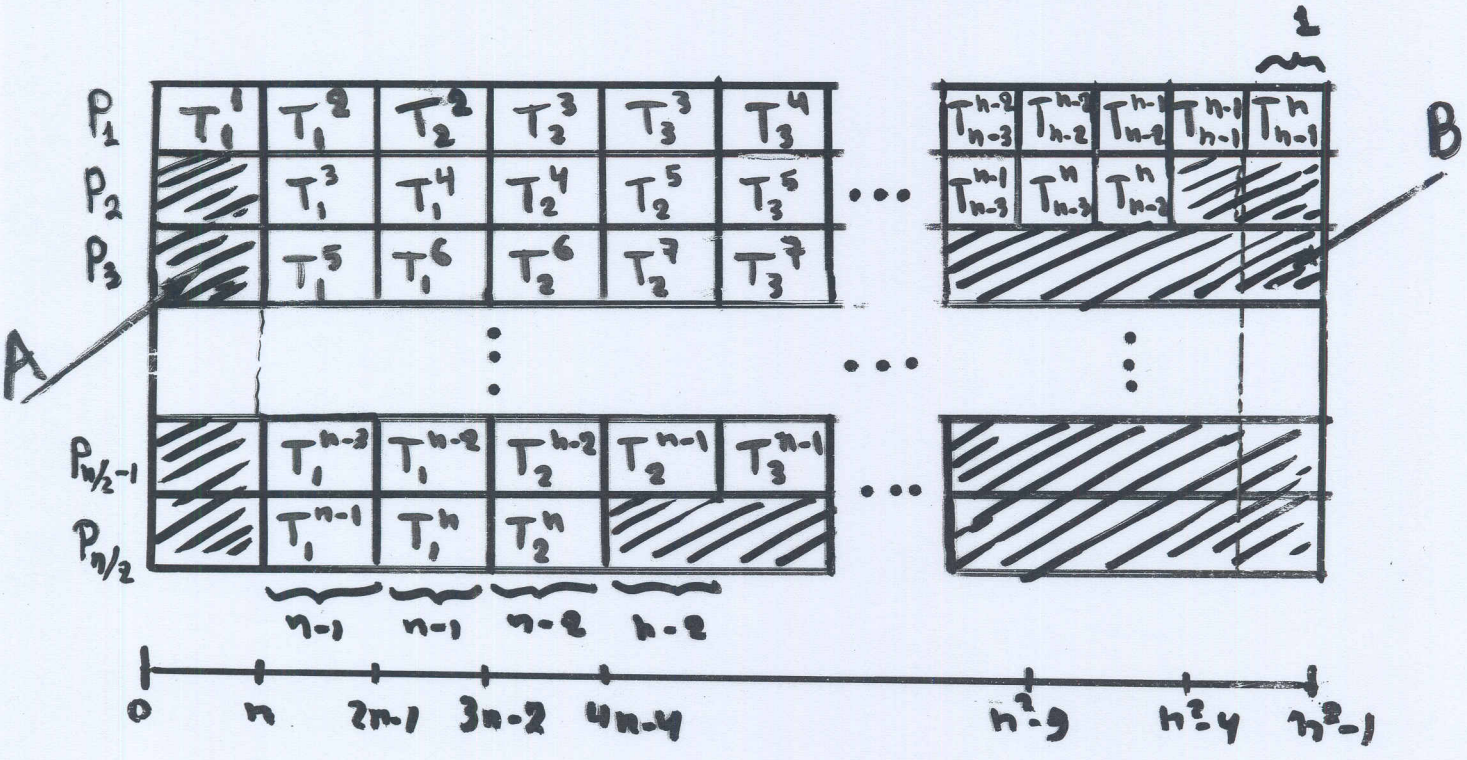
Ας υποθέσουμε ότι διατίθενται $\lceil n/2 \rceil$ processors και χρησιμοποιήσουμε την παρακάτω δρομολόγηση των tasks.

Τα tasks του S_1 δίνονται στον P_1 και τα υπόλοιπα tasks στους υπολοίπους $\lceil n/2 \rceil - 1$ processors δηλαδή τα tasks του

$$S_j : T_1^{2j-1}, T_1^{2j}, T_2^{2j}, T_2^{2j+1}, \ldots, T_{n-2(j-1)}^n$$

δίνονται στον P_j.

