Εργασία 2 - Προγραμματισμός με OpenMP

Ονοματεπώνυμο: Μάριος Γιαννόπουλος
Α.Μ.: 1115200000032

Γενικές Πληροφορίες

Υπολογιστικό Σύστημα

Όλο το έργο υλοποιήθηκε στο ίδιο υπολογιστικό περιβάλλον:

- **Όνομα Υπολογιστικού Συστήματος:** Linux12
- **Επεξεργαστής:** Intel(R) Core(TM) i5-6500 CPU @ 3.20GHz
- **Αριθμός Πυρήνων:** 4
- **Λειτουργικό Σύστημα:** Linux Ubuntu 20.04.2 LTS
- **Έκδοση Μεταγλωττιστή:** gcc (Ubuntu 9.4.0-1ubuntu1 20.04.2) 9.4.0

Οδηγίες Εκτέλεσης Python Scripts

Για την εκτέλεση των Python scripts που επεξεργάζονται τα αποτελέσματα, ακολουθήστε τα εξής βήματα:

1. Μεταβείτε στον φάκελο `scripts`.
2. Εγκαταστήστε τις απαραίτητες βιβλιοθήκες:

   ```
   pip install -r requirements.txt
   ```

3. Εκτελέστε το script που σας ενδιαφέρει:

   ```
   python <test_script>.py
   ```

Σημείωση: Όλα τα αποτελέσματα στα γραφήματα είναι από την εκτέλεση των πειραμάτων στο εργαστήριο Linux. Κάθε πείραμα εκτελέστηκε 5 φορές και τα αποτελέσματα αναφέρονται στο μέσο όρο των επαναλήψεων.

Άσκηση 2.1

Εισαγωγή

Σκοπός της παρούσας εργασίας είναι η παραλληλοποίηση του Παιχνιδιού της Ζωής (Game of Life) με χρήση της βιβλιοθήκης OpenMP. Το παιχνίδι, που σχεδιάστηκε από τον John Conway το 1970, είναι ένα μαθηματικό μοντέλο τοπικών κανόνων που παράγει πολύπλοκα μοτίβα. Στο πλαίσιο της εργασίας, υλοποιήθηκε τόσο σειριακή όσο και παράλληλη έκδοση του αλγορίθμου, ενώ τα πειράματα εκτελέστηκαν σε διαφορετικά μεγέθη πλεγμάτων και αριθμούς νημάτων.

Συγχρονισμός

Για την υλοποίηση της παράλληλης έκδοσης, χρησιμοποιήθηκαν οι διευθύνσεις OpenMP για την κατανομή του έργου στα διαθέσιμα νήματα. Λόγω της φύσης της εφαρμογής, κάθε γενιά ενημερώνεται ταυτόχρονα, οπότε δεν απαιτήθηκε πρόσθετος συγχρονισμός μεταξύ των νημάτων.

