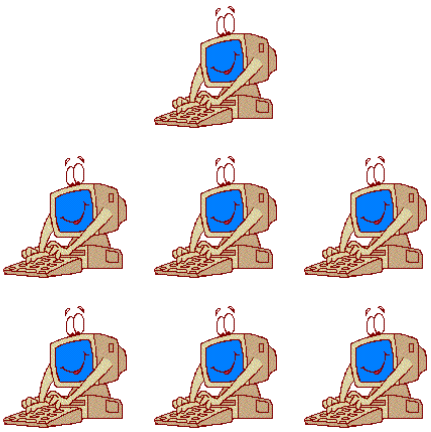
ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΑΡΑΓΩΓΗΣ & ΔΙΟΙΚΗΣΗΣ

ΕΠΙΤΑΧΥΝΣΗ ΕΞΕΛΙΚΤΙΚΩΝ ΑΛΓΟΡΙΘΜΩΝ ΜΕ ΧΡΗΣΗ ΠΑΡΑΛΛΗΛΗΣ ΕΠΕΞΕΡΓΑΣΙΑΣ



Διατριβή που υπεβλήθη για τη μερική ικανοποίηση των απαιτήσεων για την απόκτηση Μεταπτυχιακού Διπλώματος Ειδίκευσης

υπό
Μαρίνα Σ. Ντιπτένη

Χανιά
Αύγουστος 2006

Η διατριβή της Μαρίνας Σ. Ντιπτένη, εγκρίνεται

Ιωάννης Κ. Νικολός
Λέκτορας, Επιβλέπων _____

Βασίλειος Σ. Κουϊκόγλου
Καθηγητής _____

Νικόλαος Χρ. Τσουρβελούδης
Επίκουρος Καθηγητής _____

ΠΕΡΙΕΧΟΜΕΝΑ

ΕΥΡΕΤΗΡΙΟ ΕΙΚΟΝΩΝ

ΕΥΧΑΡΙΣΤΙΕΣ

Σε αυτό το σημείο θα ήθελα να ευχαριστήσω όλους όσους βοήθησαν στην ολοκλήρωση αυτής της εργασίας.

Αρχικά θέλω να ευχαριστήσω τον επιβλέποντα καθηγητή μου, Λέκτορα Ιωάννη Κ. Νικολό, για την καθοδήγηση, την πολύτιμη βοήθεια του, καθώς επίσης και την εμπιστοσύνη που μου έχει δείξει, όλα τα χρόνια της συνεργασίας μας.

Ιδιαίτερα πολύτιμη ήταν η συμβολή του φίλου μου και συνεργάτη Ιωάννη Βαλάκου, ο οποίος στάθηκε αρωγός σε όλες τις δυσκολίες που συνάντησα κατά τη διάρκεια εκπόνησης της εργασίας μου. Τον ευχαριστώ για τη βοήθεια του, τη συμπαράσταση και το χρόνο που μου αφιέρωσε.

Θα ήθελα επίσης να ευχαριστήσω το φίλο μου και συνάδελφο Σωτήρη Σαρακηνό, για την καθημερινή συνεργασία, την υποστήριξη και τη βοήθεια που μου προσέφερε οποιαδήποτε στιγμή τη χρειάστηκα.

Ένα μεγάλο ευχαριστώ οφείλω σε όλους όσους μου προσέφεραν πολύτιμες πληροφορίες, κατά τη διάρκεια της αναζήτησης μεθόδων για την υλοποίηση της εργασίας μου.

Τέλος, θα ήθελα να ευχαριστήσω ιδιαίτερα τους γονείς μου Μαρία και Σταύρο, τον αδερφό μου Ηλία και τη γιαγιά μου Κατίνα, για την αγάπη τους και την υποστήριξη τους όλα τα χρόνια των σπουδών μου, και θα ήθελα να αφιερώσω την εργασία αυτή στην οικογένεια μου και στη μνήμη του παππού μου Θανάση.

Ντιπτένη Μαρίνα
Αύγουστος 2006

ΣΥΝΤΟΜΟ ΒΙΟΓΡΑΦΙΚΟ ΣΗΜΕΙΩΜΑ

Η Μαρίνα Σ. Ντιπτένη γεννήθηκε στη Βέροια το 1981, όπου μεγάλωσε και ολοκλήρωσε τη βασική της εκπαίδευση στο Ενιαίο Πολυκλαδικό Λύκειο Βέροιας. Το 1999, εισήχθη στο Τμήμα Μηχανικών Παραγωγής και Διοίκησης του Πολυτεχνείου Κρήτης. Το 2004 αποφοίτησε με βαθμό 7,65 και εισήχθη στο Μεταπτυχιακό Πρόγραμμα Ειδίκευσης του ιδίου τμήματος, στον τομέα Συστημάτων Παραγωγής. Με την εργασία αυτή ολοκληρώνει τις μεταπτυχιακές σπουδές της.

ΠΕΡΙΛΗΨΗ

Η βασική ιδέα των παράλληλων προγραμμάτων είναι ο διαχωρισμός ενός μεγάλου προβλήματος σε μικρότερα υποπροβλήματα και η ταυτόχρονη επίλυση τους με χρήση παράλληλων επεξεργαστών. Αυτή η τακτική του διαχωρισμού και της επίλυσης, μπορεί να εφαρμοστεί στους Εξελικτικούς Αλγόριθμους με πολλούς τρόπους και υπάρχει πλήθος επιτυχημένων εφαρμογών. Κάποιες παράλληλες μέθοδοι χρησιμοποιούν έναν ενιαίο πληθυσμό, ενώ άλλες χωρίζουν τον πληθυσμό σε μεμονωμένους υποπληθυσμούς. Κάποιες μέθοδοι εκμεταλλεύονται μαζικά υπολογιστές παράλληλης αρχιτεκτονικής, ενώ άλλες εφαρμόζονται καλύτερα σε πολύ-υπολογιστές με λιγότερα και πιο ικανά στοιχεία επεξεργασίας, που συνδέονται με ένα πιο αργό δίκτυο.

Για την ανάπτυξη Παράλληλων Εξελικτικών Αλγορίθμων δεν είναι αναγκαία η ύπαρξη ειδικού υπολογιστικού υλικού. Εν γένει, οι απαιτήσεις για την επικοινωνία όλων των αλγορίθμων είναι χαμηλές και μικρού κόστους, όπως νέφη υπολογιστών, σταθμοί εργασίας, ή επικοινωνίες βασισμένες στο διαδίκτυο, οι οποίες μπορούν να αποδειχθούν ιδιαίτερα πρακτικές. Πολλές φορές, δεδομένου ότι κάποιοι παράλληλοι αλγόριθμοι έχουν ενδιαφέρουσες ιδιότητες ανεξάρτητα από τις εφαρμογές τους, η προσομοίωση τους σε σειριακούς επεξεργαστές μπορεί να αποδειχθεί εξίσου αποτελεσματική.

Σκοπός της παρούσας εργασίας είναι η ανάπτυξη λογισμικού Παράλληλου Διαφορικού Εξελικτικού (ΠΔΕ) αλγορίθμου, χωρίς ιδιαίτερες απαιτήσεις σε υπολογιστικό υλικό, για τη μείωση του χρόνου βελτιστοποίησης πραγματικών προβλημάτων με χρονοβόρες συναρτήσεις προσαρμογής. Στα κεφάλαια που ακολουθούν περιγράφονται όλα εκείνα τα στοιχεία που βοηθούν στην κατανόηση των Παράλληλων Εξελικτικών Αλγορίθμων (ΠΕΑ) και αναλύονται δύο εναλλακτικοί τρόποι παραλληλοποίησης ενός Διαφορικού Εξελικτικού αλγόριθμου, παρουσιάζοντας παράλληλα κάποιες εφαρμογές. Συγκεκριμένα στο πρώτο κεφάλαιο παρατίθεται μια πλήρης περιγραφή των ΠΕΑ, περιγράφοντας τα σημαντικότερα στοιχεία τους και τις κατηγορίες στις οποίες χωρίζονται. Το δεύτερο κεφάλαιο παρουσιάζει την παράλληλη αρχιτεκτονική των υπολογιστών και αναφέρει εργαλεία που χρησιμοποιούνται για την παραλληλοποίηση αλγορίθμων. Στο τρίτο κεφάλαιο περιγράφεται αναλυτικά η λειτουργία του ΔΕ αλγόριθμου και αναφέρονται κάποιες παράλληλες εφαρμογές σε αυτό τον χώρο. Στο Κεφάλαιο 4 παρουσιάζεται η ανάπτυξη ενός απλού παράλληλου ΔΕ μαζί με κάποιες εφαρμογές σε προβλήματα βελτιστοποίησης αεροτομών, όπου η επικοινωνία πραγματοποιείται με τη βοήθεια αρχείων κειμένου μέσω Windows, χωρίς τη χρήση κάποιου πρωτόκολλου επικοινωνίας. Ο αλγόριθμος αυτός αποδείχτηκε αρκετά ασταθής λόγω της παρουσίας ταυτόχρονης προσπέλασης αρχείων. Για το λόγο αυτό ακολουθήθηκε μια εναλλακτική προσέγγιση παραλληλοποίησης του ΔΕ αλγορίθμου με τη χρήση του προτύπου MPI, η οποία παρουσιάζεται στο Κεφάλαιο 5 μαζί με κάποια βασικά στοιχεία για την κατανόηση του προτύπου και τις εφαρμογές που πραγματοποιήθηκαν. Τέλος στο Κεφάλαιο 6 αναφέρονται τα συμπεράσματα που προέκυψαν από την παρούσα εργασία.

ΚΕΦΑΛΑΙΟ 1

ΠΑΡΑΛΛΗΛΟΙ ΕΞΕΛΙΚΤΙΚΟΙ ΑΛΓΟΡΙΘΜΟΙ

1.1 Γενικά

Οι Εξελικτικοί Αλγόριθμοι (ΕΑ) είναι στοχαστικές μέθοδοι αναζήτησης, οι οποίες έχουν εφαρμοστεί επιτυχώς σε πολλά προβλήματα αναζήτησης, βελτιστοποίησης και μηχανικής μάθησης [1], [2]. Σε αντίθεση με τις περισσότερες τεχνικές βελτιστοποίησης, οι ΕΑ χρησιμοποιούν έναν πληθυσμό λύσεων, τον οποίο διαχειρίζονται με ανταγωνιστικό τρόπο, εφαρμόζοντας κάποιους τελεστές διαφοροποίησης, για την εύρεση μιας αρκετά ικανοποιητικής, αν όχι ολικά βέλτιστης, λύσης. Άλλες ευρετικές μέθοδοι, όπως η προσομοιωμένη ανόπτηση [3], εμφανίζουν περιορισμένες δυνατότητες σε προβλήματα με πολύ μεγάλο χώρο εφικτών λύσεων και έχουν αυξημένη πιθανότητα, από τη φύση τους, εγκλωβισμού σε τοπικά βέλτιστα.

Σε πολύπλοκα προβλήματα, όπως είναι πραγματικά προβλήματα βελτιστοποίησης σχεδίασης, η εκτέλεση του αναπαραγωγικού κύκλου ενός απλού ΕΑ για μεγάλους πληθυσμούς απαιτεί σημαντικούς υπολογιστικούς πόρους, τόσο σε μνήμη όσο και σε χρόνο επεξεργασίας, με αποτέλεσμα πολλές φορές ο συνολικός χρόνος επεξεργασίας να υπολογίζεται σε μέρες ή ακόμα και σε μήνες. Το γεγονός αυτό προφανώς είναι μη αποδεκτό, οπότε πρέπει να αναζητηθούν πιο αποδοτικές λύσεις. Η αντιμετώπιση λοιπόν των μεγάλων υπολογιστικών απαιτήσεων των ΕΑ μπορεί να επιτευχθεί με τους ακόλουθους τρόπους:

- Χρησιμοποίηση ειδικών τελεστών, προσαρμοσμένων στο πρόβλημα, που επιταχύνουν τη σύγκλιση.
- Χρησιμοποίηση Υβριδικών Αλγορίθμων [4], [5].
- Χρησιμοποίηση προσεγγιστικών μοντέλων για την συνάρτηση προσαρμογής (surrogate models) [6].
- Παράλληλα μοντέλα Εξελικτικών Αλγορίθμων [7].

Παρόλο που πολλές ομάδες ΕΑ, όπως είναι οι Γενετικοί Αλγόριθμοι (ΓΑ) και οι Εξελικτικές Στρατηγικές (ΕΣ), χρησιμοποιούν αριθμητικές τιμές για την περιγραφή των υποψήφιων λύσεων, πολλοί άλλοι τύποι ΕΑ, όπως είναι ο Γενετικός και ο Εξελικτικός Προγραμματισμός (ΓΠ και ΕΠ) χρησιμοποιούν άτομα τα οποία έχουν περίπλοκη εσωτερική δομή δεδομένων, ώστε να μπορούν να αναπαραστήσουν το πρόβλημα με έναν πιο αποδοτικό τρόπο (π.χ. συμβολικά δέντρα). Το γεγονός αυτό καθιστά την αξιολόγηση του κάθε ατόμου αρκετά πολύπλοκή. Εν γένει, ο υπολογισμός της συνάρτησης προσαρμογής για την αξιολόγηση κάθε ατόμου του πληθυσμού είναι τις περισσότερες φορές η πιο χρονοβόρα διαδικασία ενός ΕΑ. Για το λόγο αυτό υπάρχει διαρκής έρευνα για το σχεδιασμό αποτελεσματικών και χρονικά αποδοτικών ΕΑ. Στον τομέα που αφορά τα παράλληλα μοντέλα ΕΑ υπάρχει ένας μεγάλος αριθμός εφαρμογών και αλγορίθμων [7].

Οι ΕΑ από τη φύση τους επιδέχονται εύκολα μετατροπή ώστε η επεξεργασία να εκτελείται παράλληλα. Ο λόγος είναι ότι διαθέτουν σε κάθε γενιά πληθυσμό λύσεων,

οι οποίες αντί να αξιολογούνται σειριακά, μπορούν να αξιολογούνται παράλληλα. Υπάρχει όμως και πιο απλή εφαρμογή της παράλληλης επεξεργασίας στους ΕΑ. Επειδή είναι στοχαστικοί αλγόριθμοι, η κάθε λύση εξαρτάται από την εκάστοτε εκτέλεση του προγράμματος, οπότε για διαφορετικές εκτελέσεις προκύπτουν και διαφορετικές λύσεις. Για να εξαχθούν ασφαλή συμπεράσματα απαιτούνται αρκετές διαφορετικές εκτελέσεις, ώστε να υπάρχει ένα στατιστικά ικανό δείγμα. Οι εκτελέσεις αυτές μπορούν να γίνονται παράλληλα σε διαφορετικούς Η/Υ [7].

Η επιτάχυνση όμως του αλγορίθμου δεν αποτελεί τη μόνη πρόκληση για τους ΠΕΑ. Χρησιμοποιώντας κάποιον ΠΕΑ συχνά οδηγούμαστε όχι μόνο σε γρηγορότερο αλγόριθμο, αλλά και σε βελτίωση της αριθμητικής του απόδοσης. Μια πραγματικά ενδιαφέρουσα παρατήρηση είναι ότι η χρησιμοποίηση δομημένου πληθυσμού, με χωρική κατανομή των ατόμων, είτε στη μορφή μιας ομάδας νησιών (island model), είτε ενός πλέγματος διάχυσης (diffusion grid), έχει ως αποτέλεσμα τη γενική βελτίωση του αλγορίθμου. Συνεπώς πολλοί χρήστες επιλέγουν μοντέλα δομημένου πληθυσμού, χωρίς να χρησιμοποιούν μηχανήματα παράλληλης επεξεργασίας, επιτυγχάνοντας και πάλι καλύτερα αποτελέσματα σε σχέση με τους παραδοσιακούς σειριακούς ΕΑ [7].

Στις κλασσικές μελέτες πάνω στους ΠΕΑ, γίνεται η υπόθεση ότι το εκάστοτε μοντέλο αναπαρίσταται απ' ευθείας πάνω στο παράλληλο υπολογιστικό υλικό, επομένως δεν υπάρχει κάποιος διαχωρισμός ανάμεσα στο μοντέλο και στο μέσο που χρησιμοποιείται για την υλοποίησή του. Παρόλα αυτά από τη στιγμή που θα σχεδιαστεί ένα δομημένο μοντέλο πληθυσμού, αυτό μπορεί να εφαρμοστεί σε οποιοδήποτε μονό επεξεργαστή ή μηχάνημα παράλληλης αρχιτεκτονικής [7].

1.2 Ιστορική Αναδρομή Εξελικτικών και Παράλληλων Εξελικτικών Αλγορίθμων

Οι Εξελικτικοί Αλγόριθμοι αποτελούν μια κατηγορία μεθόδων βελτιστοποίησης που βασίζουν τη λειτουργία τους στη μίμηση των διαδικασιών της φυσικής εξέλιξης. Ανάγουν τη γέννησή τους στα τέλη της δεκαετίας του 1950, με την παρουσίαση εργασιών από διαφορετικούς ερευνητές [8], [9], [10], [11]. Όμως, για περίπου τρεις δεκαετίες παρέμεναν στην αφάνεια και η διεθνής επιστημονική κοινότητα αγνοούσε την ύπαρξή τους. Ο λόγος ήταν η απουσία ικανών Η/Υ, που θα μπορούσαν να εκμεταλλευτούν τις δυνατότητες των ΕΑ, αλλά και τα προβλήματα και οι ατέλειες των πρώτων προσεγγίσεων στο θέμα. Το τοπίο άρχισε να αλλάζει μετά τη δεκαετία του 1960, όταν παρουσιάστηκαν εργασίες όπως αυτές των Holland [12], Rechenberg [13], Schwefel [14] και Fogel [15]. Οι παραπάνω εργασίες ακολουθούσαν διαφορετικές προσεγγίσεις στο θέμα, είχαν όμως ως κοινό παρονομαστή τη χρησιμοποίηση ενός πληθυσμού πιθανών λύσεων, οι οποίες εξελισσόμενες με τη χρήση τεχνικών δανεισμένων από τη γενετική, κατέληγαν σε (σχεδόν) βέλτιστες λύσεις. Οι ΕΑ δε χρησιμοποιήθηκαν από όλους τους ερευνητές μόνο ως εργαλεία βελτιστοποίησης, αλλά και ως μέθοδοι προσομοίωσης της φυσικής διαδικασίας της εξέλιξης, αλλά και γενικότερα της συμπεριφοράς ζωντανών οργανισμών [16].

Οι ΕΑ εμφανίζονται σε τρεις διαφορετικές μορφές, οι οποίες ακολουθούν διακριτή πορεία, αλλά με ισχυρές αλληλεπιδράσεις μεταξύ τους: τους Γενετικούς Αλγορίθμους, τον Εξελικτικό Προγραμματισμό και τις Εξελικτικές Στρατηγικές. Ως παρακλάδι των Γενετικών Αλγορίθμων εξελίχθηκε πρόσφατα ο Γενετικός Προγραμματισμός [16].

Οι ΓΑ εφευρέθηκαν από τον John Holland [12] κατά τη δεκαετία του ’60 και αναπτύχθηκαν στην αρχή από τον ίδιο και τους φοιτητές του στο πανεπιστήμιο του Michigan την δεκαετία 1960-1970. Σε αντίθεση με τις υπόλοιπες τεχνικές, ο Holland είχε ως στόχο, όχι το σχεδιασμό αλγορίθμων που να επιλύουν συγκεκριμένα προβλήματα, αλλά να εξετάσει κατά τρόπο γενικό το φαινόμενο της προσαρμογής, όπως αυτό παρατηρείται στη φύση και να αναπτύξει τρόπους, ώστε οι μηχανισμοί της φυσικής προσαρμογής να προσαρμοσθούν σε υπολογιστικά συστήματα. Ο Holland [17] παρουσιάζει τους ΓΑ ως μια αφηρημένη έννοια, που πηγάζει από τη βιολογική εξέλιξη και εκθέτει το θεωρητικό περίγραμμα της προσαρμογής των ΓΑ. Τελικά όμως η κύρια εφαρμογή τους αποδείχθηκε η βελτιστοποίηση προβλημάτων. Λόγω του αρχικού σκοπού για τον οποίο αναπτύχθηκαν, προσομοιάζουν περισσότερο από τις υπόλοιπες τεχνικές στη φυσική γενετική διαδικασία [16].

Ο Holland με την εργασία του, πρώτος εισήγαγε έναν αλγόριθμο, ο οποίος βασίζεται σε πληθυσμό πιθανών λύσεων, οι οποίες υπόκεινται σε τελεστές που αντιστοιχούν σε φυσικές διεργασίες όπως η μετάλλαξη και η διασταύρωση. Ο Rechenberg χρησιμοποιούσε «πληθυσμό» αποτελούμενο από δύο μόνο λύσεις, πατέρα και υιό, με τη δεύτερη να αποτελεί μεταλλαγμένη μορφή της πρώτης. Επίσης, ο Holland ήταν ο πρώτος που παρουσίασε ένα θεωρητικό υπόβαθρο για την επεξήγηση της λειτουργίας των ΓΑ [16].

Ο ΕΠ αναπτύχθηκε από τον Fogel [16] με σκοπό να διερευνήσει τη δυνατότητα εξέλιξης της τεχνητής νοημοσύνης με την έννοια η μηχανή να μπορεί να προβλέπει αλλαγές στο περιβάλλον της και να αντιδρά κατάλληλα. Οι εξελισσόμενοι πληθυσμοί είναι Μηχανές Πεπερασμένης Κατάστασης (Finite State Machines), η δομή των οποίων μπορεί να μεταβληθεί μέσα από την εξελικτική διαδικασία, με σκοπό να καταστούν ικανές να προβλέπουν γεγονότα με βάση την εκπαίδευση σε προηγούμενες παρατηρήσεις. Μία ΜΠΚ έχει την ικανότητα να μετατρέπει μια σειρά συμβόλων εισόδου σε μία σειρά συμβόλων εξόδου, με βάση μία πεπερασμένη λίστα καταστάσεων και μία πεπερασμένη λίστα κανόνων. Η ιδέα ήταν οι μηχανές να εξελίσσονται μαζί με το περιβάλλον τους, ώστε να προσαρμόζονται σε αυτό. Η αξιολόγηση της ΜΠΚ σε σχέση με το περιβάλλον μετράται με βάση την ποιότητα πρόβλεψης που επιτυγχάνει. Κάθε σύμβολο εξόδου συγκρίνεται με το επόμενο σύμβολο εισόδου και η ποιότητα της πρόβλεψης ποσοτικοποιείται με τη βοήθεια κάποιας συνάρτησης [16].

Το 1965 ο Ingo Rechenberg, φοιτητής ακόμη, εισήγαγε τις ΕΣ ως μεθόδους βελτιστοποίησης των παραμέτρων αεροδυναμικών σωμάτων κατά τη διάρκεια πειραμάτων σε αεροδυναμική σήραγγα [14], [18]. Η ιδέα του Rechenberg εξελίχθηκε περαιτέρω από τον Schwefel [19], [20]. Οι ΕΣ χρησιμοποιούν κωδικοποίηση πραγματικών αριθμών και μόνο τη μετάλλαξη ως τεχνική διαφοροποίησης των χρωμοσωμάτων, ενώ προσομοιάζουν με τεχνικές ανάβασης. Ο χώρος των ΕΣ παρέμεινε ένας ενεργός χώρος έρευνας και αναπτύχθηκε ανεξάρτητα από αυτόν των ΓΑ, αν και πρόσφατα έχουν αρχίσει να αλληλεπιδρούν μεταξύ τους [16].

Ο ΓΠ αναπτύχθηκε πρόσφατα από τον Koza [21], ο οποίος προτείνει αντί να λύνεται ένα πρόβλημα με εξελικτικές μεθόδους, είναι καλύτερα να γίνεται εξερεύνηση των πιθανών διαφορετικών προγραμμάτων που μπορούν να λύσουν το πρόβλημα. Έτσι, στο ΓΠ οι διακριτές λύσεις που ανταγωνίζονται μεταξύ τους είναι κάποιο είδος προγραμμάτων (που περιέχουν τόσο δομές δεδομένων όσο και συναρτήσεις). Λόγω

του παραπάνω χαρακτηριστικού διαφέρουν αρκετά από τους τυπικούς ΓΑ και ως εκ τούτου μπορούν να θεωρηθούν ως ξεχωριστή κατηγορία ΕΑ [16].

Εκτός από τις παραπάνω γενικές κατηγορίες ΕΑ, υπάρχει μία πληθώρα τροποποιήσεων και υποκατηγοριών, ανάλογα με το πρόβλημα που αντιμετωπίζεται. Η ανοικτή μορφή των ΕΑ επιτρέπει την ελεύθερη τροποποίηση των επί μέρους λειτουργιών τους, κάτι που περιορίζεται μόνο από τη φαντασία του εκάστοτε ερευνητή.

Κατά τη δεκαετία του 1980, με την ανάπτυξη των Η/Υ οι ΕΑ βρήκαν μεγάλη απήχηση και χρησιμοποιήθηκαν για την επίλυση δύσκολων προβλημάτων βελτιστοποίησης. Η εγγενής παράλληλη και κατανεμημένη φύση των ΕΑ δεν διέφυγε της προσοχής των διαφόρων ερευνητών. Ο Holland [1] έκανε τα πρώτα βήματα το 1975 με τον καθορισμό μιας παράλληλης υπολογιστικής αρχιτεκτονικής για αναπαραγωγικές διαδικασίες. Στην ουσία όμως οι πρώτες ιδέες για χρησιμοποίηση πολλαπλών ανταγωνιστικών υποπληθυσμών δόθηκαν από τον Bossert [22] το 1967, ο οποίος πρότεινε την τεχνική αυτή για την αύξηση της διαφοροποίησης του πληθυσμού και την πρόληψη της πρόωρης σύγκλισης σε τοπικά βέλτιστα. Βέβαια παρόλο που οι βασικές ιδέες έγιναν γρήγορα κατανοητές, η τεχνολογία που θα υποστήριζε παράλληλες και κατανεμημένες υπολογιστικές τεχνικές στο χώρο των Η/Υ ήταν στα σπάργανα τη δεκαετία του ’60. Έπρεπε να φτάσει η δεκαετία του ’80 για να υπάρξουν οι πρώτες εφαρμογές ΠΕΑ.

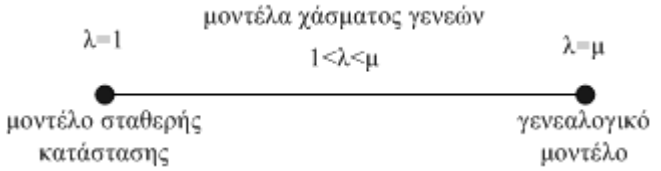
Ο Grefenstette [23] το 1981, είναι από τους πρώτους που εξετάζουν διάφορα ζητήματα που συνδέονται με την παράλληλη εφαρμογή ΓΑ. Επίσης ο Grosso [24] το 1985 είναι από τους πρώτους που εφαρμόζει τεχνικές χωρικού διαχωρισμού του πληθυσμού. Ακολούθησαν οι Tanese [25] και Cohoon [26], οι οποίοι με περισσότερο συστηματικές μελέτες το 1987 προτείνουν την παράλληλη αρχιτεκτονική του υπερκύβου, στην οποία υποπληθυσμοί τοποθετούνται στους κόμβους ενός υπερκύβου. Εξίσου σημαντική ήταν και η δουλειά των Petttey και Leuze [27] το 1989, οι οποίοι ήταν οι πρώτοι που προσπάθησαν να δώσουν μια θεωρητική βάση στη δυναμική των παράλληλων ΓΑ. Στις ΕΣ, ο Rudolph [28] εφάρμοσε ένα από τα πρώτα κατανεμημένα μοντέλα, ενώ στον ΕΠ, ορόσημο αποτέλεσε ο Duncan [29].

Όλες οι προηγούμενες μελέτες αναφέρονται σε μοντέλα, τα οποία ονομάστηκαν αραιής διάταξης (coarse grain model) ή μοντέλα νήσων (island model). Ένα διαφορετικό μοντέλο χωρικής κατανομής εισάχθηκε με την εργασία του Gorges-Schleuter [30] το 1989, το οποίο ονομάζεται μοντέλο πυκνής διάταξης (fine grained model), κυψελοειδές μοντέλο (cellular model) ή μοντέλο διάχυσης (diffusion model). Βασίζεται σε ένα χωρικά κατανεμημένο πληθυσμό, στον οποίο οι γενετικές αλληλεπιδράσεις μπορούν να λάβουν χώρα στη γειτονική περιοχή γύρω από κάθε άτομο. Παρεμφερής ερευνητική εργασία πραγματοποιήθηκε την ίδια περίοδο και από τους Manderick και Spiessen [31].

Εν γένει οι ΠΕΑ χωρίζονται σε δύο μεγάλες κατηγορίες, τους πανμικτικούς ΕΑ, όπου ο πληθυσμός είναι ενιαίος, και τους δομημένους ΕΑ, όπου ο πληθυσμός κατανέμεται χωρικά. Και οι δύο αυτές κατηγορίες μελετώνται αναλυτικά στη συνέχεια.

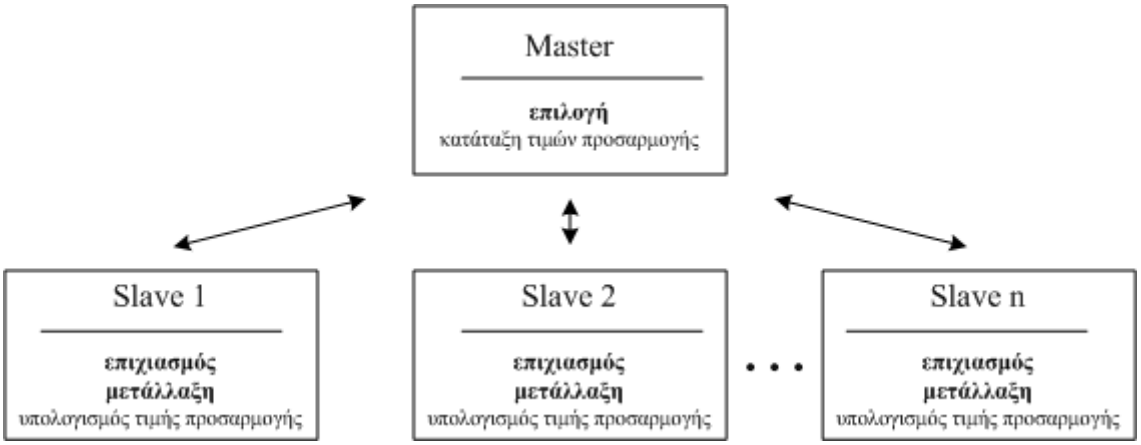
1.3 Πανμικτικοί Εξελικτικοί Αλγόριθμοι (Panmictic EAs)

Οι πανμικτικοί ΕΑ αποτελούν την κλασσική περίπτωση ΕΑ όπου ο πληθυσμός είναι ενιαίος και η διαδικασία της επιλογής πραγματοποιείται σε ολόκληρο τον πληθυσμό. Υπάρχουν δύο δημοφιλής κατηγορίες πανμικτικών ΕΑ, οι οποίοι διαφοροποιούνται κατά την αναπαραγωγική διαδικασία [32]. Το πρώτο καλείται γενεαλογικό μοντέλο (generational model) στο οποίο ένας καινούριος πληθυσμός από λ άτομα αντικαθιστά τον παλιό (***Εικόνα 1.1***: δεξί μέρος του σχήματος όπου μ είναι το μέγεθος του πληθυσμού). Ο δεύτερος τύπος καλείται σταθερής κατάστασης (steady state), όπου συνήθως ένα ή δύο νέα άτομα δημιουργούνται σε κάθε βήμα του αλγορίθμου και εισέρχονται στον πληθυσμό, και συνεπώς μπορούν να συνυπάρξουν με τους γονείς τους. Στην ενδιάμεση περιοχή υπάρχει πληθώρα μοντέλων επιλογής τα οποία γενικώς ονομάζονται αλγόριθμοι χάσματος γενεών (generation gap algorithms), στους οποίους ένα δεδομένο ποσοστό των ατόμων αντικαθίσταται με ένα νέο (***Εικόνα 1.1***). Προφανώς, τα σταθερής κατάστασης και γενεαλογικά μοντέλα επιλογής αποτελούν δύο ειδικές υποκατηγορίες των αλγορίθμων χάσματος γενεών [7].



Εικόνα 1.1: Μοντέλα Πανμικτικών Εξελικτικών Αλγορίθμων [7].

Σε αυτό τον τύπο ΕΑ μπορεί να εφαρμοστεί παραλληλισμός με τη χρήση του μοντέλου ολικού παραλληλισμού (global parallelism). Σύμφωνα με αυτό, τα άτομα του πληθυσμού αξιολογούνται παράλληλα σε διαφορετικούς επεξεργαστές, ενώ η διαδικασία της επιλογής πραγματοποιείται σειριακά στον κεντρικό Η/Υ, που ελέγχει τη συνολική διαδικασία του ΕΑ [33]. Το μοντέλο αυτό συνδυάζεται συνήθως με αρχιτεκτονική δικτύου «Master-Slave» από όπου παίρνει και το όνομα του (***Εικόνα 1.2***). Ο αλγόριθμος είναι ακριβώς ο ίδιος όπως και στη σειριακή εφαρμογή, μόνο που είναι ταχύτερος, ειδικά σε περιπτώσεις που η αξιολόγηση της κάθε λύσης είναι ιδιαίτερα χρονοβόρα. Μπορεί να υπάρχει ένας επεξεργαστής για κάθε άτομο του πληθυσμού, ή σε κάθε επεξεργαστή να αντιστοιχούν περισσότερα άτομα του πληθυσμού, μοιρασμένα με τρόπο ώστε να επιτυγχάνεται ομοιόμορφος φόρτος.



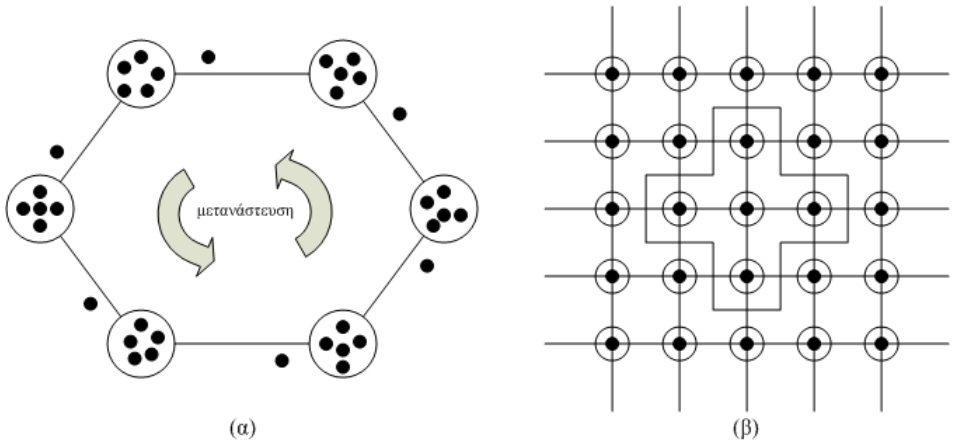
Εικόνα 1.2: Αρχιτεκτονική δικτύου τύπου «Master-Slave».