Για την περίπτωση n = άρτιος η δρομολόγηση αυτή φαίνεται στο παρακάτω σχήμα:



Αφού η δρομολόγηση αυτή έχει μήκος ίσο με $L(S_1) = n^2 - 1$ είναι <u>optimal</u> για $n/2$ processors και η <u>απόδοση</u>

$$E_p = \lim_{n\to\infty} \frac{S_p}{p} = \lim \frac{n^3/3 + O(n^2)}{(n^2-1)\,n/2} = \frac{2}{3}$$

και αυτή η απόδοση επιτυγχάνεται με ακρίβεια 2% για σχετικά μικρό n ($n \geq 50$).

Τίθεται τώρα το ακόλουθο ενδιαφέρον ερώτημα:

<u>Υπάρχει δρομολόγηση μήκους n^2-1 με $p < \lceil n/2 \rceil$ processors ??</u>

Ορίζουμε <u>υπολογιστικό εμβαδό</u> (CA) μιας δρομολόγησης του συστήματος των tasks

$$CA = \underline{\text{αριθμός processors}} \times \underline{\text{μήκος δρομολόγησης}} - \underline{\text{εμβαδό που όλοι οι processors δεν εργάζονται}}.$$

Στο παράδειγμα μας έχουμε:

$$\boxed{CA} = \underbrace{(n^2-1)p}_{\text{συνολικό εμβαδό}} - \underbrace{(p-1)n}_{A} - 2\sum_{j=2}^{p-1}(p-j)j - \underbrace{(p-1)\cdot 1}_{B}$$

$$= (n^2-1)p - (p-1)(n-1) - \frac{p^3-p}{3}$$

Η <u>συνολική ποσότητα εργασίας</u> για το σύστημα των tasks είναι:

$$\boxed{TW} = \frac{n^3}{3} + \frac{2n}{3} - 1$$

Άρα ένα κάτω φράγμα του αριθμού των processors που απαιτούνται για μία δρομολόγηση μήκους n^2-1 είναι <u>ο μικρότερος p</u>:

$$\boxed{CA \geq TW}$$

Για μικρές άρτιες τιμές του n ο ελάχιστος αριθμός processors p είναι:

$2 \le n \le 8$	$p = n/2$
$10 \le n \le 14$	$p = n/2 - 1$
$16 \le n \le 22$	$p = n/2 - 2$
$24 \le n \le 28$	$p = n/2 - 3$
$30 \le n \le 34$	$p = n/2 - 4$
$36 \le n$	$p \le n/2 - 5$

Για μεγάλες τιμές του n θεωρούμε $p = \alpha \cdot n$ και προσδιορίζουμε το α:

