ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΚΕΝΤΡΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΤΕ

Ανάπτυξη Πολιτισμικών Αλγορίθμων ή Αλγορίθμων Κουλτούρας (Cultural Algorithms) και Εφαρμογή τους σε Προβλήματα Βελτιστοποίησης

Πτυχιακή εργασία του

ΤΑΣΣΙΟΥ ΜΑΡΙΟΥ 2860

Επιβλέπων : Δρ. Σπυρίδων Α. Καζαρλής.

ΣΕΡΡΕΣ, ΜΑΙΟΣ 2018

<u>Υπεύθυνη Δήλωση</u>

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της πτυχιακής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της, είναι πλήρως αναγνωρισμένη και αναφέρεται στην πτυχιακή εργασία. Έχω αναφέρει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η πτυχιακή εργασία προετοιμάστηκε από εμένα προσωπικά ειδικά για τις απαιτήσεις του προγράμματος σπουδών του Τμήματος Μηχανικών Πληροφορικής ΤΕ του Τ.Ε.Ι. Κεντρικής Μακεδονίας.

Περιεχόμενα

Λίστα Εικόνων 5
Λίστα Πινάκων 5
Κεφάλαιο 1 Θεωρητικά στοιχεία της πτυχιακής 6
1.1 Εισαγωγή 6
1.2 Εξελικτική Υπολογιστική 7
1.3 Αλγόριθμοι Κουλτούρας (Cultural Algorithms) 10
Κεφάλαιο 2 Ανάλυση και Υλοποίηση του Αλγόριθμου Κουλτούρας 14
2.1 Τρόπος υλοποίησης Αλγόριθμου Κουλτούρας και Μέσα υλοποίησης 14
2.1.1 Ο Γενετικός Αλγόριθμος 14
2.1.2 Πολιτισμικός Αλγόριθμος 16
2.1.3 Γραφικό Περιβάλλον 16
2.1.4 Προβλήματα προς επίλυση 16
2.1.5 Παράμετροι Αλγορίθμου 16
2.2 Ο τρόπος λειτουργίας του κώδικα (user manual) 17
2.3 Τεχνική Περιγραφή του κώδικα (technical reference) 20
2.3.1 Βασική φόρμα ή κύριο μενού 21
2.3.2 Φόρμα Εμφάνισης Αποτελεσμάτων 26
2.3.3 Αρχείο VisualFunction 30
2.3.4 Αρχεία Functions 33
2.4 Προβλήματα επίλυσης 52
2.4.1 Περιγραφή της συνάρτησης Rosenbrock 52
2.4.2 Περιγραφή της συνάρτησης Rastrigin 53
2.4.3 Περιγραφή της συνάρτησης Schwefel 54
2.4.4 Περιγραφή του ρεαλιστικού προβλήματος. 55
2.5 Περιγραφή των πειραμάτων 56
Κεφάλαιο 3 Αποτελέσματα και συμπεράσματα 57
3.1 Αποτελέσματα πειραμάτων 57
3.1.1 Αποτελέσματα πειραμάτων για τη συνάρτηση Rosenbrock 57
3.1.2 Αποτελέσματα πειραμάτων για τη συνάρτηση Rastrigin 57
3.1.3 Αποτελέσματα πειραμάτων για τη συνάρτηση Schwefel 58
3.1.4 Αποτελέσμτα πειραμάτων για τη συνάρτηση του Παραλληλόγραμμου 59
3.2 Εξαγωγή συμπερασμάτων για την απόδοση του αλγόριθμου και την επίδραση των παραμέτρων 59

3.2.1 Σχολιασμός των πειραμάτων για την συνάρτηση Rosenbrock 59
3.2.2 Σχολιασμός των πειραμάτων για την συνάρτηση Rastrigin 60
3.2.3 Σχολιασμός των πειραμάτων για την συνάρτηση Schwefel 60
3.2.4 Σχολιασμός των πειραμάτων για το Πρόβλημα παραλληλογράμμου. 61
3.2.5 Παρουσίαση των καλύτερων παραμέτρων για κάθε πρόλημα. 61
Κεφάλαιο 4 Συμπεράσματα από την πραγματοποίηση της πτυχιακής, προτάσεις για μελλοντικές επεκτάσεις 62
4.1 Συμπεράσματα χρήσης των Αλγορίθμων Κουλτούρας 62
4.1.1 Συγκριτικός πίνακας των αποτελεσμάτων για τη συνάρτηση Rosenbrock 62
4.1.2 Συγκριτικός πίνακας των αποτελεσμάτων για τη συνάρτηση Rastrigin..... 63
4.1.3 Συγκριτικός πίνακας των αποτελεσμάτων για τη συνάρτηση Schwefel 63
4.1.4 Συγκριτικός πίνακας των αποτελεσμάτων για τη συνάρτηση του Παραλληλόγραμμου 64
4.2 Μελλοντικές Επεκτάσεις 65
Βιβλιογραφία 66
Παράθεση κώδικα 67

Λίστα Εικόνων

Εικόνα 1: Λογικό διάγραμμα της Εξελικτικής Διαδικασίας 9
Εικόνα 2: Ο λογικός ψευδο κώδικας του Αλγόριθμου Κουλτούρας 11
Εικόνα 3: Λογικό Διάγραμμα του Πολιτισμικού Αλγορίθμου 12
Εικόνα 4: Λογικό διάγραμμα του Γενετικού Αλγορίθμου 16
Εικόνα 5: Βασική Φόρμα ή Κύριο Μενού της Εφαρμογής 18
Εικόνα 6: Φόρμα Εμφάνισης των Αποτελεσμάτων 19
Εικόνα 7: Αρχείο Αποτελεσμάτων 20
Εικόνα 8: Γραφική παράσταση της συνάρτησης Rosenbrock 53
Εικόνα 9: Γραφική Παράσταση της Συνάρτησης Rastrigin 54
Εικόνα 10: Γραφική Παράσταση της Συνάρτησης Schwefel 55

Λίστα Πινάκων

Πίνακας 1: Παραλληλισμός των εννοιών της Εξελικτικής Υπολογιστικής με αυτών της Βιολογικής Εξέλιξης 8
Πίνακας 2: Περιγραφή των Controllers του Basic.h 25
Πίνακας 3: Περιγραφή των Controllers του View.h 29
Πίνακας 4: Αποτελέσματα του αλγόριθμου κουλτούρας για τη συνάρτηση Rosenbrock 57
Πίνακας 5: Αποτελέσματα του αλγόριθμου κουλτούρας για τη συνάρτηση Rastrigin 58
Πίνακας 6: Αποτελέσματα του αλγόριθμου κουλτούρας για τη συνάρτηση Schwefel 58
Πίνακας 7: Αποτελέσματα του αλγόριθμου κουλτούρας για τη συνάρτηση του Παραλληλόγραμμου 59
Πίνακας 8: Βέλτιστες τιμές παραμέτρων για κάθε πρόβλημα 61
Πίνακας 9: Συγκριτικός πίνακας των αποτελεσμάτων για τη συνάρτηση Rosenbrock ... 62
Πίνακας 10: Συγκριτικός πίνακας των αποτελεσμάτων για τη συνάρτηση Rastrigin 63
Πίνακας 11: Συγκριτικός πίνακας των αποτελεσμάτων για τη συνάρτηση Schwefel 64
Πίνακας 12: Συγκριτικός πίνακας των αποτελεσμάτων για τη συνάρτηση του Παραλληλόγραμμου 64

l

Κεφάλαιο 1 Θεωρητικά στοιχεία της πτυχιακής

1.1 Εισαγωγή

Στην παρούσα πτυχιακή θα ασχοληθούμε με έναν κλάδο της Εξελικτικής Υπολογιστικής ο οποίος είναι οι Πολιτισμικοί Αλγόριθμοι ή αλλιώς Αλγόριθμοι κουλτούρας (Cultural Algorithms).