Με βάση λοιπόν τη μέθοδο του ολικού παραλληλισμού διατηρείται ένας απλός πληθυσμός και οι υπολογισμοί των ατόμων και οι εφαρμογές των γενετικών τελεστών εφαρμόζονται παράλληλα. Ένας απλός τρόπος για να γίνει αυτό είναι να δημιουργείται η επόμενη γενιά από την προηγούμενη σε παράλληλο βρόχο. Τα περισσότερα βήματα του βρόχου (υπολογισμός, επιχιασμός, μετάλλαξη) μπορούν να εκτελεστούν παράλληλα. Ένας απλός «Master» επεξεργαστής επιβλέπει τον συνολικό πληθυσμό και κάνει την επιλογή. Οι «Slave» επεξεργαστές λαμβάνουν τα άτομα που είναι επανασυνδιασμένα για να φτιάξουν την νέα γενιά. Αυτή η νέα γενιά πρέπει να αξιολογηθεί πριν επιστρέψει στον «Master» επεξεργαστή [34].

1.4 Δομημένοι Εξελικτικοί Αλγόριθμοι (Structured EAs)

Υπάρχει μια μεγάλη παράδοση στη χρησιμοποίηση δομημένων πληθυσμών σε εξελικτικές εφαρμογές και πιο ειδικά σε παράλληλες μεθόδους. Οι πιο διαδεδομένοι τύποι δομημένων ΕΑ είναι οι κατανεμημένοι ΕΑ (distributed EAs) και οι κυψελοειδείς ΕΑ (cellular EAs) και χρησιμοποιούνται ευρέως σε διαδικασίες βελτιστοποίησης [35].

Η διαδικασία δόμησης ενός απλού πληθυσμού μπορεί να επιτευχθεί καταμερίζοντας τον σε διάφορους υποπληθυσμούς, όπου μπορούν να τοποθετηθούν είτε σε νησιά για την εκτέλεση σποραδικών ανταλλαγών ατόμων (***Εικόνα 1.3(α)***), είτε να έχουν τη μορφή γειτονιών (***Εικόνα 1.3(β)***).



Εικόνα 1.3: Μοντέλα δομημένων ΕΑ. (α) Κατανεμημένος ΕΑ. (β) Κυψελοειδής ΕΑ [7].

Οι κατανεμημένοι ΕΑ είναι εν γένει σύνθετοι αλγόριθμοι. Στηρίζονται σε διάφορους υποπληθυσμούς που ανταλλάσουν άτομα περιοδικά. Αυτή η ανταλλαγή ατόμων καλείται μετανάστευση και ελέγχεται από διάφορες παραμέτρους. Αυτού του τύπου οι ΕΑ είναι πολύ δημοφιλείς αλλά είναι και πιο δύσκολο να ελεγχθούν λόγω της διαδικασίας της μετανάστευσης. Συγκεκριμένα, πρέπει να καθοριστούν ο αριθμός και το μέγεθος των υποπληθυσμών, η συχνότητα μετανάστευσης, ο αριθμός και ο προορισμός των μεταναστών και η μέθοδος που χρησιμοποιείται για την επιλογή των ατόμων που θα μεταναστεύσουν [7].

Οι κατανεμημένοι ΕΑ είναι γνωστοί με διαφορετικά ονόματα. Το όνομα κατανεμημένοι ΕΑ υιοθετήθηκε επειδή συχνά εφαρμόζονται σε υπολογιστές MIMD (βλ. §2.1) με κατανεμημένη μνήμη. Συχνά αναφέρονται και ως πολλαπλού πληθυσμού επειδή στηρίζονται στην ταυτόχρονη εξέλιξη πολλών πληθυσμών.

Δεδομένου ότι η αναλογία υπολογισμών - επικοινωνίας είναι συνήθως αρκετά υψηλή αναφέρονται και ως αραιής διάταξης ΕΑ. Επίσης, οι ΕΑ πολλαπλών πληθυσμών προσομοιάζουν το μοντέλο νήσων, θεωρώντας διαχωρισμό του πληθυσμού σε γεωγραφικά κατανεμημένους υποπληθυσμούς με μετανάστευση μεταξύ των πληθυσμών. Εάν η μετανάστευση είναι περιορισμένη μεταξύ των γειτονικών πληθυσμών το μοντέλο πληθυσμού ονομάζεται «stepping stone» [34].

Γενικά παρατηρείται ότι ένα κατανεμημένο μοντέλο, είτε αυτό τρέχει σε πολλαπλούς επεξεργαστές, είτε όχι, είναι συνήθως γρηγορότερο από έναν πανμικτικό ΕΑ. Βασικός λόγος για αυτό είναι ότι ο χρόνος εκτέλεσης και ο αριθμός των υπολογισμών μειώνονται σημαντικά χάρη στις ξεχωριστές αναζητήσεις σε διάφορες περιοχές του χώρου λύσεων. Επίσης είναι περισσότερο ελκυστικοί σε σχέση με τους σειριακούς ΕΑ, γιατί χρησιμοποιούν μικρότερου μεγέθους πληθυσμούς και συνεπώς αναμένεται να συγκλίνουν ταχύτερα αν τους αναθέσουμε σε διαφορετικούς επεξεργαστές. Ωστόσο, όταν συγκρίνουμε τη συμπεριφορά των σειριακών και των παράλληλων αλγορίθμων, πρέπει να λάβουμε υπόψη και την ποιότητα των λύσεων που βρίσκονται σε κάθε περίπτωση. Ενώ είναι αλήθεια ότι μικρότεροι υποπληθυσμοί συγκλίνουν ταχύτερα είναι επίσης γεγονός ότι η ποιότητα των λύσεων μπορεί να είναι φτωχή [7].

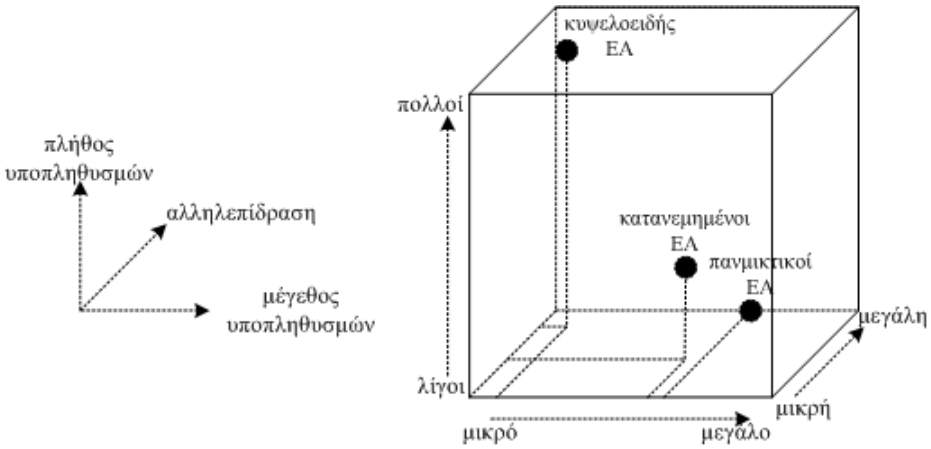
Στους κατανεμημένους ΕΑ, όπως αναφέρθηκε, απαιτούνται προαιρετικές παράμετροι, οι οποίες ελέγχουν τη μετανάστευση και το πώς οι μετανάστες επιλέγονται/ενσωματώνονται από/προς το νησί πηγή/προορισμός [36], [37]. Στους κυψελοειδείς ΕΑ η ύπαρξη επικαλυπτόμενων μικρών γειτονιών βοηθάει στην εξερεύνηση του χώρου αναζήτησης. Αυτοί οι δύο τύποι ΕΑ παρέχουν γενικά καλύτερη περιγραφή του χώρου αναζήτησης και βελτιώνουν στις περισσότερες περιπτώσεις την αριθμητική συμπεριφορά καθώς και την ταχύτητα του βασικού αλγορίθμου [7].

Οι κυψελοειδείς ΕΑ, αναφέρονται έτσι επειδή είναι μια κλάση αυτόματων κυττάρων με στοχαστικούς κανόνες γενετικής μεταλλαγής, και είναι επίσης γνωστοί ως πυκνής διάταξης αλγόριθμοι. Αυτή η κλάση των ΠΕΑ καλείται και μοντέλο διάχυσης, επειδή η διάδοση των καλών ιδιοτήτων στον πληθυσμό είναι ανάλογη της τυχαίας διάχυσης των μορίων σε κάποια ρευστά. Αποτελούνται από απλό χωρικά δομημένο πληθυσμό. Η δομή του πληθυσμού είναι συνήθως ορθογώνιο πλέγμα δύο διαστάσεων και υπάρχει ένα άτομο για κάθε κόμβο του πλέγματος. Η ιδανική περίπτωση είναι να υπάρχει ένας επεξεργαστής για κάθε άτομο, έτσι ώστε ο υπολογισμός της προσαρμογής να εκτελείται ταυτόχρονα για όλα τα άτομα. Η επιλογή και το ζευγάρωμα περιορίζονται σε μια μικρή περιοχή γύρω από το άτομο. Οι περιοχές αυτές επικαλύπτονται έτσι ώστε τελικά οι καλές ιδιότητες των ατόμων που υπερέχουν να μπορούν να διαδοθούν σε όλο τον πληθυσμό. Οι κυψελοειδείς ΕΑ εφαρμόζονται καλά σε μαζικούς παράλληλους SIMD υπολογιστές (βλ. §2.1), όπου εκτελείται η ίδια απλή εντολή σε όλους τους επεξεργαστές, αλλά είναι επίσης πιθανό να εφαρμοστούν αποτελεσματικά σε αραιής διάταξης MIMD [38] υπολογιστές (βλ. §2.1) [7].

Η συμπεριφορά των ΕΑ με χωρικά κατανεμημένους πληθυσμούς είναι ενδιαφέρουσα, ανεξάρτητα αν εφαρμόζονται σε σειριακούς ή παράλληλους υπολογιστές. Έχοντας γεωγραφικά κατανεμημένο πληθυσμό μπορούν να παρέχουν κάποια πλεονεκτήματα που είναι ανεξάρτητα από την χρήση πολλαπλών επεξεργαστών.

Η κύρια διαφορά των κυψελοειδών ΕΑ σε σχέση με τους πανμικτικούς έγκειται στο γεγονός της δόμησης του πληθυσμού και της διασποράς. Στους κυψελοειδείς ΕΑ ο αναπαραγωγικός κύκλος πραγματοποιείται μέσα σε κάθε μια από τις ομάδες ατόμων. Κάθε άτομο έχει τη δική του ομάδα από υποψήφια άτομα προς διασταύρωση, που καθορίζονται από γειτονικά άτομα, ενώ ταυτόχρονα ένα άτομο μπορεί να ανήκει σε πολλές ομάδες. Στους πανμικτικούς ΕΑ η επιλογή των ατόμων γίνεται ανάμεσα στα άτομα ολόκληρου του πληθυσμού [7].

Είναι σημαντικό επίσης να αναφερθεί ότι οι πανμικτικοί ΕΑ δεν επηρεάζουν γενικά τη συμπεριφορά του αλγορίθμου. Οι διάφοροι τύποι των δομημένων ΕΑ που αναφέρθηκαν αλλάζουν τον τρόπο με τον οποίο λειτουργούν οι ΕΑ. Επίσης στην πρώτη κατηγορία μπορούν να ζευγαρώσουν οποιαδήποτε δύο άτομα του πληθυσμού, ενώ στις άλλες δύο περιορίζονται στα άτομα που είναι γεωγραφικά κοντά.



Εικόνα 1.4: Κύβος τρισδιάστατης απεικόνισης ΕΑ [7].

Στην ***Εικόνα 1.4*** παρατηρείται μια τρισδιάστατη απεικόνιση των διαφόρων τύπων των ΕΑ βασισμένη στον αριθμό των υποπληθυσμών, το πλήθος των ατόμων σε κάθε έναν και το βαθμό αλληλεπίδρασης ανάμεσά τους.

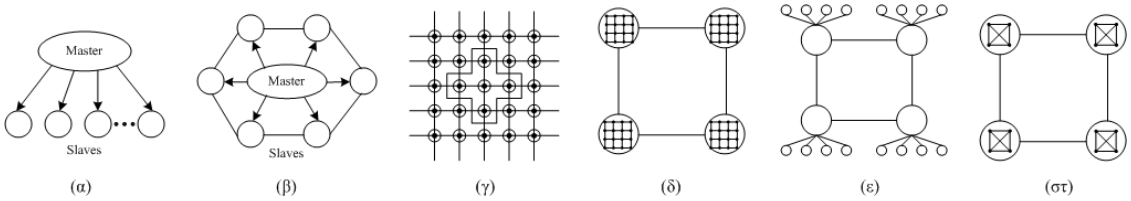
Παρόλο που οι κατανεμημένοι και οι κυψελοειδείς ΕΑ αποτελούν μοντέλα δομημένων ΕΑ παρουσιάζουν αρκετές διαφορές μεταξύ τους. Ένας κυψελοειδής ΕΑ αποτελείται από πολλούς υποπληθυσμούς που όμως συνήθως περιέχουν ένα μόνο άτομο, σε αντίθεση με τους κατανεμημένους ΕΑ που ο πληθυσμός κατανέμεται σε λιγότερους υποπληθυσμούς αποτελούμενους όμως από περισσότερα άτομα. Σε έναν κατανεμημένο ΕΑ οι υποπληθυσμοί συνδέονται χαλαρά, η αλληλεπίδραση δηλαδή μεταξύ τους είναι μικρή, ενώ σε έναν κυψελοειδή ΕΑ συνδέονται πιο σφικτά. Ο κύβος της ***Εικόνας 1.4*** παρουσιάζει ένα γενικότερο τρόπο κατάταξη δομημένων ΕΑ συμπεριλαμβάνοντας και τους πανμικτικούς ΕΑ για την καλύτερη κατανόηση της συμπεριφοράς τους [7].

1.5 Υβριδικά Μοντέλα Παράλληλων Εξελικτικών Αλγορίθμων

Ο συνδυασμός δύο διαφορετικών μοντέλων (ή ακόμα και ίδιων) ΠΕΑ μπορεί να οδηγήσει σε αλγόριθμους με πολύ περισσότερες δυνατότητες, καθώς ένας τέτοιος συνδυασμός εκμεταλλεύεται τα πλεονεκτήματα των δύο διαφορετικών αλγορίθμων. Αυτού του τύπου οι ΠΕΑ ονομάζονται Υβριδικά μοντέλα και βρίσκουν πλήθος εφαρμογών, καθώς υπάρχει μεγάλος αριθμός δυνατών συνδυασμών και μπορούν να

προσαρμόζονται στον εκάστοτε τύπο προβλήματος. Ένα από τα γνωστότερα Υβριδικά μοντέλα είναι τα Ιεραρχικά Υβρίδια [39].

Αυτό το μοντέλο παράλληλων αλγορίθμων συνδυάζει κατανεμημένους ΕΑ με πανμικτικούς ή κυψελοειδείς ΕΑ. Ονομάζονται Ιεραρχικά Υβρίδια ΠΕΑ επειδή στα υψηλότερα επίπεδα χρησιμοποιούν πολλαπλού πληθυσμού αλγορίθμους, ενώ στα κατώτερα απλού πληθυσμού. Ένας Ιεραρχικός ΠΕΑ συνδυάζει τα πλεονεκτήματα των στοιχείων του και συμπεριφέρεται καλύτερα από ότι αυτά μόνα τους. Οι Ιεραρχικοί αλγόριθμοι συμπεριφέροντε καλά σε εφαρμογές σε συμμετρικούς πολύ-επεξεργαστές [34].



Εικόνα 1.5: Διαφορετικά μοντέλα ΠΕΑ και υβριδικά μοντέλα σε δύο επίπεδα. (α) Μοντέλο ολικής παραλληλοποίησης. (β) Μοντέλο αραιής διάταξης. (γ) Μοντέλο πυκνής διάταξης. (δ) Υβριδικό μοντέλο αραιής και πυκνής διάταξης. (ε) Υβριδικό μοντέλο αραιής διάταξης και ολικής [7].

Μερικές από τις πιο γνωστές παράλληλες εφαρμογές παρουσιάζονται στην ***Εικόνα 1.5***. Το μοντέλο ολικής παραλληλοποίησης (global parallelization) (***Εικόνα 1.5 (α)***) δείχνει μια πανμικτικού τύπου εξέλιξη με υπολογισμούς που εκτελούνται παράλληλα. Στην ***Εικόνα 1.5 (β)*** παρουσιάζεται το μοντέλο νησιών με πολλαπλούς πληθυσμούς, στο οποίο διάφοροι υποαλγόριθμοι ΕΑ τρέχουν παράλληλα και χαρακτηρίζεται και ως αραιής διάταξης (coarse grained). Ο πυκνής διάταξης (fine grained) ή κυψελοειδής ΕΑ παρουσιάζεται στην ***Εικόνα 2.2 (γ)***. Στην ***Εικόνα 2.2 (δ) (ε)*** και ***(στ)*** φαίνονται υβριδικοί αλγόριθμοι, οι οποίοι υλοποιούν προσέγγιση παράλληλης εφαρμογής σε δύο επίπεδα [7].

1.6 Σύγχρονοι και Ασύγχρονοι Παράλληλοι Εξελικτικοί Αλγόριθμοι

Στους πανμικτικούς ΕΑ η εφαρμογή της διαδικασίας συνήθως είναι σύγχρονη (synchronous application) όμως μπορεί να υπάρξει και ασύγχρονη εφαρμογή (asynchronous application) που σε αρκετές περιπτώσεις επιφέρει καλύτερα αποτελέσματα. Ο αλγόριθμος είναι σύγχρονος, όταν ο «Master» επεξεργαστής περιμένει να λάβει τις τιμές της συνάρτησης προσαρμογής για όλα τα άτομα του πληθυσμού πριν επεξεργαστεί την επόμενη γενιά. Για να εκτελεστεί επομένως το σειριακό τμήμα του αλγορίθμου όπου εφαρμόζεται ο τελεστής της επιλογής, θα πρέπει να ολοκληρωθούν όλες οι αξιολογήσεις. Παρόλο που οι περισσότερες εφαρμογές ΠΕΑ είναι σύγχρονες, είναι πιθανό να εφαρμοστούν και ασύγχρονοι ΠΕΑ, όπου ο κεντρικός Η/Υ εκτελεί το σειριακό τμήμα του αλγορίθμου περιοδικά, εφαρμόζοντας τη διαδικασία επιλογής στα εκάστοτε άτομα που υπάρχουν και μπορεί να ανήκουν σε διαφορετικές γενιές [40],[41].

Εν γένει υπάρχουν σύγχρονες και ασύγχρονες μέθοδοι για κάθε τύπο παράλληλων αλγορίθμων. Στους σύγχρονους αλγορίθμους οι πολύ-επεξεργαστές έχουν να αντιμετωπίσουν πάντα άτομα τις ίδιας γενιάς και υπάρχει μια μορφή επικοινωνίας για το συγχρονισμό της διαδικασίας. Αυτό σημαίνει ότι ο γρηγορότερος επεξεργαστής

πρέπει να περιμένει τον πιο αργό να τελειώσει για να συνεχίσει με την επόμενη γενιά. Στην περίπτωση των ασύγχρονων αλγορίθμων η συμπεριφορά είναι διαφορετική και είναι δύσκολο να εκτιμηθούν ή να επαναληφθούν τα ίδια αποτελέσματα επειδή η επικοινωνία συμβαίνει σε τυχαίες στιγμές.

1.7 Εναλλακτική Ταξινόμηση Παράλληλων Εξελικτικών Αλγορίθμων

Σε αυτό το σημείο παρουσιάζεται μια δομημένη ταξινόμηση των ΠΕΑ από τρεις βασικές οπτικές γωνίες: βάσει μοντέλου, τύπου και εφαρμογής [7].

(1) Βάσει μοντέλου

Αυτή η κατηγορία προσομοιάζει τον διαχωρισμό που έγινε παραπάνω χωρίζοντας τους ΠΕΑ σε αραιής διάταξης (κατανεμημένοι ΕΑ) και πυκνής διάταξης ΠΕΑ (κυψελοειδείς ΕΑ) [7].

Αραιής διάταξης ΠΕΑ
Είναι η πιο δημοφιλής μέθοδος και έχουν γραφτεί πολλά άρθρα που περιγράφουν τις εφαρμογές της. Σε έναν αραιής διάταξης ΠΕΑ, ο πληθυσμός διαιρείται σε διάφορους υποπληθυσμούς. Κάθε υποπληθυσμός ανατίθεται σε διαφορετικό επεξεργαστή (νησί). Κάθε επεξεργαστής τρέχει έναν σειριακό ΕΑ στον πληθυσμό του. Τα άτομα σε έναν υποπληθυσμό είναι σχετικά απομονωμένα με τα άτομα άλλων υποπληθυσμών. Οι μεμονωμένοι πληθυσμοί βοηθούν στην διατήρηση της γενετικής ποικιλομορφίας. Συνεπώς ο πληθυσμός κάθε νησιού εξερευνά διαφορετικό μέρος του χώρου αναζήτησης. Περιστασιακά, καλά άτομα μεταναστεύουν από κάποιον πληθυσμό σε κάποιον άλλο [7].

Οι περισσότερες εφαρμογές τρέχουν τον ίδιο ΕΑ σε κάθε νησί. Κάποιες εξαιρέσεις είναι [7]:

- Διαφορετικές κωδικοποιήσεις με διαφορετικό μέγεθος πληθυσμών σε ξεχωριστά νησιά.
- Διαφορετικές τιμές μετάλλαξης.
- Πληθυσμοί συν-λειτουργίας, όπου επιτρέπεται στους πληθυσμούς να αναμιγνύονται χρησιμοποιώντας διαφορετικό αριθμό λειτουργιών και παραμέτρων.

Παρά την προσπάθεια να δοθούν κάποιες θεωρητικές θεμελιώσεις, οι παράμετροι εφαρμόζονται διαισθητικά. Σημαντικές επιλογές που απαιτούνται από τον χρήστη είναι οι ακόλουθες [7]:

- Ποιοι επεξεργαστές ανταλλάσουν μεταξύ τους άτομα.
- Πόσο συχνά οι επεξεργαστές ανταλλάσουν άτομα.
- Το πλήθος των ατόμων που ανταλλάσουν οι επεξεργαστές.
- Ποια στρατηγική χρησιμοποιείται για την επιλογή των ατόμων που θα μεταναστεύσουν.

Πυκνής διάταξης ΠΕΑ
Σε έναν πυκνής διάταξης ΠΕΑ συνήθως ένα άτομο ανατίθεται σε κάθε επεξεργαστή. Τα άτομα επιτρέπεται να ζευγαρώσουν μόνο με γειτονικά άτομα (demes). Παρόλο οι περισσότερες εφαρμογές προτείνουν ένα σχετικά μικρό μέγεθος «γειτονιάς», κριτική παράμετρος είναι η αναλογία της ακτίνας της «γειτονιάς» με το μέγεθος του βασικού

πλέγματος. Το σχήμα των «γειτονιών»» μπορεί να είναι σταυρός, τετράγωνο, γραμμή, κ.α.. Οι «γειτονιές» μπορεί να επικαλύπτονται ανάλογα με το μέγεθος και το σχήμα τους. Γι' αυτό τα καλά άτομα επιτρέπεται να διαδίδονται σε όλο τον πληθυσμό. Σημαντικές επιλογές είναι το μέγεθος των «γειτονιών», η τοπολογία των επεξεργαστών καθώς και το σχήμα επανατοποθέτησης των ατόμων [7].

(2) Βάσει τύπου

Με βάση τον τύπο τους οι ΠΕΑ κατατάσσονται σε τρεις βασικές κατηγορίες [42], [35].

- Προσανατολισμένοι σε εφαρμογές (Application oriented): Είναι κλειστά συστήματα σχεδιασμένα για να κρύψουν τις λεπτομέρειες των ΕΑ ώστε να βοηθήσουν το χρήστη να αναπτύξει εφαρμογές σε συγκεκριμένα πεδία. Κάποιοι από αυτούς είναι χρήσιμοι σε διάφορες περιπτώσεις, όπως προγραμματισμός εργασιών ή τηλεπικοινωνίες. Άλλοι είναι περισσότερο προσανατολισμένοι σε συγκεκριμένη εφαρμογή, όπως π.χ. σε χρηματοοικονομικά. Συνήθως έχουν κάποιο βοηθητικό μενού και εύκολη παραμετροποίηση.
- Προσανατολισμένοι στον αλγόριθμο (Algorithm oriented): Βασίζονται σε συγκεκριμένους αλγορίθμους. Ο πηγαίος κώδικας είναι διαθέσιμος, ώστε να παρέχεται εύκολη ενσωμάτωση σε νέες εφαρμογές. Αυτή η κλάση μπορεί περαιτέρω να διαχωριστεί στις ακόλουθες κατηγορίες:
    - Συγκεκριμένος αλγόριθμος: Περιέχουν έναν απλό ΕΑ. Οι χρήστες του είναι είτε προγραμματιστές συστημάτων για τη σχεδίαση εφαρμογών, είτε ερευνητές ΕΑ.
    - Βιβλιοθήκες αλγορίθμων: Υποστηρίζουν μια ομάδα αλγορίθμων σε τυποποίηση βιβλιοθήκης. Είναι υψηλά παραμετροποιημένοι και περιέχουν πολλούς διαφορετικούς τελεστές ώστε να βοηθήσουν σε μελλοντικές εφαρμογές.
- Εργαλειοθήκες (Toolkits): Αποτελούν ευέλικτα περιβάλλοντα για προγραμματισμό διαφορετικών ΕΑ και εφαρμογών. Μπορούν να υποδιαιρεθούν στις ακόλουθες κατηγορίες:
    - Εκπαιδευτικά: Χρησιμοποιούνται για να εισάγουν τις έννοιες των ΕΑ σε αρχάριους χρήστες. Οι βασικές τεχνικές για την παρακολούθηση της εκτέλεσης και των αποτελεσμάτων, κατά τη διάρκεια της εξέλιξης μπορούν να διαχειριστούν εύκολα.
    - Γενικού σκοπού: Χρήσιμα για την τροποποίηση, ανάπτυξη και επίβλεψη πλήθους τελεστών, αλγορίθμων και εφαρμογών.

(3) Βάσει εφαρμογής

Οι ΠΕΑ και οι κατανεμημένοι ΕΑ έχουν φανεί στην πράξη πολύ χρήσιμοι σε ένα πλήθος από βιομηχανικές και εμπορικές εφαρμογές. Πολλά προβλήματα της καθημερινής ζωής μπορεί να χρειαστούν μέρες ή ακόμα και μήνες υπολογιστικού χρόνου για την επίλυσή τους σε σειριακά συστήματα. Παρόλο που ο πραγματικός χρόνος πολυπλοκότητας ενός προβλήματος δεν μπορεί να εξαλειφθεί χρησιμοποιώντας ένα πεπερασμένο πλήθος παράλληλων μηχανημάτων, η παράλληλη λογική συχνά επιτρέπει την μείωση του χρόνου αυτού σε λογικά επίπεδα. Αυτό μπορεί να είναι πολύ σημαντικό σε βιομηχανικό ή εμπορικό επίπεδο, όπου ο χρόνος για την εύρεση μιας λύσης παίζει καθοριστικό ρόλο είτε στη διαδικασία λήψης

αποφάσεων, είτε σε θέματα ανταγωνισμού. Επιπλέον, τα νέα μοντέλα δομημένου πληθυσμού συχνά επιτρέπουν καλύτερη εξερεύνηση εναλλακτικών λύσεων του χώρου σχεδίασης. Παρακάτω παρουσιάζονται μερικά από τα σημαντικότερα πεδία, όπου εφαρμόζονται ΠΕΑ σε πραγματικά προβλήματα με επιτυχία [7]:

- Συνδυαστική βελτιστοποίηση και επιχειρησιακή έρευνα.
- Σχεδίαση δικτύων τηλεπικοινωνίας.
- Χρηματοοικονομικές εφαρμογές.
- Σχεδίαση αναλογικών ηλεκτρονικών κυκλωμάτων.
- Σχεδίαση κυκλωμάτων VLSI.
- Μηχανολογική σχεδίαση.

1.8 Πλεονεκτήματα και Μειονεκτήματα Παράλληλης Εφαρμογής Εξελικτικού Αλγόριθμου

Τα αναμενόμενα πλεονεκτήματα από την παράλληλη εφαρμογή ενός ΕΑ μπορούν να συνοψιστούν ως ακολούθως [7]:

- Βρίσκουν εναλλακτικές λύσεις για το ίδιο πρόβλημα.
- Πραγματοποιείται παράλληλη έρευνα σε πολλαπλά σημεία του χώρου.
- Η παραλληλοποίηση πραγματοποιείται σχετικά εύκολα, χρησιμοποιώντας νησιά ή γειτονιές.
- Η έρευνα είναι πιο αποδοτική ακόμα και αν δεν χρησιμοποιείται παράλληλο υπολογιστικό υλικό.
- Επιτυγχάνεται, στις περισσότερες περιπτώσεις, υψηλότερη αποδοτικότητα από τους σειριακούς ΕΑ.
- Η συνεργασία με άλλες διαδικασίες έρευνας είναι εύκολη.
- Επιτυγχάνεται σημαντική επιτάχυνση με τη χρήση πολλαπλών επεξεργαστών.

Αυτά τα πλεονεκτήματα έρχονται σε αντίθεση με στοιχεία άλλων ευρετικών αλγορίθμων, οι οποίοι συνήθως απαιτούν τεχνητούς περιορισμούς, ψάχνουν μόνο από ένα σημείο σε κάποιο νέο ή βρίσκουν μια απλή λύση τη φορά και έχουν εν γένει μεγαλύτερη πιθανότητα να εγκλωβιστούν σε τοπικά βέλτιστα σε σχέση με τους ΕΑ [7].

Παρόλα αυτά αν και οποιοσδήποτε ΕΑ μπορεί να τρέξει με παράλληλο τρόπο, η υψηλή αποδοτικότητα δεν επιτυγχάνεται πάντα. Η ανάλυση ενός ΠΕΑ απαιτεί τις περισσότερες φορές τη χρησιμοποίηση μιας πολύπλοκης και ετερογενούς ομάδας δοκιμαστικών προβλημάτων για εκτέλεση πειραμάτων. Επίσης πρέπει να δοθεί ιδιαίτερη προσοχή στο κατά πόσο τα πειράματα θα μπορούν να επαναληφθούν για μελλοντικές προεκτάσεις. Τέλος παραμένουν αναπάντητα μερικά ερωτήματα που σχετίζονται με τη φυσική, αριθμητική καθώς και με την παράλληλη εκτέλεση των μοντέλων. Πρέπει επίσης να επισημανθεί ότι χρειάζονται επιπλέον παράμετροι για τον καθορισμό της ευρετικής ικανότητας ενός ΠΕΑ, επομένως απαιτείται περαιτέρω μελέτη για την πλήρη κατανόηση της σημαντικότητάς τους [7].

Έχουν πραγματοποιηθεί πολλές προσεγγίσεις με σκοπό να βρεθεί η καλύτερη τοπολογία για έναν αλγόριθμο πολλαπλού πληθυσμού. Τα αποτελέσματα είναι συχνά ανεπαρκή, δεδομένου ότι κάθε φορά ένας υπερκύβος είναι πιο αποδοτικός από ένα δακτύλιο. Το ίδιο ισχύει και για τα ρυθμό και τη συχνότητα μετανάστευσης. Κάποιες επιπλέον έρευνες έχουν δείξει πρακτικά για ένα ταξινομημένο σύνολο προβλημάτων

και μοντέλων κατανομής ότι κάποιες παράμετροι πρέπει να μένουν σταθερές και κάποιες άλλες να είναι ελεύθερες, με σκοπό να αλλάζουν τη συμπεριφορά του αλγορίθμου. Ένα συμπέρασμα, που δεν ισχύει σε όλες τις περιπτώσεις, είναι ότι η μετανάστευση του καλύτερου ατόμου του νησιού επιφέρει αισθητή βελτίωση στη συνάρτηση βελτιστοποίησης, παρόλο που η μετανάστευση ενός τυχαίου ατόμου μπορεί να είναι επιθυμητή για συνδυαστική βελτιστοποίηση και προβλήματα υψηλής συσχέτισης παραμέτρων (επιστατικά), δεδομένου ότι η πρόωρη σύγκλιση δεν επιβάλλεται στους αλγορίθμους κατανομής. Τέλος τα τετράγωνα πλέγματα παρέχουν γρηγορότερους υπολογισμούς σε μη επιστατικά προβλήματα [7].