Πειραματική Διαδικασία

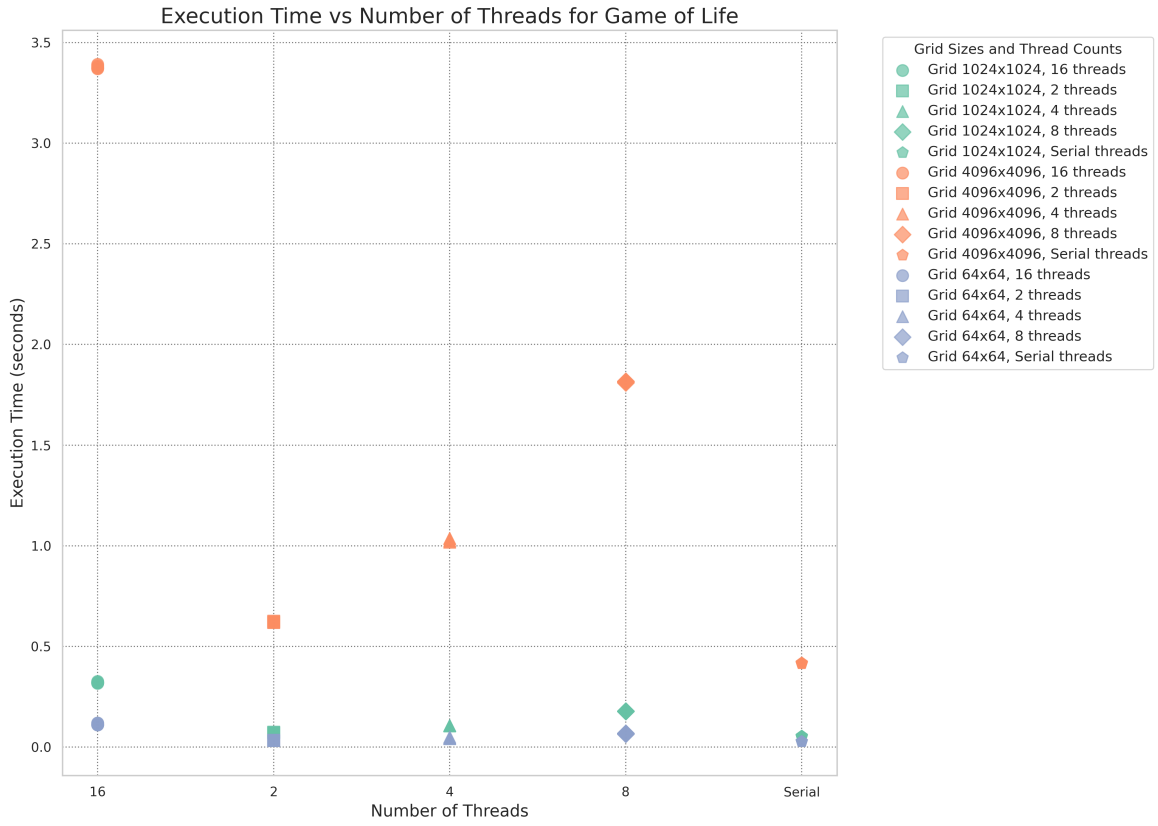
- **Παραμετροποίηση:**
  - Μέγεθος πλέγματος: 64×64, 1024×1024, 4096×4096.
  - Αριθμός γενιών: 1000.
  - Αριθμός νημάτων: 2, 4, 8, 16.
- **Εκτέλεση:**
  - Τα πειράματα εκτελέστηκαν 5 φορές για κάθε συνδυασμό παραμέτρων.
  - Καταγράφηκε ο χρόνος εκτέλεσης για κάθε πείραμα.
  - Τα δεδομένα αποθηκεύτηκαν σε CSV αρχείο.
- **Αυτοματοποίηση:**
  - Αναπτύχθηκαν Python scripts για την εκτέλεση των πειραμάτων και την καταγραφή των δεδομένων.
  - Χρησιμοποιήθηκαν Python scripts για την επεξεργασία των αποτελεσμάτων και τη δημιουργία γραφημάτων.

Αποτελέσματα

- **Σύγκριση Σειριακού και Παράλληλου Αλγορίθμου:**
  - Για μικρά πλέγματα (64×64), η σειριακή έκδοση ήταν ταχύτερη λόγω του overhead της παραλληλοποίησης.
  - Για μεγαλύτερα πλέγματα (1024×1024 και 4096×4096), ο παράλληλος αλγόριθμος απέδωσε καλύτερα.
- **Επιτάχυνση:**
  - Η ταχύτητα εκτέλεσης βελτιώθηκε σημαντικά με την αύξηση του αριθμού των νημάτων, μέχρι να φτάσει το όριο των πυρήνων του επεξεργαστή.
  - Το μέγεθος του πλέγματος επηρέασε σημαντικά την απόδοση της παραλληλοποίησης.

Γραφήματα

Το παρακάτω γράφημα παρουσιάζει τη σχέση μεταξύ του μεγέθους του πλέγματος, του αριθμού των νημάτων και του χρόνου εκτέλεσης:



Γράφημα 1: Χρόνος Εκτέλεσης ανά Μέγεθος Πλέγματος και Αριθμό Νημάτων

Συμπεράσματα

1. Η παραλληλοποίηση είναι αποδοτική για μεγάλα μεγέθη πλεγμάτων.
2. Η απόδοση του αλγορίθμου περιορίζεται από τον αριθμό των διαθέσιμων πυρήνων.
3. Για μικρά μεγέθη πλεγμάτων, η σειριακή έκδοση παραμένει αποδοτικότερη λόγω του overhead της παραλληλοποίησης.

Η εργασία κατέδειξε τη σημασία της παραλληλοποίησης για μεγάλα προβλήματα, αλλά και τους περιορισμούς της λόγω του hardware.