Οι Αλγόριθμοι Κουλτούρας προσομοιώνουν τον Πολιτισμό των ανθρώπων δηλαδή την Κουλτούρα και την Παράδοση τους. Όπως γνωρίζουμε ο Πολιτισμός είναι ένα σύστημα συσσώρευσης γνώσεων και εμπειρίας, το οποίο εξελίσσεται συνέχεια με την πάροδο του χρόνου.

Όπως γνωρίζουμε ένας απόγονος εξαρτάται από τις γενετικές του καταβολές, δηλαδή το DNA που έχει κληρονομήσει από τους γονείς του καθώς και το περιβάλλον με το οποίο αλληλοεπιδρά, δηλαδή την κοινωνία στην οποία μεγαλώνει.

Συνεπώς μπορούμε να πούμε ότι ο απόγονος μιας γενιάς έχει την ευκαιρία να διδαχθεί ή να δεχθεί την αλληλεπίδραση από τον Πολιτισμό στον οποίο μεγαλώνει αλλά ταυτόχρονα έχει την ευκαιρία να εμπλουτίσει τον Πολιτισμό έτσι, ώστε να βοηθήσει μελλοντικά τις επόμενες γενιές. Αυτή η διαδικασία, δηλαδή η αλληλεπίδραση του ατόμου με το περιβάλλον, είναι μια αμφίδρομη διαδικασία.

Οι Αλγόριθμοι Κουλτούρας είναι επέκταση των Γενετικών Αλγορίθμων. Βασικά χρησιμοποιούν το μοντέλο των Γενετικών Αλγορίθμων και επιπλέον έναν χώρο μνήμης (belief space). Ο χώρος μνήμης είναι το μέρος στο οποίο αποθηκεύεται η αποκτώμενη εμπειρία κάθε γενιάς σύμφωνα με την επίλυση του προβλήματος, η οποία με την σειρά της είναι διαθέσιμη στους απογόνους και μπορεί να επιταχύνει σημαντικά την αναζήτηση προς το βέλτιστο. Κατ' αυτόν τον τρόπο, οι απόγονοι που παράγονται σε κάθε γενιά έχουν την δυνατότητα να αλληλοεπιδράσουν με τον χώρο μνήμης και να βελτιώσουν την ποιότητά τους με απώτερο σκοπό την γρηγορότερη αναζήτηση του βέλτιστου.

Στα πλαίσια της παρούσας πτυχιακής εργασίας θα αναπτυχθούν Αλγόριθμοι Κουλτούρας σε περιβάλλον Οπτικού Προγραμματισμού (Borland Builder C++). Επιπλέον, θα ελεγχθεί η απόδοσή τους εφαρμόζοντας μια σειρά από προβλήματα βελτιστοποίησης, τα οποία αποτελούν προβλήματα δοκιμών (benchmark problems). Τα προβλήματα βελτιστοποίησης που θα διεξαχθούν είναι τα εξής:

1. Rosenbrock Function
2. Rastrigin Function
3. Schwefel Funcrion
4. Ένα ρεαλιστικό πρόβλημα το οποίο είναι ένα Παραλληλόγραμμο σταθερού όγκου και ελάχιστης επιφάνειας.

Στο Κεφάλαιο 2 θα γίνει εκτενέστερη περιγραφή των προαναφερθέντων συναρτήσεων.

Όσον αναφορά την διαδικασία εκτέλεσης των δοκιμών, χρησιμοποιήσαμε τον συνδυασμό διαφόρων παραμέτρων του Αλγορίθμου και για κάθε συνδυασμό τρέξαμε τον Αλγόριθμο 10 φορές, διότι είναι στοχαστικός. Τα αποτελέσματα πάρθηκαν με βάση τον μέσο όρο των αποτελεσμάτων μετά από τις 10 φορές εκτέλεσης του Αλγόριθμου για κάθε συνδυασμό παραμέτρων και για κάθε πρόβλημα. Στο Κεφάλαιο 2 γίνεται πλήρης αναφορά στις παραμέτρους που συνετέλεσαν στην παραγωγή των αποτελεσμάτων. Στο Κεφάλαιο 3 παρουσιάζονται τα συμπεράσματα σχετικά με τις παραμέτρους που οδήγησαν το κάθε πρόβλημα στην βέλτιστη τιμή.

1.2 Εξελικτική Υπολογιστική

Η Εξελικτική Υπολογιστική είναι ένας κλάδος της Υπολογιστικής Ευφυίας που αφορά υπολογιστικές μεθόδους εμπνευσμένες από τη διαδικασία και τους μηχανισμούς βιολογικής εξέλιξης. Δηλαδή, την βιολογική εξέλιξη των ειδών στην φύση καθώς και άλλες βιολογικές τεχνικές και συστήματα, όπως για παράδειγμα το ανοσοποιητικό σύστημα. Οι μηχανισμοί εξέλιξης περιγράφουν τον τρόπο με τον οποίο η εξέλιξη πραγματοποιείται μέσω της τροποποίησης και της διάδοσης του γενετικού υλικού (πρωτεΐνες). Ο κύριος σκοπός είναι η αναζήτηση και η εύρεση των βέλτιστων λύσεων σε προβλήματα που είναι δύσκολα να επιλυθούν. Επιπλέον, με την χρήση της μπορούν να δημιουργηθούν και τεχνητά συστήματα με το πλεονέκτημα να αυτό-εξελίσσονται και να προσαρμόζονται μόνα τους στο περιβάλλον. Συνεπώς, χρησιμοποιώντας τις βασικές αρχές της εξέλιξης που είναι ο ανασυνδυασμός των γενετικών χαρακτηριστικών των γονέων, η μετάλλαξη των χρωμοσωμάτων, η αντικατάσταση του πληθυσμού των γονέων από τους απογόνους και η επικοινωνία μεταξύ των ατόμων και των πληθυσμών μπορούν να συμβάλλουν:

- Στην επίτευξη βελτιστοποίησης σε δύσκολα προβλήματα με μεγάλους χώρους λύσεων.
- Στην επίτευξη αυτόματης παραγωγής Software.
- Στην επίτευξη αυτό-εξελισσόμενου Hardware.
- Στην προσομοίωση συστημάτων.
- Στην βελτιστοποίηση παραμέτρων άλλων αλγορίθμων και τεχνικών.