ΚΕΦΑΛΑΙΟ 2

ΠΑΡΑΛΛΗΛΗ ΕΠΕΞΕΡΓΑΣΙΑ

2.1 Παράλληλη Αρχιτεκτονική Υπολογιστών

Κάθε υπολογιστής, είτε σειριακός είτε παράλληλος, λειτουργεί εκτελώντας εντολές σε δεδομένα. Μια ροή εντολών (αλγόριθμος) καθορίζει το τι θα κάνει ο υπολογιστής σε κάθε βήμα. Μια ροή δεδομένων (είσοδος στον αλγόριθμο) επηρεάζεται από αυτές τις εντολές. Μια ευρέως διαδεδομένη ταξινόμηση των Η/Υ, σύμφωνα με τον Michael J.Flynn (μοντέλο Flynn) [43], βασίζεται στο πλήθος των ταυτόχρονων ροών εντολών και δεδομένων, που γίνονται αντιληπτά από τον επεξεργαστή κατά τη διάρκεια της εκτέλεσης ενός προγράμματος (***Εικόνα 2.1***). Ανάλογα με το αν είναι μία ή πολλές οι ροές εντολών και δεδομένων, οι υπολογιστές χωρίζονται σε τέσσερις κατηγορίες [44]:

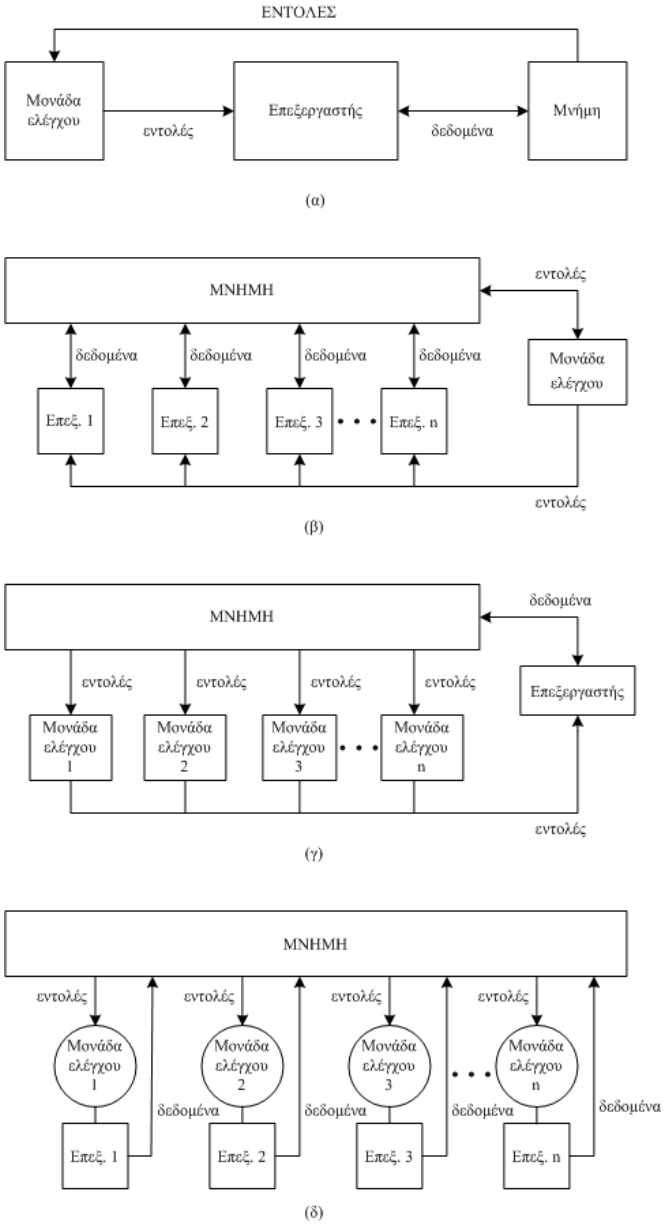
- **SISD** (Single Instruction Single Data stream): Μια Εντολή Μια ροή Δεδομένων.
- **SIMD** (Single Instruction Multiple Data stream): Μια Εντολή Πολλές ροές Δεδομένων.
- **MISD** (Multiple Instruction Single Data stream): Πολλές Εντολές Μια ροή Δεδομένων.
- **MIMD** (Multiple Instruction Multiple Data stream): Πολλές Εντολές Πολλές ροές Δεδομένων.

Η κατηγορία SISD αντιστοιχεί στον κλασσικό προσωπικό Η/Υ ή σταθμό εργασίας. Ένας υπολογιστής SISD αποτελείται από έναν επεξεργαστή, που λαμβάνει μονή ροή εντολών, η οποία αντιστοιχεί σε μονή ροή δεδομένων (***Εικόνα 2.1 (α)***). Σε κάθε βήμα, η μονάδα ελέγχου εκπέμπει μία εντολή, η οποία αντιστοιχεί σε ένα δεδομένο που λαμβάνεται από τη μονάδα μνήμης. Σχεδόν όλοι οι υπολογιστές που χρησιμοποιούνται σήμερα ανήκουν σε αυτό το μοντέλο, που προτάθηκε από τον John von Neumann στα τέλη της δεκαετίας του ‘40. Οι αλγόριθμοι που τρέχουν σε υπολογιστές SISD καλούνται σειριακοί, καθώς δεν περιέχουν καμία τεχνική παραλληλοποίησης.

Ένας υπολογιστής SIMD αποτελείται από *n* πανομοιότυπους επεξεργαστές, κάθε ένας από τους οποίους έχει τη δική του τοπική μνήμη, που μπορεί να αποθηκεύσει δεδομένα. Όλοι οι επεξεργαστές δουλεύουν υπό τον έλεγχο μονής ροής εντολών, που προέρχεται από μια κεντρική μονάδα ελέγχου. Υπάρχουν *n* ροές δεδομένων, μια για κάθε επεξεργαστή (***Εικόνα 2.1 (β)***). Οι επεξεργαστές λειτουργούν σύγχρονα, σε κάθε βήμα δηλαδή όλοι οι επεξεργαστές εκτελούν την ίδια εντολή σε διαφορετικό στοιχείο δεδομένων. Εκμεταλλεύονται την χωρική παραλληλία, που μπορεί να υπάρχει σε κάποιο πρόβλημα, ενώ είναι κατάλληλοι για μεγάλες και ομοιόμορφες δομές δεδομένων [44].

Οι SIMD υπολογιστές είναι πολύ πιο ασταθείς από τους MISD υπολογιστές. Πλήθος προβλημάτων, που καλύπτουν μια μεγάλη περιοχή εφαρμογών, μπορούν να λυθούν με παράλληλους αλγορίθμους σε υπολογιστές SIMD. Ένα άλλο ενδιαφέρον χαρακτηριστικό είναι ότι οι αλγόριθμοι για αυτούς τους υπολογιστές είναι σχετικά

εύκολο να σχεδιαστούν, να αναλυθούν και να εφαρμοστούν. Από την άλλη πλευρά, μόνο προβλήματα που μπορούν να υποδιαιρεθούν σε ένα σύνολο πανομοιότυπων υποπροβλημάτων και έπειτα να επιλυθούν ταυτόχρονα από τους διάφορους επεξεργαστές χρησιμοποιώντας το ίδιο σύνολο εντολών, μπορούν να αντιμετωπιστούν από SIMD υπολογιστές. Υπάρχουν πολλοί υπολογισμοί που δεν ταιριάζουν σε αυτόν τον τύπο, αυτά τα προβλήματα συνήθως υποδιαιρούνται σε διαφορετικά υποπροβλήματα και λύνονται χρησιμοποιώντας MIMD υπολογιστές [44].



Εικόνα 2.1: Ταξινόμηση της αρχιτεκτονικής των υπολογιστών βάσει του μοντέλου Flyn. (α) Αρχιτεκτονική SISD. (β)Αρχιτεκτονική SIMD. (γ) Αρχιτεκτονική MISD. (δ) Αρχιτεκτονική MIMD [7].

Στην κατηγορία MIMD πολλοί επεξεργαστές λειτουργούν σε συνεργασία, μέσω κάποιας μορφής διασύνδεσης. Η κλάση των παράλληλων υπολογιστών είναι η πιο γενική και η πιο αποτελεσματική στην κατάταξη Flynn. Εδώ υπάρχουν *n* επεξεργαστές, *n* ροές εντολών και *n* ροές δεδομένων. Κάθε επεξεργαστής έχει τη δική

του μονάδα ελέγχου και την δική του τοπική μνήμη, καθιστώντας τους πιο αποτελεσματικούς σε σχέση με αυτούς που χρησιμοποιούνται στους SIMD υπολογιστές (***Εικόνα 2.1 (δ)***). Κάθε επεξεργαστής λειτουργεί υπό τον έλεγχο απλής ροής εντολών, που προέρχεται από τη δική του μονάδα ελέγχου, γι' αυτό ενδεχομένως όλοι οι επεξεργαστές εκτελούν διαφορετικά προγράμματα με διαφορετικά δεδομένα, καθώς επιλύουν διαφορετικά υποπροβλήματα του αρχικού προβλήματος. Αυτό σημαίνει ότι συνήθως οι επεξεργαστές λειτουργούν ασύγχρονα. Το μοντέλο MIMD της παράλληλης υπολογιστικής είναι το πιο γενικό και αποτελεσματικό. Οι υπολογιστές αυτής της κλάσης χρησιμοποιούνται για να λύσουν παράλληλα τα προβλήματα εκείνα τα οποία δεν έχουν κανονική δομή, που απαιτείται από το μοντέλο SIMD. Από την άλλη, οι ασύγχρονοι αλγόριθμοι είναι δύσκολο να σχεδιαστούν, να αναλυθούν και να εφαρμοστούν [44].

Η κατηγορία MISD γενικά βρίσκει ελάχιστες εφαρμογές. Σε αυτή την κατηγορία *n* επεξεργαστές, κάθε ένας από τους οποίους έχει την δική του μονάδα ελέγχου μοιράζονται μια κοινή μονάδα μνήμης (***Εικόνα 2.1 (γ)***). Σε κάθε βήμα, ένα στοιχείο δεδομένου, που λαμβάνεται από την μνήμη, επεξεργάζεται ταυτόχρονα από όλους τους επεξεργαστές, κάθε ένας ανάλογα με την εντολή που έχει λάβει από την μονάδα ελέγχου του. Η παραλληλοποίηση επιτυγχάνεται επιτρέποντας στους επεξεργαστές να επεξεργάζονται διαφορετικά το ίδιο δεδομένο. Αυτή η κλάση των υπολογιστών προσαρμόζεται φυσικά στους υπολογισμούς που απαιτούν μια είσοδος να υποστεί διαφορετικές επεξεργασίες, η κάθε μια από τις οποίες εφαρμόζεται στην αρχική είσοδο, π.χ. προβλήματα ταξινόμησης. Το είδος των υπολογισμών που μπορούν να εκτελεστούν αποτελεσματικά στους MIMD υπολογιστές είναι μάλλον εξειδικευμένο, όπως είναι η επεξεργασία σήματος [44].

Όσον αφορά στην κατανομή της μνήμης μεταξύ των επεξεργαστών, υπάρχουν δύο κατηγορίες συστημάτων [7]:

- Διαμοιρασμένης μνήμης (Shared Memory)
- Κατανεμημένης μνήμης (Distributed Memory)

Στην πρώτη κατηγορία υπάρχει κεντρική μνήμη, στην οποία αναφέρονται όλοι οι επεξεργαστές, οπότε όλα τα δεδομένα είναι προσπελάσιμα από όλους τους επεξεργαστές. Στην περίπτωση αυτή χρησιμοποιούνται ταχύτατες μνήμες προσωρινής αποθήκευσης δεδομένων (cache memories) σε κάθε επεξεργαστή, για να επιταχυνθεί η προσπέλαση της κεντρικής μνήμης. Στην κατηγορία αυτή ανήκουν μεγάλα συστήματα με πολλούς επεξεργαστές σε κοινή πλατφόρμα [7].

Στη δεύτερη κατηγορία ανήκουν τα δίκτυα (νέφη - clusters) από σταθμούς εργασίας. Στην κατηγορία αυτή απαιτείται κάποια μορφή επικοινωνίας μεταξύ των επεξεργαστών, για την αποστολή και λήψη δεδομένων. Η απόδοση περιορίζεται από την ταχύτητα της επικοινωνίας, από το διαφορετικό φορτίο του κάθε επεξεργαστή, από τον διαφορετικό τύπο ή το διαφορετικό λειτουργικό κάθε μηχανήματος [7].

Στην παράλληλη αρχιτεκτονική Η/Υ πρέπει να προστεθεί πλέον και ο Παγκόσμιος Ιστός (World Wide Web), ως μια μορφή διαμοιρασμένης υπολογιστικής ισχύος. Η χρησιμοποίηση διαμοιρασμένων υπολογιστικών πόρων μέσω του Παγκόσμιου Ιστού, έτσι ώστε να συμπεριφέροντε ως ένας Η/Υ, ονομάζεται metacomputing.

Η χρησιμοποίηση του Παγκόσμιου Ιστού για παράλληλη επεξεργασία εμπεριέχει πολλά προβλήματα, τα οποία συνδέονται με [7]:

- Την ύπαρξη πολύ μεγάλου αριθμού Η/Υ.
- Τη διακοπή της επικοινωνίας μεταξύ των υπολογιστών.
- Την ύπαρξη διαφορετικών τύπων μηχανών.
- Την ύπαρξη διαφορετικών λειτουργικών συστημάτων.
- Την ύπαρξη διαφορετικών πρωτοκόλλων επικοινωνίας.
- Την ύπαρξη περιορισμών πρόσβασης (π.χ. τοίχος προστασίας κ.λπ.).

2.2 Εργαλεία Παραλληλοποίησης Εξελικτικού Αλγόριθμου

Σε αυτό το σημείο θα παρουσιαστούν μερικά δημοφιλή εργαλεία επικοινωνίας για την ανάπτυξη ενός ΕΑ [7]. Οι περισσότερες εργαλειοθήκες (toolkits) ΠΕΑ εφαρμόζονται χρησιμοποιώντας το μοντέλο επικοινωνίας μεταφοράς μηνύματος (message passing model), το οποίο έχει φυσικά πλεονεκτήματα σε πολύ-υπολογιστές που έχουν κατανεμημένη μνήμη και αποτελούν το πιο σύνηθες είδος Η/Υ σε πανεπιστήμια και ερευνητικά ινστιτούτα.

Στο μοντέλο μεταφοράς μηνύματος οι διεργασίες στον ίδιο επεξεργαστή ή σε διαφορετικούς ως προς τη φύση επεξεργαστές, πραγματοποιούνται στέλνοντας μηνύματα η μια στην άλλη, χρησιμοποιώντας ένα μέσο μεταφοράς, όπως είναι μια απλή ή εξειδικευμένη δικτυακή σύνδεση. Οι δύο βασικές αρχές είναι οι ρουτίνες αποστολής και λήψης (send/receive).

Socket: Το περιβάλλον διεπαφής (interface) socket [45] είναι ευρέως διαδεδομένο εργαλείο προγραμματισμού μεταφοράς μηνυμάτων. Ένα σύνολο δομών δεδομένων και συναρτήσεων επιτρέπουν στον προγραμματιστή να πραγματοποιήσει αλληλεπιδραστικές συνδέσεις μεταξύ δύο υπολογιστών χρησιμοποιώντας το πρωτόκολλο TCP για την εφαρμογή κατανεμημένων εφαρμογών γενικής φύσεως.

PVM (Parallel Virtual Machine): Το PVM [46] αποτελεί λογισμικό το οποίο επιτρέπει την εφαρμογή ενός ετερογενούς δικτύου παράλληλων και σειριακών υπολογιστών σαν ένα ενιαίο, γενικό και ευέλικτο παράλληλο υπολογιστικό μέσο.

MPI (Message Passing Interface): Το MPI [47] αποτελεί μια βιβλιοθήκη από ρουτίνες μεταφοράς μηνυμάτων παρόμοιες με το PVM αλλά περισσότερο πλήρης. Αυτού του είδους το API (Application Protocol Interface) πρωτοεμφανίστηκε στα μέσα της δεκαετίας του ‘90 από μια μεγάλη ομάδα ατόμων που αποτελούσαν κυρίως ακαδημαϊκοί, κυβερνητικοί καθώς και άνθρωποι από το χώρο της βιομηχανίας. Αυτό το περιβάλλον διεπαφής αντικατοπτρίζει τις εμπειρίες των παραπάνω ανθρώπων με προγενέστερες βιβλιοθήκες μεταφοράς μηνυμάτων όπως είναι το PVM. Βασικός στόχος ήταν η ανάπτυξη μιας ενιαίας βιβλιοθήκης η οποία θα μπορούσε να εφαρμοστεί επαρκώς σε μια πληθώρα από συστήματα πολλαπλών επεξεργαστών. Το MPI αποτελεί πλέον πρότυπο και υπάρχουν διάφορα λογισμικά εφαρμογής του όπως το MPICH και το LAM/MPI.

Οι ρουτίνες του MPI υποστηρίζουν επικοινωνίες τύπου «process to process», επικοινωνίες ομάδων, το στήσιμο και τη διαχείριση ομάδων επικοινωνίας καθώς και την αλληλεπίδραση με το περιβάλλον. Επιπρόσθετα, στο άμεσο μέλλον θα δοθεί η

δυνατότητα στους χρήστες του MPI για υπηρεσίες σε WAN (Wide Area Network) δίκτυα. Εκτενέστερη αναφορά στο MPI γίνεται στο Κεφάλαιο 5.

Java – RMI (Remote Method Invocation): Η εφαρμογή RPC (Remote Procedure Calls) στη Java καλείται Java – RMI [48]. Το RPC επιτρέπει τη λειτουργία απομακρυσμένων υπηρεσιών (remote services) με διαφανή τρόπο και επιπλέον είναι ένα φυσικό βήμα για τους προγραμματιστές που χρησιμοποιούν σειριακή λογική. Το RMI στη Java επιτρέπει σε μια εφαρμογή να χρησιμοποιεί απομακρυσμένες υπηρεσίες με πρόσθετα πλεονεκτήματα να βρίσκεται σε ανεξάρτητη πλατφόρμα και να έχει πρόσβαση σε όλα τα υπόλοιπα χρήσιμα χαρακτηριστικά της Java, όταν έχει να κάνει με κατανεμημένη υπολογιστική και γενικά το Internet.

Το μοντέλο πελάτη/διακομιστή (client/server), το οποίο χρησιμοποιείται από το Java – RMI, συχνά είναι σχετικά αργό στις παρούσες εφαρμογές της Java, τουλάχιστον σε επιστημονικούς υπολογισμούς, που είναι αρκετά σημαντικός παράγοντας στους αλγόριθμους βελτιστοποίησης.

CORBA (Common Object Request Broker Architecture): Παρόλο που πολλά συστήματα εφαρμόζουν «πολυδιαχωρισμό» (multithreading) για παράλληλα και/ή κατανεμημένα προγράμματα, κάποιες υψηλοτέρου επιπέδου έρευνες έχουν επικεντρωθεί στη συγχώνευση υπαρχόντων ή μελλοντικών εφαρμογών, έτσι ώστε να μπορούν να δουλεύουν απρόσκοπτα σε κατανεμημένο περιβάλλον. Τέτοια συστήματα λογισμικού δημιούργησαν τον όρο «middleware», από τα οποία το CORBA είναι ένα από τα γνωστότερα [49]. Το CORBA βασίζεται σε αντικειμενοστραφείς τεχνολογίες. Είναι μια συλλογή συγκεκριμένων εργαλείων, που επιλύουν λειτουργικά προβλήματα σε κατανεμημένα συστήματα.

Το CORBA είναι πολύ σημαντικό λογισμικό επειδή επιταχύνει εφαρμογές, επιτρέποντας στους πελάτες να χρησιμοποιήσουν λειτουργίες σε κατανεμημένα αντικείμενα (distributed objects) χωρίς να λάβουν υπόψη την τοποθεσία του αντικείμενου, τη γλώσσα προγραμματισμού, το λειτουργικό σύστημα, το πρωτόκολλο επικοινωνίας ή τη σύνθεση του Η/Υ.

Globus: Είναι ένα νέο και πολλά υποσχόμενο ερευνητικό έργο, που παρέχει ένα πλήρες σύνολο εργαλείων για το χτίσιμο «metacomputing» εφαρμογών [50]. Χτίζεται πάνω στις υπηρεσίες που παρέχονται από προγενέστερα συστήματα, όπως είναι το PVM, το MPI και το Legion και επεκτείνει ευρέως τις δυνατότητές τους. Η εργαλειοθήκη του Globus αποτελείται από διάφορους υποκώδικες (modules), που χρησιμοποιούνται για την εφαρμογή υψηλού επιπέδου υπηρεσιών. Μια τέτοια υπηρεσία είναι η AWARE (Adaptive Wide Area Resource Environment). Περιέχει ένα ολοκληρωμένο σύνολο υπηρεσιών συμπεριλαμβανομένων «metacomputing enabled» περιβαλλόντων αλληλεπίδρασης σε μια εφαρμογή MPI βιβλιοθήκης, διάφορων γλωσσών προγραμματισμού, καθώς και εργαλεία κατασκευής εικονικών περιβαλλόντων (CVEs – Constructing Virtual Enviroments).

OpenMP: Το openMP αποτελεί ένα σύνολο από οδηγίες μεταγλωττιστών (compiler directives) και βιβλιοθήκες ρουτινών τα οποία χρησιμοποιούνται για να εκφράσουν διαδικασίες παραλληλίας διαμοιρασμένης μνήμης (shared memory parallelism). Ο προγραμματιστής χρησιμοποιεί τις κατάλληλες οδηγίες σε ένα σειριακό πρόγραμμα

για να πει στον μεταγλωττιστή ποια μέρη του προγράμματος εκτελούνται παράλληλα και για να προσδιορίσει τα σημεία συγχρονισμού.

ΚΕΦΑΛΑΙΟ 3

ΠΑΡΑΛΛΗΛΟΙ ΔΙΑΦΟΡΙΚΟΙ ΕΞΕΛΙΚΤΙΚΟΙ

3.1 Γενικά

Όπως έχει ήδη αναφερθεί, στους ΕΑ, η διαδικασία αξιολόγησης χρωμοσωμάτων μέσω κάποιας συνάρτησης προσαρμογής, είναι αρκετές φορές μια ιδιαίτερα χρονοβόρα διαδικασία. Έτσι η απαίτηση για υψηλότερη απόδοση, χαμηλότερο κόστος και διατήρηση της παραγωγικότητας οδήγησε τους ερευνητές στο χώρο της παράλληλης επεξεργασίας. Η υψηλή αποδοτικότητα των απλών Διαφορικών Εξελικτικών αλγορίθμων σε πραγματικά προβλήματα ήταν ένα κίνητρο για την εφαρμογή παράλληλης επεξεργασίας σε αυτούς.

Στο Διαφορικό Εξελικτικό αλγόριθμο εισάγεται ένας νέος τελεστής μετάλλαξης, που βασίζεται σε μια τριάδα τυχαία επιλεγμένων διαφορετικών ατόμων [51]. Για κάθε άτομο του εκάστοτε πληθυσμού επιλέγεται μια τριάδα των τυχαίων ατόμων και παράγεται ένα νέο διάνυσμα παραμέτρων προσθέτοντας την σταθμισμένη διαφορά των δύο ατόμων της τριάδας στο τρίτο (δότης). Με αυτόν τον τρόπο παράγεται ένα τροποποιημένο άτομο το οποίο μαζί με το υπό εξέλιξη άτομο του πληθυσμού υπόκειται σε επιχιασμό, παράγοντας κατ' αυτόν τον τρόπο την τελική υποψήφια λύση. Ο πληθυσμός της επόμενης γενιάς επιλέγεται ανάμεσα στον υπάρχοντα πληθυσμό και τις αντίστοιχες υποψήφιες λύσεις που έχουν προέλθει από τη διαδικασία της μετάλλαξης και του επιχιασμού. Εάν το υποψήφιο διάνυσμα παραμέτρων κάθε ατόμου του πληθυσμού δίνει καλύτερη τιμή αντικειμενικής συνάρτησης από το αντίστοιχο υπάρχων τότε το τελευταίο αντικαθίσταται από το πρώτο. Η βασική διαφορά του τελεστή επιλογής του ΔΕ αλγορίθμου από τους υπόλοιπους Εξελικτικούς Αλγορίθμους εντοπίζεται στο γεγονός ότι κάθε υποψήφια λύση δε συγκρίνεται με όλα τα άτομα του πληθυσμού παρά μόνο με το αντίστοιχο του στον υπάρχοντα πληθυσμό, το οποίο και αντικαθιστά εάν δίνει καλύτερη λύση. Το χαρακτηριστικό αυτό είναι πολύ σημαντικό για την παραλληλοποίηση του ΔΕ αλγορίθμου καθώς επιτρέπει εύκολη εφαρμογή ασύγχρονης διαδικασίας. Εάν κάθε άτομο ανατεθεί σε έναν επεξεργαστή τότε αυτός μπορεί να προχωρήσει στον υπολογισμό ενός νέου ατόμου, κάθε φορά που ολοκληρώνει τον υπολογισμό του προγόνου του. Η επικοινωνία μεταξύ των διαφορετικών επεξεργαστών πραγματοποιείται μόνο για την εφαρμογή του τελεστή μετάλλαξης, ο οποίος δεν απαιτεί την ύπαρξη γενιάς με την στενή έννοια, καθώς μπορεί στον υπάρχοντα πληθυσμό να υπάρχουν άτομα που ανήκουν σε διαφορετικές γενιές.

3.2 Ιστορική Αναδρομή Διαφορικών Εξελικτικών Αλγορίθμων

Οι Διαφορικοί Εξελικτικοί αλγόριθμοι δημιουργήθηκαν από τις προσπάθειες του Ken Price να επιλύσει το πρόβλημα των πολυωνύμων Chebychev που είχε προταθεί από τον Rainer Storn [52]. Μία αιφνίδια πρόοδος συνέβη όταν ο Ken Price σκέφτηκε να χρησιμοποιήσει διαφορές διανυσμάτων για να διαταράξει τον πληθυσμό. Προσομοιώσεις σε υπολογιστές και από τις δύο μεριές απέφεραν ουσιαστικές βελτιώσεις, που συνέθεσαν την σημερινή εικόνα των ΔΕ αλγορίθμων. Οι ΔΕ είναι αλγόριθμοι ευπροσάρμοστοι, εύρωστοι και αποτελεσματικοί, που τα τελευταία

χρόνια έχουν βρει εφαρμογή τόσο σε πραγματικά, όσο και σε μαθηματικά προβλήματα βελτιστοποίησης.

Οι ΔΕ αλγόριθμοι ανήκουν στην ευρύτερη ομάδα των Εξελικτικών Αλγορίθμων και έχουν παρουσιάσει ελκυστικά χαρακτηριστικά γνωρίσματα κατά τη βελτιστοποίηση συνεχών συναρτήσεων [53]. Ο Rainer Storn παρουσιάζει τους ΔΕ αλγορίθμους να υπερτερούν έναντι των άλλων εξελικτικών μεθόδων και τους προσδίδει ιδιαίτερη υπολογιστική ευελιξία. Οι ΔΕ παρουσιάζουν συνεχή πρόοδο από τα πρώτα χρόνια εξέλιξης των συγκεκριμένων αλγορίθμων, (1994 - 1996), ενώ η επιστημονική κοινότητα συνεχίζει να εργάζεται πάνω στην εξέλιξή τους. Οι ΔΕ αλγόριθμοι κέρδισαν την 3η θέση στο First International Contest on Evolutionary Computation - 1stICEO, που πραγματοποιήθηκε στη Nagoya τον Μάιο του 1996 (οι πρώτες δύο θέσεις δόθηκαν σε μη-γενετικού τύπου αλγορίθμους που δεν είναι πάντοτε εφαρμόσιμοι, αλλά επέλυσαν το δοθέν πρόβλημα πιο γρήγορα από τους ΔΕ αλγορίθμους). Οι ΔΕ αλγόριθμοι, αποδείχτηκαν δηλαδή, οι καλύτεροι γενετικού τύπου αλγόριθμοι για την επίλυση πραγματικών τιμών συναρτήσεων του $1^{ου}$ ICEO, υπερνικώντας άλλους ως τότε γνωστούς και ευρέως χρησιμοποιούμενους αλγορίθμους [52].

3.3 Ο Διαφορικός Εξελικτικός Αλγόριθμος

Ο Διαφορικός Εξελικτικός αλγόριθμος αποτελεί ένα τύπο ΕΣ, που αναπτύχθηκε με τέτοιο τρόπο ώστε να μπορεί να χειρίζεται προβλήματα βελτιστοποίησης ορισμένα σε συνεχή πεδία, τα οποία απαντώνται συχνά στη μηχανολογική σχεδίαση [54], [55]. Ο κλασικός ΔΕ αλγόριθμος εξελίσσει ένα σταθερό μέγεθος πληθυσμού, ο οποίος αρχικοποιείται γεννώντας τυχαίες τιμές για τις παραμέτρους των χρωμοσωμάτων μέσα σε προκαθορισμένα όρια. Έπειτα από την αρχικοποίηση του πληθυσμού, ξεκινάει μια επαναληπτική διαδικασία, όπου σε κάθε επανάληψη (γενιά), παράγεται ένας νέος πληθυσμός μέχρι να ικανοποιηθεί μια συνθήκη τερματισμού. Σε κάθε γενιά, κάθε άτομο του πληθυσμού μπορεί να αντικατασταθεί από ένα νέο άτομο. Το νέο άτομο είναι γραμμικός συνδυασμός μεταξύ ενός τυχαία επιλεγμένου ατόμου και τη διαφορά δύο άλλων τυχαία επιλεγμένων ατόμων. Εκτός από το μέγεθος του πληθυσμού άλλες παράμετροι του αλγορίθμου είναι η πιθανότητα επιλογής, του επιχιασμού και ένας παράγοντας που ενισχύει τον όρο της διαφοράς. Εμπειρικές μελέτες έδειξαν ότι η σύγκλιση του ΔΕ αλγόριθμου εξαρτάται σημαντικά από αυτές τις παραμέτρους. Στη συνέχεια δίνεται μια αναλυτική περιγραφή του αλγορίθμου.

Δεδομένης μιας αντικειμενικής συνάρτησης $f(X):\mathbb{R}^n \to \mathbb{R}$, στόχος της βελτιστοποίησης είναι η ελαχιστοποίηση της τιμής της αντικειμενικής συνάρτησης βελτιστοποιώντας τις τιμές των παραμέτρων του $X=(x_1,x_2,...,x_n) \quad x \in \mathbb{R}$, όπου X το διάνυσμα που αποτελείται από τις n παραμέτρους της αντικειμενικής συνάρτησης. Αυτές οι παράμετροι παίρνουν τιμές μεταξύ καθορισμένων άνω και κάτω ορίων, $x_j^{(L)} \le x_j \le x_j^{(U)} \quad j=1,...,n$.

Ο ΔΕ αλγόριθμος εφαρμόζει πραγματική κωδικοποίηση για τις τιμές των παραμέτρων της αντικειμενικής συνάρτησης. Προκειμένου να υπάρξει κάποιο σημείο εκκίνησης για τον αλγόριθμο, πραγματοποιείται αρχικοποίηση του πληθυσμού. Συχνά η μόνη διαθέσιμη πληροφορία είναι τα όρια των παραμέτρων, γι' αυτό η αρχικοποίηση γίνεται δίνοντας τυχαίες τιμές στις παραμέτρους μεταξύ των δοσμένων ορίων: $x_{i,j}^{(o)} = r \cdot (x_j^{(U)} - x_j^{(L)}) + x_j^{(L)}$ όπου $i=1,...,n_{\pi\lambda\eta\theta.}$, $j=1,...,n$ και r είναι τυχαία τιμή

ομοιόμορφης κατανομής στο διάστημα [0, 1]. Το σχήμα αναπαραγωγής του πληθυσμού του ΔΕ παράγει έναν προσωρινό πληθυσμό ως ακολούθως:

$$x'^{(G+1)}_{i,j} = \begin{cases} x^{(G)}_{C_i,j} + F(x^{(G)}_{A_i,j} - x^{(G)}_{B_i,j}) & \varepsilon\alpha\nu \quad r \le Cr \quad \forall \quad j = 1,...,n \\ x^{(G)}_{i,j} & \alpha\lambda\lambda\iota\omega\varsigma \end{cases}$$

όπου

$i = 1,...n_{\pi\lambda\eta\theta.}, \quad j = 1,...,n$

$A = 1,...n_{\pi\lambda\eta\theta.}, \quad B = 1,...n_{\pi\lambda\eta\theta.}, \quad C = 1,...n_{\pi\lambda\eta\theta.}, \quad A_i \neq B_i \neq C_i \neq i$

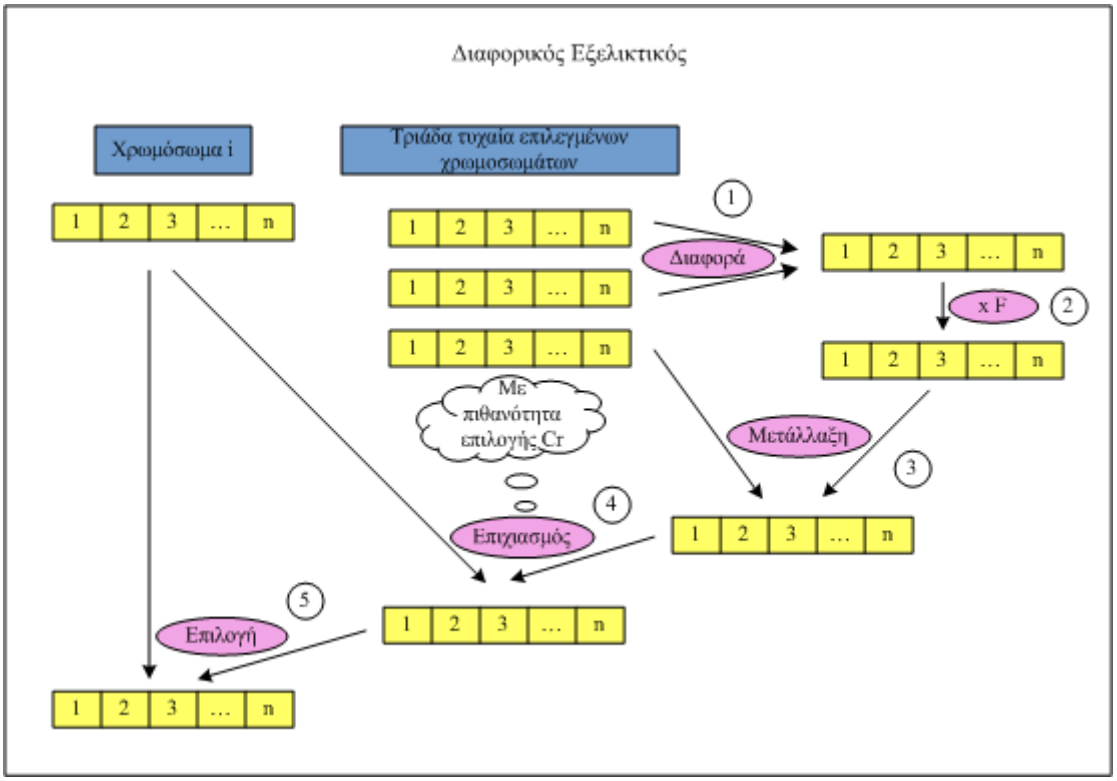
$Cr \in [0,1], \quad F \in [0,2], \quad r \in [0,1]$

Για κάθε υπό εξέλιξη χρωμόσωμα τα A, B και C είναι τυχαία επιλεγμένα χρωμοσώματα του πληθυσμού, τα οποία είναι διαφορετικά μεταξύ τους και διαφορετικά από το υπό εξέλιξη χρωμόσωμα. Ο τυχαίος αριθμός *r* γεννάται για κάθε παράμετρο του χρωμοσώματος. Οι *F* και *Cr* είναι παράμετροι ελέγχου του ΔΕ, οι οποίες παραμένουν σταθερές κατά τη διαδικασία, επηρεάζοντας την ταχύτητα σύγκλισης και την ευρωστία του αλγορίθμου. Εξαρτώνται από την αντικειμενική συνάρτηση, τα χαρακτηριστικά του προβλήματος και το μέγεθος του πληθυσμού. Οι βέλτιστες τιμές αυτών των παραμέτρων λαμβάνονται μέσω διαδικασίας δοκιμής και σφάλματος. Το σχήμα επιλογής του ΔΕ αλγορίθμου περιγράφεται ως ακολούθως:

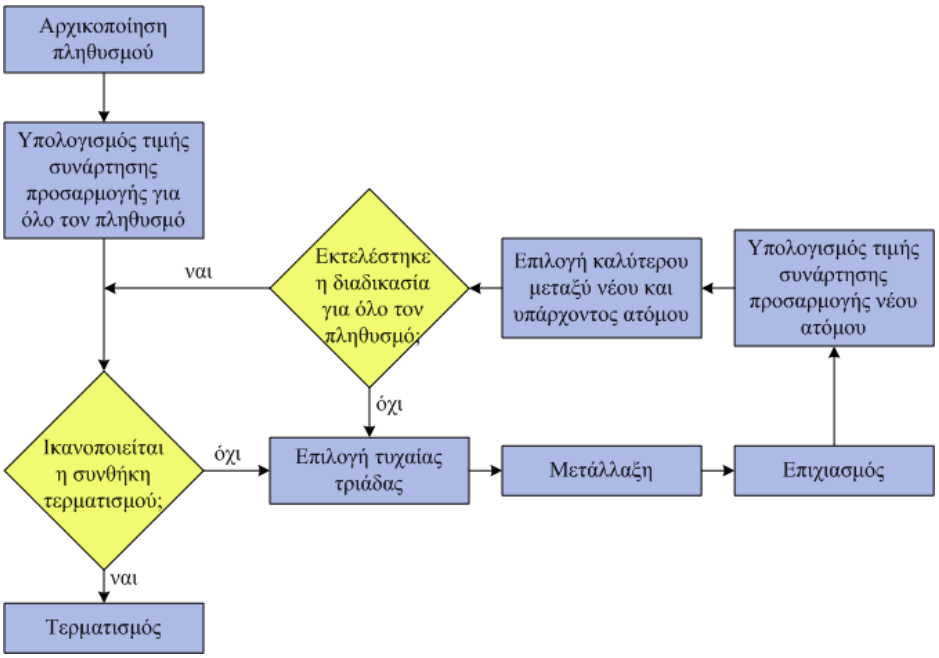
$$X^{(G+1)}_i = \begin{cases} X'^{(G+1)}_i & \varepsilon\alpha\nu \quad f(X'^{(G+1)}_i) \le f(X^{(G)}_i) \\ X^{(G)}_i & \alpha\lambda\lambda\iota\omega\varsigma \end{cases}$$

Το σχήμα επιλέγει το προσωρινό ή το υπάρχον χρωμόσωμα με βάση την καλύτερη τιμή της συνάρτησης κόστους. Γι' αυτό όλα τα άτομα της επόμενης γενιάς είναι εξίσου καλά ή καλύτερα από τα αντίστοιχα τους στον υπάρχοντα πληθυσμό. Η γενική διαδικασία που ακολουθείται σε έναν Διαφορικό Εξελικτικό αλγόριθμο απεικονίζεται στην ***Εικόνα 3.1***. Στην ***Εικόνα 3.2*** που ακολουθεί, παρουσιάζεται ένα διάγραμμα ροής των βασικών λειτουργιών του Διαφορικού Εξελικτικού Αλγορίθμου, όπου φαίνεται η σειριακή του εφαρμογή.