Άσκηση 2.2

Εισαγωγή

Η παρούσα εργασία εξετάζει την παραλληλοποίηση της διαδικασίας επίλυσης γραμμικών συστημάτων με τη χρήση της απαλοιφής Gauss και της αντικατάστασης προς τα πίσω. Η απαλοιφή Gauss μετατρέπει ένα σύστημα σε άνω τριγωνική μορφή, ενώ η αντικατάσταση προς τα πίσω υπολογίζει τις λύσεις. Η εργασία περιλαμβάνει σειριακές και παράλληλες υλοποιήσεις τόσο για τον *"κατά γραμμή"* όσο και για τον *"κατά στήλη"* αλγόριθμο. Η αξιολόγηση των υλοποιήσεων γίνεται μέσω πειραμάτων σε διάφορα μεγέθη συστημάτων και αριθμούς νημάτων.

Προσεγγίσεις Παραλληλοποίησης

1. **Αλγόριθμος "Κατά Γραμμή":**
   - Ο εξωτερικός βρόχος που διατρέχει τις γραμμές μπορεί να παραλληλοποιηθεί.
   - Ο εσωτερικός βρόχος παρουσιάζει εξαρτήσεις δεδομένων που καθιστούν την παραλληλοποίησή του πιο περίπλοκη.
2. **Αλγόριθμος "Κατά Στήλη":**
   - Ο εξωτερικός βρόχος δεν παρουσιάζει εξαρτήσεις δεδομένων και παραλληλοποιείται εύκολα.
   - Ο εσωτερικός βρόχος επηρεάζεται από την εξάρτηση μεταξύ των στηλών, γεγονός που απαιτεί προσεκτικό χειρισμό.

Για την παραλληλοποίηση, χρησιμοποιήθηκε το OpenMP, αξιοποιώντας οδηγίες όπως `#pragma omp parallel for` για τον έλεγχο των βρόχων.

Πειραματική Διαδικασία

- **Παραμετροποίηση:**
  - Μέγεθος συστήματος (n): 100, 1000, 5000, 10000.
  - Αριθμός νημάτων: 2, 4, 8, 16.
  - Τύποι Αλγορίθμων:
    - "Κατά Γραμμή" και "Κατά Στήλη".
    - Σειριακή και Παράλληλη Υλοποίηση.
  - Χρονοπρογραμματισμός (Schedule): static, dynamic, guided, runtime.
- **Διαδικασία Εκτέλεσης:**
  - Κάθε πείραμα εκτελέστηκε 5 φορές και μετρήθηκε ο μέσος χρόνος εκτέλεσης.
  - Τα δεδομένα αποθηκεύτηκαν σε CSV για ανάλυση.
- **Αυτοματοποίηση:**
  - Αναπτύχθηκαν scripts σε Python για την αυτοματοποίηση των πειραμάτων, την επεξεργασία των αποτελεσμάτων και τη δημιουργία γραφημάτων.

Αποτελέσματα

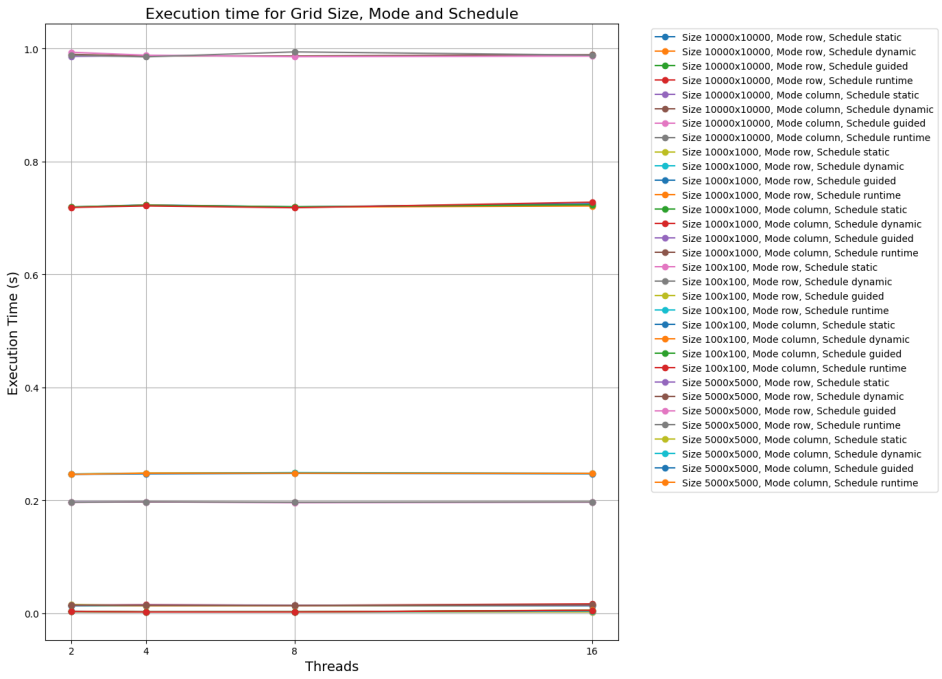
- **Σύγκριση Σειριακού και Παράλληλου Αλγορίθμου:**
  - Για μικρά μεγέθη πινάκων (100), ο σειριακός αλγόριθμος είναι συχνά ταχύτερος λόγω του overhead της παραλληλοποίησης.
  - Για μεγαλύτερα μεγέθη πινάκων (1000 και άνω), οι παράλληλοι αλγόριθμοι υπερέχουν.
- **Επιδόσεις ανά Schedule:**
  - Το *static* schedule είχε σταθερή απόδοση και ήταν αποδοτικό για μικρά grids.
  - Το *dynamic* schedule είχε πλεονέκτημα σε ανομοιόμορφα workloads, αλλά παρουσίαζε μεγαλύτερο overhead.
  - Το *guided* schedule προσέφερε καλή ισορροπία για μεγάλα grids.