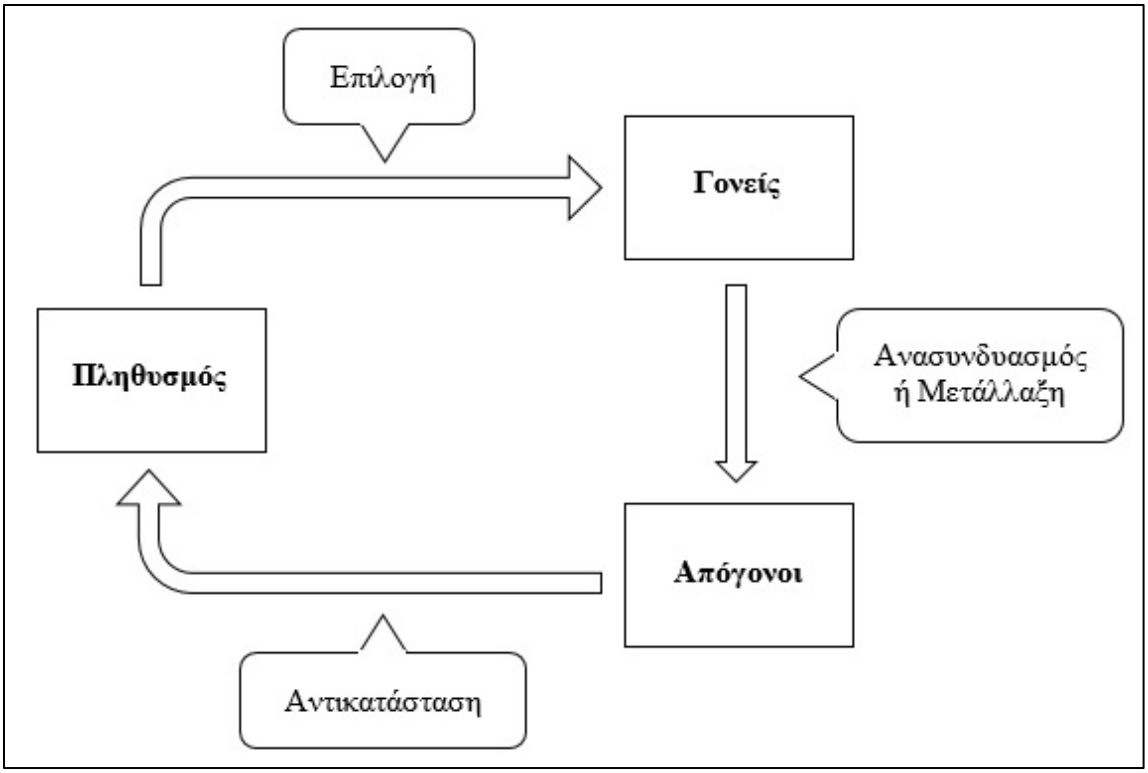
Όσον αφορά την ιστορία της Εξελικτικής Υπολογιστικής έως σήμερα, αξίζει να σημειωθεί ότι οι πρώτες προσπάθειές υλοποίησης των αρχών της έγιναν στα τέλη της δεκαετίας του 50 από τους Box, Friedman, Bledsoe και Bremerman. Το 1962, ο J. Holland ανέφερε για πρώτη φορά τους Γενετικούς Αλγορίθμους, όπου και το 1972 δημοσίευσε την εργασία “Adaptation in Natural and Artificial Systems” στην οποία θεμελιώνονται οι βασικές αρχές των Γενετικών Αλγορίθμων. Το 1962, ο L. Fogel διατύπωσε τον Εξελικτικό Προγραμματισμό και το 1965 ο I. Rechenberg και ο H.-P. Schwefel διατύπωσαν την Εξελικτική Στρατηγική. Μετά το 1978 παρουσιάστηκε το πρώτο Classifier System από τους J. Holland και J. Reitman. Το 1989 παρουσιάστηκε ο Γενετικός Προγραμματισμός από τον J. Koza και το 1992 παρουσιάστηκε το Εξελισσόμενο λογισμικό από τον Hugo De Garis. Ως τελευταίο ορόσημο αυτής της σύντομης ιστορικής αναδρομής είναι η παρουσίαση της Βελτιστοποίησης αποικιών μυρμηγκιών το 1992 από τον Marco Dorigo.

Η Εξελικτική Υπολογιστική χρησιμοποιεί κάποιες έννοιες οι οποίες είναι εμπνευσμένες από την βιολογική εξέλιξη. Πιο συγκεκριμένα δημιουργεί έναν παραλληλισμό αυτών, ο οποίος και παρουσιάζεται στον Πίνακα 1 που παρατίθεται στη συνέχεια.

Βιολογική Εξέλιξη	**Προσέγγιση Εξελικτικής Υπολογιστικής**
Άτομο	Υποψήφια Λύση
Επιδόσεις	Ποιότητα
Περιβάλλον	Πρόβλημα
Φυσική επιλογή	Πιθανοτική επιλογή
Αναπαραγωγή	Ανασυνδυασμός
Γενιές Ατόμων	Γενιές Λύσεων
Βελτίωση είδους	Βελτιστοποίηση

Πίνακας 1: Παραλληλισμός των εννοιών της Εξελικτικής Υπολογιστικής με αυτών της Βιολογικής Εξέλιξης

Η Εξελικτική Υπολογιστική χρησιμοποιεί κωδικοποιημένες λύσεις. Αυτό σημαίνει ότι όλες οι μεταβλητές μεταφράζονται σε συμβολοσειρές από 0 και 1 ή αλλιώς χρησιμοποιείται το δυαδικό σύστημα για την αναπαράστασή τους. Ένα βασικό χαρακτηριστικό είναι η Συνάρτηση Ποιότητας (Fitness Function), η οποία αξιολογεί τις λύσεις του αλγορίθμου σε κάθε γενιά. Η συνάρτηση ποιότητας πρέπει να τηρεί τρείς προϋποθέσεις. Πρώτον, πρέπει να είναι γρήγορη και αυτό γιατί χρησιμοποιείται για κάθε παραγόμενη λύση. Δεύτερον, πρέπει να αντιπροσωπεύει ακριβώς το πραγματικό πρόβλημα και τρίτον, πρέπει να μην περιέχει θόρυβο, δηλαδή δεν πρέπει να επιστρέφει ίδια λύση ή την ίδια ποιότητα παραπάνω από μία φορά. Στην Εικόνα 1 παρουσιάζεται το σχεδιάγραμμα το οποίο αντιστοιχεί στον κύκλο που ακολουθεί η εξέλιξη.



Εικόνα 1: Λογικό διάγραμμα της Εξελικτικής Διαδικασίας

Βέβαια, η διαδικασία της εξέλιξης πρέπει να τηρεί και κάποιο κριτήριο τερματισμού. Επομένως, θα πρέπει να υλοποιηθεί με κάποιο από τα παρακάτω κριτήρια. Αυτά είναι:

- Απόλυτο όριο γενεών, για παράδειγμα 500 γενιές.
- Τερματισμός μετά από κάποιο όριο βέλτιστης λύσης που επιτεύχθηκε.
- Τερματισμός μετά από την σύγκλιση του πληθυσμού στην ίδια λύση μετά από συγκεκριμένο αριθμό γενεών.
- Τερματισμός μετά από κάποιο πεπερασμένο όριο πραγματικού χρόνου.
- Τερματισμός μετά από επίτευξη των περιορισμών, όπως σε προβλήματα με περιορισμούς.

Όπως αντιλαμβανόμαστε η παραπάνω διαδικασία βασίζεται στην τυχαιότητα. Δεν μπορεί να μας εγγυηθεί ότι θα βρεθεί η βέλτιστη λύση σε πεπερασμένο χρόνο. Για αυτό και οι αλγόριθμοι που επιλύουν την παραπάνω διαδικασία ονομάζονται Ευριστικοί (heuristic). Συνεπώς, όλο το ενδιαφέρον στρέφεται προς την αύξηση της απόδοσης των Εξελικτικών Αλγορίθμων ως προς τον χρόνο και τις γενιές που μπορούν να επιτύχουν καλύτερες λύσεις.