Στους ΔΕ Αλγορίθμους, ένα από τα βασικά προβλήματα που αντιμετωπίζονται είναι το ότι δύναται να αποδοθεί κάποια λύση, η οποία δεν ανήκει στο πεδίο των εφικτών λύσεων, αλλά αποδόθηκε από τον αλγόριθμο, λόγω του ότι με τη χρήση της λύσης αυτής επιτυγχάνεται στη συνάρτηση προσαρμογής καλύτερη και πιο ικανοποιητική τιμή από ότι με τις υπόλοιπες εναλλακτικές. Οι αλγόριθμοι αυτοί δηλαδή στην απλή τους μορφή δεν μπορούν να αντιμετωπίσουν προβλήματα βελτιστοποίησης με περιορισμούς. Ένας τρόπος για να αντιμετωπίσουν προβλήματα με περιορισμούς είναι η επέμβαση στη διαδικασία επιλογής. Συγκεκριμένα, το πιθανό χρωμόσωμα του ενδιάμεσου πληθυσμού αντικαθιστά το αντίστοιχο αρχικό αν και μόνο αν ικανοποιεί όλους τους δοθέντες περιορισμούς και βελτιώνει την τιμή της συνάρτησης κόστους. Με τον τρόπο αυτό, πραγματοποιείται αύξηση της πίεσης εύρεσης της περιοχής εφικτών λύσεων και βελτιώνεται η αποδοτικότητα και η αξιοπιστία του αλγορίθμου.



Εικόνα 3.1: Περιγραφή λειτουργίας Διαφορικού Εξελικτικού Αλγορίθμου.



Εικόνα 3.2: Διάγραμμα ροής του Διαφορικού Εξελικτικού Αλγορίθμου.

Τα πιο σημαντικά πλεονεκτήματα των ΔΕ αλγορίθμων είναι τα παρακάτω:

- Δεν απαιτείται μία αρχική εφικτή λύση για την έναρξη της λειτουργίας τους.
- Η χρήση τους είναι εύκολη, αφού απαιτείται μόνο η αρχική επιλογή των παραμέτρων που χαρακτηρίζουν τον αλγόριθμο, και οι οποίες στη συνέχεια παραμένουν συνήθως αμετάβλητοι.
- Είναι εύρωστοι και αξιόπιστοι.

- Αποδίδουν λύση σε αρκετά μικρό χρονικό διάστημα.
- Είναι απλοί στη χρήση τους.
- Η λειτουργία τους γίνεται εύκολα κατανοητή.
- Είναι εύκολοι στον προγραμματισμό.

3.4 Εφαρμογές Παράλληλων Διαφορικών Εξελικτικών Αλγορίθμων

Υπάρχουν λιγοστές αναφορές στην ανοικτή βιβλιογραφία για παράλληλους ΔΕ αλγόριθμους. Η παραλληλοποίηση ΔΕ αλγορίθμων, χρησιμοποιώντας δίκτυο με τοπολογία δακτυλίου, μπορεί να βελτιώσει τόσο την ταχύτητα όσο και την απόδοση της μεθόδου. Τα αποτελέσματα έδειξαν ότι το μέγεθος ανταλλαγής πληροφοριών μεταξύ των υποπληθυσμών των διαφορετικών επεξεργαστών, καθορίζει σημαντικά την απόδοση του αλγορίθμου [56].

Μια άλλη προσέγγιση παρουσιάζει παράλληλη εφαρμογή ΔΕ αλγορίθμου, σε τοπικό δίκτυο υπολογιστών, που προσφέρει επιτάχυνση της διαδικασίας βελτιστοποίησης χρονοβόρων αντικειμενικών συναρτήσεων με μεγάλη απαίτηση σε υπολογιστικούς πόρους. Η εφαρμογή πραγματοποιείται χρησιμοποιώντας κοινόχρηστα αρχεία δίσκου, εγκαθιστώντας ασύγχρονη επικοινωνία μεταξύ του «Master» και των «Slave» διαδικασιών, χωρίς την χρήση κάποιου πρωτοκόλλου επικοινωνίας, όπως PVM ή MPI [57].

Βελτίωση τόσο του χρόνου επεξεργασίας όσο και της απόδοσης του αλγόριθμου παρουσιάζεται και στην περίπτωση αραιής διάταξης παραλληλοποίησης προσαρμοστικού ΔΕ αλγόριθμου, που βασίζεται σε πολύ-πληθυσμιακό μοντέλο, με τυχαίας σύνδεσης τοπολογία και εφαρμόζεται σε δίκτυο υπολογιστών. Αυτή η προσέγγιση μπορεί να πετύχει σημαντική επιτάχυνση, ενώ το ολικό βέλτιστο εντοπίζεται με μεγαλύτερη πιθανότητα, ακόμα και αν υπάρχουν πολλά παρόμοια υποβέλτιστα. Η βελτίωση της σύγκλισης οφείλεται κυρίως στην μετανάστευση ατόμων μεταξύ των υποπληθυσμών, με αποτέλεσμα να διατηρείται η ποικιλομορφία του πληθυσμού και να αποφεύγεται η πρόωρη σύγκλιση ακόμα και αν εφαρμοστεί σε σειριακή εφαρμογή [58].

Μια άλλη δυνατότητα ενός πολύ-πληθυσμιακού ΔΕ αλγορίθμου είναι η εύρεση όλων των βέλτιστων μιας πολυκριτήριας συνάρτησης. Η εξερεύνηση όλου του χώρου έρευνας μπορεί να διασφαλιστεί από μια ελεγχόμενη αρχικοποίηση των υποπληθυσμών, ενώ κάθε διαφορετικός ΔΕ αλγόριθμος διασφαλίζει την εκμετάλλευση του χώρου. Τα βέλτιστα που εντοπίζονται, αποθηκεύονται σε ένα αρχείο, το οποίο επίσης παίζει το ρόλο προσωρινής μνήμης επικοινωνίας μεταξύ των υποπληθυσμών [59].

ΚΕΦΑΛΑΙΟ 4

ΕΝΑΣ ΑΠΛΟΣ ΠΑΡΑΛΛΗΛΟΣ ΔΙΑΦΟΡΙΚΟΣ ΕΞΕΛΙΚΤΙΚΟΣ ΑΛΓΟΡΙΘΜΟΣ

4.1 Γενικά

Στο κεφάλαιο αυτό παρουσιάζεται η πρώτη προσπάθεια που έγινε για την παραλληλοποίηση ενός ΔΕ αλγορίθμου, με σκοπό την πραγματοποίηση μιας εύκολης και εύχρηστης εφαρμογής, χωρίς ιδιαίτερες απαιτήσεις σε υπολογιστικό υλικό ή πολύπλοκα λογισμικά.

Για την επίτευξη αυτού του εγχειρήματος χρησιμοποιήθηκε απλό δίκτυο προσωπικών υπολογιστών και το λογισμικό αναπτύχθηκε σε Microsoft Visual Basic 6.0. Για την επικοινωνία μεταξύ των προγραμμάτων δεν χρησιμοποιήθηκε κάποιο πρωτόκολλο, προκειμένου να γίνει όσο το δυνατόν πιο εύχρηστη και πιο απλή η εφαρμογή, και χωρίς να απαιτείται εγκατάσταση λογισμικού στους υπολογιστές. Κάθε πρόγραμμα λαμβάνει τις επιθυμητές πληροφορίες μέσω απλών αρχείων κειμένου.

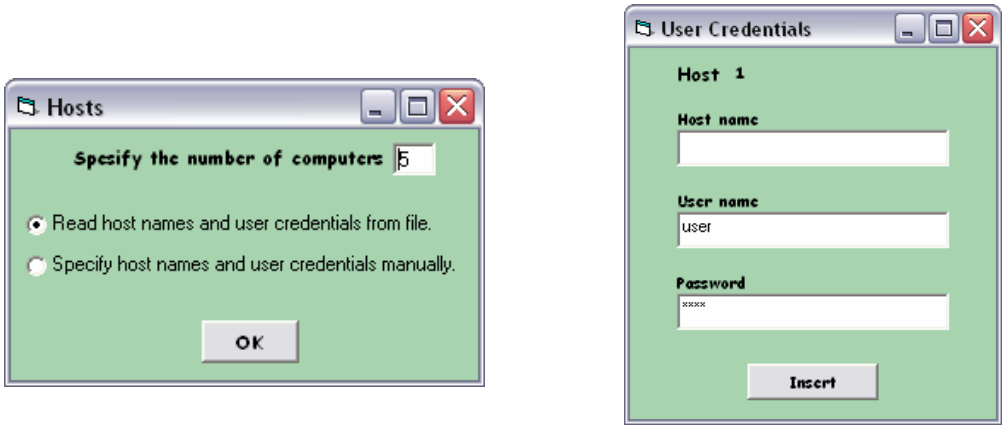
Σε αυτή την προσπάθεια προέκυψαν δύο κυρίως προβλήματα, το πρώτο εκ των οποίων αφορούσε στην αυτόματη εκτέλεση προγραμμάτων σε απομακρυσμένους υπολογιστές, ελέγχοντας τη διαδικασία μέσω ενός κεντρικού, το οποίο τελικά επιλύθηκε με την βοήθεια των PsTools [60] (βλ. §4.7). Το δεύτερο και βασικότερο πρόβλημα αφορούσε στην ταυτόχρονη προσπέλαση αρχείων, που λαμβάνει χώρα κατά τη διάρκεια εκτέλεσης του προγράμματος, το οποίο αν και μειώθηκε σε μεγάλο βαθμό δεν εξαλείφθηκε τελείως, γεγονός που καθιστά το πρόγραμμα ασταθές. Στη συνέχεια παρουσιάζεται η λειτουργία του λογισμικού που αναπτύχθηκε, ενώ αναλύονται όλες οι επιμέρους διαδικασίες του.

4.2 Παρουσίαση Λογισμικού

Το λογισμικό που αναπτύχθηκε έχει τη δυνατότητα να επιλύει πραγματικά προβλήματα χρησιμοποιώντας ένα Διαφορικό Εξελικτικό αλγόριθμο σε παράλληλη ασύγχρονη εφαρμογή. Ο χρήστης αλληλεπιδρά με το πρόγραμμα μέσω φόρμας, στην οποία μπορεί να καθορίσει τις σημαντικότερες παραμέτρους του αλγορίθμου. Για την επίλυση του προβλήματος απαιτείται ένα (ή περισσότερα εάν είναι αναγκαία) εκτελέσιμο αρχείο της συνάρτησης προσαρμογής και δύο αρχεία κειμένου με τις απαραίτητες πληροφορίες για την εκτέλεση του προγράμματος.

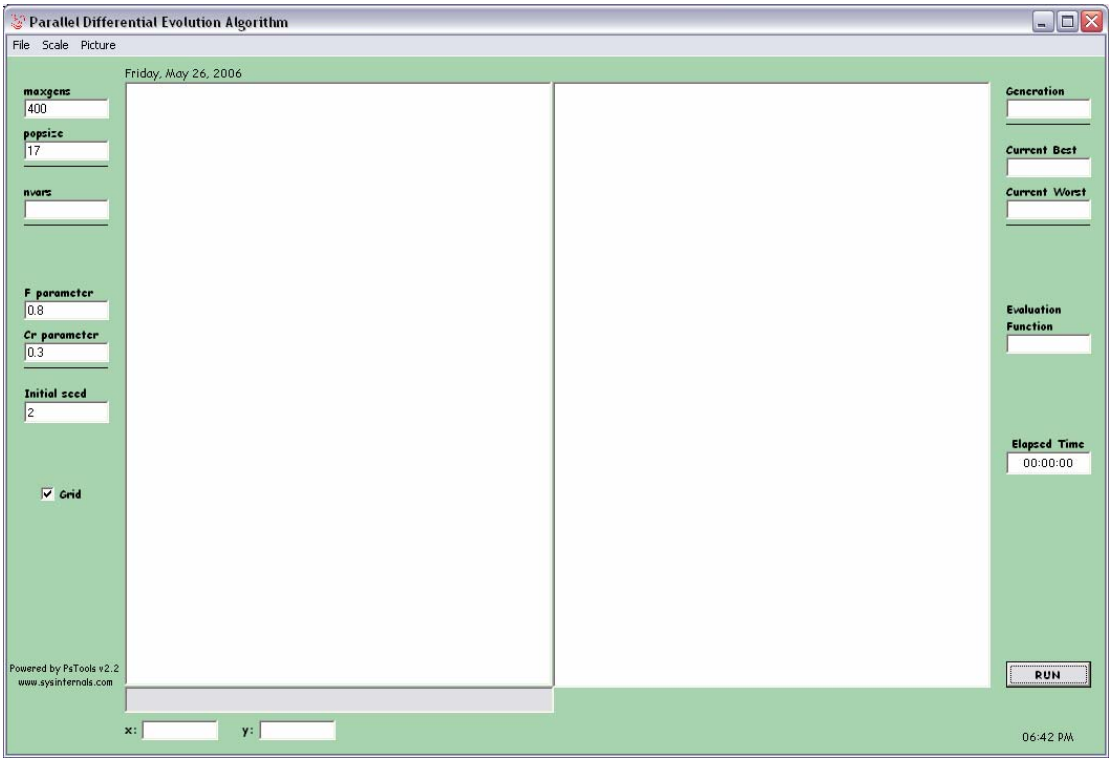
Το συγκεκριμένο λογισμικό αποτελείται από δύο επιμέρους προγράμματα, το κύριο που καλείται «Master» και το δευτερεύον που καλείται «Slave». Με την εκτέλεση του «Master» εμφανίζεται στην οθόνη η φόρμα της ***Εικόνας 4.1(α)*** όπου ο χρήστης δίνει το πλήθος των υπολογιστών του δικτύου και επιλέγει αν θα δώσει τα στοιχεία του δικτύου μέσω αρχείου ή σε φόρμα (***Εικόνα 4.1(β)***). Στη συνέχεια εμφανίζεται η βασική φόρμα του προγράμματος (***Εικόνα 4.2***) όπου ο χρήστης μπορεί να επιλέξει το πλήθος των γενεών, το μέγεθος του πληθυσμού και να καθορίσει τις παραμέτρους F και Cr του Διαφορικού Εξελικτικού αλγορίθμου (***Εικόνα 4.3 (α)***). Μετά την εκκίνηση του προγράμματος, εμφανίζεται στη φόρμα το πλήθος των ανεξάρτητων μεταβλητών

σχεδίασης του προβλήματος καθώς επίσης και η συνάρτηση προσαρμογής που υπολογίζεται κάθε στιγμή.



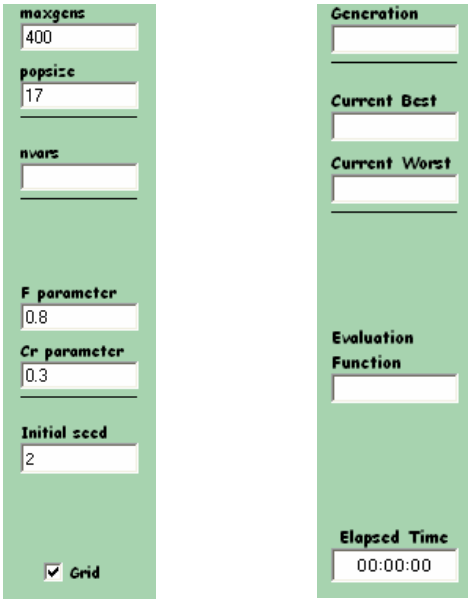
(α) *(β)*

Εικόνα 4.1: (α) Φόρμα εκκίνησης του προγράμματος όπου καθορίζεται το πλήθος των υπολογιστών. (β) Φόρμα εισαγωγής στοιχείων δικτύου.



Εικόνα 4.2: Φόρμα μέσω της οποίας αλληλεπιδρά ο χρήστης με τον αλγόριθμο.

Επίσης εμφανίζεται η γενιά στην οποία βρίσκεται ο αλγόριθμος και η τιμή της καλύτερης και της χειρότερης λύσης κάθε γενιάς (***Εικόνα 4.3 (β)***), ενώ υπάρχει ένα χρονόμετρο το οποίο υπολογίζει τη διάρκεια εκτέλεσης του προγράμματος. Τα δύο παράθυρα στο κέντρο της φόρμας παρουσιάζουν την σύγκλιση του αλγορίθμου σε πραγματική κλίμακα (αριστερά) και λογαριθμική κλίμακα (δεξιά).



(α) *(β)*

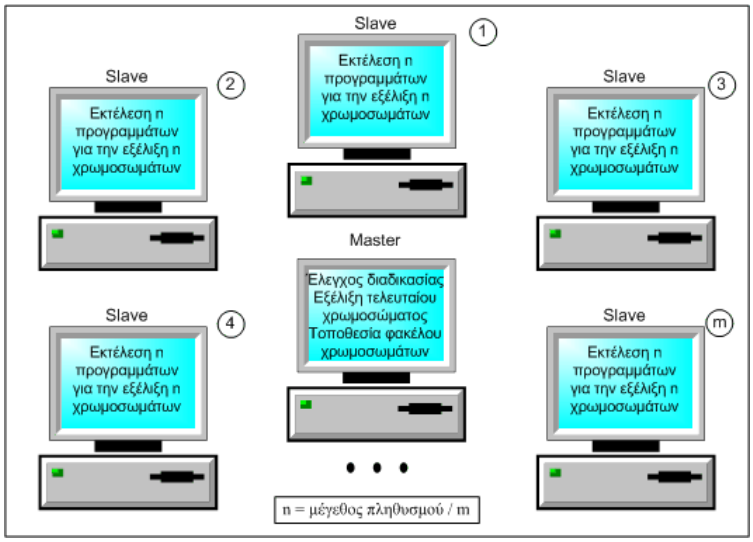
Εικόνα 4.3: (α) Δεξί μέρος φόρμας όπου ο χρήστης μπορεί να επέμβει στις παραμέτρους του αλγόριθμου. (β) Αριστερό μέρος φόρμας όπου εμφανίζεται η εκάστοτε γενιά, η καλύτερη και η χειρότερη λύση κάθε γενιάς, η συνάρτηση προσαρμογής και ο χρόνος εκτέλεσης.

4.3 Δομή κώδικα

Στη συγκεκριμένη προσέγγιση παράλληλου Διαφορικού Εξελικτικού αλγορίθμου εφαρμόστηκε μοντέλο «Master-Slave» σε ασύγχρονη υλοποίηση. Η εφαρμογή έγινε σε τοπικό δίκτυο υπολογιστών (LAN) με λειτουργικό σύστημα Windows XP Professional. Στην ***Εικόνα 4.4*** φαίνεται η τοπολογία του δικτύου και η κατανομή των προγραμμάτων σε αυτούς. Το πακέτο του προγράμματος, όπως αναφέρθηκε, αποτελείται από το πρόγραμμα «Master» και το υποπρόγραμμα «Slave» το οποίο αναπαράγεται τόσες φορές όσο το μέγεθος του πληθυσμού μείον ένα. Εξωτερικά απαιτείται ένα αρχείο κειμένου με το πλήθος και τα όρια των μεταβλητών σχεδίασης, ένα αρχείο κειμένου με το πλήθος και τα ονόματα των εκτελέσιμων αρχείων, βάσει των οποίων θα αξιολογούνται τα χρωμοσώματα και θα υπολογίζεται η τιμή της συνάρτησης προσαρμογής. Φυσικά απαιτούνται να υπάρχουν και τα αντίστοιχα εκτελέσιμα αρχεία. Προαιρετικά παρέχεται ένα αρχείο κειμένου με τα ονόματα των υπολογιστών και τα στοιχεία των λογαριασμών, προκειμένου να μην πληκτρολογούνται κάθε φορά στην αντίστοιχη φόρμα.

Η διαδικασία είναι πλήρως αυτοματοποιημένη και ο χρήστης αλληλεπιδρά μόνο με τον κεντρικό υπολογιστή του δικτύου, στον οποίο τοποθετείται ο «Master». Οι απομακρυσμένοι υπολογιστές δεν απαιτούν καμία εγκατάσταση λογισμικού, καθώς όλα τα απαραίτητα εκτελέσιμα αρχεία τοποθετούνται σε αυτούς μέσω δικτύου με εντολή του λογισμικού «Master». Επιπλέον, όλες οι εντολές για την εκκίνηση των λογισμικών «Slave» δίνονται από το «Master» μέσω ενός πακέτου εντολών δικτύου, (τα PsTools) των οποίων η λειτουργία περιγράφεται στη συνέχεια. Έπειτα από την τοποθέτηση του «Master» στον κεντρικό υπολογιστή του δικτύου αυτός αναλαμβάνει να διεκπεραιώσει όλες τις απαραίτητες διεργασίες για την εκτέλεση του προγράμματος, όπως να διανείμει τα λογισμικά «Slave» στους απομακρυσμένους υπολογιστές, να αναθέσει σε κάθε «Slave» το άτομο του πληθυσμού το οποίο θα

εξελίσσει, καθώς επίσης και να ελέγχει την πορεία της εκτέλεσης του συνολικού αλγορίθμου. Μια επιπλέον λειτουργία του λογισμικού «Master» είναι να εξελίσσει το τελευταίο άτομο του πληθυσμού. Τα λογισμικά «Slave» εκτελούν τον Διαφορικό Εξελικτικό αλγόριθμο για να εξελίξουν μόνο το άτομο του πληθυσμού που τους έχει ανατεθεί.



Εικόνα 4.4: Τοπολογία δικτύου υπολογιστών.

Για την κατανομή των λογισμικών «Slave» στους υπολογιστές του δικτύου διαιρείται ο πληθυσμός με το πλήθος των απομακρυσμένων υπολογιστών και σε κάθε έναν υπολογιστή ανατίθεται αριθμός λογισμικών «Slave» ίσος με το μέγεθος του αντίστοιχου υποπληθυσμού. Εάν ο πληθυσμός δεν διαιρείται ακριβώς με το πλήθος των υπολογιστών, το υπόλοιπο τοποθετείται στον τελευταίο υπολογιστή.

Άμεση επικοινωνία μεταξύ του λογισμικού «Master» και των λογισμικών «Slave» μετά την έναρξη του προγράμματος δεν υπάρχει και η απαιτούμενη ενημέρωση για την εξέλιξη των χρωμοσωμάτων γίνεται αποκλειστικά μέσω αρχείων κειμένου, τα οποία είναι τοποθετημένα σε ειδικό φάκελο (φάκελος χρωμοσωμάτων), σε περιοχή του κεντρικού υπολογιστή.

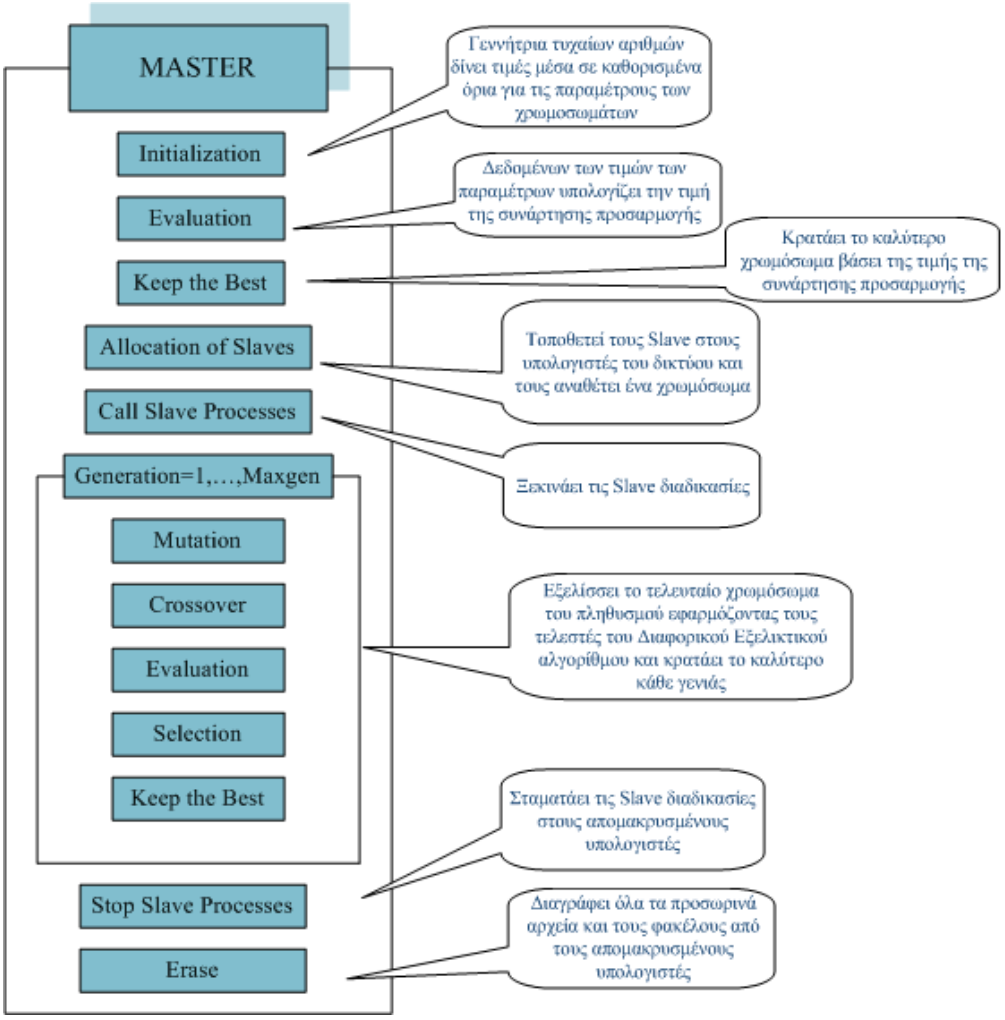
4.4 Λειτουργία «Master»

Το λογισμικό «Master» είναι υπεύθυνο για την λειτουργία ολόκληρου του αλγορίθμου. Αρχικά δημιουργεί τον πρώτο πληθυσμό, γεννώντας τυχαίες τιμές μεταξύ των προκαθορισμένων ορίων για τις μεταβλητές σχεδίασης (γονίδια των χρωμοσωμάτων). Για αυτόν τον πληθυσμό υπολογίζει την τιμή της συνάρτησης προσαρμογής κάθε χρωμοσώματος, εγγράφει στον φάκελο χρωμοσωμάτων αρχεία κειμένου, που κάθε ένα περιέχει τις τιμές των μεταβλητών σχεδίασης και την τιμή της συνάρτησης προσαρμογής του χρωμοσώματος, ενώ φυλάσσει σε κατάλληλη θέση το χρωμόσωμα με την καλύτερη (μικρότερη) τιμή της συνάρτησης προσαρμογής.

Στ συνέχεια ισοκατανέμει τα λογισμικά «Slave» στους απομακρυσμένους υπολογιστές, δίνοντας τους και τις απαραίτητες πληροφορίες, ώστε να μπορέσουν να ξεκινήσουν τη διαδικασία. Κάθε πακέτο που εγγράφεται στους απομακρυσμένους υπολογιστές για την λειτουργία του εκάστοτε λογισμικού «Slave» περιλαμβάνει το

εκτελέσιμο αρχείο του «Slave», ένα αρχείο κειμένου με τις απαραίτητες πληροφορίες, καθώς και τα εκτελέσιμα αρχεία για τον υπολογισμό της συνάρτησης κόστους.

Επόμενο βήμα του κώδικα είναι να καλέσει όλα τα «Slave» στους απομακρυσμένους υπολογιστές, έτσι ώστε να εκκινήσουν τη διαδικασία εξέλιξης του ατόμου του πληθυσμού που έχουν αναλάβει. Μετά την κλήση ενός λογισμικού «Slave» αυτό λειτουργεί ανεξάρτητα, επικοινωνώντας μόνο με τον φάκελο χρωμοσωμάτων, εωσότου τερματιστεί από τον «Master».



Εικόνα 4.5: Διάγραμμα λειτουργίας λογισμικού «Master», που εκτελείται στον κεντρικό υπολογιστή.

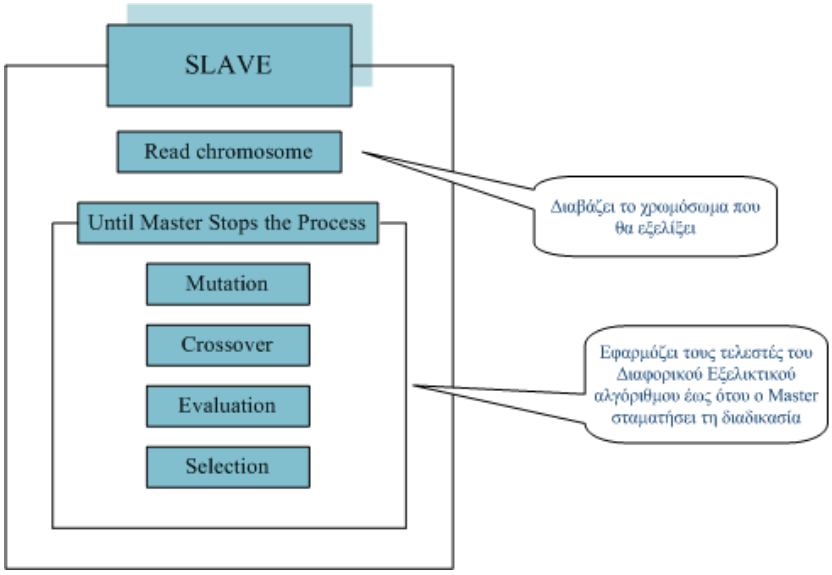
Στη συνέχεια το λογισμικό «Master» εκκινεί τη διαδικασία εξέλιξης του τελευταίου ατόμου του πληθυσμού, εφαρμόζοντας σε αυτό τον ΔΕ αλγόριθμο. Τη διαδικασία αυτή την εκτελεί για συγκεκριμένο πλήθος γενιών, που καθορίζεται από τον χρήστη. Στην ουσία αυτό το πλήθος γενεών αποτελεί και τη συνθήκη τερματισμού ολόκληρου του αλγορίθμου. Σε κάθε γενιά αρχικά εκτελείται μετάλλαξη βάσει της διαφοράς, πολλαπλασιασμένη με ένα συντελεστή F, δύο τυχαία επιλεγμένων χρωμοσωμάτων και ενός τρίτου τυχαία επιλεγμένου χρωμοσώματος. Έπειτα γίνεται επιχιασμός μεταξύ του χρωμοσώματος που προκύπτει από τη μετάλλαξη και του προς εξέλιξη χρωμοσώματος με μια πιθανότητα επιλογής Cr για κάθε παράμετρο. Βάσει της συνάρτησης κόστους, επιλέγεται το καλύτερο από το προκύπτον και το προς εξέλιξη χρωμόσωμα. Στο τέλος κάθε γενιάς το λογισμικό «Master» αναζητά στο φάκελο

χρωμοσωμάτων το καλύτερο χρωμόσωμα από τα υπάρχοντα, τα οποία μπορεί να ανήκουν σε διαφορετικές γενιές, αφού ο αλγόριθμος είναι ασύγχρονος.

Εφόσον το «Master» εκτελέσει το πλήθος των γενεών που του έχει ανατεθεί, τερματίζει τη λειτουργία των «Slave» και εξάγει ως αποτέλεσμα το συνολικά καλύτερο χρωμόσωμα, εγγράφοντας σε ένα αρχείο κειμένου τις τιμές των παραμέτρων του και την τιμή της αντίστοιχης συνάρτησης κόστους. Επίσης παράγει ένα ακόμα αρχείο, που περιέχει το ιστορικό σύγκλισης του αλγορίθμου κατά τη διάρκεια εκτέλεσης του, εγγράφοντας το καλύτερο και το χειρότερο άτομο κάθε γενιάς (ως γενιά ορίζεται η διαδοχή του τελευταίου χρωμοσώματος του πληθυσμού το οποίο διαχειρίζεται το λογισμικό «Master»). Τέλος διαγράφει από τους απομακρυσμένους υπολογιστές τα αρχεία των «Slave» και από όλο το δίκτυο οποιοδήποτε άλλο αρχείο δημιουργήθηκε για τις ανάγκες εκτέλεσης του προγράμματος. Στην ***Εικόνα 4.5*** δίνεται το διάγραμμα της λειτουργίας του λογισμικού «Master».