– Το *runtime* επέτρεψε τη δυναμική ρύθμιση αλλά παρουσίασε αστάθεια σε ορισμένες περιπτώσεις.

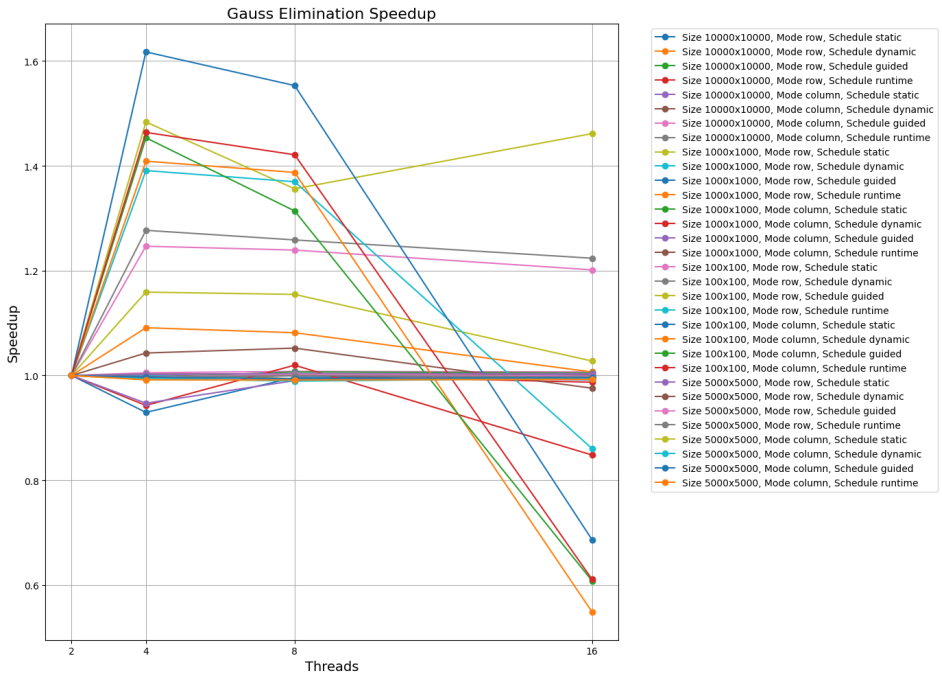
- **Γενικές Παρατηρήσεις:**
  – Ο "κατά γραμμή" αλγόριθμος είχε γενικά καλύτερη απόδοση λόγω μικρότερου υπολογιστικού κόστους.
  – Το μέγεθος του πίνακα και ο αριθμός των νημάτων επηρέασαν σημαντικά την απόδοση.