Σε αυτό το σημείο αξίζει να γίνει σύντομη αναφορά και στις τεχνικές της Εξελικτικής Υπολογιστικής. Αυτές ειναι:

- Εξελικτικές Στρατηγικές
- Εξελικτικός Προγραμματισμός
- Γενετικός Προγραμματισμός
- Γενετικοί Αλγόριθμοι

1.3 Αλγόριθμοι Κουλτούρας (Cultural Algorithms)

Ο Πολιτισμικός Αλγόριθμος (Π.Α) είναι μια επέκταση στο πεδίο του Εξελικτικού Υπολογισμού και μπορεί να θεωρηθεί ως Μετά-Εξελικτικός Αλγόριθμος. Ευρέως ανήκει στον τομέα της Υπολογιστικής Ευφυίας και σχετίζεται με άλλες επεκτάσεις υψηλής τάξης του Εξελικτικού Υπολογισμού, όπως είναι οι Μεμετικοί Αλγόριθμοι.

Ο Πολιτισμικός Αλγόριθμος εμπνέεται από την αρχή της πολιτισμικής εξέλιξης. Ο πολιτισμός περιλαμβάνει τις συνήθειες, τη γνώση, τις πεποιθήσεις, τα ήθη και τα έθιμα ενός μέλους της κοινωνίας. Ο πολιτισμός δεν υπάρχει ανεξάρτητα από τον πολιτισμικό περιβάλλον και μπορούν μεταξύ τους να αλληλοεπιδράσουν μέσω θετικής ή αρνητικής ανάδρασης. Η μελέτη της αλληλεπίδρασης του πολιτισμού με το περιβάλλον αναφέρεται ως Πολιτισμική Οικολογία.

Ο Πολιτισμικός Αλγόριθμος μπορεί να εξηγηθεί στα πλαίσια ενός εμπνευσμένου Σύστηματος. Καθώς εξελίσσεται η εξελικτική διαδικασία, τα άτομα συσσωρεύουν πληροφορίες σχετικά με τον κόσμο (περιβάλλον) οι οποίες μεταδίδονται σε άλλα άτομα στον πληθυσμό. Συλλογικά αυτές οι πληροφορίες είναι γνώσεις τις οποίες όλα τα μέλη του πληθυσμού μπορούν να εκμεταλλευτούν. Σε αυτό το σημείο μπορεί να υπάρξει θετική και αρνητική ανάδραση. Θετική ανάδραση υπάρχει όταν:

- Οι πολιτισμικές γνώσεις δείχνουν χρήσιμες περιοχές του περιβάλλοντος.
- Οι πληροφορίες διαβιβάζονται μεταξύ των γενεών.
- Υπάρχει η εκμετάλλευση των πληροφοριών από τους απογόνους.
- Υπάρχει η εκκαθάριση πληροφοριών διότι καλύτερες πληροφορίες παίρνουν την θέση των παλιών που δεν είναι τόσο καλές.
- Υπάρχει η προσαρμοστικότητα των πληροφοριών ανάλογα με τις καταστάσεις που αλλάζουν.

Επιπρόσθετα, ενδέχεται να υπάρξει και αρνητική ανάδραση μέσω των πολιτισμικών γνώσεων.

Ο στόχος του αλγορίθμου σχετικά με την επεξεργασία των πληροφοριών είναι η βελτίωση της εκμάθησης ή η σύγκλιση μιας ένθετης τεχνικής αναζήτησης (συνήθως ένας εξελικτικός αλγόριθμος) ο οποίος χρησιμοποιεί μίας υψηλότερης τάξης πολιτισμική εξέλιξη. Συνεπώς, ο αλγόριθμος λειτουργεί σε δύο επίπεδα, το επίπεδο πληθυσμού και το πολιτισμικό επίπεδο.

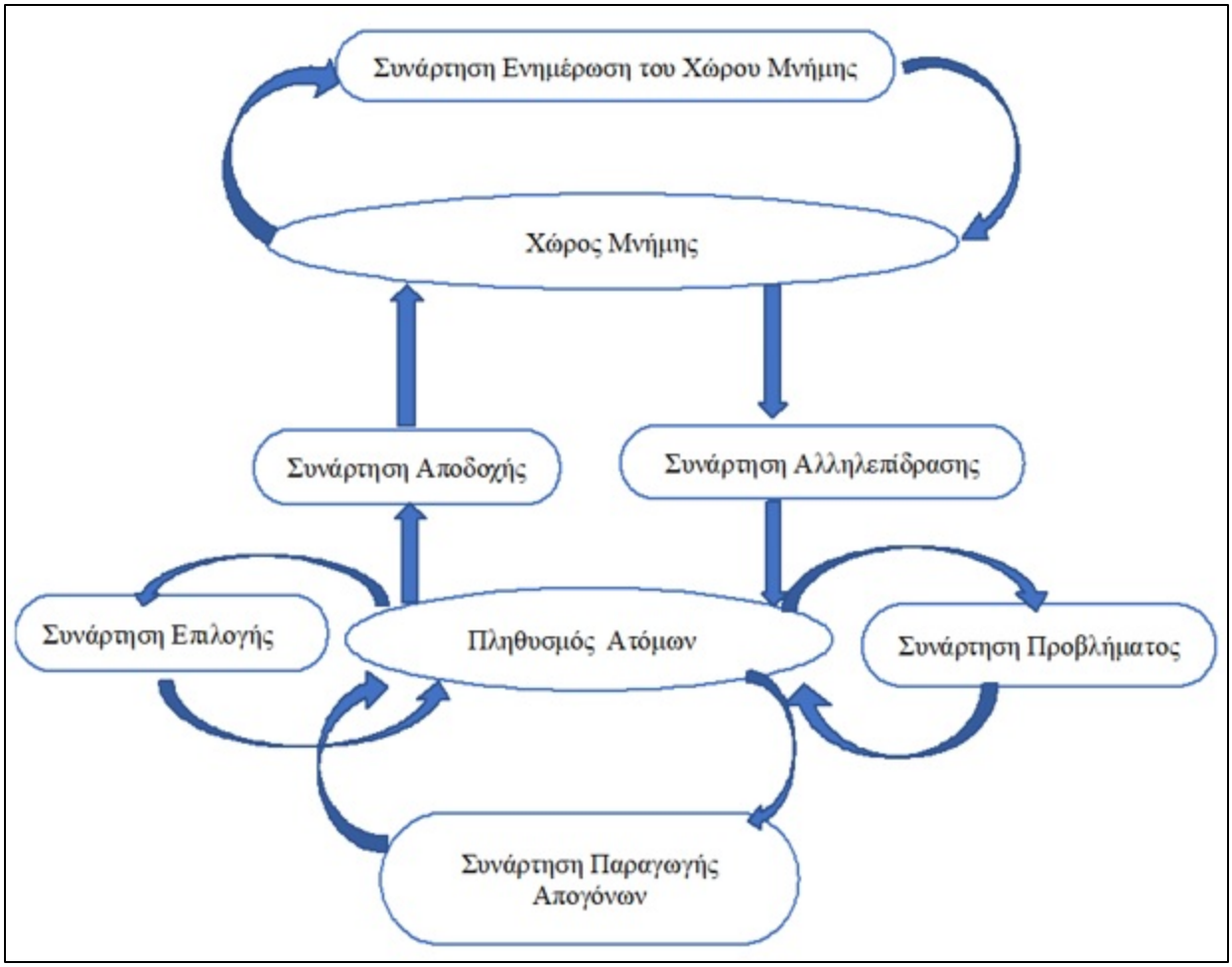
Το επίπεδο του πληθυσμού είναι σαν μια εξελικτική αναζήτηση, στην οποία τα άτομα εκπροσωπούν τις υποψήφιες λύσεις, οι οποίες κωδικοποιούνται σε συμβολοσειρές και αξιολογούνται με την χρήση της Συνάρτησης Ποιότητας.