4.5 Λειτουργία «Slave»

Το λογισμικό «Slave» αποτελεί ένα εκτελέσιμο αρχείο, το οποίο με την έναρξη του προγράμματος τοποθετείται μαζί με τα εκτελέσιμα αρχεία της συνάρτησης κόστους και το αρχείο με τις απαιτούμενες πληροφορίες, με εντολή του «Master», σε συγκεκριμένη τοποθεσία στους απομακρυσμένους υπολογιστές.



Εικόνα 4.6: Διάγραμμα λειτουργίας λογισμικού «Slave», που εκτελείται στους απομακρυσμένους υπολογιστές.

Αρχικά διαβάζει τις πληροφορίες που χρειάζεται για την εκτέλεση του προγράμματος από το αντίστοιχο αρχείο κειμένου. Αυτό το αρχείο περιέχει το χρωμόσωμα το οποίο καλείται να αναλάβει να εξελίξει, το πλήθος των παραμέτρων του χρωμοσώματος, τις παραμέτρους *F* και *Cr*, που απαιτούνται για τον ΔΕ αλγόριθμο, την τοποθεσία του φάκελου χρωμοσωμάτων στον κεντρικό υπολογιστή, το πλήθος των εκτελέσιμων αρχείων συνάρτησης προσαρμογής καθώς και τα ονόματά τους.

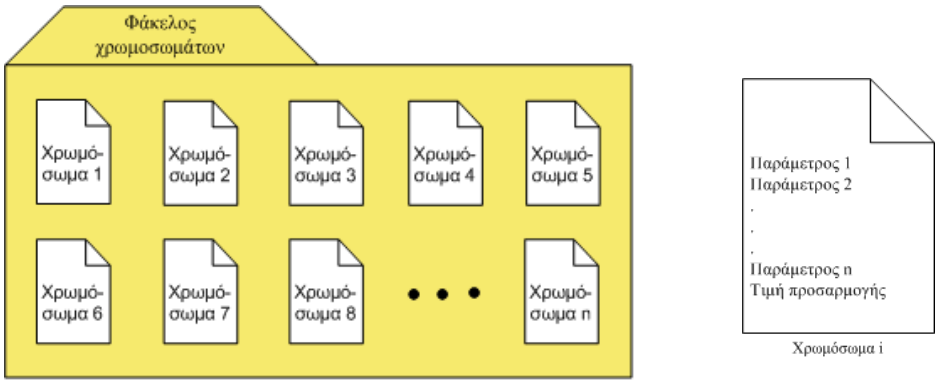
Στην συνέχεια εκτελείται ένας ατέρμων βρόγχος εωσότου η λειτουργία του «Slave» να διακοπεί κατόπιν εντολής του «Master». Μέσα σε αυτόν το βρόγχο ο κώδικας

διαβάζει τις παραμέτρους του χρωμοσώματος που του αντιστοιχεί και εφαρμόζει τον ΔΕ αλγόριθμο.

Ο «Slave» αλγόριθμος, από τη στιγμή που ξεκινάει, λειτουργεί τελείως ανεξάρτητα από τον «Master» και τους υπόλοιπους «Slave» και αλληλεπιδρά μόνο με τον φάκελο χρωμοσωμάτων, από όπου διαβάζει και γράφει το χρωμόσωμα που του αντιστοιχεί και από όπου διαβάζει τα τρία τυχαία χρωμοσώματα που απαιτούνται για την υλοποίηση του ΔΕ αλγόριθμου. Το πλήθος των επαναλήψεων που εκτελεί τον ΔΕ αλγόριθμο δεν είναι ίδιο με το πλήθος των γενεών του «Master», εξαιτίας της ασύγχρονης υλοποίησης του παράλληλου ΔΕ και εξαρτάται από την διάρκεια εκτέλεσης του προγράμματος και τους διαθέσιμους πόρους του συστήματος. Στην ***Εικόνα 4.6*** δίνεται το διάγραμμα της λειτουργίας του λογισμικού «Slave», που εκτελεί κάθε απομακρυσμένος υπολογιστής για κάθε χρωμόσωμα.

4.6 Επικοινωνία και Αλληλεπίδραση

Όπως έχει αναφερθεί, δεν υπάρχει απευθείας επικοινωνία μεταξύ των προγραμμάτων μετά την έναρξη του προγράμματος. Η μόνη αλληλεπίδραση των προγραμμάτων είναι με τον φάκελο χρωμοσωμάτων (***Εικόνα 4.7(α)***), από τον οποίο λαμβάνουν τις πληροφορίες που απαιτούν και τον οποίο ενημερώνουν για την εξέλιξη των χρωμοσωμάτων. Περιέχει αρχεία κειμένου ίσα στον αριθμό με το μέγεθος του πληθυσμού. Κάθε αρχείο κειμένου αντιστοιχεί σε ένα χρωμόσωμα και περιέχει τις τιμές των παραμέτρων του χρωμοσώματος καθώς και την τιμή της συνάρτησης κόστους για τις παραμέτρους αυτές (***Εικόνα 4.7(β)***).



(α) ***(β)***

Εικόνα 4.7: (α) Φάκελος χρωμοσωμάτων, που περιέχει τα αρχεία κειμένου με τις πληροφορίες για τα χρωμοσώματα. (β) Δομή αρχείου κειμένου, που χρησιμοποιείται για την επικοινωνία μεταξύ των προγραμμάτων.

Ο φάκελος χρωμοσωμάτων δημιουργείται από το λογισμικό «Master» κατά την αρχικοποίηση, διαγράφεται κατά την λήξη και είναι προσβάσιμος από όλους τους υπολογιστές του δικτύου. Από αυτή την τοποθεσία κάθε αλγόριθμος διαβάζει τις μεταβλητές σχεδίασης του χρωμοσώματος που του έχει ανατεθεί, καθώς επίσης λαμβάνει και τα τρία τυχαία άτομα για την υλοποίηση του ΔΕ αλγόριθμου. Μετά το πέρας του ΔΕ, κάθε αλγόριθμος ενημερώνει τον φάκελο χρωμοσωμάτων για τυχόν εξέλιξη του ατόμου του. Ο κάθε αλγόριθμος εξελίσσει ξεχωριστά το δικό του χρωμόσωμα, με βάση τα τρέχοντα χρωμοσώματα που υπάρχουν στον φάκελο χρωμοσωμάτων τη στιγμή που τα χρειάζεται, τα οποία μπορεί να μην ανήκουν στην ίδια γενιά, καθώς ο αλγόριθμος είναι ασύγχρονος.

4.7 PsTools

Τα PsTools αποτελούν μια ολοκληρωμένη σουίτα εφαρμογών, που επιτρέπει την απομακρυσμένη διαχείριση συστημάτων Η/Υ καθώς επίσης και του τοπικού συστήματος. Πιο συγκεκριμένα τα PsTools είναι μια συλλογή από εντολές, οι οποίες μπορούν να χρησιμοποιηθούν απευθείας από το αντίστοιχο περιβάλλον γραμμής εντολών (command prompt) των Windows. Αξίζει να σημειωθεί ότι δεν απαιτούν καμία εγκατάσταση, είναι εξαιρετικά απλά στη χρήση και επίσης δε χρειάζεται καν να υπάρχουν στους απομακρυσμένους υπολογιστές [60].

Οι εντολές που χρησιμοποιήθηκαν εδώ είναι η PsExec, που επιτρέπει την απομακρυσμένη εκτέλεση εφαρμογών σε συστήματα, με ολοκληρωμένο τρόπο και παρέχοντας πλήρη αλληλεπίδραση σε εφαρμογές κονσόλας και η PsKill, η οποία είναι μια εντολή τερματισμού, που επιτρέπει τον τερματισμό εφαρμογών τόσο στο τοπικό όσο και στα απομακρυσμένα συστήματα. Όπως όλες οι εντολές των PsTools οι PsExec και PsKill δεν απαιτούν καμία εγκατάσταση λογισμικού στα απομακρυσμένα συστήματα.

4.8 Εφαρμογές

4.8.1 Διερεύνηση παραμέτρων F και Cr Διαφορικού Εξελικτικού Αλγόριθμου

Ο κώδικας που αναπτύχθηκε για την παραλληλοποίηση του ΔΕ αλγορίθμου εφαρμόστηκε αρχικά σε πέντε μαθηματικές συναρτήσεις (***Πίνακας 4.1***) για να διερευνηθούν οι παράμετροι F και Cr. Εκτελέστηκαν τρεξίματα για όλους τους δυνατούς συνδυασμούς $F \in [0,1]$ και $Cr \in [0,1]$ για κάθε συνάρτηση, ώστε να εντοπιστούν οι τιμές εκείνες για τις οποίες παρουσιάζει καλύτερη συμπεριφορά ο αλγόριθμος. Λόγω έλλειψης υπολογιστικών πόρων τα τρεξίματα έγιναν με πληθυσμό 10 ατόμων, ενώ κάθε χρωμόσωμα αποτελούταν από 5 ανεξάρτητες μεταβλητές σχεδίασης. Η διάρκεια της εκτέλεσης ήταν 400 γενιές. Τα αποτελέσματα φαίνονται στους πίνακες που ακολουθούν.

Όνομα Συνάρτησης	Τύπος Συνάρτησης	Όρια Μεταβλητών Σχεδίασης
Sphere	$f(x) = \sum_{i=1}^{n} x_i^2$	$-100 \le x_i \le 100$
Rosenbrock	$f(x) = \sum_{i=1}^{n} (100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2)$	$-100 \le x_i \le 100$
Griewank	$f(x) = \frac{1}{4000}\sum_{i=1}^{n}(x_i - 100)^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i - 100}{\sqrt{i}}\right) + 1$	$-600 \le x_i \le 600$
Rastrigin	$f(x) = \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i) + 10)$	$-10 \le x_i \le 10$
Ackley	$f(x) = 20 + e - 20e^{-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}} - e^{\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)}$	$-32.768 \le x_i \le 32.768$

Πίνακας 4.1: Δοκιμαστικές συναρτήσεις κόστους.

Cr \ F	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
0.1	↓	↓	↓	↓	↑	↑	↑	↑	↑	↑
0.2	↓	↓	↓	↓	↓	↑	↑	↑	↑	↑
0.3	↓	↓	↓	↓	↓	↑	↑	↑	↑	↑
0.4	↓	↓	↓	↓	↓	↑	↑	↑	↑	↑
0.5	↓	↓	↓	↓	↓	↑	↑	↑	↑	↑
0.6	↓	↓	↓	↓	↓	↓	↑	↑	↑	↑
0.7	↓	↓	↓	↓	↓	↓	↓	↑	↑	↑
0.8	↓	↓	↓	↓	↓	↓	↓	↑	↑	↑
0.9	↓	↓	↓	↓	↓	↓	↓	↓	↑	↑
1.0	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓

Πίνακας 4.2: Αποτελέσματα για τη συνάρτηση Sphere (ff0).Τα βέλη προς τα πάνω δηλώνουν επιτυχή σύγκλιση.

Η πρώτη συνάρτηση προσαρμογής που χρησιμοποιήθηκε ήταν η συνάρτηση Sphere. Η συνάρτηση Sphere είναι μια εύκολη σχετικά συνάρτηση η οποία παρουσίασε τα καλύτερα αποτελέσματα. Στον ***Πίνακα 4.2*** δίνονται τα αποτελέσματα των εκτελέσεων για όλους τους συνδυασμούς *F* και *Cr*. Όπως είναι φανερό ο αλγόριθμος παρουσίασε σύγκλιση για τιμές του *F* μεγαλύτερες του 0.5 και τιμές του *Cr* σχετικά μικρές που όμως φτάνουν έως 0.9 για μεγάλες τιμές του *F*.

Cr \ F	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
0.1	↓	↓	↓	↓	↓	↑	↑	↑	↑	↑
0.2	↓	↓	↓	↓	↓	↑	↑	↑	↑	↑
0.3	↓	↓	↓	↓	↓	↑	↑	↑	↑	↑
0.4	↓	↓	↓	↓	↓	↓	↓	↑	↑	↑
0.5	↓	↓	↓	↓	↓	↓	↓	↑	↑	↑
0.6	↓	↓	↓	↓	↓	↓	↓	↑	↑	↑
0.7	↓	↓	↓	↓	↓	↓	↓	↑	↑	↑
0.8	↓	↓	↓	↓	↓	↓	↓	↑	↑	↑
0.9	↓	↓	↓	↓	↓	↓	↓	↓	↓	↑
1.0	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓

Πίνακας 4.3: Αποτελέσματα για τη συνάρτηση Ackley (ff1). Τα βέλη προς τα πάνω δηλώνουν επιτυχή σύγκλιση.

Η συνάρτηση Ackley παρουσίασε επίσης σχετικά καλά αποτελέσματα. Ο αλγόριθμος συνέκλινε όπως φαίνεται στον ***Πίνακα 4.3*** για τιμές του *F* μεγαλύτερες του 0.6 και τιμές του *Cr* από 0.3 έως και 0.9.

Ο αλγόριθμος, χρησιμοποιώντας για συνάρτηση προσαρμογής την συνάρτηση Rastrigin, συνέκλινε για πολύ λιγότερους συνδυασμούς των παραμέτρων του ΔΕ αλγορίθμου, που όμως παραμένουν για το *F* σε τιμές μεγαλύτερες του 0.6, ενώ για το *Cr* τιμές έως 0.3. Στον ***Πίνακα 4.4*** δίνονται τα αποτελέσματα της σύγκλισης για όλους τους δυνατούς συνδυασμούς.

Cr \ F	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
0.1	↓	↓	↓	↓	↓	↑	↑	↑	↑	↑
0.2	↓	↓	↓	↓	↓	↑	↑	↑	↑	↓
0.3	↓	↓	↓	↓	↓	↓	↑	↑	↑	↓
0.4	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
0.5	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
0.6	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
0.7	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
0.8	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
0.9	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
1.0	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓

Πίνακας 4.4: Αποτελέσματα για τη συνάρτηση Rastrigin (ff5). Τα βέλη προς τα πάνω δηλώνουν επιτυχή σύγκλιση.

Cr \ F	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
0.1	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
0.2	↓	↓	↓	↓	↓	↑	↓	↓	↓	↓
0.3	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
0.4	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
0.5	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
0.6	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
0.7	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
0.8	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
0.9	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
1.0	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓

Πίνακας 4.5:Αποτελέσματα για τη συνάρτηση Griewank (ff4). Τα βέλη προς τα πάνω δηλώνουν επιτυχή σύγκλιση.

Cr \ F	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
0.1	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
0.2	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
0.3	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
0.4	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
0.5	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
0.6	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
0.7	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
0.8	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
0.9	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
1.0	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓

Πίνακας 4.6: Αποτελέσματα για τη συνάρτηση Rosenbrock (ff2). Τα βέλη προς τα πάνω δηλώνουν επιτυχή σύγκλιση.

Οι επόμενες δύο συναρτήσεις δεν έδωσαν ικανοποιητικά αποτελέσματα, αφού χρησιμοποιώντας την Griewank ο αλγόριθμος συνέκλινε μόνο μια φορά (***Πίνακας 4.5)***, ενώ με την συνάρτηση Rosenbrock ο αλγόριθμος δεν παρουσίασε σύγκλιση για κανένα συνδυασμό τιμών των *F* και *Cr* (***Πίνακας 4.6***). Αυτό οφείλεται ίσως στη μεγαλύτερη πολυπλοκότητα των συναρτήσεων αλλά κυρίως στο μικρό μέγεθος του πληθυσμού που χρησιμοποιήθηκε.

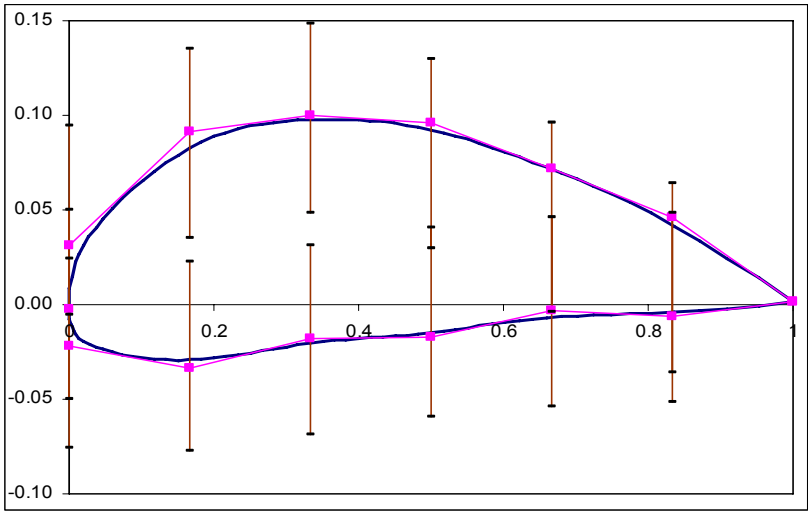
Με την παραπάνω μελέτη έγινε φανερό ότι η καλύτερη συμπεριφορά εντοπίζεται για μεγάλες τιμές του F και μικρές τιμές του Cr. Συνεπώς οι συνδυασμοί για $F \in [0.6, 0.9]$ και $Cr \in [0.1, 0.3]$ δίνουν στις περισσότερες των περιπτώσεων ικανοποιητικά αποτελέσματα. Οι συναρτήσεις Rosenbrock και Griewank παρόλο που δεν παρουσίασαν σύγκλιση, παρουσίασαν καλύτερη συμπεριφορά για τις τιμές των *F* και *Cr* που προαναφέρθηκαν.

Πρέπει να σημειωθεί εδώ, ότι για τις παραπάνω συναρτήσεις η χρονική απόδοση του αλγορίθμου σε σχέση με τον σειριακό δεν ήταν ικανοποιητική, λόγω της πολύ μικρής διάρκειας εκτέλεσης των συναρτήσεων. Ο χρόνος που απαιτείται για τις συνδέσεις και την επικοινωνία των υπολογιστών μέσω αρχείων κειμένου είναι σημαντικά υπολογίσιμος σε σχέση με αυτόν της εκτέλεσης των συναρτήσεων προσαρμογής, γεγονός που καθιστά την συγκεκριμένη παράλληλη εφαρμογή αναποτελεσματική για αυτού του είδους των προβλημάτων. Όσον αφορά στην ποιοτική απόδοση του αλγορίθμου, τα αποτελέσματα ήταν εξίσου καλά και κάποιες φορές καλύτερα, σε σχέση με αυτά του σειριακού.

4.8.2 Σύγκριση παράλληλου και σειριακού Διαφορικού Εξελικτικού αλγορίθμου

Προκειμένου να εξεταστεί περεταίρω η συμπεριφορά του παράλληλου ΔΕ αλγόριθμου, ο κώδικας δοκιμάστηκε σε σύγκριση με την σειριακή εφαρμογή του ίδιου ΔΕ αλγόριθμου σε δύο πραγματικά προβλήματα βελτιστοποίησης σχεδίασης αεροτομών.

Για τον παράλληλο κώδικα, χρησιμοποιήθηκαν 5 υπολογιστές, από τους οποίους ο κεντρικός αποτελείται από έναν επεξεργαστή ενώ οι υπόλοιποι από δύο επεξεργαστές ο καθένας (σύνολο 9 επεξεργαστών).

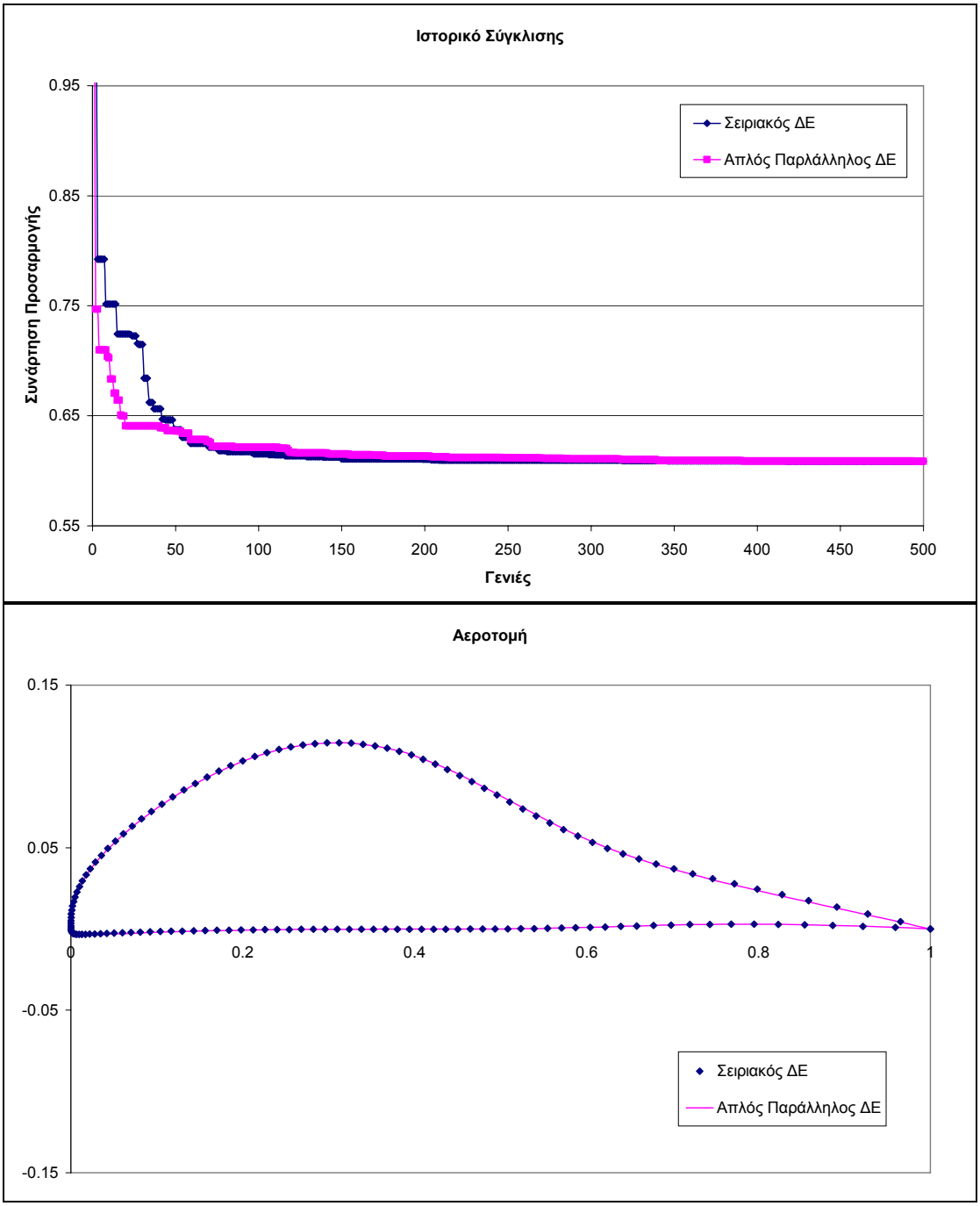


Εικόνα 4.8: Αναπαράσταση αεροτομής με τη χρήση καμπύλης B-Spline.

Η παραμετροποίηση της αεροτομής βασίζεται σε μια ανοιχτή B-Spline καμπύλη 3ου βαθμού με *2M+1* σημεία ελέγχου, όπου *M* είναι το πλήθος των σημείων ελέγχου σε κάθε πλευρά της αεροτομής. Το αρχικό και το τελικό σημείο ελέγχου έχουν τοποθετηθεί στο x =1.0. Τρία διαδοχικά σημεία είναι τοποθετημένα στο x =0 (***Εικόνα 4.8***). Με αυτόν τον τρόπο η ακμή προσβολής της καμπύλης που δημιουργείται βρίσκεται πάντα πάνω στον άξονα y. Το πρώτο και το τελευταίο σημείο ελέγχου, που αντιστοιχούν στην ακμή εκφυγής της αεροτομής, είναι σταθερά κατά τη διάρκεια της διαδικασίας βελτιστοποίησης. Οι x συντεταγμένες των υπόλοιπων σημείων ελέγχου είναι επίσης σταθερές και μόνο οι y συντεταγμένες είναι ελεύθερες να κινηθούν, δημιουργώντας έτσι *2M-1* ανεξάρτητες μεταβλητές σχεδίασης. Όσον αφορά στις συντεταγμένες x, τα σημεία ελέγχου είναι ομοιόμορφα κατανεμημένα κατά μήκος του άξονα x μεταξύ ακμής προσβολής και ακμής εκφυγής. Όλα τα σημεία ελέγχου, εκτός από τα σταθερά, επιτρέπεται να κινούνται σε κάθετα τμήματα, με το εύρος τους να καθορίζεται από τον χρήστη (***Εικόνα 4.8***).

Και στα δύο πειράματα χρησιμοποιήθηκαν 11 ανεξάρτητες μεταβλητές σχεδίασης. Κάθε υποψήφια λύση μεταφράζεται αυτόματα σε αεροτομή, μέσω παραμετρικών B-Spline συναρτήσεων [61]. Το πρώτο πρόβλημα ορίζεται ως βελτιστοποίηση του σχήματος μιας αεροτομής, με σκοπό την μεγιστοποίηση της άνωσης και την ελαχιστοποίηση της οπισθέλκουσας, κάτω από προκαθορισμένες συνθήκες λειτουργίας. Οι συνθήκες ροής που χρησιμοποιήθηκαν είναι οι ακόλουθες: $a_\infty = 8^o$, $\mathrm{Re}_c = 1200000$, $M_\infty = 0.18$. Το λογισμικό αξιολόγησης είναι το XFOIL του M. Drela [62], το οποίο συνδυάζει την ταχύτητα και την ακρίβεια των μεθόδων ιδιόμορφων σημείων υψηλής τάξης (high order panel methods) με πλήρως συζευγμένη μέθοδο αλληλεπίδρασης μη συνεκτικής ροής με οριακό στρώμα (fully-coupled viscous-inviscid interaction method). Το λογισμικό είναι κατάλληλο για την επίλυση ροών μικρών αριθμών Reynolds, όπως στην συγκεκριμένη περίπτωση. Τα διαστήματα μέσα στα οποία κινούνται οι μεταβλητές σχεδίασης καθορίζονται έτσι ώστε να ικανοποιείται η απαίτηση για διατήρηση του μέγιστου πάχους της βέλτιστης αεροτομής όσο το δυνατό πιο κοντά σε μια προκαθορισμένη τιμή. Σε αυτό το πρόβλημα το μέγιστο πάχος που τέθηκε ως στόχος ήταν το 12%. Το πρόβλημα διατυπώθηκε ως πρόβλημα ελαχιστοποίησης και ως συνάρτηση κόστους ορίστηκε η $f = 100 \cdot C_d / C_l$. Το ιστορικό σύγκλισης του σειριακού και του παράλληλου αλγορίθμου καθώς και οι καλύτερες λύσεις (αεροτομές) που προέκυψαν δίνονται στην ***Εικόνα 4.9***.

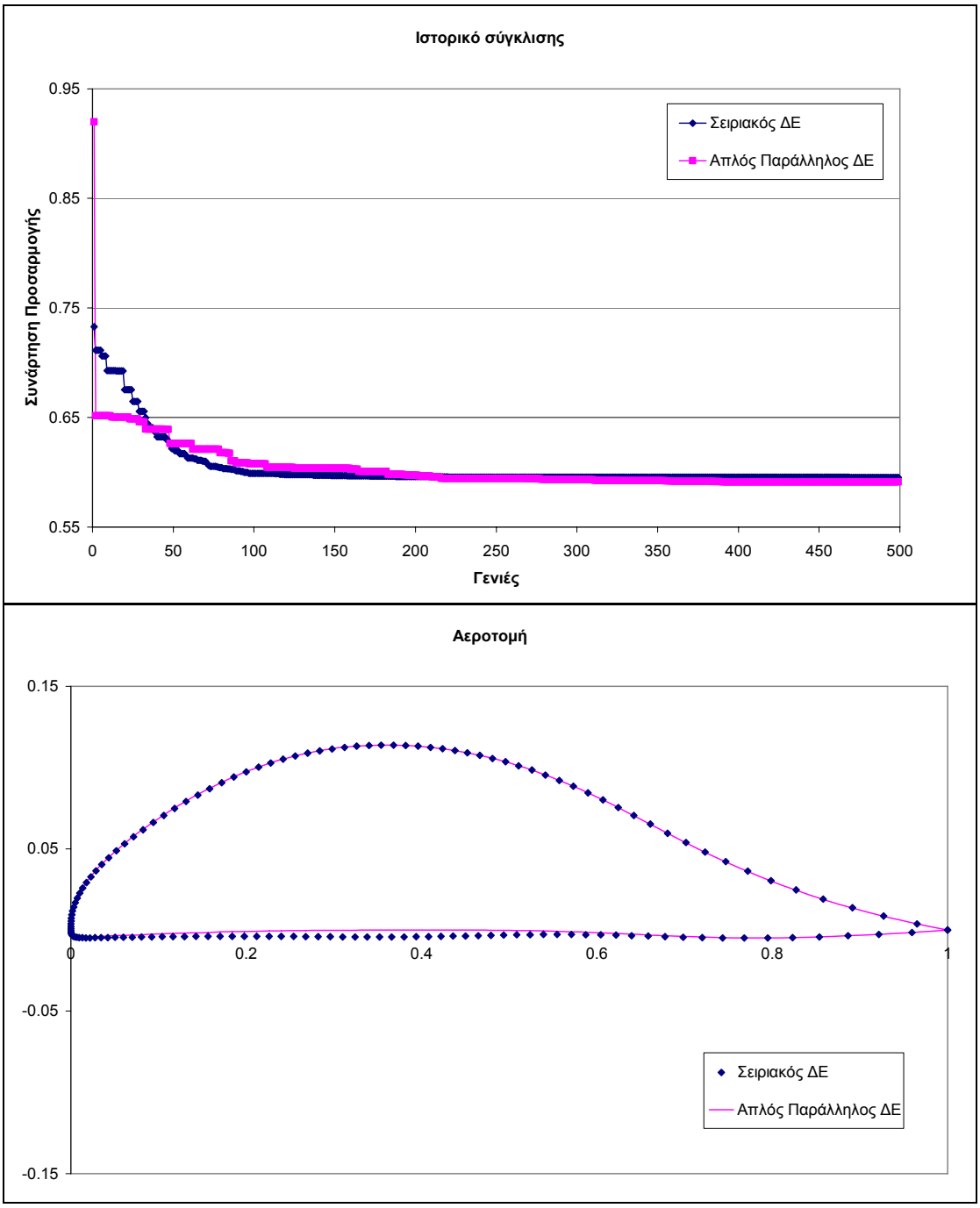
Για τους δύο αλγόριθμους χρησιμοποιήθηκε το ίδιο μέγεθος πληθυσμού ίσο με 17, το ίδιο πλήθος γενεών ίσο με 500 και οι τιμές των παραμέτρων F και Cr ήταν 0.8 και 0.3 αντίστοιχα. Και οι δύο αλγόριθμοι συνέκλιναν σε σχεδόν όμοιες λύσεις, ωστόσο ο σειριακός ολοκλήρωσε την βελτιστοποίηση (500 γενιές) σε 2906 δευτερόλεπτα, ενώ ο παράλληλος σε 758 δευτερόλεπτα. Η επιτάχυνση που επέφερε ο παράλληλος αλγόριθμος ήταν 3.83 φορές έναντι του σειριακού. Θα πρέπει να σημειωθεί εδώ ότι στην παράλληλη εκτέλεση το πλήθος των επεξεργαστών που χρησιμοποιήθηκαν ήταν 9 έναντι του ενός επεξεργαστή του σειριακού. Η επιτάχυνση που επιτεύχθηκε δεν ήταν ανάλογη του πλήθους των επεξεργαστών λόγω της μικρής διάρκειας εκτέλεσης του λογισμικού υπολογισμού της συνάρτησης κόστους σε σχέση με τον χρόνο επικοινωνίας μεταξύ των υπολογιστών.



Εικόνα 4.9: Ιστορικό σύγκλισης για το πρώτο πρόβλημα βελτιστοποίησης αεροτομής (επάνω). Οι καλύτερες λύσεις (αεροτομές) που υπολογίστηκαν με τους δύο αλγορίθμους (κάτω).

Το δεύτερο πρόβλημα ορίζεται ως βελτιστοποίηση του σχήματος μιας αεροτομής, με σκοπό την μεγιστοποίηση της άνωσης και την ελαχιστοποίηση της οπισθέλκουσας, για διαφορετικές γωνίες προσβολής. Οι συνθήκες ροής που χρησιμοποιήθηκαν είναι οι ακόλουθες: $a_{\infty,i} = 3^o, 4^o, 5^o, 6^o, 7^o$, $\mathrm{Re}_c = 1500000$, $M_\infty = 0.18$. Κάθε υποψήφια λύση αξιολογείται για κάθε μια από τις παραπάνω γωνίες προσβολής, χρησιμοποιώντας το ίδιο λογισμικό όπως και στο πρώτο πείραμα. Χρησιμοποιήθηκαν επίσης τα ίδια όρια για τις μεταβλητές σχεδίασης των σημείων ελέγχου, το ίδιο μέγεθος πληθυσμού και το ίδιο πλήθος γενεών τόσο για το σειριακό όσο και για τον παράλληλο αλγόριθμο.

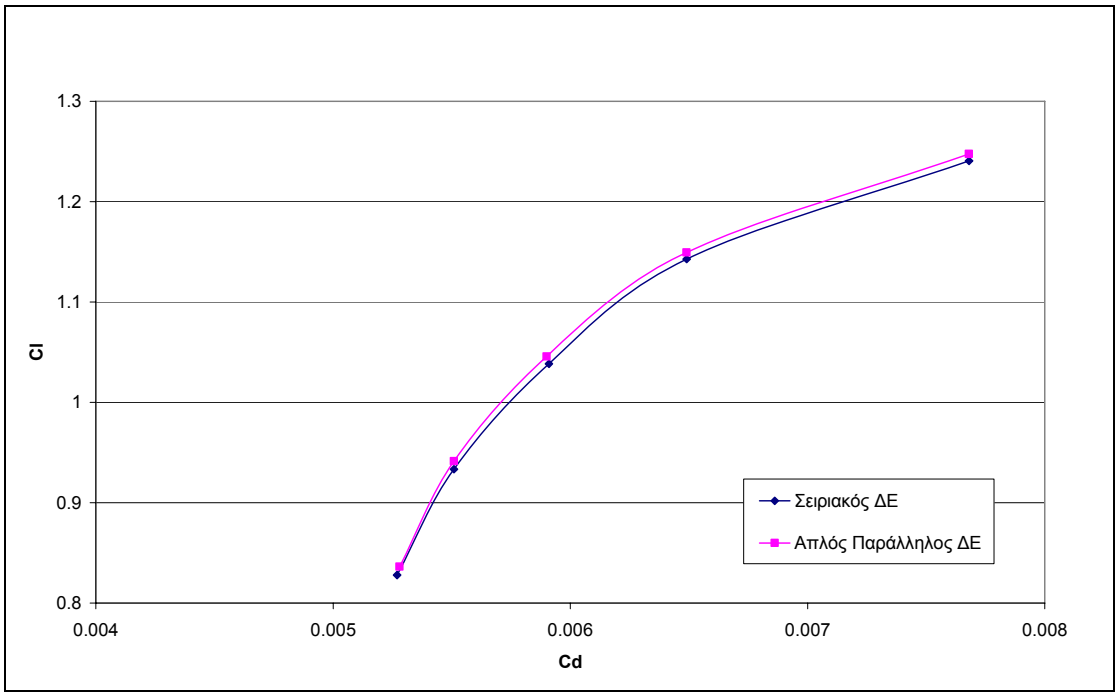
Το πρόβλημα διατυπώθηκε ως πρόβλημα ελαχιστοποίησης και ως συνάρτηση κόστους ορίστηκε η $f = 100 \cdot \sum_{i=1}^{5} C_{d,i} \Big/ \sum_{i=1}^{5} C_{l,i}$.