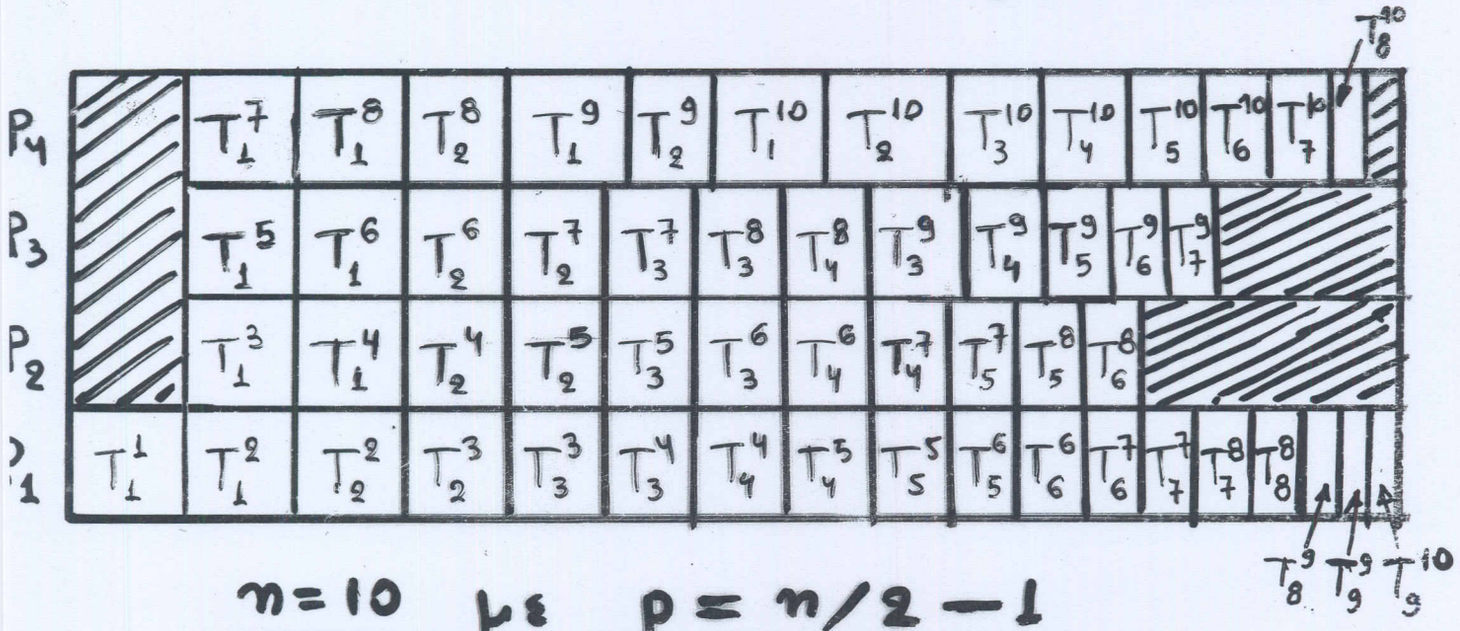
$$\lim_{n \to \infty} \frac{CA}{TW} = 1 \iff 3\alpha - \alpha^3 = 1$$

μια προσεγγιστική λύση είναι

$\alpha = 0.34729$.

** Τονίζεται ότι η τιμή αυτή του α είναι μόνο ένα κάτω φράγμα και δεν είναι γνωστό αν αυτό επιτυγχάνεται στην πράξη.

Στο παρακάτω σχήμα παρίσταται μιά δρομολόγηση μήκους n^2-1 χρησιμοποιώντας $n/2-1$ processors για $n=10$.



$n=10$ με $p=n/2-1$

Έχουν επιτευχθεί δρομολογήσεις για όλες τις άρτιες τιμές $n \leq 50$.

(π.χ. $n=50$ με $n/2-5$ processors).

** Αν το κάτω φράγμα ($\alpha = 0.34729$) ήταν κατορθωτό σ'όλες τις περιπτώσεις, τότε για μεγάλο n και $p = \alpha \cdot n$ processors.

$$E_p = \lim_{p\to\infty} \frac{S_p}{p} = \frac{n^3/3}{(n^2-1)\alpha n} = \frac{1}{3\alpha} \simeq \underline{0.9598}$$

Οταν $p \leq \lceil n/2 \rceil$ τότε ακολουθείται η παρακάτω μέθοδος:

Δίνονται στον <u>processor P_j</u> τα tasks που περιέχονται στα

<u>$S_j, S_{j+p}, \ldots, S_{j+lp}$</u>

όπου $l := \max_{r \in Z} (j + rp)$

έτσι ώστε $j + lp \leq \lceil n/2 \rceil$.

και $S_j = \left(T_1^{2j-1}, T_1^{2j}, T_2^{2j}, T_2^{2j+1}, \ldots, T_{n-2(j-1)}^{n} \right)$

Ας εξετάσουμε πρώτα τη περίπτωση όπου <u>$\lceil n/4 \rceil \leq p \leq \lceil n/2 \rceil$</u>, τότε ο P_1 επεξεργάζεται μόνο τα tasks του S_1 και του S_{1+p}.

Αυτή η δρομολόγηση έχει μήκος:

$$\underline{L} = L(S_1) + \sum_{T_k^j \in S_{1+p}} W(T_k^j) =$$

$$= \underline{2n^2 - 4p^2 + O(n)}$$