Το δεύτερο επίπεδο είναι η γνώση ή ο χώρος μνήμης όπου υπάρχουν οι πληροφορίες που έχουν αποκτηθεί από γενιές και έχουν αποθηκευτεί. Ο συγκεκριμένος χώρος μνήμης είναι προσβάσιμος από την τρέχουσα γενιά. Επομένως, όπως είναι κατανοητό χρησιμοποιείται ένα πρωτόκολλο επικοινωνίας το οποίο επιτρέπει στους δύο χώρους να αλληλοεπιδρούν μεταξύ τους.

```
CULTURAL ALGORITHM
begin
t=0;
Initialize Population POP(0);
Initialize Belief Network BLF(0);
Initialize Communication Channel CHL(0);
Evaluate (POP(0));
t=1;
repeat
        Communicate (POP(0), BLF(t));
        Adjust (BLF(t));
        Communicate (BLF(t), POP(t));
        Modulate Fitness (BLF(t), POP(t));
        t = t+1;
        Select POP(t) from POP(t-1);
        Evolve (POP(t));
        Evaluate (POP(t));
until (termination condition)
end
```

Εικόνα 2: Ο λογικός ψευδο κώδικας του Αλγόριθμου Κουλτούρας

Ο αλγόριθμος εστιάζει πάνω στην δομή των γνώσεων που έχουν καταγράφει σύμφωνα με τους διαφορετικούς τύπους γνώσης ανάλογα με την φύση του προβλήματος. Για παράδειγμα, η δομή αυτή μπορεί να χρησιμοποιηθεί για την καταγραφή της καλύτερης υποψηφίας λύσης που βρέθηκε καθώς και άλλες γενικευμένες πληροφορίες όσον αναφορά τις περιοχές αναζήτησης του χώρου που αναμένεται να αποδώσουν περισσότερο (αποτέλεσματα καλών υποψήφιων λύσεων). Αυτή η πολιτισμική γνώση ανακαλύπτεται από τον πληθυσμό χρησιμοποιώντας την εξελικτική αναζήτηση και με τη σειρά της χρησιμοποιείται για να επηρεάσει τις επόμενες γενιές. Η συνάρτηση αποδοχής των γνώσεων περιορίζει την επικοινωνία του πληθυσμού με τον χώρο μνήμης στην περίπτωση που η τρέχουσα γνώση δεν είναι τόσο καλή όσο αυτές που ήδη είναι αποθηκευμένες. Στην Εικόνα 2 παρουσιάζεται ο ψευδο-κώδικας του Αλγορίθμου Κουλτούρας. Ο αλγόριθμος είναι αφηρημένος, παρέχοντας ευελιξία στην ερμηνεία των διαδικασιών όπως την αποδοχή των πληροφοριών, την δομή της βάσης γνώσεων ή χώρου μνήμης και την ειδική ενσωματωμένη εξελικτική ικανότητα του αλγορίθμου.



Εικόνα 3: Λογικό Διάγραμμα του Πολιτισμικού Αλγορίθμου

Ο Πολιτισμικός Αλγόριθμος αρχικά χρησιμοποιήθηκε ως εργαλείο προσομοίωσης για να διερευνήσει την Πολιτισμική Οικολογία. Έχει προσαρμοστεί ως αλγόριθμος βελτιστοποίησης για μια μεγάλη ποικιλία προβλημάτων βελτιστοποίησης χωρίς περιορισμούς, για την συνδυαστική βελτιστοποίηση, και για την βελτιστοποίηση συνεχών συναρτήσεων.

Η δομή του χώρου μνήμης παρέχει έναν μηχανισμό ενσωμάτωσης των πληροφορίων για την εξελικτική αναζήτηση σύμφωνα με το συγκεκριμένο πρόβλημα που εκτελείται.

Οι λειτουργίες αποδοχής που ελέγχουν τη ροή των πληροφοριών στον χώρο μνήμης είναι συνήθως άπληστες, συμπεριλαβάνοντας μόνο την τρέχουσα καλύτερη πληροφορία από την τρέχουσα γενιά και χωρίς αντικατάσταση των υπαρχουσών γνώσεων, παρά μόνο εάν υπάρχει βελτίωση. Οι λειτουργίες αποδοχής είναι επί των πλείστων ντετερμινιστικές, αν και έχουν γίνει έρευνες για Πιθανότικές και Ασαφείς λειτουργίες αποδοχής.

Ο Πολιτισμικός Αλγόριθμος προτάθηκε από τον Reynolds το 1994 ο οποίος συνδυάστηκε με τον Version Space Algorithm (χρήση δυαδικής συμβολοσειράς βασισμένος στον γενετικό αλγόριθμο), όπου γενικευμένες λύσεις των ατόμων γνωστοποιήθηκαν ως πολιτισμικές γνώσεις με τη μορφή schema patterns (συμβολοσειρές των 1, 0 και #, όπου το "#" αντιπροσωπεύει τον χαρακτήρα μπαλαντέρ).

Κεφάλαιο 2 Ανάλυση και Υλοποίηση του Αλγόριθμου Κουλτούρας

Στο Κεφάλαιο 2, αρχικά παρουσιάζεται ο τρόπος υλοποίησης του Αλγορίθμου Κουλτούρας και το γραφικό περιβάλλον της εφαρμογής, η οποία επιτρέπει στο χρήστη να εκτελέσει το πρόγραμμα με διάφορες παραμέτρους και να δει τα αποτελέσματα. Έπειτα ο τρόπος υλοποίησης του κώδικα, καθώς επίσης και η περιγραφή αυτού αναλύονται εκτενώς. Τελικό αντικείμενο αυτού του κεφαλαίου είναι η ανάλυση των προβλημάτων επίλυσης που χρησιμοποιούνται και η περιγραφή των πειραμάτων που εκτελούνται.

2.1 Τρόπος υλοποίησης Αλγόριθμου Κουλτούρας και Μέσα υλοποίησης

2.1.1 Ο Γενετικός Αλγόριθμος

Οι Γενετικοί Αλγόριθμοι (Γ.Α) όπως αναφέρεται και στο Κεφάλαιο 1 ανήκουν στον τομέα της Εξελικτικής Υπολογιστικής. Είναι το πιο δημοφιλές είδος Εξελικτικού Αλγορίθμου και χρησιμοποιούνται για γενικής φύσεως βελτιστοποιήσεις και αναζήτησή λύσεων. Παρουσιάστηκαν το 1962 από τον John Holland και έχουν χρησιμοποιηθεί σε πολλές και διαφορετικές εφαρμογές όπως για παράδειγμα «Βελτιστοποίηση Αεροτομής» και «Βέλτιστη Κοπή Υλικού» . Επιπλέον, χρησιμοποιήθηκαν με άλλες διαφορετικές παραλλαγές και τροποποιήσεις, όπως για παράδειγμα ως υβριδικά σχήματα με άλλους αλγορίθμους. Η χρήση τους είναι γενική, μπορούν να χρησιμοποιηθούν σχεδόν παντού και εκτελούνται offline διότι απαιτούν σημαντικό υπολογιστικό χρόνο.