Εικόνα 4.10: Ιστορικό σύγκλισης για το δεύτερο πρόβλημα βελτιστοποίησης αεροτομής (επάνω). Οι καλύτερες λύσεις (αεροτομές) που υπολογίστηκαν με τους δύο αλγορίθμους (κάτω).

Όπως και στο πρώτο πείραμα και οι δύο αλγόριθμοι συνέκλιναν σε σχεδόν όμοιες λύσεις με τον σειριακό να απαιτεί 4463 δευτερόλεπτα για την εκτέλεση, ενώ ο παράλληλος χρειάστηκε μόνο 823 δευτερόλεπτα Η επιτάχυνση που επέφερε ο παράλληλος αλγόριθμος ήταν 5.42 φορές έναντι του σειριακού, για 9 επεξεργαστές. Το ιστορικό σύγκλισης της διαδικασίας καθώς και οι καλύτερες λύσεις για τους δύο

αλγορίθμους δίνονται στην ***Εικόνα 4.10***, ενώ τα αποτελέσματα άνωσης - οπισθέλκουσας (διάγραμμα πολικών) για τις καλύτερες λύσεις που προέκυψαν από τον σειριακό και τον παράλληλο αλγόριθμο δίνονται στη ***Εικόνα 4.11***.



Εικόνα 4.11: Διάγραμμα άνωσης - οπισθέλκουσας για τις καλύτερες λύσεις που προέκυψαν από τον σειριακό και τον παράλληλο αλγόριθμο.

4.9 Συμπεράσματα

Στη συγκεκριμένη προσπάθεια παραλληλοποίησης του ΔΕ αλγόριθμου, δύο ήταν τα βασικά προβλήματα που παρουσιάστηκαν. Το πρώτο, έγκειται στην κλήση και εκτέλεση των «Slave» αλγορίθμων, από τον «Master», στους απομακρυσμένους υπολογιστές, καθώς κάτι τέτοιο αντιβαίνει στη λογική ασφάλειας των Windows. Αρχικά έγιναν κάποιες προσπάθειες χρησιμοποίησης του πρωτοκόλλου Telnet των Windows, που όμως δεν επέφεραν αποτελέσματα, καθώς δεν ήταν δυνατή η σύνδεση περισσότερων από δύο υπολογιστές. Εν τέλει, χρησιμοποιήθηκαν τα εκτελέσιμα αρχεία για εκτέλεση και διακοπή απομακρυσμένων προγραμμάτων που προσφέρει το ελεύθερο λογισμικό PsTools, τα οποία βοήθησαν να ξεπεραστεί αυτό το πρόβλημα.

Το δεύτερο πρόβλημα που χρειάστηκε να αντιμετωπιστεί, ήταν η ταυτόχρονη προσπέλαση αρχείων. Όπως αναφέρθηκε, η επικοινωνία πραγματοποιήθηκε με την παρουσία ενός κοινόχρηστου φακέλου χρωμοσωμάτων, στον οποίο είχαν όλοι οι επιμέρους υπολογιστές πρόσβαση. Ο ασύγχρονος χαρακτήρας της παράλληλης εφαρμογής και ο τελεστής μετάλλαξης του ΔΕ αλγόριθμου, που απαιτεί την λήψη τριών τυχαίων χρωμοσωμάτων για την μετάλλαξη του χρωμοσώματος, έκανε το φαινόμενο αρκετά έντονο. Με τη χρήση ελέγχων «χρήσης αρχείου» και διαχείριση σφαλμάτων το πρόβλημα μειώθηκε σημαντικά, αλλά δεν κατέστη δυνατό να εξαλειφθεί τελείως.

Εν γένει, το λογισμικό παράλληλου ΔΕ αλγορίθμου που αναπτύχθηκε, παρόλο που παρουσιάζει κάποια αστάθεια λόγω του προβλήματος που περιγράφηκε, σε συναρτήσεις με σημαντικό χρόνο υπολογισμού της συνάρτησης προσαρμογής, δίνει

πολύ ικανοποιητικά αποτελέσματα και ως προς το χρόνο και ως προς την ποιότητα της λύσης, σε σύγκριση με τον ίδιο ΔΕ αλγόριθμο σε σειριακή εφαρμογή.

ΚΕΦΑΛΑΙΟ 5

ΠΑΡΑΛΛΗΛΟΣ ΔΙΑΦΟΡΙΚΟΣ ΕΞΕΛΙΚΤΙΚΟΣ ΑΛΓΟΡΙΘΜΟΣ ΜΕ ΧΡΗΣΗ ΤΟΥ ΠΡΟΤΥΠΟΥ MPI

5.1 Γενικά

Η δεύτερη προσπάθεια παραλληλοποίησης ΔΕ αλγορίθμου υλοποιήθηκε χρησιμοποιώντας το πρότυπο MPI-2, προκειμένου να πραγματοποιηθεί η επικοινωνία μεταξύ των υπολογιστών και να αποφευχθούν τα προβλήματα ταυτόχρονης προσπέλασης αρχείων. Ο ΔΕ αλγόριθμος, στον οποίο εφαρμόστηκε η παραλληλοποίηση, ήταν ο ίδιος με την προηγούμενη προσπάθεια στο ίδιο δίκτυο υπολογιστών, ενώ σε αυτή την περίπτωση η εφαρμογή ήταν σύγχρονη και το λογισμικό αναπτύχθηκε σε Compaq Visual Fortran 6.0. Δεν υλοποιήθηκε κάποιο γραφικό περιβάλλον, συνεπώς οι απαραίτητες πληροφορίες εισάγονται στο πρόγραμμα μέσω αρχείων κειμένου και τα αποτελέσματα καθώς και η σύγκλιση του αλγορίθμου εξάγονται και αυτά σε αρχεία.

Η ανάπτυξη παράλληλου προγράμματος με τη χρήση του προτύπου MPI ήταν πιο αποτελεσματική, καθώς δημιουργήθηκε ένα σταθερό πρόγραμμα χωρίς προβλήματα επικοινωνίας και με ικανοποιητική απόδοση όσον αφορά στο χρόνο εκτέλεσης. Θα πρέπει να τονιστεί ότι για την εκτέλεση της εφαρμογής αυτής, απαιτείται η εγκατάσταση λογισμικού σε όλους του υπολογιστές του δικτύου και η ενσωμάτωση των ρουτινών του MPI-2 μέσα στον κώδικα του ΔΕ αλγορίθμου.

5.2 Πρότυπο MPI

Το πρότυπο MPI αποτελεί στην ουσία ένα σύνολο ρουτινών οι οποίες καλούνται από τον χρήστη μέσα στο πρόγραμμα, σε συγκεκριμένα σημεία, για να εκτελέσουν συγκεκριμένες διαδικασίες επικοινωνίας μεταξύ των επεξεργαστών [63]. Το πρότυπο MPI-2 επιτρέπει επιπλέον, τη δυναμική εισαγωγή νέων υπολογιστών κατά τη διάρκεια της εκτέλεσης του προγράμματος. Στη συνέχεια θα αναφερόμαστε και στα δύο αυτά πρότυπα ως MPI, καθώς τα στοιχεία που θα παρατεθούν αφορούν και στα δύο.

5.2.1 Ιστορική Αναδρομή του MPI

Η διαδικασία της δημιουργίας ενός προτύπου, που θα ενεργοποιήσει τη δυνατότητα ανάπτυξης εφαρμογών μεταφοράς μηνύματος, ξεκίνησε στο Workshop on Message Passing Standardization τον Απρίλιο του 1992 και στο MPI Forum, που οργανώθηκε στο συνέδριο Supercomputing το 1992. Κατά τη διάρκεια των δεκαοκτώ επόμενων μηνών η ομάδα ανάπτυξης MPI συνέχισε τις συναντήσεις της και η πρώτη έκδοση του προτύπου MPI (MPI Standard Version 1.0) ολοκληρώθηκε το Μάιο του 1994 [64], [65]. Το πρότυπο αυτό, αναπτύχθηκε προκειμένου να ικανοποιήσει τις ανάγκες της κοινότητας για ένα ανοικτό και ευέλικτο πρότυπο, με ρουτίνες μεταφοράς μηνύματος για την ενδοεπικοινωνία μεταξύ των επεξεργαστών σε παράλληλους υπολογιστές. Την άνοιξη του 1995 διατέθηκε η έκδοση 1.1, η οποία περιείχε κάποιες διευκρινήσεις και βελτιώσεις [66]. Μια ακόμα συνάντηση της ομάδας ανάπτυξης

MPI πραγματοποιήθηκε το 1998, για να διορθώσει μικροπροβλήματα και να προσθέσει κάποια επιπλέον χαρακτηριστικά, το αποτέλεσμα της οποίας ήταν το πρότυπο MPI-2 [67].

Στη συνέχεια αναπτύσσονται τα βασικά στοιχεία για το MPI. Ο απαιτητικός αναγνώστης μπορεί να αναζητήσει περισσότερες λεπτομέρειες στις αναφορές [68], [69].

5.2.2 Σύντομη Περίληψη του Προτύπου MPI

Το MPI είναι ένα εργαλείο προγραμματιστικών εφαρμογών μεταφοράς μηνύματος, που περιλαμβάνει πρωτόκολλο και εννοιολογικές προδιαγραφές για το πώς τα στοιχεία του πρέπει να συμπεριφέρονται σε κάθε εφαρμογή. Το MPI μπορεί να αντιμετωπίσει τόσο μεταφορά μηνύματος μεταξύ ζεύγους «εργασιών», όσο και ομαδικούς χειρισμούς. Επιπλέον, το MPI παρέχει υποστήριξη στις διαδικασίες σε δύο επίπεδα. Πρώτον, οι διαδικασίες ονομάζονται βάσει της κατάταξης της ομάδας στην οποία λαμβάνει χώρα η επικοινωνία. Δεύτερον, εικονικές τοπολογίες, επιτρέπουν την ονομασία των διαδικασιών βάσει γραφήματος ή Καρτεσιανού επιπέδου, το οποίο βοηθάει στην εννοιολογική συσχέτιση των διαδικασιών για την κατάλληλη μεταφορά μηνύματος με πιο αποδοτικό τρόπο. Οι δίαυλοι επικοινωνίας, οι οποίοι διακινούν τις πληροφορίες, παρέχουν ένα σημαντικό μέτρο ασφάλειας το οποίο είναι απαραίτητο και χρήσιμο για την ανάπτυξη παράλληλου κώδικα προσανατολισμένου σε βιβλιοθήκες [63].

Το MPI παρέχει επίσης τρεις εναλλακτικές κλάσεις υπηρεσιών, περιβαλλοντική αναζήτηση πληροφορίας, βασικές πληροφορίες που αφορούν στο χρόνο για την μέτρηση της απόδοσης της εφαρμογής και δημιουργία περιβάλλοντος για εξωτερική παρακολούθηση της διαδικασίας. Το MPI πραγματοποιεί ετερογενή μετατροπή δεδομένων, απαιτώντας τον καθορισμό του τύπου δεδομένων σε κάθε διαδικασία επικοινωνίας. Παρέχει τόσο τους ενσωματωμένους, όσο και άλλους καθορισμένους από τον χρήστη τύπους δεδομένων.

Το MPI ολοκληρώνει τη λειτουργικότητα του με τη δυνατότητα δημιουργίας αντικειμένων, με καθορισμένα εργαλεία κατασκευής και καταστροφής, γεγονός που του δίνει ένα προσανατολισμένο σε αντικείμενα χαρακτήρα. Τα αντικείμενα αυτά περιέχουν ομάδες στοιχείων, εργαλείων επικοινωνίας και δίνουν τη δυνατότητα για την ανάπτυξη ασύγχρονων εφαρμογών.

Το MPI παρέχει υποστήριξη τόσο σε SPMD όσο και MPMD αρχιτεκτονικές παράλληλου προγραμματισμού. Επιπλέον, μπορεί να υποστηρίξει εσωτερικούς υπολογισμούς μέσω διαδικασιών ενδοεπικοινωνίας. Η μορφή της ροής δεδομένων των υπολογισμών μπορεί επίσης να καθοριστεί και να πραγματοποιηθεί μέσω διαύλων ενδοεπικοινωνίας.

Σημαντικό σημείο για την κατανόηση του MPI αποτελεί το πώς ένα πρόγραμμα αναπαράγεται σε μεμονωμένα προγράμματα (threads) τα οποία εκτελούνται ξεχωριστά σε κάθε επεξεργαστή. Στην ουσία, όταν η εντολή εκτέλεσης του MPI καλέσει το παράλληλο πρόγραμμα, *ν* πανομοιότυπα αντίτυπα του προγράμματος «στέλνονται» να εκτελεστούν στους υπολογιστές του δικτύου. Τα προγράμματα αυτά, που προκύπτουν από την αναπαραγωγή του αρχικού, θα αναφέρονται ως «εργασίες»

και το πλήθος τους ορίζεται από τον χρήστη. Εάν δεν καθοριστεί κάποιου είδους επικοινωνία μεταξύ των «εργασιών» από το χρήστη, τα υποπρογράμματα αυτά τρέχουν ανεξάρτητα χωρίς να γνωρίζει κανένα την ύπαρξη του άλλου. Κάθε «εργασία» εκτελεί τη διαδικασία του προγράμματος όπως θα γινόταν σε ένα σειριακό υπολογιστή, εκτελώντας το πρόγραμμα γραμμή-γραμμή.

Όπως αναφέρθηκε παραπάνω, το MPI αποτελεί ένα σύνολο ρουτινών. Συνολικά, πάνω από εκατό διαφορετικές ρουτίνες εμπεριέχονται στα πρότυπα MPI και MPI-2. Παρόλα αυτά το MPI μπορεί να χρησιμοποιηθεί και από αρχάριους, αφού ένα μικρό σύνολο ρουτινών αρκεί για τυπικές εφαρμογές παράλληλων προγραμμάτων [70]. Στην πραγματικότητα οι βασικές ρουτίνες, που απαιτούνται για την υλοποίηση ενός απλού παράλληλου προγράμματος, είναι έξι, οι δύο από τις οποίες απλά «ανοίγουν» και «κλείνουν» το MPI. Οι έξι βασικές ρουτίνες του MPI είναι οι ακόλουθες [70]:

MPI_INIT(...)	Αρχικοποιεί το MPI
MPI_COMM_SIZE(...)	Καθορίζει το πλήθος των «εργασιών»
MPI_COMM_RANK(...)	Ταυτοποιεί τις «εργασίες»
MPI_SEND(...)	Στέλνει δεδομένα
MPI_RECV(...)	Λαμβάνει δεδομένα
MPI_FINALIZE(...)	Τερματίζει το MPI

Κάθε μια από τις παραπάνω εισάγεται, με τα απαιτούμενα ορίσματα, στον κώδικα για να εκτελέσει την επιθυμητή διαδικασία. Οι ρουτίνες επικοινωνίας μπορούν να διαχειριστούν ακεραίους, απλής η διπλής ακρίβειας πραγματικούς αριθμούς ή πίνακες οποιασδήποτε διάστασης. Στο Παράρτημα δίνονται λεπτομερείς οδηγίες για την ανάπτυξη και εκτέλεση ενός τυπικού παράλληλου προγράμματος, ενώ στο [71] ο αναγνώστης μπορεί να βρει όλες τις ρουτίνες του προτύπου MPI.

Οι ρουτίνες του MPI επιτρέπουν στον χρήστη να δίνει οδηγίες για τον τρόπο με τον οποίο θα εκτελεστεί το πρόγραμμα. Υπάρχουν συγκεκριμένες τεχνικές που χρησιμοποιούνται στα προγράμματα MPI, οι οποίες μπορούν να ορίσουν ποια κομμάτια του κώδικα θα εκτελεστούν από συγκεκριμένες «εργασίες». Για παράδειγμα, όταν καλείται η υπορουτίνα MPI_COMM_RANK, πραγματοποιεί μια κατάταξη και επιστρέφει ένα αναγνωριστικό για κάθε «εργασία», συνήθως αναφέρεται ως «myid», το οποίο μετράει από 0 έως *ν*, όπου *ν* το πλήθος των «εργασιών» που έχουν ανατεθεί στο δίκτυο. Συνεπώς, με κάποια εντολή «if» (π.χ. «if myid=0») μπορεί ο χρήστης να καθορίσει κάποιο σημείο του κώδικα που θέλει να εκτελεστεί από συγκεκριμένη «εργασία». Θα πρέπει εδώ να δοθεί προσοχή δεδομένου ότι κάτι τέτοιο μπορεί να προκαλέσει απώλεια συγχρονισμού μεταξύ των επιμέρους προγραμμάτων. Αυτό μπορεί να λυθεί χρησιμοποιώντας μια υπορουτίνα συγχρονισμού, π.χ. η MPI_BARRIER, η οποία περιμένει μέχρι όλες οι «εργασίες» να φτάσουν σε αυτό το σημείο του κώδικα.

Στις περισσότερες περιπτώσεις τα επιμέρους προγράμματα απαιτούν την ύπαρξη επικοινωνίας μεταξύ τους, κατά τη διάρκεια εκτέλεσης τους, για την ανταλλαγή πληροφοριών. Σε αυτήν την περίπτωση, πρέπει να χρησιμοποιηθούν κατάλληλές ρουτίνες επικοινωνίας (π.χ. MPI_SEND, MPI_RECV, MPI_BCAST), οι οποίες θα παρεμβληθούν στον κώδικα και θα διοχετεύσουν τις πληροφορίες με τον τρόπο που έχει καθοριστεί. Και εδώ πρέπει να δοθεί προσοχή, καθώς εάν οι «εργασίες» δεν

βρίσκονται στο ίδιο σημείο τη στιγμή της επικοινωνίας, μπορεί να προκληθούν προβλήματα.

5.2.3 Πακέτο Εφαρμογής του Προτύπου MPI - MPICH

Ένα από τα πιο δημοφιλή πακέτα εφαρμογής του προτύπου MPI είναι το λογισμικό MPICH (το πακέτο εφαρμογής του προτύπου MPI-2 είναι το λογισμικό MPICH2), το οποίο διατίθεται ελεύθερα [72].

Η ανάπτυξη λογισμικού που θα παρείχε μια εύχρηστη εφαρμογή του προτύπου MPI ξεκίνησε την ίδια περίοδο που αναπτυσσόταν η διαδικασία MPI. Η κύρια ιδέα ήταν να υπάρχει πρώιμη ανάδραση στις αποφάσεις που λαμβάνονταν από την ομάδα ανάπτυξης του MPI και να αναπτύσσεται άμεσα μια εφαρμογή, η οποία θα προσφέρει τη δυνατότητα πειραματισμού πάνω στα στοιχεία που είναι υπό ανάπτυξη. Στόχος της εφαρμογής ήταν, να περικλείει όλα τα συστήματα που υποστηρίζουν το μοντέλο μεταφοράς μηνύματος. Το MPICH αποτελεί μια πλήρη εφαρμογή του MPI, που σχεδιάστηκε για να είναι εύχρηστη και παράλληλα αποδοτική. Τα γράμματα «CH» στο MPICH προέρχονται από την λέξη χαμαιλέοντας (Chameleon), που συμβολίζει την συμβατότητα με κάθε περιβάλλον, γεγονός που το καθιστά εύχρηστο. Ένας δεύτερος στόχος από την αρχή της ανάπτυξης του, ήταν να θυσιαστεί όσο το δυνατότερο λιγότερη αποδοτικότητα, προκειμένου να παραμείνει εύχρηστο [63].

Το MPICH αποτελεί τόσο πεδίο έρευνας, όσο και πεδίο ανάπτυξης λογισμικού. Ως πεδίο έρευνας, έχει ως στόχο να εξερευνήσει μεθόδους, ώστε να γεφυρώσει το χάσμα μεταξύ του προγραμματιστή παράλληλων προγραμμάτων και της συμπεριφοράς του υπολογιστικού υλικού. Στο MPICH, υιοθετείται ο περιορισμός ότι το προγραμματιστικό περιβάλλον διεπαφής θα είναι το MPI, απορρίπτοντας περιορισμούς που αφορούν στην αρχιτεκτονική του μηχανήματος και διατηρώντας υψηλή απόδοση (μετρημένη σε όρους bandwith και latency για λειτουργίες μεταφοράς μηνύματος). Ως πεδίο ανάπτυξης λογισμικού, στόχος είναι να προάγει την υιοθέτηση του προτύπου MPI, παρέχοντας στους χρήστες δωρεάν, υψηλής απόδοσης εφαρμογή σε πλήθος πλατφορμών. Η αρχιτεκτονική του MPICH, καθώς και ο βαθμός στον οποίο επιτυγχάνονται οι στόχοι αυτοί συζητείται στο [63].

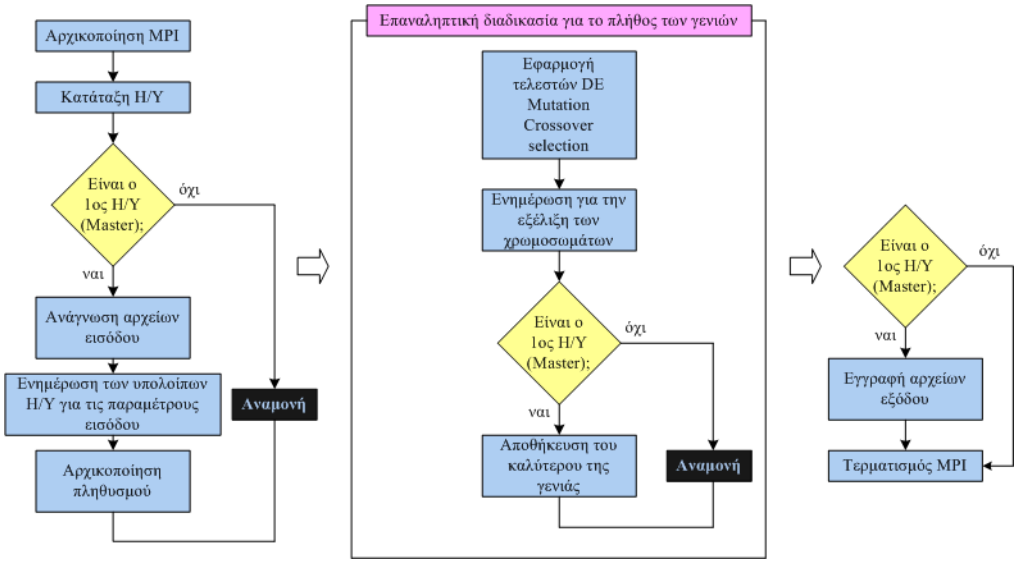
Κάποια άλλα πακέτα εφαρμογών MPI είναι τα ακόλουθα:

- Το LAM [73] διατίθεται από το Κέντρο Supercomputer Ohio και βρίσκει εφαρμογή σε ετερογενή δίκτυα Sun, DEC, SGI, IBM και σταθμούς εργασίας HP.
- Το CHIMP-MPI [74] διατίθεται από το Κέντρο Παράλληλης Υπολογιστικής του Εδιμβούργου και βρίσκει εφαρμογή σε δίκτυα Sun, SGI, DEC, IBM και σταθμούς εργασίας HP, σε μηχανήματα Meiko Computing Surface και σε Fujitsu AP-1000. Η ανάπτυξη του έχει βασιστεί στο CHIMP [75].
- Έρευνα πάνω σε συστήματα μεταφοράς μηνύματος συμπεριλαμβανομένου του MPI έχει γίνει στο Πολυτεχνείο του Μονάχου [76], [77].
- Το Unify [78], που αναπτύχθηκε και παρέχεται από το Πανεπιστήμιο του Μισισιπή, έχει βασίσει το MPI σε μια έκδοση του PVM [79], η οποία έχει τροποποιηθεί κατάλληλα. Το Unify επιτρέπει τον συνδυασμό του MPI και του PVM στο ίδιο πρόγραμμα.

5.3 Παρουσίαση και Δομή Λογισμικού

Το λογισμικό παράλληλου ΔΕ αλγόριθμου, που αναπτύχθηκε για τις ανάγκες αυτής της εργασίας, χρησιμοποιώντας το πρότυπο MPI-2 για την επικοινωνία μεταξύ των «εργασιών», αποτελεί σύγχρονη εφαρμογή του ίδιου απλού ΔΕ αλγορίθμου που αναπτύχθηκε στο Κεφάλαιο 4.

Όπως αναφέρθηκε παραπάνω δε δημιουργήθηκε κάποιο γραφικό περιβάλλον, συνεπώς για την είσοδο όλων των απαιτούμενων πληροφοριών καθώς και την έξοδο των αποτελεσμάτων χρησιμοποιούνται αρχεία κειμένου. Τα αρχεία που απαιτούνται είναι: ένα αρχείο με τις παραμέτρους του ΔΕ αλγόριθμου (πληθυσμό, γενιές, *F*, *Cr*), ένα αρχείο με το πλήθος των ανεξάρτητων μεταβλητών σχεδίασης του προβλήματος, συμπεριλαμβανομένων των ορίων τους, ένα αρχείο με το πλήθος των υπολογιστών του δικτύου με τα ονόματά τους, ένα αρχείο με το πλήθος των εκτελέσιμων αρχείων για τον υπολογισμό της συνάρτησης κόστους και τα ονόματα τους και τέλος τα εκτελέσιμα αρχεία που απαιτούνται για τον υπολογισμό της συνάρτησης κόστους. Μετά την εκτέλεση του προγράμματος παράγεται ένα αρχείο με όλα τα στοιχεία του τρεξίματος, συμπεριλαμβανομένου του χρόνου εκτέλεσης, ένα αρχείο με τις τιμές των μεταβλητών σχεδίασης του καλύτερου χρωμοσώματος και ένα αρχείο με το ιστορικό σύγκλισης του αλγορίθμου.



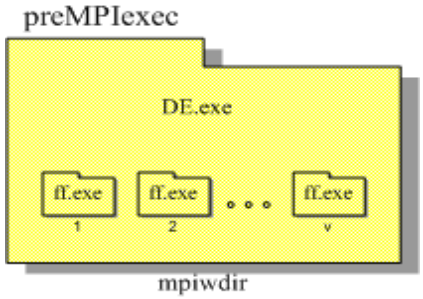
Εικόνα 5.1: Διάγραμμα λειτουργίας λογισμικού.

Τα προγράμματα MPI-2 για την εκτέλεση τους απαιτούν κλήση, μέσω γραμμής εντολών, του αρχείου mpiexec.exe (***Εικόνα 5.2***), το οποίο παρέχεται στο χρήστη με την εγκατάσταση του MPICH2 (βλ. Παράρτημα). Το εκτελέσιμο αρχείο του παράλληλου ΔΕ αλγόριθμου πρέπει να βρίσκεται σε κάθε υπολογιστή, ενώ το εκτελέσιμο της συνάρτησης κόστους, εφόσον ανατίθεται πάνω από ένα χρωμόσωμα σε κάθε υπολογιστή, πρέπει να βρίσκεται σε διαφορετικούς φακέλους για κάθε «εργασία». Εδώ, σε αντιστοιχία με τα προγράμματα «Slave» που περιγράφηκαν στο προηγούμενο κεφάλαιο, κάθε χρωμόσωμα ανατίθεται σε μια «εργασία». Για την εκτέλεση προγραμμάτων MPI-2 το πρόγραμμα αντιγράφεται σε όλους τους απομακρυσμένους υπολογιστές του δικτύου. Αν και κάτι τέτοιο δεν είναι απαραίτητο, ακολουθείται αυτή η τακτική, διαφορετικά υπάρχει περίπτωση να προκληθούν προβλήματα.

Για την αυτοματοποίηση της διαδικασίας κατασκευάστηκαν δύο βοηθητικά προγράμματα, το πρώτο εκ των οποίων εκτελείται πριν το κυρίως πρόγραμμα (preMPIexec), ενώ το δεύτερο μετά το πέρας του (postMPIexec). Το πρώτο δημιουργεί σε κάθε απομακρυσμένο υπολογιστή ένα φάκελο εργασίας, όπου αντιγράφει μέσα το κυρίως πρόγραμμα και δημιουργεί υποφακέλους με το εκτελέσιμο αρχείο της συνάρτησης κόστους, για κάθε χρωμόσωμα που ανατίθεται στον κάθε υπολογιστή (***Εικόνα 5.3***). Το δεύτερο διαγράφει όλα τα αρχεία που δημιουργήθηκαν για τις ανάγκες του προγράμματος στους απομακρυσμένους υπολογιστές.

Εικόνα 5.2: Κλήση mpiexec μέσω γραμμής εντολών για την εκτέλεση προγράμματος MPI.

Ο πληθυσμός, που ορίζεται από τον χρήστη, ισοκατανέμεται στους υπολογιστές, ενώ εάν υπάρχει υπόλοιπο, αυτό τοποθετείται στον τελευταίο υπολογιστή. Με την κλήση της εντολής mpiexec του πακέτου εφαρμογών MPICH2 τα εκτελέσιμα του προγράμματος, που έχουν τοποθετηθεί σε όλους τους υπολογιστές, εκκινούν τόσες φορές όσα τα χρωμοσώματα που έχουν ανατεθεί στον κάθε υπολογιστή και το κάθε ένα («εργασία») αναλαμβάνει την εξέλιξη ενός χρωμοσώματος, εφαρμόζοντας σε αυτό τους τελεστές του ΔΕ αλγόριθμου.

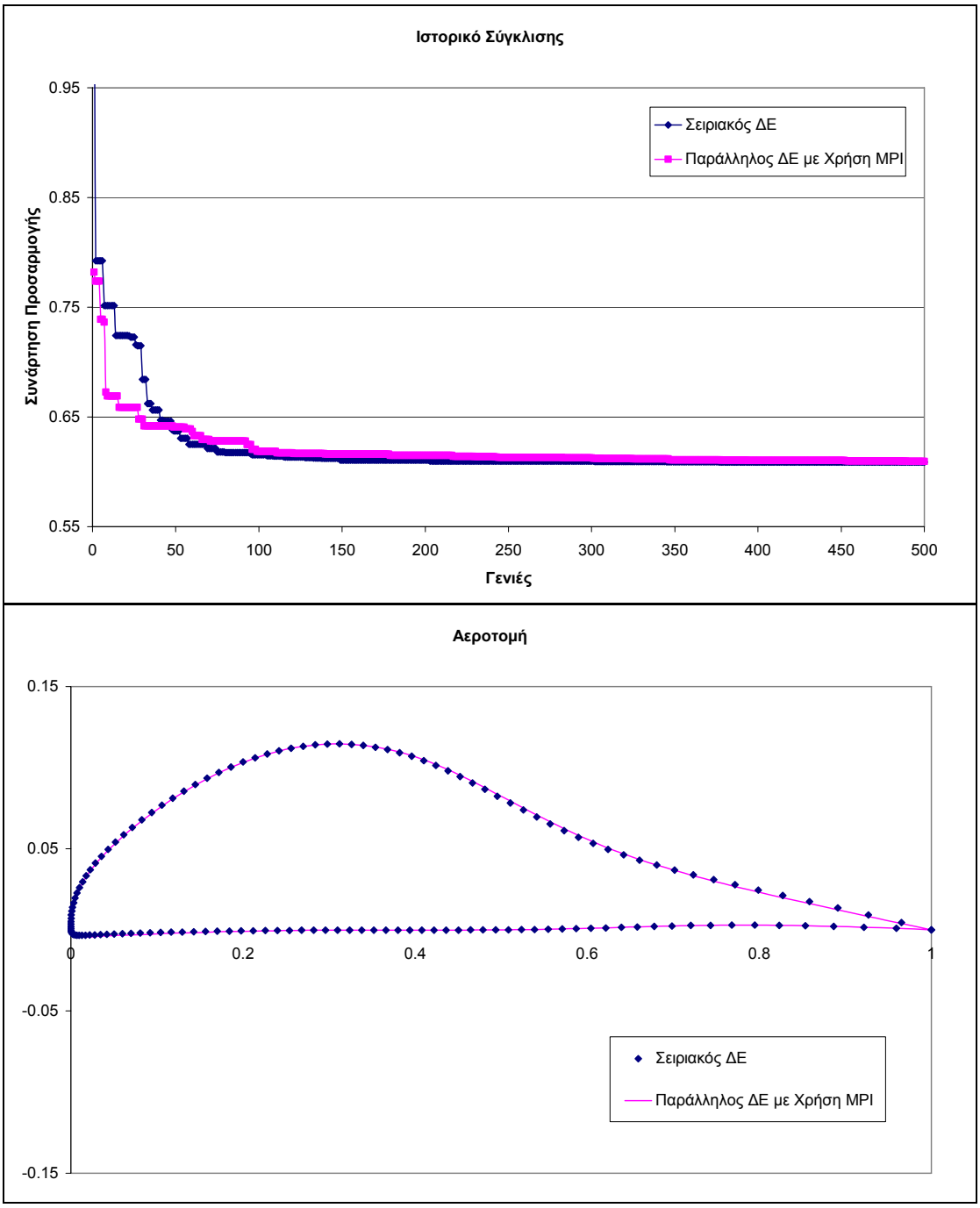


Εικόνα 5.3: Το πρόγραμμα preMPIexec δημιουργεί σε κάθε απομακρυσμένο υπολογιστή ένα φάκελο εργασίας.