Γραφήματα

Τα παρακάτω γραφήματα παρουσιάζουν την επίδραση του μεγέθους του πίνακα, του αριθμού των νημάτων, του τύπου του αλγορίθμου και του schedule στον χρόνο εκτέλεσης:



Γράφημα 2: Χρόνος Εκτέλεσης κατά Μέγεθος Πίνακα, Αλγόριθμο και Τύπο Εκτέλεσης



Γράφημα 3: Επίδραση του Schedule στον Χρόνο Εκτέλεσης

Συμπεράσματα

1. Η παραλληλοποίηση βελτιώνει σημαντικά τον χρόνο εκτέλεσης για μεγάλους πίνακες.
2. Ο "κατά γραμμή" αλγόριθμος υπερτερεί σε απόδοση λόγω μικρότερης πολυπλοκότητας.
3. Το μέγεθος του πίνακα και ο αριθμός των νημάτων επηρεάζουν την επεκτασιμότητα και την απόδοση.
4. Οι παράμετροι προγραμματισμού, όπως το schedule, μπορούν να βελτιστοποιήσουν περαιτέρω τον χρόνο εκτέλεσης.

Η εργασία ανέδειξε την αποδοτικότητα της παραλληλοποίησης για μεγάλα συστήματα, ενώ τόνισε τη σημασία της βέλτιστης χρήσης των εργαλείων OpenMP για τη μέγιστη αξιοποίηση του hardware.

Άσκηση 2.3

Εισαγωγή

Σκοπός της παρούσας εργασίας είναι η παραλληλοποίηση του Παιχνιδιού της Ζωής (Game of Life) χρησιμοποιώντας την οδηγία `task` της βιβλιοθήκης OpenMP. Η παραλληλοποίηση με τη χρήση της οδηγίας `task` επιτρέπει την κατανομή του υπολογιστικού έργου σε επιμέρους μονάδες εργασίας, οι οποίες μπορούν να εκτελούνται ασύγχρονα. Στο πλαίσιο της εργασίας, υλοποιήθηκε μια παράλληλη έκδοση του αλγορίθμου με την οδηγία `task`, η οποία συγκρίνεται με την προηγούμενη υλοποίηση της Άσκησης 2.1.

Συγχρονισμός

Για την υλοποίηση της παράλληλης έκδοσης με την οδηγία `task`, χρησιμοποιήθηκε η κατάλληλη κατανομή εργασιών μεταξύ των νημάτων, με στόχο την αποτελεσματική παράλληλη εκτέλεση. Κάθε γενιά του

παιχνιδιού υποδιαιρείται σε μικρότερα καθήκοντα, τα οποία ανατίθενται σε διαφορετικά νήματα μέσω `task`. Η χρήση των `task` εξασφαλίζει τη σωστή εκτέλεση του αλγορίθμου χωρίς να απαιτείται άλλος συγχρονισμός μεταξύ των νημάτων.