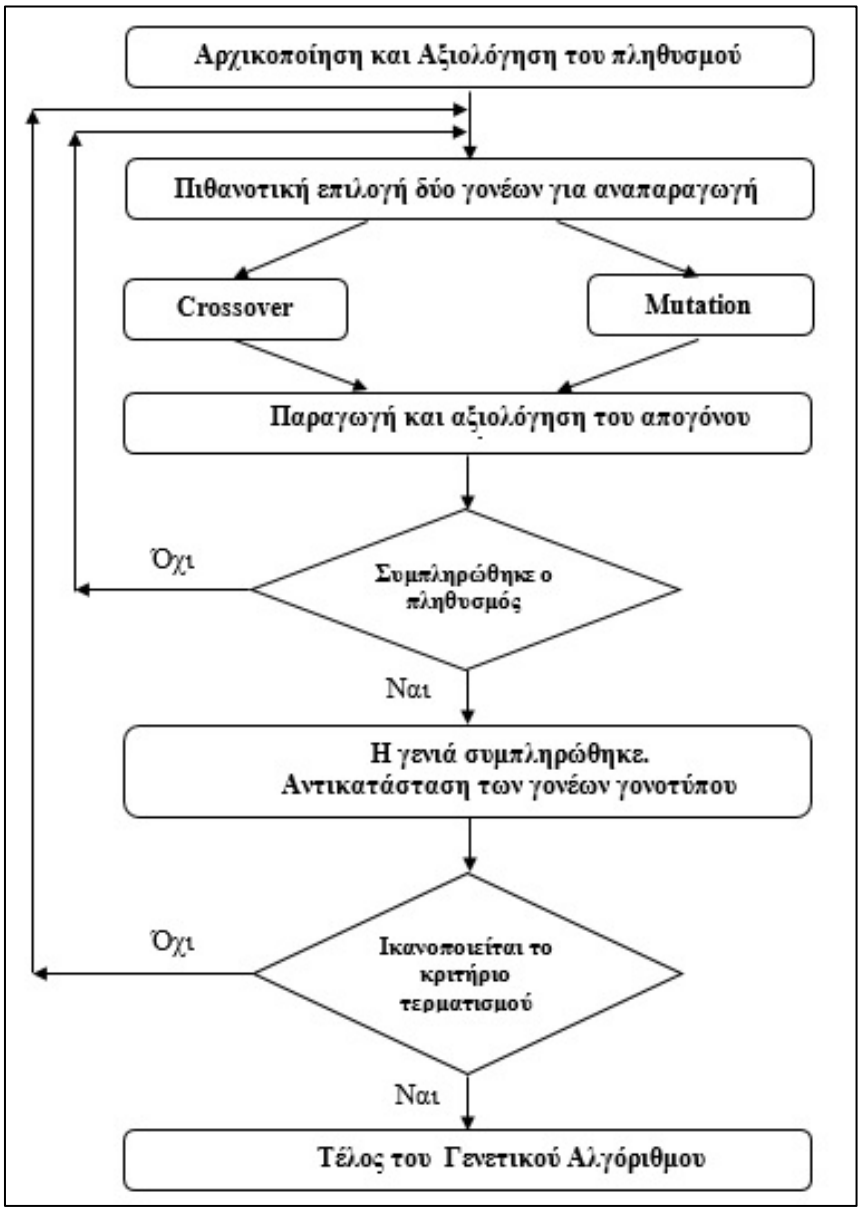
Συνεπώς όσον αναφορά την παρούσα υλοποίηση θα χρησιμοποιήσουμε τις βασικές αρχές του Γενετικού Αλγορίθμου. Ο Γενετικός Αλγόριθμος στηρίζεται σε 5 βασικές αρχές οι οποίες είναι:

1. Δημιουργία του Αρχικού Πληθυσμού χρησιμοποιώντας συμβολοσειρές από 0 και 1
2. Δημιουργία της Συνάρτησης Ποιότητας που θα αξιολογεί τις παραγόμενες λύσεις
3. Δημιουργία της Συνάρτησης Επιλογής των καλύτερων ατόμων-λύσεων του πληθυσμού για αναπαραγωγή.
4. Δημιουργία και χρήση των τελεστών αναπαραγωγής δηλαδή του Ανασυνδυασμού (Crossover) και της Μετάλλαξης (Mutation).
5. Αντικατάσταση των γονέων από τους απογόνους (ολική ή μερική)

Αξίζει να σημειωθεί ότι σχετικά με τους τελεστές αναπαραγωγής θα χρησιμοποιήσουμε όπως θα δούμε και παρακάτω διαφορετικές τιμές για την Μετάλλαξη έτσι, ώστε να πειραματιστούμε με τα προβλήματα που πρέπει να βελτιστοποιήσουμε. Ωστόσο, ο Ανασυνδυασμός θα παραμείνει σταθερός στην πιθανότητα 0.9. Αυτό γιατί όταν ο τελεστής του Ανασυνδυασμού έχει μεγάλη πιθανότητα εκτέλεσης συμβαίνει σύγκλιση του πληθυσμού, ενώ με τον τελεστή της Μετάλλαξης συμβαίνει απόκλιση του πληθυσμού. Συνεπώς, χρειαζόμαστε την Μετάλλαξη όχι με μεγάλη πιθανότητα εκτέλεσής, διότι με την επιλογή μεγάλης πιθανότητας δε θα βρεθεί η βέλτιστη λύση.

Ακόμη σχετικά με την μέθοδο επιλογής των γονέων θα χρησιμοποιήσουμε τον Τροχό της Ρουλέτας (Roulette Wheel Parent Selection). Αυτή η μέθοδος βασίζεται στην επιλογή των γονέων με βάση την ποιότητά τους. Τα πλεονεκτήματα της είναι δύο. Είναι εύκολη στο να

υλοποιηθεί και η πιθανότητα της επιλογής είναι απευθείας ανάλογη της ποιότητας των ατόμων. Τα μειονεκτήματα της είναι αρκετά. Αρχικά είναι πολύ εύκολο να οδηγήσει σε πρώιμή σύγκλιση, διότι τα άτομα που έχουν καλές ποιότητες μπορούν να κυριεύσουν πολύ γρήγορα στον πληθυσμό. Δεύτερον σε μικρούς πληθυσμούς συμβαίνουν στατιστικά σφάλματα. Τρίτον σε περιπτώσεις όπου οι ποιότητες των ατόμων του πληθυσμού είναι παρόμοιες δεν υπάρχει η πίεση της επιλογής «Selection Pressure». Τέταρτόν υπάρχει ο περιορισμός σύμφωνα με τον οποίο οι τιμές πρέπει να είναι μόνο θετικές. Θα χρησιμοποιήσουμε τον τελεστή αναρρίχησης (Hill Climb) έτσι, ώστε ο αλγόριθμος να μπορεί να ξεπερνάει εύκολα τα τοπικά μέγιστα και να μην σταματάει εκεί πιστεύοντας ότι έχει βρει την βέλτιστη λύση. Τέλος, ο γενετικός αλγόριθμος θα χρησιμοποιεί και τον ελιτισμό έτσι ώστε κάθε νέα γενιά να έχει την καλύτερη λύση της προηγούμενης γενιάς. Στην Εικόνα 4 παρουσιάζεται το λογικό διάγραμμα του Γενετικού Αλγόριθμου.



Εικόνα 4: Λογικό διάγραμμα του Γενετικού Αλγορίθμου

2.1.2 Πολιτισμικός Αλγόριθμος

Μετά την περιγραφή της υλοποίησης του Γ.Α στην παράγραφο 2.1.1, τώρα ήρθε η σειρά της περιγραφής του Πολιτισμικού Αλγορίθμου. Στο υποκεφάλαιο 1.3 έγινε λεπτομερής αναφορά στον Π.Α. Συνεπώς, τα βήματα που θα ακολουθήσουμε για να υλοποιήσουμε τον Π.Α είναι τα εξής:

1. Δημιουργία του Χώρου Μνήμης (Belief Space).
2. Δημιουργία της Συνάρτησης που θα εισάγει θα ταξινομεί και θα ενημερώνει τις καλύτερες ποιότητες στον Χώρο Μνήμης.
3. Δημιουργία της συνάρτησης που θα αλληλοεπιδρά ο Γ.Α με τον Χώρο Μνήμης.