Αρχικά, κάθε «εργασία» παίρνει κάποιο αναγνωριστικό (0, 1, 2, …, *ν*). Χάρη σε αυτό (χρησιμοποιώντας την τεχνική «if»), δίδεται η δυνατότητα στο πρόγραμμα να εκτελεί συγκεκριμένα κομμάτια για συγκεκριμένες «εργασίες», συνεπώς και για την πρώτη «εργασία» (if myid=0), που εδώ παίζει το ρόλο του «Master». Με αυτόν τον τρόπο η πρώτη εργασία με την εκκίνηση του προγράμματος, διαβάζει από τα αρχεία εισόδου τις πληροφορίες που απαιτούνται και μέσω εντολών MPI-2 τις μεταβιβάζει σε όλες τις υπόλοιπες. Στην συνέχεια όλες οι «εργασίες» συνεχίζουν τη εκτέλεση του προγράμματος παράλληλα και σύγχρονα, ενώ επικοινωνία μεταξύ τους πραγματοποιείται, μέσω εντολών MPI-2, στο τέλος κάθε γενιάς για να ενημερώσουν και να ενημερωθούν για την εξέλιξη των χρωμοσωμάτων. Μετά το πέρας των γενιών η πρώτη «εργασία» γράφει τα αρχεία εξόδου και το πρόγραμμα τερματίζεται.

5.4 Εφαρμογές

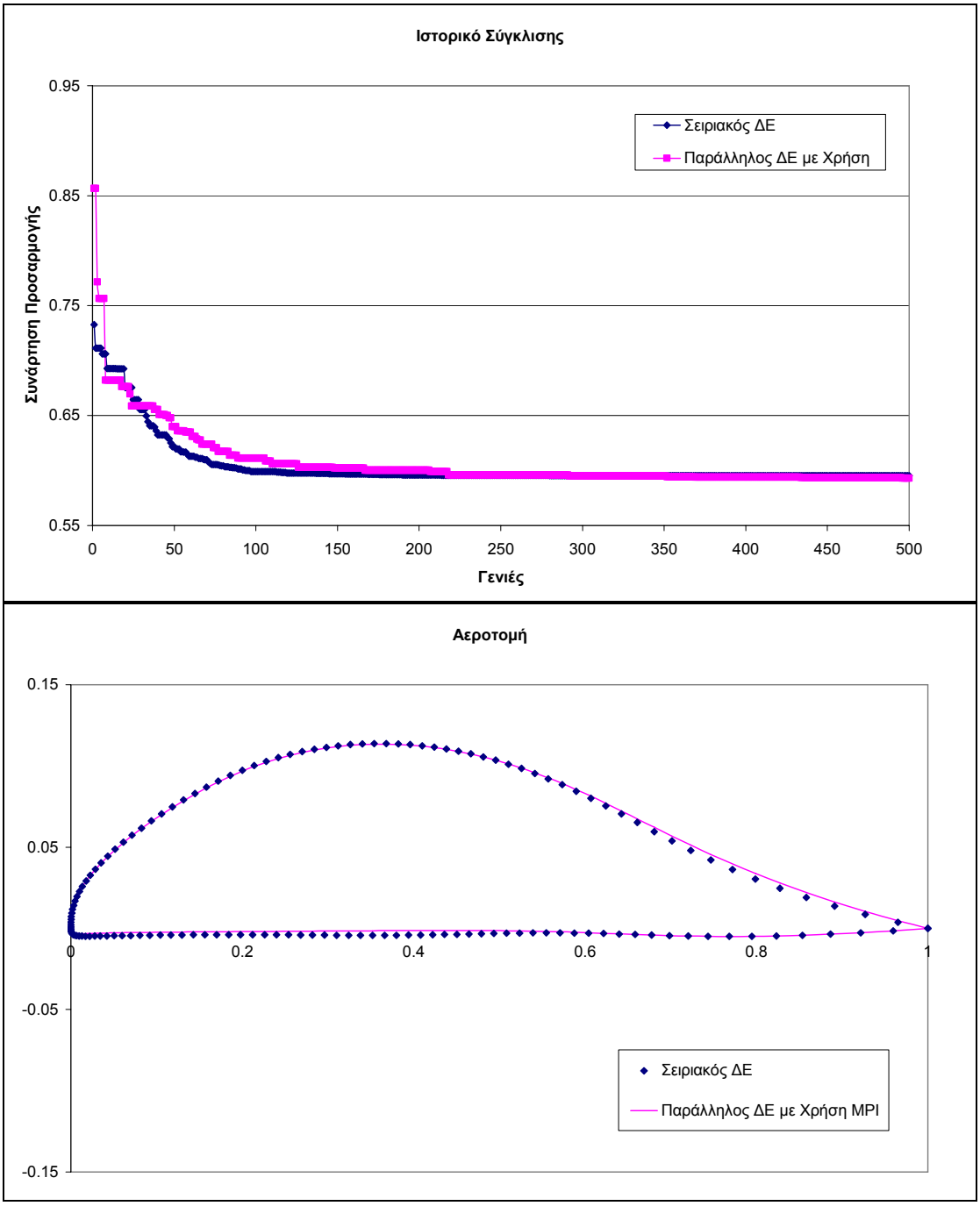
5.4.1 Σύγκριση παράλληλου και σειριακού Διαφορικού Εξελικτικού αλγορίθμου



Εικόνα 5.4: Ιστορικό σύγκλισης για το πρώτο πρόβλημα βελτιστοποίησης αεροτομής (επάνω). Οι καλύτερες λύσεις (αεροτομές) που υπολογίστηκαν με τους δύο αλγορίθμους (κάτω).

Ο κώδικας που αναπτύχθηκε δοκιμάστηκε στα ίδια πραγματικά προβλήματα που παρουσιάστηκαν στην § 4.8.2, χρησιμοποιώντας το ίδιο δίκτυο υπολογιστών, αποτελούμενο από 5 Η/Υ και 9 επεξεργαστές, προκειμένου να εξεταστεί η συμπεριφορά του αρχικά σε σύγκριση με τον σειριακό ΔΕ αλγόριθμο και στη συνέχεια με τον παράλληλο κώδικα που παρουσιάστηκε στο Κεφάλαιο 4.

Όπως έχει ήδη αναφερθεί, το πρώτο πρόβλημα αφορά στη βελτιστοποίηση του σχήματος μιας αεροτομής, με σκοπό την μεγιστοποίηση της άνωσης και την ελαχιστοποίηση της οπισθέλκουσας, κάτω από προκαθορισμένες συνθήκες ροής ($a_\infty = 8^o$, $\mathrm{Re}_c = 1200000$, $M_\infty = 0.18$). Το πρόβλημα αποτελεί πρόβλημα ελαχιστοποίησης με συνάρτηση κόστους την $f = 100 \cdot C_d / C_l$. Το ιστορικό σύγκλισης του σειριακού και του παράλληλου αλγορίθμου καθώς και οι καλύτερες λύσεις (αεροτομές) που προέκυψαν δίνονται στην ***Εικόνα 5.4***.

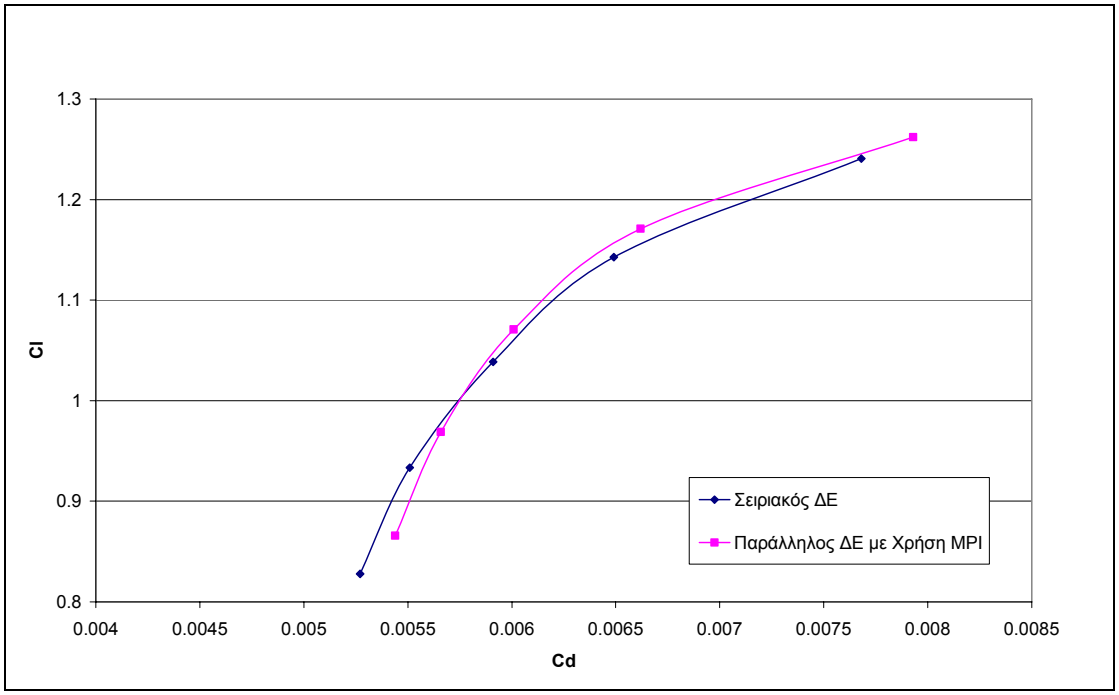


Εικόνα 5.5: Ιστορικό σύγκλισης για το δεύτερο πρόβλημα βελτιστοποίησης αεροτομής (πάνω). Οι καλύτερες λύσεις (αεροτομές) που υπολογίστηκαν με τους δύο αλγορίθμους (κάτω).

Οι παράμετροι που χρησιμοποιήθηκαν όμοια με πριν ήταν, μέγεθος πληθυσμού = 17, πλήθος γενεών = 500, F = 0.8 και Cr = 0.3. Και οι δύο αλγόριθμοι συνέκλιναν σε σχεδόν όμοιες λύσεις, ωστόσο ο σειριακός ολοκλήρωσε την βελτιστοποίηση (500 γενιές) σε 2906 δευτερόλεπτα, ενώ ο παράλληλος σε 807 δευτερόλεπτα. Η επιτάχυνση που επέφερε ο παράλληλος αλγόριθμος ήταν 3.6 φορές έναντι του σειριακού. Η επιτάχυνση που επιτεύχθηκε, όπως συνέβη και με τον παράλληλο ΔΕ αλγόριθμο του Κεφαλαίου 4, δεν ήταν ανάλογη του πλήθους των επεξεργαστών λόγω της μικρής διάρκειας εκτέλεσης της συνάρτησης προσαρμογής σε σχέση με τον χρόνο επικοινωνίας μεταξύ των υπολογιστών.

Το δεύτερο πρόβλημα αφορά στη βελτιστοποίηση του σχήματος μιας αεροτομής, με σκοπό την μεγιστοποίηση της άνωσης και την ελαχιστοποίηση της οπισθέλκουσας, για διαφορετικές γωνίες προσβολής ($a_{\infty,i} = 3^o, 4^o, 5^o, 6^o, 7^o$, $\mathrm{Re}_c = 1500000$, $M_\infty = 0.18$). Το πρόβλημα αποτελεί πρόβλημα ελαχιστοποίησης με συνάρτηση κόστους την $f = 100 \cdot \sum_{i=1}^{5} C_{d,i} \Big/ \sum_{i=1}^{5} C_{l,i}$.

Οι δύο αλγόριθμοι συνέκλιναν σε σχεδόν όμοιες λύσεις με τον σειριακό να απαιτεί 4463 δευτερόλεπτα για την εκτέλεση, ενώ ο παράλληλος χρειάστηκε 1320 δευτερόλεπτα, το οποίο αντιστοιχεί σε επιτάχυνση 3.38 φορών για 9 επεξεργαστές. Το ιστορικό σύγκλισης της διαδικασίας καθώς και οι καλύτερες λύσεις για τους δύο αλγορίθμους δίνονται στην ***Εικόνα 5.5***, ενώ το διάγραμμα άνωσης – οπισθέλκουσας για τις καλύτερες λύσεις που προέκυψαν δίνονται στη ***Εικόνα 5.6***.

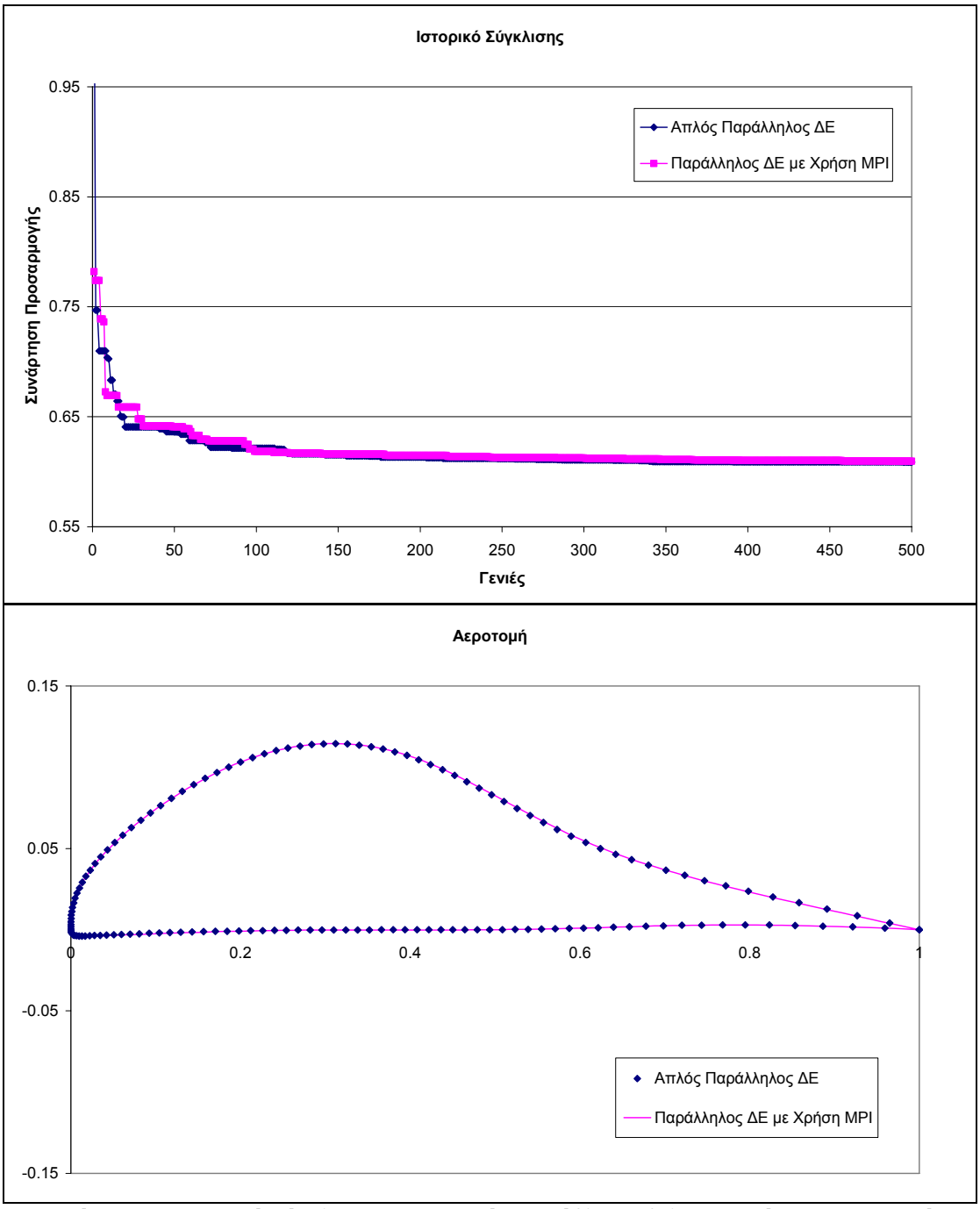


Εικόνα 5.6: Διάγραμμα άνωσης - οπισθέλκουσας για τις καλύτερες λύσεις που προέκυψαν από τον σειριακό και τον παράλληλο αλγόριθμο.

5.4.2 Σύγκριση των δύο παράλληλων Διαφορικών Εξελικτικών αλγορίθμων

Εδώ γίνεται σύγκριση μεταξύ των δύο παράλληλων προγραμμάτων, που αναπτύχθηκαν και παρουσιάστηκαν στα Κεφάλαια 4 και 5. Η σύγκριση γίνεται με

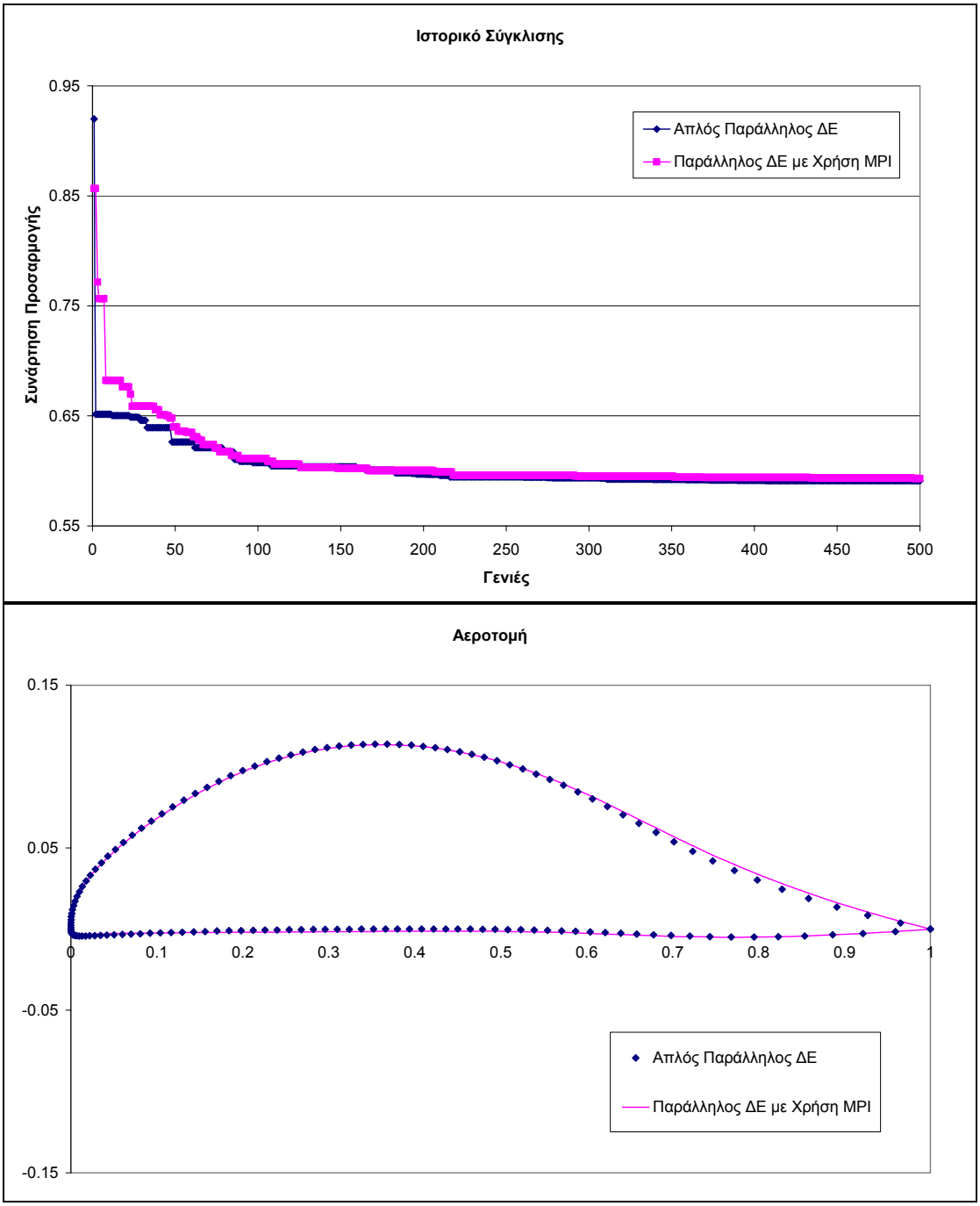
βάση τη συμπεριφορά τους ως προς τον χρόνο και την ποιότητα της λύσης στα δύο πραγματικά προβλήματα που παρουσιάστηκαν παραπάνω. Στην ***Εικόνα 5.7*** δίνεται το ιστορικό της σύγκλισης για τους δύο αλγορίθμους στο πρώτο πρόβλημα βελτιστοποίησης σχήματος αεροτομής, καθώς και οι αεροτομές που προκύπτουν. Όπως φαίνεται από τη σύγκριση, τόσο η σύγκλιση όσο και η τελική λύση είναι σχεδόν όμοιες για τους δύο αλγορίθμους, καθώς η τιμή της συνάρτησης προσαρμογής της τελικής λύσης για τον απλό παράλληλο ΔΕ αλγόριθμο είναι 0.6087, ενώ για τον παράλληλο με χρήση MPI είναι 0.6095.



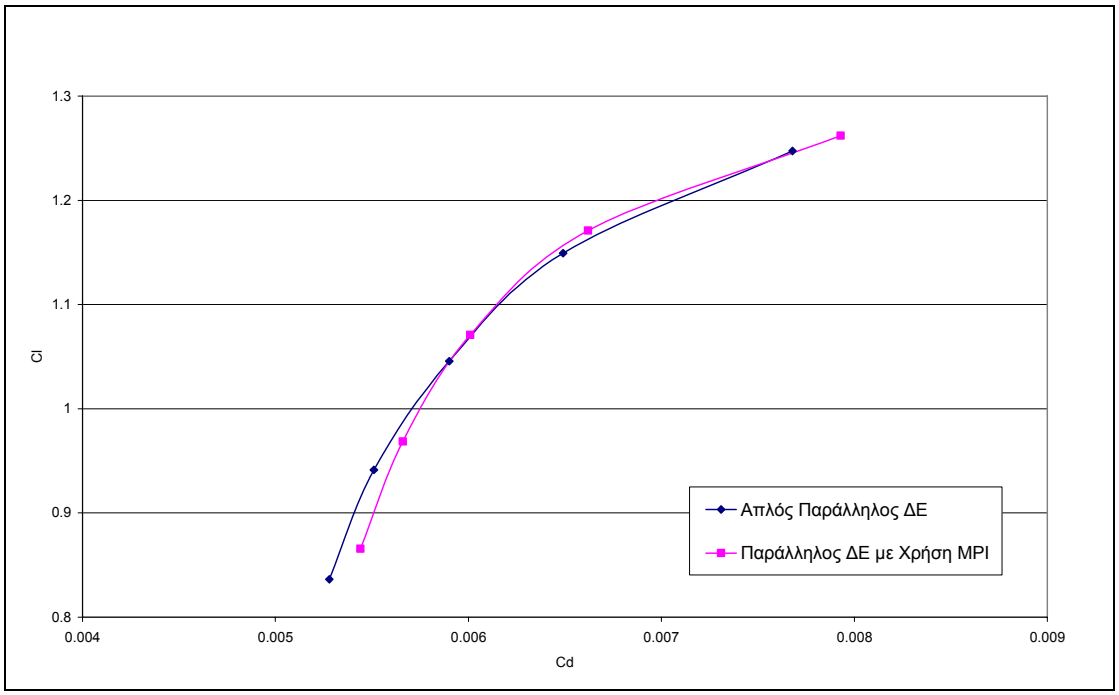
Εικόνα 5.7: Ιστορικό σύγκλισης για το πρώτο πρόβλημα βελτιστοποίησης αεροτομής (επάνω). Οι καλύτερες λύσεις (αεροτομές) που υπολογίστηκαν με τους δύο αλγόριθμους (κάτω).

Ο χρόνος που απαίτησε ο πρώτος ήταν 758 δευτερόλεπτα, ενώ ο δεύτερος χρειάστηκε 807 δευτερόλεπτα. Ο απλός παράλληλος ΔΕ αλγόριθμος ήταν 1.06 φορές πιο γρήγορος, γεγονός που οφείλεται τόσο στον διαφορετικό τρόπο υλοποίησης της παραλληλοποίησης αλλά κυρίως στον ασύγχρονο χαρακτήρα του πρώτου.

Στις ***Εικόνες 5.8*** και ***5.9*** δίνεται το ιστορικό της σύγκλισης με την καλύτερη λύση (αεροτομή) και τα διαγράμματα άνωσης - οπισθέλκουσας για τους δύο παράλληλους αλγόριθμους, στο δεύτερο πρόβλημα βελτιστοποίησης σχήματος αεροτομής για διαφορετικές γωνίες προσβολής.



Εικόνα 5.8: Ιστορικό σύγκλισης για το δεύτερο πρόβλημα βελτιστοποίησης αεροτομής (επάνω). Οι καλύτερες λύσεις (αεροτομές) που υπολογίστηκαν με τους δύο αλγόριθμους (κάτω).



Εικόνα 5.9: Διάγραμμα άνωσης - οπισθέλκουσας για τις καλύτερες λύσεις που προέκυψαν για τους δύο παράλληλους αλγορίθμους.

Και σε αυτή την εφαρμογή η σύγκλιση και η τελική λύση είναι σχεδόν όμοιες για τους δύο αλγορίθμους, με τιμές συνάρτησης προσαρμογής της τελικής λύσης για τον απλό παράλληλο ΔΕ αλγόριθμο 0.5911, ενώ για τον παράλληλο με χρήση MPI 0.5930. Ο χρόνος που απαίτησε ο πρώτος ήταν 823 δευτερόλεπτα, ενώ ο δεύτερος χρειάστηκε 1320 δευτερόλεπτα. Ο απλός παράλληλος ΔΕ αλγόριθμος ήταν 1.6 φορές πιο γρήγορος έναντι αυτού με χρήση MPI.

5.5 Συμπεράσματα

Η παραλληλοποίηση του ΔΕ αλγόριθμου με χρήση του προτύπου MPI επέφερε ικανοποιητική επιτάχυνση σε σχέση με τον αντίστοιχο σειριακό ΔΕ αλγόριθμο. Ο αλγόριθμος που αναπτύχθηκε είναι αρκετά σταθερός και η επικοινωνία μεταξύ των υπολογιστών πραγματοποιείται χωρίς προβλήματα.

Πρέπει να σημειωθεί εδώ ότι προβλήματα δύναται να δημιουργηθούν στην αρχή ή στο τέλος της εκτέλεσης του προγράμματος κατά το στάδιο της αρχικοποίησης του MPI ή τερματισμού του, τα οποία μπορεί να οφείλονται στο λογισμικό υλοποίησης του MPI (MPICH), στο περιβάλλον των Windows ή σε συνδυασμό τους. Εξάλλου η εφαρμογή του προτύπου MPI σε περιβάλλον Windows είναι ένα τομέας που γνωρίζει ανάπτυξη τα τελευταία χρόνια. Παρόλα αυτά δεν είναι σύνηθες φαινόμενο και δεν επηρεάζουν σημαντικά την βελτιστοποίηση καθώς στο στάδιο της αρχικοποίησης δε σπαταλάται χρόνος, ενώ στο στάδιο του τερματισμού έχουν ήδη εξαχθεί τα αποτελέσματα.

Οι δύο παράλληλοι ΔΕ αλγόριθμοι παρουσιάζουν σχεδόν όμοια αποτελέσματα όσον αφορά στην τελική λύση, ενώ ο απλός παράλληλος ΔΕ αλγόριθμος εμφανίζεται λίγο γρηγορότερος έναντι αυτού με χρήση της βιβλιοθήκης MPI. Το γεγονός αυτό καθιστά και τους δύο αλγόριθμου εξίσου αποδοτικούς με τη μικρή αυτή χρονική διαφορά να

οφείλεται στον ασύγχρονο χαρακτήρα του πρώτου. Πρέπει επίσης να σημειωθεί ότι λόγω του διαφορετικού τρόπου υλοποίησης της παραλληλοποίησης, ο χρόνος που απαιτείται για την επικοινωνία μεταξύ των υπολογιστών του δικτύου διαφέρει και συμβάλει στην διαφοροποίηση του χρόνου εκτέλεσης.

ΚΕΦΑΛΑΙΟ 6

ΣΥΜΠΕΡΑΣΜΑΤΑ

Τα συμπεράσματα που προέκυψαν κατά την ανάπτυξη και εφαρμογή των λογισμικών που αναπτύχθηκαν και παρουσιάστηκαν στα Κεφάλαια 4 και 5 συνοψίζονται παρακάτω.

Η αποδοτικότητα ενός παράλληλου Διαφορικού Εξελικτικού αλγόριθμου εξαρτάται από την διαδικασία παραλληλοποίησης καθώς επίσης και από το εκάστοτε πρόβλημα. Εάν ο χρόνος που απαιτείται για επικοινωνία μεταξύ των υπολογιστών του δικτύου, από την παράλληλη εφαρμογή, είναι συγκρίσιμος σε σχέση με αυτόν που απαιτεί ο υπολογισμός της συνάρτησης κόστους του προβλήματος για κάθε υπολογισμό, τότε η παράλληλη εφαρμογή δε συνιστάται, καθώς όχι μόνο δεν επιταχύνεται η διαδικασία αλλά σε κάποιες περιπτώσεις μπορεί να παρατηρηθεί επιβράδυνση. Με λίγα λόγια, προβλήματα με συναρτήσεις προσαρμογής πολύ μικρού χρόνου εκτέλεσης (π.χ. μαθηματικές συναρτήσεις όπως της § 4.8.1) παρουσιάζουν καλύτερη απόδοση ως προς το χρόνο χρησιμοποιώντας το ΔΕ αλγόριθμο σε σειριακή εφαρμογή. Παρόλα αυτά, πολλές φορές η παράλληλη εφαρμογή σε τέτοιου είδους συναρτήσεις οδηγεί σε καλύτερα αποτελέσματα ως προς την ποιότητα της λύσης.

Αντίθετα με τα παραπάνω, σε προβλήματα με χρονοβόρες συναρτήσεις προσαρμογής, τα αποτελέσματα που επιτυγχάνονται με την εφαρμογή του παράλληλου ΔΕ αλγόριθμου είναι εντυπωσιακά καθώς ο χρόνος εκτέλεσης δύναται να μειωθεί πάρα πολύ. Όσο περισσότερο χρόνο απαιτεί η συνάρτηση κόστους για κάθε άτομο του πληθυσμού τόσο μειώνεται ο χρόνος εκτέλεσης της παράλληλης εφαρμογής σε σχέση με αυτόν της σειριακής.

Άλλοι παράγοντες που επηρεάζουν την χρονική απόδοση του αλγορίθμου είναι το πλήθος των υπολογιστών του δικτύου σε σχέση με το μέγεθος του πληθυσμού. Η ιδανική περίπτωση εντοπίζεται στην παρουσία ενός μόνο ατόμου σε κάθε επεξεργαστή, ενώ όσο περισσότερα άτομα αναλαμβάνει να εξελίξει κάθε επεξεργαστής τόσο μειώνεται η χρονική απόδοση του παράλληλου αλγορίθμου.

Οι παράγοντες F και Cr, που αποτελούν παραμέτρους του Διαφορικού Εξελικτικού αλγόριθμου, επηρεάζουν σημαντικά την απόδοση ως προς την ποιότητα της λύσης, ενώ δεν επηρεάζουν καθόλου την χρονική απόδοση. Βάσει της διερεύνησης που έγινε στην § 4.8.1 προτείνονται κάποιες τιμές αυτών των παραμέτρων, που επιφέρουν ικανοποιητικά αποτελέσματα.

Τέλος, ο σύγχρονος ή ασύγχρονος χαρακτήρας του παράλληλου αλγορίθμου επηρεάζει την ταχύτητα του αλγορίθμου, ενώ δύναται να επηρεάσει και την ποιότητα της λύσης, με την ασύγχρονη εφαρμογή να δείχνει ότι υπερέχει έναντι της σύγχρονης.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] J. H. Holland, *Adaptation in Natural and Artificial Systems*, Ann Arbor, MI: Univ. Michigan Press, 1975.

[2] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Reading, MA: Addison-Wesley, 1989.

[3] P. J. Van Laarhoven and E. H. L. Aarts, *Simulated Annealing: Theory and Applications*, Norwell, MA: Kluwer, 1988.

[4] A. Martinez-Estudillo, C. Hervas-Martinez, F. Martinez-Estudillo, and N. Garcia-Pedrajas, "Hybrid method based on clustering for evolutionary algorithms with local search", *IEEE Transactions on Systems, Man and Cybernetics Part B: Cybernetics*, Vol. XX, N0. Y, 2004.

[5] G. Raidl, *Hybrid Evolutionary Algorithms for Combinatorial Optimization*, Technical University of Wien, November 2002.

[6] J. Søndergaard, *Optimization Using Surrogate Models by the Space Mapping Technique*, Ph.D. Thesis, Technical University of Denmark, Kgs. Lyngby, 2003.

[7] E. Alba, M. Tomassini, "Parallelism and Evolutionary Algorithms", *IEEE Transactions on Evolutionary Computation*, Vol. 6, No. 5, October 2002.

[8] H. J. Bremermann, "Optimization through evolution and recombination", in M. C. Yovits et al. (Eds.), *Self-Orginizing Systems*, Washington, DC: Spartan, 1962.

[9] R. M. Friedberg, "A learning machine: Part I", *IBM J.*, vol. 2, no. 1, pp. 2-13, Jan. 1958.

[10] R. M. Friedberg, B. Dunham, and J. H. North, "A learning Machine: Part II", IBM J., vol. 3, no. 7, pp. 282-287, July 1959.

[11] G. E. Box, "Evolutionary Operation: A method for increasing industrial productivity", *Appl. Statistics*, vol. VI, no. 2, pp. 81-101, 1957.

[12] J. H. Holland, "Outline for a logical theory of adaptive systems", *J. Assoc. Comput. Mach.*, vol. 3, pp. 297-314, 1962.

[13] I. Rechenberg, *Cybernetic solution path of an experimental problem*, Royal Aircraft Establishment, Library translation No. 1122, Farnborough, Hants., U.K., Aug. 1965.

[14] H.-P. Schwefel, *Projekt MHD-Staustrahlrohr: Experimentelle Optimierung einer Zweiphasenduse*, Teil I, Technischer Bericht 11.034/68, 35 AEG Forschungsinstitut, Berlin, Germany, Oct. 1968.

[15] L. J. Fogel, "Autonomous automata", *Ind. Res*., vol. 4, pp. 14-19, 1962.

[16] T. Back, U. Hammel and H.-P. Schwefel, "Evolutionary Computation: Comments on the History and Current State", *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, April 1997.

[17] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*, MIT Press, 1992.

[18] I. Rechenberg, *Evolutionstrategie*, Stuttgart: Frommann-Holzboog, 1973.

[19] H.-P. Schwefel, *Evolutionsstrategie und numerische Optimierung*, Ph.D. thesis, T.U. Berlin, 1975.

[20] H.-P. Schwefel, *Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie*, Basel: Birkhauser, 1977.

[21] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press, 1992.

[22] W. Bossert, "Mathematical optimization: Are there abstract limits on natural selection?", in P. S. Moorhead and M. M. Kaplan, (Eds.), *Mathematical*

Challenges to the Neo-Darwinian Interpretation of Evolution, Philadelphia, PA: Wistar Inst. Press, pp. 35–46, 1967.

[23] J. J. Grefenstette, *Parallel adaptive algorithms for function optimization*, Vanderbilt Univ., Nashville, TN, Tech. Rep. CS-81-19, 1981.