Πειραματική Διαδικασία

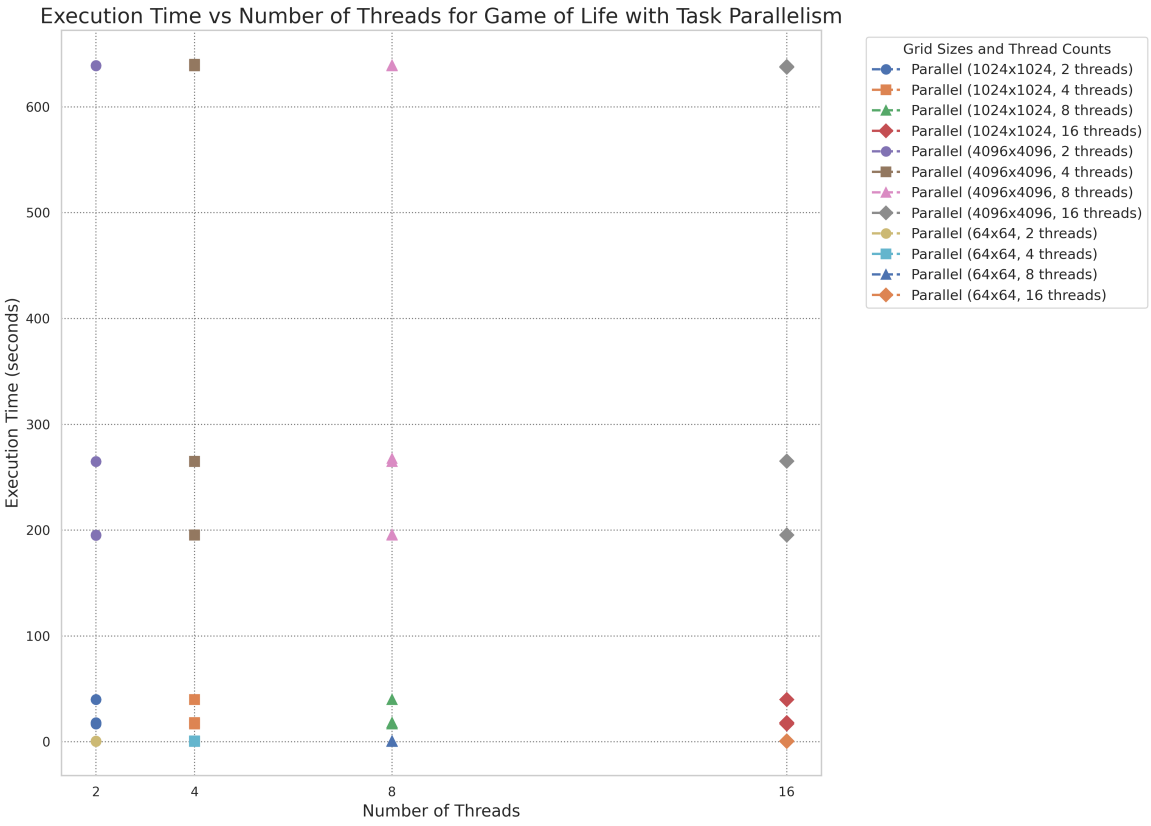
- **Παραμετροποίηση:**
  - Μέγεθος πλέγματος: 64×64, 1024×1024, 4096×4096.
  - Αριθμός γενιών: 1000.
  - Αριθμός νημάτων: 2, 4, 8, 16.
- **Εκτέλεση:**
  - Τα πειράματα εκτελέστηκαν 5 φορές για κάθε συνδυασμό παραμέτρων.
  - Καταγράφηκε ο χρόνος εκτέλεσης για κάθε πείραμα.
  - Τα δεδομένα αποθηκεύτηκαν σε CSV αρχείο.
- **Αυτοματοποίηση:**
  - Αναπτύχθηκαν Python scripts για την εκτέλεση των πειραμάτων και την καταγραφή των δεδομένων.
  - Χρησιμοποιήθηκαν Python scripts για την επεξεργασία των αποτελεσμάτων και τη δημιουργία γραφημάτων.

Αποτελέσματα

- **Σύγκριση Σειριακού και Παράλληλου Αλγορίθμου (OpenMP task):**
  - Για μικρά πλέγματα (64×64), η σειριακή έκδοση ήταν ταχύτερη λόγω του overhead της παραλληλοποίησης.
  - Για μεγαλύτερα πλέγματα (1024×1024 και 4096×4096), ο παράλληλος αλγόριθμος με την οδηγία `task` απέδωσε καλύτερα από τον σειριακό.
- **Επιτάχυνση:**
  - Η ταχύτητα εκτέλεσης βελτιώθηκε με την αύξηση του αριθμού των νημάτων, μέχρι να φτάσει το όριο των πυρήνων του επεξεργαστή.
  - Το μέγεθος του πλέγματος επηρέασε την απόδοση, με μεγαλύτερα πλέγματα να επωφελούνται περισσότερο από την παραλληλοποίηση.

Γραφήματα

Το παρακάτω γράφημα παρουσιάζει τη σχέση μεταξύ του μεγέθους του πλέγματος, του αριθμού των νημάτων και του χρόνου εκτέλεσης για την παράλληλη υλοποίηση με την οδηγία `task`:



Γράφημα 4: Χρόνος Εκτέλεσης ανά Μέγεθος Πλέγματος και Αριθμό Νημάτων με OpenMP Task

Συμπεράσματα

1. Η παραλληλοποίηση με την οδηγία `task` αποδείχθηκε αποδοτική για μεγάλα μεγέθη πλεγμάτων.
2. Η απόδοση της παραλληλοποίησης περιορίζεται από τον αριθμό των διαθέσιμων πυρήνων του επεξεργαστή.
3. Για μικρά μεγέθη πλεγμάτων, η σειριακή έκδοση παραμένει αποδοτικότερη λόγω του overhead της παραλληλοποίησης.
4. Η παραλληλία με `task` προσφέρει μεγαλύτερη ευελιξία στην κατανομή των εργασιών και την αξιοποίηση των πόρων του υπολογιστή.

Η εργασία κατέδειξε τη σημασία της παραλληλοποίησης με τη χρήση της οδηγίας `task`, αλλά και τις προκλήσεις που σχετίζονται με τη διαχείριση των πόρων του υπολογιστή.