2.1.3 Γραφικό Περιβάλλον

Για την δημιουργία του γραφικού περιβάλλοντος εργαστήκαμε με το ίδιο πρόγραμμα που χρησιμοποιήσαμε για την ανάπτυξη του κώδικα, δηλαδή την Borland C++ Builder 6. Χρησιμοποιήσαμε την βιβλιοθήκη VCL (Visual Component Library). Με την συγκεκριμένη βιβλιοθήκη μπορέσαμε να κατασκευάσουμε τις φόρμες της εφαρμογής καθώς και να χρησιμοποιήσουμε διάφορα συστατικά, όπως για παράδειγμα πίνακες.

2.1.4 Προβλήματα προς επίλυση

Η βασική ιδέα της συγκεκριμένης πτυχιακής είναι να την χρησιμοποιήσουμε για την επίλυση κάποιων συγκεκριμένων προβλημάτων. Όπως αναφέρεται και στην Εισαγωγή αυτά είναι οι συναρτήσεις Rosenbrock , Rastrigin, Schwefel και ένα ρεαλιστικό πρόβλημα το οποίο είναι ένα Παραλληλόγραμμο. Σκοπός μας είναι να βαθμολογήσουμε τις επιδόσεις του Πολιτισμικού Αλγορίθμου χρησιμοποιώντας τα παραπάνω προβλήματα βελτιστοποίησης.

2.1.5 Παράμετροι Αλγορίθμου

Στο τελευταίο σκέλος της πτυχιακής θα ασχοληθούμε με την εξαγωγή των συμπερασμάτων. Αυτά τα συμπεράσματα απορρέουν μετά από πολλές επαναλήψεις του αλγορίθμου για διαφορετικές παραμέτρους για το ίδιο πρόβλημα, διότι το κάθε πρόβλημα είναι διαφορετικής φύσεως. Οι παράμετροι που λάβαμε υπόψιν για την βαθμολόγηση και την εξαγωγή των συμπερασμάτων κάθε συνάρτησης είναι οι εξής:

1. Μέγεθος του πληθυσμού.
2. Αριθμός γενεών.
3. Μέγεθος χώρου μνήμης.
4. Μέγεθος συμβολοσειράς των ατόμων.
5. Επιλογή προβλήματος.
6. Πιθανότητα μετάλλαξης.
7. Τρόπος αλληλεπίδρασης του Γ.Α με τον χώρο μνήμης.
8. Τελεστής Αναρρίχησης.

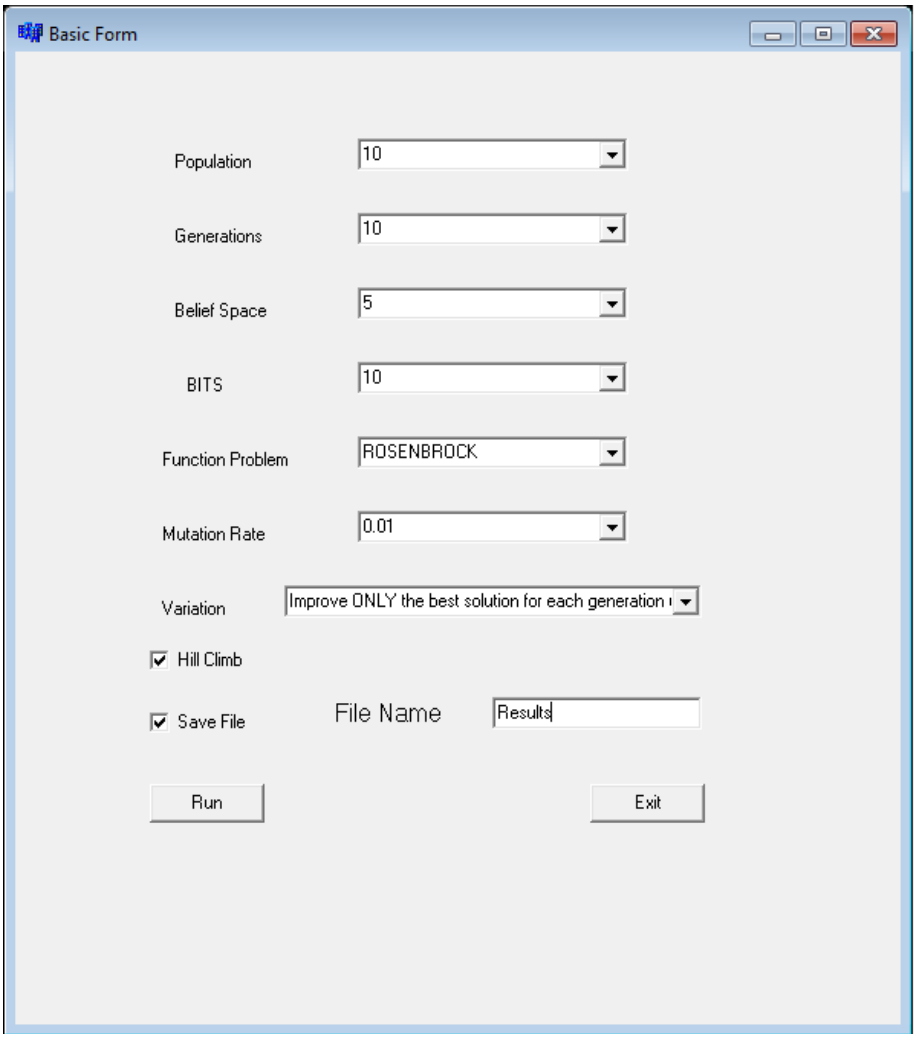
2.1.6 Μέσα υλοποίησης

Για την δημιουργία της παρούσας πτυχιακής εργαστήκαμε εξ ολοκλήρου με το πρόγραμμα ανάπτυξης λογισμικού Borland C++ Builder 6. Το συγκεκριμένο πρόγραμμα έκανε την εμφάνισή του το 2001. Ήταν ένα πολύ ισχυρό εργαλείο για τους προγραμματιστές καθώς προσέφερε πολλές τεχνολογίες και δυνατότητες για παράδειγμα βάσεις δεδομένων, ανάπτυξη γραφικού περιβάλλοντος και πολλά πρωτόκολλα όπως το TCP/IP και άλλα.

2.2 Ο τρόπος λειτουργίας του κώδικα (user manual)

Σε αυτήν την ενότητα θα περιγράψουμε τον τρόπο λειτουργίας της εφαρμογής ως προς την οπτική γωνία του χρήστη. Αρχικά η εφαρμογή μας στηρίζεται σε δύο φόρμες. Η πρώτη φόρμα (Εικόνα 5) είναι το κύριο μενού της εφαρμογής όπου ο χρήστης πληκτρολογεί και επιλέγει τις παραμέτρους του αλγορίθμου. Εκτός από αυτό, έχει την δυνατότητα να αποθηκεύσει τα αποτελέσματα του αλγορίθμου σε αρχείο και να το ονομάσει όπως επιθυμεί. Η δεύτερη φόρμα (Εικόνα 6) της εφαρμογής μας παρουσιάζει τα αποτελέσματα του αλγορίθμου. Επιπλέον, μας παρουσιάζει τις παραμέτρους που ο χρήστης εισήγαγε, τον χρόνο εκτέλεσης του αλγορίθμου, τον χώρο μνήμης. Μία ακόμα λειτουργία που μας επιτρέπει είναι να ανοίξουμε το αρχείο που επιλέξαμε να σώσουμε τα αποτελέσματα. Εδώ πρέπει να σημειωθεί πως σε αυτήν την φόρμα βλέπουμε τα αποτελέσματα του αλγορίθμου για κάθε γενιά, ενώ στο αρχείο έχουμε μια πλήρη εικόνα των αποτελεσμάτων για κάθε γενιά.