[24] P. B. Grosso, *Computer simulation of genetic adaptation: Parallel subcomponent interaction in a multilocus model*, Ph.D. dissertation, Univ. Michigan, Ann Arbor, 1985.

[25] R. Tanese, "Parallel genetic algorithms for a hypercube", in J. J. Grefenstette, (Ed.), *Proc. 2nd Int. Conf. Genetic Algorithms*, p. 177, 1987.

[26] J. P. Cohoon, S. U. Hegde, W. N. Martin, and D. Richards, "Punctuated equilibria: A parallel genetic algorithm", in J. J. Grefenstette (Ed.), *Proc. 2nd Int. Conf. Genetic Algorithms*, pp. 148–154, 1987.

[27] C. C. Pettey and M. R. Leuze, "A theoretical investigation of a parallel genetic algorithm", in J. D. Schaffer (Ed.), *Proc. 3rd Int. Conf. Genetic Algorithms*, pp. 398–405, 1989.

[28] G. Rudolph, "Global optimization by means of distributed evolution strategies", in H.-P. Schwefel and R.Manner (Eds.), *Parallel Problem Solving from Nature*, vol. 496, pp. 209-213, 1991.

[29] B. S. Duncan, "Parallel Evolutionary Progamming", in D. B. Fogel and W. Ahmar, (Eds.), *Proc. 2nd Annu. Conf. Evolutionary Programming*, La Jolla, CA: Evolut. Prog. Soc., pp. 202-208, 1993.

[30] M. Gorges-Schleuter, "ASPARAGOS an asynchronous parallel genetic optimization strategy", in J. D. Schaffer (Ed.), *Proc. 3rd Int. Conf. Genetic Algorithms*, pp. 422–427, 1989.

[31] B. Manderick and P. Spiessens, "Fine-grained parallel genetic algorithms", in J. D. Schaffer (Ed.), *Proc. 3rd Int. Conf. Genetic Algorithms*, pp. 428–433, 1989.

[32] G. Syswerda, "A study of reproduction in generational and steady-state genetic algorithms", in G. J. E. Rawlins, (Ed.), *Foundations of Genetic Algorithms*, San Mateo, CA: Morgan Kaufmann, pp. 94–101, 1991.

[33] G. Levine, *Users Guide to the PGAPack Parallel Genetic Algorithm Library*, Argonne Nat. Lab. Math. Comput. Sci. Div., Tech. Rep. ANL-95/18, Jan. 31, 1995.

[34] E. Cantú-Paz, *Designing Efficient and Accurate Parallel Genetic Algorithm*, Ph.D. thesis, University of Illinois, 1999.

[35] E. Alba and J. M. Troya, "A survey of parallel distributed genetic algorithms", *Complexity*, vol. 4, no. 4, pp. 31–52, 1999.

[36] T. C. Belding, "The distributed genetic algorithm revisited", in L. J. Eshelman, (Ed.), *Proc. 6th Int. Conf. Genetic Algorithms*, pp. 114–121, 1995.

[37] R. Tanese, "Distributed genetic algorithms", in J. D. Schaffer, (Ed.), *ICGA-3*, pp. 434–439, 1989.

[38] T. Maruyama, T. Hirose, and A. Konagaya, "A fine-grained parallel genetic algorithm for distributed parallel systems", in S. Forrest (Ed.), *Proc. 5th Int. Conf. Genetic Algorithms*, pp. 184–190, 1993.

[39] E. D. De Jong, D. Thierens, R. A. Watson, "Hierarchical Genetic Algorithms", in Yao, X., Burke, E., Lozano, J. A., Smith, J., Merelo-Guervos, J. J., Bullinaria, J. A., Rowe, J., Tino, P., Kaban, A., and Sonwefel H.-P. (Eds.), *Parallel Problem Solving from Nature – PPSN VIII*, Vol. 3242 of CNCS, Birmingham, UK. Springer – Verlang, 2004.

[40] E. Alba , J. M. Troya, "An Analysis of Synchronous and Asynchronous Parallel Distributed Genetic Algorithms with Structured and Panmictic Islands", in

Proceedings of the 11 IPPS/SPDP'99 Workshops Held in Conjunction with the 13th International Parallel Processing Symposium and 10th Symposium on Parallel and Distributed Processing, pp.248-256, April 12-16, 1999.

[41] E. Alba and J. M. Troya. "Analyzing synchronous and asynchronous parallel distributed genetic algorithms", *FGCS*, 17, pp. 451–465, January 2001.

[42] J. L. Ribeiro-Filho, C. Alippi, and P. Treleaven, "Genetic algorithm programming environments", in J. Stender (Ed.), *Parallel Genetic Algorithms: Theory and Applications*, Amsterdam, The Netherlands: IOS Press, pp. 65–83, 1993.

[43] K. Hwang, *Advanced Computer Architecture*, New York: McGraw Hill, 1993.

[44] S. Tommesani, "Programming Models", in http://www.tommesani.com/ProgrammingModels.html, 2000.

[45] D. E. Comer and D. L. Stevens, *Internetworking with TCP/IP (Volume III)*, Englewood Cliffs, NJ: Prentice-Hall, 1993.

[46] V. S. Sunderam, "PVM: A framework for parallel distributed computing", *J. Concurr. Practice and Experience*, vol. 2, no. 4, pp. 315–339, 1990.

[47] Message Passing Interface Forum, "MPI: A message-passing interface standard", *Int. J. Supercomput. Applic.*, vol. 8, no. 3–4, pp. 165–414, 1994.

[48] JavaSoft. RMI: The JDK 1.1 Specification (1997). Available in javasoft.com/products/jdk/1.1/docs/guide/rmi/index.html.

[49] A. Umar, *Object-Oriented Client/Server Internet Environments*, Englewood Cliffs, NJ: Prentice-Hall, 1997.

[50] I. Foster and C. Kesselmann, "Globus: A metacomputing infrastructure toolkit", *J. Supercomput. Applic.*, vol. 11, no. 2, pp. 115–128, 1997.

[51] M. S. Ntipteni, I. M. Valakos, I. K. Nikolos, "An Asynchronous Parallel Differential Evolution Algorithm", ERCOFTAC 2006 Conference on Design Optimization: Methods & Applications, Gran Canaria, Canary Islands, 2006.

[52] R. Storn, *Differential Evolution (DE)*, in http://icsi.berkeley.edu/~storn/index.html.

[53] R. Storn and K. Price, *Differential Evolution-a Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces*, Technical Report TR-95-012, ICSI, ftp.icsi.berkeley.edu, March 1995.

[54] R. Storn and K. Price, "Differential Evolution– A simple evolution strategy for fast optimization", *Dr. Dobb's Journal*, pp. 18–24 and 78, April 1997.

[55] J. J. Lampinen, "Solving Problems Subject to Multiple Nonlinear Constraints by the Differential Evolution", in R. M. P. Osmera, (Ed.), *Proceedings of MENDEL 2001*, 7th International Conference on Soft Computing, pp. 50-57, June 2001.

[56] D. K. Tasoulis, N. G. Pavlidis, V. P. Plagianakos, and M. N. Vrahatis, "Parallel Differential Evolution", in *Proceedings of the Congress on Evolutionary Computation (CEC 2004)*, Portland, Oregon, 2004.

[57] J. J. Lampinen, "Differential Evolution – New Naturally Parallel Approach for Engineering Design Optimization", in Bary H.V. Topping (Ed.), *Developments in Computational Mechanics with High Performance Computing*, Civil-Comp Press, Edinburgh, pp. 217-228, 1999.

[58] D. Zaharie, D. Petcu, "Parallel Implementation of multi-population Differential Evolution", in D. Grigoras et al. (Eds.), *Proceedings of the 2nd Workshop in Concurrent Information Processing and Computing (CIPC 2003)*, Sinaia, Romania, 2003.

[59] Zaharie, D., "A Multipopulation Differential Evolution Algorithm for multimodal Optimization", in R. Matousek, P. Osmer (Eds.), *Proceedings of the 10th International Conference on Soft Computing (Mendel 2004)*, Brno, 2004.
[60] Russinovich M., *PsTools*, in http://www.sysinternals.com/Utilities/PsTools.html, 2006.
[61] L. Piegl, W. Tiller, *The NURBS Book*, Springer-Verlag, Berlin, Heidelberg, 1997.
[62] M. Drela, "XFOIL: An Analysis and Design System for Low Reynolds Number Airfoils", In *Proceeding of the Conference on Low Reynolds Number Airfoil Aerodynamics*, University of Notre Dame, 1989.
[63] W. Gropp, E. Lusk, N. Doss, and A. Skjellum, *A High-Performance, Portable Implementation of the MPI, Message Passing Interface Standard*, Technical Report, Argonne Nat'l Laboratory and Mississippi State Univ., 1996.
[64] Message Passing Interface Forum, *Document for a Standard Message-Passing Interface*, Technical Report No CS-93-214 (revised), University of Tennessee, April 1994.
[65] Message Passing Interface Forum, "MPI: A Message-Passing Interface Standard", *International Journal of Supercomputer Applications*, 8(3/4):165-414, 1994.
[66] The MPI Forum, *The MPI Message-Passing Interface Standard*, in http://www.mcs.anl.gov/mpi/standard.html, 1995.
[67] Message Passing Interface Forum, *MPI-2: Extensions to the Message-Passing Interface*, 2003.
[68] W. Gropp, E. Lusk, and A. Skjellum, *Using MPI: Portable Programming with the Message-Passing Interface*, Second Edition, MIT Press, 1999.
[69] W. Gropp, E. Lusk, and R. Thakur, *Using MPI-2: Advanced Features of the Message-Passing Interface*, MIT Press, 1999.
[70] Message Passing Interface, Dr. Richard S. Miller, Department of Mechanical Engineering, Clemson University, in http://www.ces.clemson.edu/~rm/compute3.html.
[71] MPI Routines, in http://www-unix.mcs.anl.gov/mpi/www/www3/.
[72] MPICH2 Home Page, in http://www-unix.mcs.anl.gov/mpi/mpich2/.
[73] G. Burns, R. Daoud, and J. Vaigl, "LAM: An Open Cluster Environment for MPI", in J. W. Ross (Ed.), *Proceedings of Supercomputing Symposium '94*, pp. 379-386. University of Toronto, 1994.
[74] R. A. A. Bruce, J. G. Mills, and A. G. Smith, *CHIMP/MPI user guide*, Technical Report EPCC-KTP-CHIMP-V2-USER 1.2, Edinburgh Parallel Computing Centre, 1994.
[75] L. J. Clarke, R. A. Fletcher, S. M. Trewin, R. A. A. Bruce, A. G. Smith, and S. R. Chapple, "Reuse, Portability and Parallel Libraries", in *Proceedings of IFIP WG10.3-Programming Environment for Massively Parallel Distributed Systems*, 1994.
[76] G. Stellner, "Co check: Check pointing and process migration for MPI", in *Proceedings of the IPPS, IEEE Computer Society Press*, 1996.
[77] G. Stellner, and J. Pruyne, "Resource Management and Check pointing for PVM", in *Proceedings of the 2nd European PVM User' Group Meeting*, pp 131-136, Editions Hermes, Lyon, 1995.
[78] P. L. Vaughan, A. Skjellum, D. S. Reese, and F. C. Cheng, "Migrating from PVM to MPI, part I: The Unify System", in *Proceedings of the Fifth Symposium on the Frontiers of Massively Parallel Computation*, pp. 488-495, McLean,

Virginia, IEEE Computer Society Technical Committee on Computer Architecture, IEEE Computer Society Press, 1995.

[79] A. Geist, A. Beguclin, J. Dongarra, W. Jiang, B. Manchek, and V. Sunderam, *PVM: Parallel Virtual Machine – A User's Guide and Tutorial for Network Parallel Computing*, MIT Press, 1994.

ΠΑΡΑΡΤΗΜΑ Ι

ΒΑΣΙΚΕΣ ΟΔΗΓΙΕΣ ΓΙΑ ΤΟ ΠΡΟΤΥΠΟ MPI-2 ΚΑΙ ΤΟ ΛΟΓΙΣΜΙΚΟ MPICH2

Π1.1 Βασικές Ρουτίνες του Προτύπου MPI-2

Όπως αναφέρθηκε στο Κεφάλαιο 5, το πρότυπο MPI-2 αποτελείται από μεγάλο πλήθος ρουτινών, οι οποίες δίνουν τη δυνατότητα στο MPI-2 να είναι ευέλικτο και να μπορεί να εκτελεί κάθε επιθυμητή διαδικασία. Παρόλα αυτά, οι ρουτίνες που μπορούν να χρησιμοποιηθούν για την ανάπτυξη ενός βασικού παράλληλου προγράμματος είναι περιορισμένες, δίνοντας τη δυνατότητα στον αρχάριο χρήστη να αναπτύσσει με ευκολία τα δικά του παράλληλα προγράμματα. Στη συνέχεια περιγράφονται όλες οι ρουτίνες που χρησιμοποιήθηκαν στην ανάπτυξη του προγράμματος MPI-2 για την παραλληλοποίηση του ΔΕ αλγόριθμου.

Λειτουργία Ρουτινών

MPI_INIT(...)	Αρχικοποιεί το περιβάλλον εκτέλεσης MPI.
MPI_COMM_SIZE(...)	Καθορίζει το πλήθος των «εργασιών» που έχουν ανατεθεί στο δίκτυο.
MPI_COMM_RANK(...)	Καθορίζει ποια «εργασία» είναι η κάθε μια, θέτοντας έναν αναγνωριστικό αριθμό σε αυτές.
MPI_SEND(...)	Στέλνει δεδομένα.
MPI_RECV(...)	Λαμβάνει δεδομένα.
MPI_BCAST(…)	Γνωστοποιεί ένα δεδομένο από συγκεκριμένη «εργασία» σε όλες τις υπόλοιπες.
MPI_BARRIER(…)	Περιμένει μέχρι όλες οι «εργασίες» να φτάσουν σε αυτό το σημείο.
MPI_WTIME(…)	Επιστρέφει τον χρόνο της εκτέλεσης σε δευτερόλεπτα, βάσει κάποιας στιγμής στο παρελθόν.
MPI_FINALIZE(...)	Τερματίζει το περιβάλλον εκτέλεσης MPI.

Σύνταξη Ρουτινών

- ***MPI_INIT*** *(IERR)*

  Η *MPI_INIT* έχει ως όρισμα μόνο τον παράγοντα επιστροφής σφάλματος **IERR**, που έχουν όλες οι ρουτίνες του MPI στην Fortran (εκτός από τις *MPI_WTIME* και *MPI_WTICK*).

- ***MPI_COMM_SIZE*** *(MPI_COMM COMM, SIZE, IERR)*

  Παράμετροι Εισόδου
  COMM: χειριστής επικοινωνίας (αναφέρεται στο σύνολο των διαδικασιών)
  Παράμετροι Εξόδου
  SIZE: πλήθος διαδικασιών στο σύνολο comm (ακέραιος)
  IERR: παράγοντας επιστροφής σφάλματος

  π.χ. ***MPI_COMM_SIZE*** *(MPI_COMM_WORLD, numprocs, ierr)*

- ***MPI_COMM_RANK*** *(MPI_COMM COMM, RANK, IERR)*

  Παράμετροι Εισόδου
  COMM: χειριστής επικοινωνίας (αναφέρεται στο σύνολο των διαδικασιών)
  Παράμετροι Εξόδου
  RANK: αναγνωριστικός αριθμός (0, 1, 2,…, *ν*) για τις διαδικασίες στο σύνολο comm (ακέραιος)
  IERR: παράγοντας επιστροφής σφάλματος

  π.χ. ***MPI_COMM_RANK*** *(MPI_COMM_WORLD, myid, ierr)*

- ***MPI_SEND*** *(DATA, COUNT, MPI_DATATYPE, DEST, TAG, MPI_COMM COMM, IERR)*

  Παράμετροι Εισόδου
  DATA: όνομα δεδομένου προς αποστολή
  COUNT: αριθμός μονάδων δεδομένων προς αποστολή (μη αρνητικός ακέραιος)
  MPI_DATATYPE: τύπος δεδομένων για κάθε μονάδα λαμβανόμενων δεδομένων (χειριστής)
  DEST: αναγνωριστικός αριθμός της «εργασίας» στην οποία αποστέλλονται τα δεδομένα (ακέραιος)
  TAG: ετικέτα μηνύματος (ακέραιος) (η ρουτίνα *MPI_RECV* που θα λάβει το αποσταλμένο δεδομένο πρέπει να έχει την ίδια ετικέτα)
  COMM: χειριστής επικοινωνίας (αναφέρεται στο σύνολο των διαδικασιών)
  Παράμετροι Εξόδου
  IERR: παράγοντας επιστροφής σφάλματος

  π.χ. ***MPI_SEND*** *(isend, 1, MPI_INTEGER, 1, itag, MPI_COMM_WORLD, ierr)*

- ***MPI_RECV*** *(DATA, COUNT, MPI_DATATYPE, SOURCE, TAG, MPI_COMM COMM, MPI_STATUS, IERR)*

  Παράμετροι Εξόδου
  DATA: όνομα δεδομένου προς αποστολή
  STATUS: Κατάσταση αντικειμένου (κατάσταση)
  IERR: παράγοντας επιστροφής σφάλματος
  Παράμετροι Εισόδου
  COUNT: αριθμός μονάδων δεδομένων προς λήψη (μη αρνητικός ακέραιος)
  MPI_DATATYPE: τύπος δεδομένων για κάθε μονάδα λαμβανόμενων δεδομένων (χειριστής)
  SOURCE: αναγνωριστικός αριθμός της «εργασίας», η οποία αποστέλλει τα δεδομένα που θα ληφθούν (ακέραιος)
  TAG: ετικέτα μηνύματος (ακέραιος) (η ρουτίνα *MPI_SEND* που αποστέλλει το δεδομένο που λαμβάνεται πρέπει να έχει την ίδια ετικέτα)
  COMM: χειριστής επικοινωνίας (αναφέρεται στο σύνολο των διαδικασιών)

  π.χ ***MPI_RECV*** *(irecv, 1, MPI_INTEGER, 0, itag, MPI_COMM_WORLD, status, ierr)*

- ***MPI_BCAST*** *(DATA, COUNT, MPI_DATATYPE, ROOT, MPI_COMM COMM., IERR)*

  Παράμετροι Εισόδου/Εξόδου
  DATA: όνομα δεδομένου προς αποστολή
  COUNT: αριθμός μονάδων δεδομένων προς γνωστοποίηση (μη αρνητικός ακέραιος)
  MPI_DATATYPE: τύπος δεδομένων για κάθε μονάδα λαμβανόμενων δεδομένων (χειριστής)
  ROOT: αναγνωριστικός αριθμός της «εργασίας» πηγής η οποία γνωστοποιεί τα δεδομένα (ακέραιος)
  COMM: χειριστής επικοινωνίας (αναφέρεται στο σύνολο των διαδικασιών)
  IERR: παράγοντας επιστροφής σφάλματος

  π.χ. ***MPI_BCAST*** *(ibcast, 1, MPI_INTEGER, 0, MPI_COMM_WORLD, ierr)*

- ***MPI_BARRIER*** *(MPI_COMM COMM, IERR)*

  Παράμετροι Εισόδου
  COMM: χειριστής επικοινωνίας (αναφέρεται στο σύνολο των διαδικασιών)
  Παράμετροι Εξόδου
  IERR: παράγοντας επιστροφής σφάλματος

  π.χ. ***MPI_BARRIER*** *(MPI_COMM_WORLD, ierr)*

- ***MPI_WTIME*** *( )*

  Η ρουτίνα *MPI_WTIME* δεν παίρνει κανένα όρισμα, ενώ επιστρέφει τον χρόνο σε δευτερόλεπτα από κάποια στιγμή στο παρελθόν έως τώρα.

- ***MPI_FINALIZE*** *(IERR)*

  Παράμετροι Εξόδου
  IERR: παράγοντας επιστροφής σφάλματος

Πλήρης λίστα με όλες τις ρουτίνες του MPI-2 τη λειτουργία και τη σύνταξη τους παρέχεται στο http://www-unix.mcs.anl.gov/mpi/www/www3/. Στη συνέχεια δίνεται ένα παράδειγμα ενός απλού προγράμματος MPI-2, το οποίο τρέχει σε δίκτυο 2 υπολογιστών εκτελώντας 2 «εργασίες». Η πρώτη «εργασία» με myid = 0 διαβάζει από αρχείο εισόδου (input.txt) κάποια δεδομένα, ενώ σε αυτό το σημείο τοποθετείται *MPI_BARRIER* προκειμένου η δεύτερη «εργασία» να περιμένει. Στη συνέχεια η πρώτη «εργασία» ενημερώνει την άλλη εργασία για αυτές τις πληροφορίες και εγγράφουν και οι δύο το αποτέλεσμα στην οθόνη. Ακολουθώντας τις οδηγίες που δίνονται στη συνέχεια για την εγκατάσταση του λογισμικού MPICH2 και την ανάπτυξη προγραμμάτων MPI-2 σε Fortran, παρέχοντας το αντίστοιχο αρχείο εισόδου και δίνοντας σε γραμμή εντολών την ακόλουθη σύνταξη mpiexec, το παρακάτω πρόγραμμα λειτουργεί.

mpiexec –path c:\ -hosts 2 host1 1 host2 1 exmple.exe

```
      Program Example
C
      INCLUDE 'mpif.h'
C
      CHARACTER dataname1*4, dataname2*3,variable1*6
      INTEGER variable2
      INTEGER myid, numprocs, ierr
      INTEGER status(MPI_STATUS_SIZE)
C
      CALL MPI_INIT(ierr)
      CALL MPI_COMM_RANK(MPI_COMM_WORLD, myid, ierr)
      CALL MPI_COMM_SIZE(MPI_COMM_WORLD,numprocs, ierr)
C
      PRINT *, 'Process ', myid, ' of ', numprocs, ' is alive'
C
      IF (myid.eq.0) THEN
            OPEN (1, FILE = "input.txt")
            READ(1,*) dataname1, variable1
            READ(1,*) dataname2, variable2
            CLOSE(1)
      ENDIF
C
      CALL MPI_BARRIER (MPI_COMM_WORLD, ierr)
C
      IF (myid.eq.0) THEN
            CALL MPI_SEND (dataname1, 4, MPI_CHARACTER, 1, 1,
     &      MPI_COMM_WORLD, ierr)
            CALL MPI_SEND (dataname2, 3, MPI_CHARACTER, 1, 2,
     &      MPI_COMM_WORLD, ierr)
            CALL MPI_SEND (variable1, 6, MPI_CHARACTER, 1, 3,
     &      MPI_COMM_WORLD, ierr)
      ELSEIF (myid.ne.0) THEN
            CALL MPI_RECV (dataname1, 4, MPI_CHARACTER, 0, 1,
     &      MPI_COMM_WORLD, status, ierr)
            CALL MPI_RECV (dataname2, 3, MPI_CHARACTER, 0, 2,
     &      MPI_COMM_WORLD, status, ierr)
            CALL MPI_RECV (variable1, 6, MPI_CHARACTER, 0, 3,
     &      MPI_COMM_WORLD, status, ierr)
      ENDIF
C
      CALL MPI_BCAST(variable2, 1, MPI_INTEGER, 0,
     & MPI_COMM_WORLD, ierr)
C
      PRINT *, dataname1,' ', variable1
      PRINT *, dataname2, variable2
C
      CALL MPI_FINALIZE(ierr)
C
      STOP
      END
```

Π1.2 Εκτέλεση Προγράμματος MPI-2

Για την εκτέλεση παράλληλου προγράμματος με τη χρήση του πακέτου MPI-2, πρέπει να γίνει κλήση της εντολής mpiexec του λογισμικού MPICH2 μέσω γραμμής εντολών. Το εκτελέσιμο αρχείο mpiexec.exe τοποθετείται στον υπολογιστή με την εγκατάσταση του MPICH2 και βρίσκεται στην τοποθεσία C:\Program Files\MPICH2\bin. Συνεπώς πρέπει να γίνεται κλήση είτε από αυτή την τοποθεσία, είτε μεταφέροντας το στην τοποθεσία από όπου εκτελείται το πρόγραμμα. Στην εφαρμογή μας χρησιμοποιείται η δεύτερη διαδικασία, οπότε τοποθετείται στον φάκελο εργασίας του προγράμματος mpiwdir.

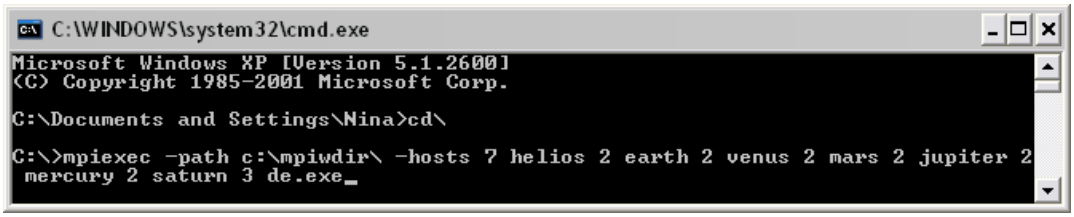
Η εντολή mpiexec έχει πολλές δυνατότητες και μπορεί να χρησιμοποιηθεί ορίζοντας ένα αρκετά μεγάλος πλήθος παραμέτρων. Εδώ θα περιοριστούμε στην περιγραφή των παραμέτρων που απαιτούνται στην σύνταξη της εντολής για την εκτέλεση του παράλληλου προγράμματος που αναπτύχθηκε. Η γενική μορφή της είναι:

mpiexec –path *pathname* –hosts *n* host1 *k* host2 *k* … hostm *q program.exe*

όπου
pathname: είναι η τοποθεσία που βρίσκεται το εκτελέσιμο πρόγραμμα MPI-2 και πρέπει να είναι κοινή για όλους τους υπολογιστές.
n: είναι το πλήθος των υπολογιστών που συμμετέχουν στο δίκτυο.
k: είναι το πλήθος των «εργασιών» που αναθέτονται σε κάθε υπολογιστή και ισούται με το μέγεθος του πληθυσμού δια το πλήθος των υπολογιστών (k = popsize / n).
q: είναι το πλήθος των εργασιών που αναθέτονται στον τελευταίο υπολογιστή και ισούται με το k συν το υπόλοιπο της διαίρεση εάν υπάρχει (q = k + mod(popsize / n).
program.exe: είναι το όνομα του εκτελέσιμου αρχείου του προγράμματος MPI-2.

Για παράδειγμα, για να εκτελεστεί το πρόγραμμα DE.exe με πληθυσμό 15, το οποίο βρίσκεται, σε όλους τους υπολογιστές του δικτύου, στον φάκελο εργασίας στην τοποθεσία C:\mpiwdir και το δίκτυο αποτελείται από 7 υπολογιστές τους helios, earth, venus, mars, jupiter, mercury και saturn θα πρέπει να γραφεί στη γραμμή εντολών το ακόλουθο (***Εικόνα Π.1***).

mpiexec -path c:\mpiwdir\ -hosts 7 helios 2 earth 2 venus 2 mars 2 jupiter 2 mercury 2 saturn 3 de.exe



Εικόνα Π.1: Σύνταξη εντολής mpiexec του λογισμικού MPICH2 στη γραμμή εντολών για την εκτέλεση του προγράμματος DE.exe με πληθυσμό 15 σε δίκτυο 7 υπολογιστών.

Επειδή όμως για μεγάλο μέγεθος πληθυσμού και μεγάλο πλήθος υπολογιστών στο δίκτυο η σύνταξη της εντολής καθίσταται χρονοβόρα και κουραστική, έχει κατασκευαστεί το πρόγραμμα preMPIexec, το οποίο εκτός από το να αντιγράφει τα απαραίτητα αρχεία στους απομακρυσμένους υπολογιστές, δημιουργεί ένα αρχείο batch, το run.bat, με το άνοιγμα του οποίου εκτελείται αυτόματα το πρόγραμμα. Το

preMPIexec διαβάζει από τα αρχεία εισόδου το μέγεθος του πληθυσμού, το πλήθος και τα ονόματα των υπολογιστών, δημιουργεί αυτόματα την σύνταξη της εντολής και την εγγράφει στο αρχείο run.bat.

Υπάρχουν και άλλοι εναλλακτικοί τρόποι για την σύνταξη της εντολής mpiexec προκειμένου να εκτελεστεί το ίδιο πρόγραμμα, όπως για παράδειγμα,

mpiexec –path c:\mpiwdir –n 15 –machinefile mf.txt de.exe

όπου 15 είναι το πλήθος των «εργασιών» που ισοδυναμεί με το μέγεθος του πληθυσμού και mf.txt ένα αρχείο κειμένου με τα ονόματα των υπολογιστών του δικτύου, η οποία όμως, στην περίπτωση που ο πληθυσμός δεν ισοκατανέμεται στους υπολογιστές, δεν συνάδει με την κατανομή των φακέλων με τα εκτελέσιμα των συναρτήσεων προσαρμογής. Περισσότερες πληροφορίες για την σύνταξη της mpiexec μπορούν να βρεθούν στο User's Guide (σελ. 12-15) που διατίθεται ελεύθερα στο http://www.mcs.anl.gov/mpi/mpich.

ΠΡΟΣΟΧΗ!! Για να εκτελεστεί ένα πρόγραμμα MPI πρέπει ο Τοίχος Προστασίας (Firewall) των Windows να είναι απενεργοποιημένος από όλους τους υπολογιστές. Οι υπολογιστές να έχουν λογαριασμούς με κοινό «username» και «password». Εφόσον το πρόγραμμα εκτελεί αντιγραφή αρχείων μέσω δικτύου πρέπει οι δίσκοι C:\ στους υπολογιστές του δικτύου να είναι "shared" και να παρέχουν πλήρη έλεγχο. (Local Disk (C:\) \ Properties \ Sharing \ Share this folder on the network & Allow network users to change my files ή Local Disk (C:\) \ Properties \ Sharing \ New Share \ name: C & Permissions: Full Control)

Π1.3 Εγκατάσταση Λογισμικού MPICH2 σε Windows XP για Fortran

Για την εγκατάσταση του λογισμικού MPICH2 σε Windows XP για εκτέλεση προγραμμάτων, που είναι γραμμένα σε Fortran, απαιτείται η εγκατάσταση των ακολούθων, τα οποία διατίθενται ελεύθερα στο διαδίκτυο στις σελίδες που αναφέρονται.

1. Microsoft .NET Framework Version 1.1 Redistributable Package (dotnetfx.exe) (http://www.microsoft.com/downloads/details.aspx?displaylang=en&FamilyID=262D25E3-F589-4842-8157-034D1E7CF3A3)
2. Microsoft .NET Framework 1.1 Service Pack 1 (NDP1.1sp1-KB867460-X86.exe) (http://www.microsoft.com/downloads/details.aspx?displaylang=en&FamilyID=A8F5654F-088E-40B2-BBDB-A83353618B38)
3. MPICH2 (mpich2-1.0.3-1win32-ia32.msi) (http://www.mcs.anl.gov/mpi/mpich)

Περισσότερες πληροφορίες για την εγκατάσταση του MPICH2 υπάρχουν στο Installer's Guide (σελ. 18-21) που διατίθεται ελεύθερα στο http://www.mcs.anl.gov/mpi/mpich.

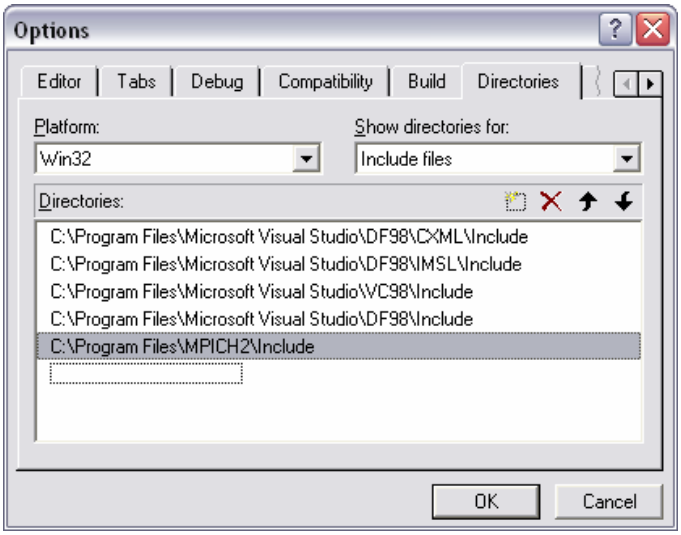
Π1.4 Ανάπτυξη Προγράμματος MPI-2 σε Visual Fortran 6 για Windows XP

Για την ανάπτυξη προγράμματος MPI-2 στο Project Workspace γίνεται προσθήκη του fmpich2s.lib, το οποίο βρίσκεται στην τοποθεσία C:\Program Files\MPICH2\lib

(Project → Add To Project → Files). Επίσης συμπεριλαμβάνεται στον κώδικα το header file mpif.h, συνεπώς στην αρχή του κώδικα εγγράφεται:

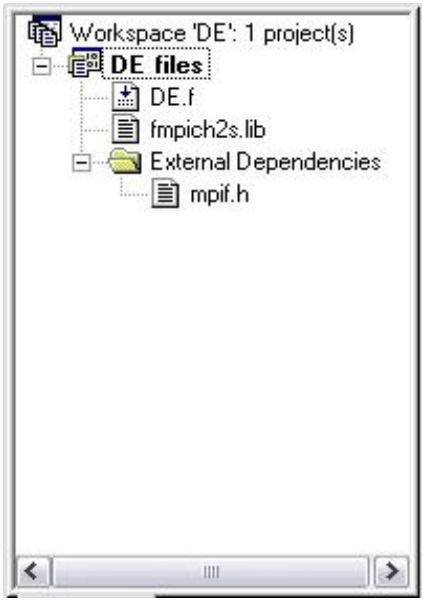
INCLUDE 'mpif.h'

Για να αναγνωρίσει ο κώδικας αυτό το αρχείο αυτό, πρέπει στο Tools\Options\ Directories να γίνει directories\new\C:\Program Files\MPICH2\Include (***Εικόνα Π.2)***



Εικόνα Π.2: Παράθυρο επιλογών της Fortran, στο οποίο δηλώνετε η τοποθεσία του αρχείου mpif.h.

Μόλις γίνει compilation του κώδικα, προστίθεται στο Project Workspace ένας φάκελος External Dependencies που περιέχει το header file mpif.h (***Εικόνα Π.3***). Περισσότερες λεπτομέρειες μπορούν να βρεθούν στο Windows Development Guide (σελ. 29) στο http://www.mcs.anl.gov/mpi/mpich.



Εικόνα Π.3: Project Workspace του προγράμματος DE.exe.

Τώρα είναι έτοιμα όλα για να γράψετε και να εκτελέσετε ένα πρόγραμμα MPI-2 σε Fortran για Windows XP. *ΚΑΛΗ ΕΠΙΤΥΧΙΑ!!*