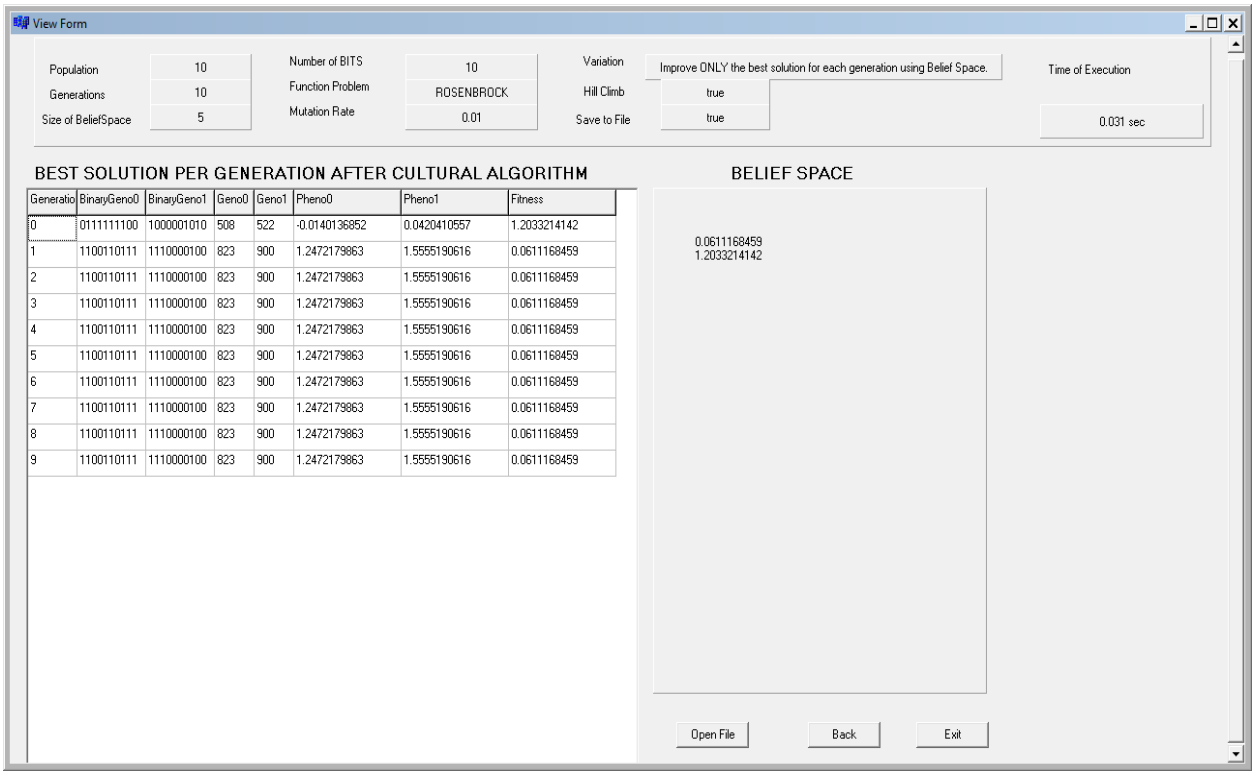
Στην Εικόνα 5 παρουσιάζεται η βασική φόρμα, η οποία είναι το αρχικό μενού της εφαρμογής. Ο χρήστης έχει την δυνατότητα να επιλέξει και να πληκτρολογήσει τις τιμές των παραμέτρων. Ακόμη έχει την επιλογή να επιλέξει αν θέλει να χρησιμοποιήσει τον τελεστή αναρρίχησής και αν θέλει να σώσει τα αποτελέσματα σε κάποιο αρχείο με τον τίτλο της επιλογής του. Αν ο χρήστης δεν επιλέξει τιμές για τις παραμέτρους, οι παράμετροι θα πάρουν κάποιες αρχικές τιμές ειδάλλως ο αλγόριθμος δεν θα μπορέσει να τρέξει. Ακόμη τα στοιχεία που πληκτρολογεί ο χρήστης ελέγχονται για την εγκυρότητά τους. Για παράδειγμα, ο χρήστης δεν μπορεί να χρησιμοποιήσει έναν πραγματικό αριθμό για την τιμή που θα εισάγει στην παράμετρο του πληθυσμού διότι η τιμή θα πρέπει να είναι ακέραια. Επίσης, στο κάτω μέρος βλέπουμε δύο κουμπιά. Αυτά είναι, το κουμπί «Run» που είναι η εκτέλεση της εφαρμογής, δηλαδή κατευθείαν καλείται ο αλγόριθμος ο οποίος παίρνει σαν εισόδους του τις παραμέτρους που έθεσε ο χρήστης και, το κουμπί «Exit» που είναι η έξοδος που τερματίζει την εφαρμογή.



Εικόνα 5: Βασική Φόρμα ή Κύριο Μενού της Εφαρμογής

Στην Εικόνα 6 παρουσιάζεται η δεύτερη φόρμα. Σε αυτή εμφανίζονται τα αποτελέσματα του αλγορίθμου στον χρήστη σύμφωνα με τις τιμές που επέλεξε για κάθε παράμετρο του αλγορίθμου. Συνεπώς στο πάνω μέρος της φόρμας βλέπουμε τις τιμές των παραμέτρων που επιλέχθηκαν από τον χρήστη, καθώς και τον χρόνο εκτέλεσης του αλγορίθμου, δηλαδή πόσο χρόνο χρειάστηκε για να ικανοποιήσει το κριτήριο του τερματισμού σε σχέση με τις παραμέτρους που του εισήγαγε ο χρήστης. Από κάτω βλέπουμε έναν πίνακα ο οποίος μας εμφανίζει την καλύτερη λύση κάθε γενιάς. Αυτό που παρατηρούμε στο συγκεκριμένο στιγμιότυπο είναι ένας πίνακας με 10 γραμμές και, αν κοιτάξουμε στο πάνω μέρος βλέπουμε ότι έχουμε εξίσου εισάγει την τιμή 10 στην παράμετρο Γενιές. Επομένως, σε αυτόν τον πίνακα αν τον δούμε λίγο πιο προσεκτικά βλέπουμε την ποιότητα της καλύτερης λύσης, βλέπουμε ακόμη τον γενότυπο, τον φαινότυπο και την δυαδική ακολουθία (DNA) των δύο ατόμων του πληθυσμού που παρήγαγαν την συγκεκριμένη λύση. Αν μετρήσουμε τη δυαδική ακολουθία αντιλαμβανόμαστε ότι και αυτός ο αριθμός είναι ο αριθμός της παραμέτρου BITS που θέσαμε. Στην συνέχεια, στα δεξιά βλέπουμε το χώρο μνήμης (Belief Space) της εφαρμογής. Ο συγκεκριμένος πίνακας είναι ταξινομημένος και μπορεί να διαφέρει στο μέγεθος του.

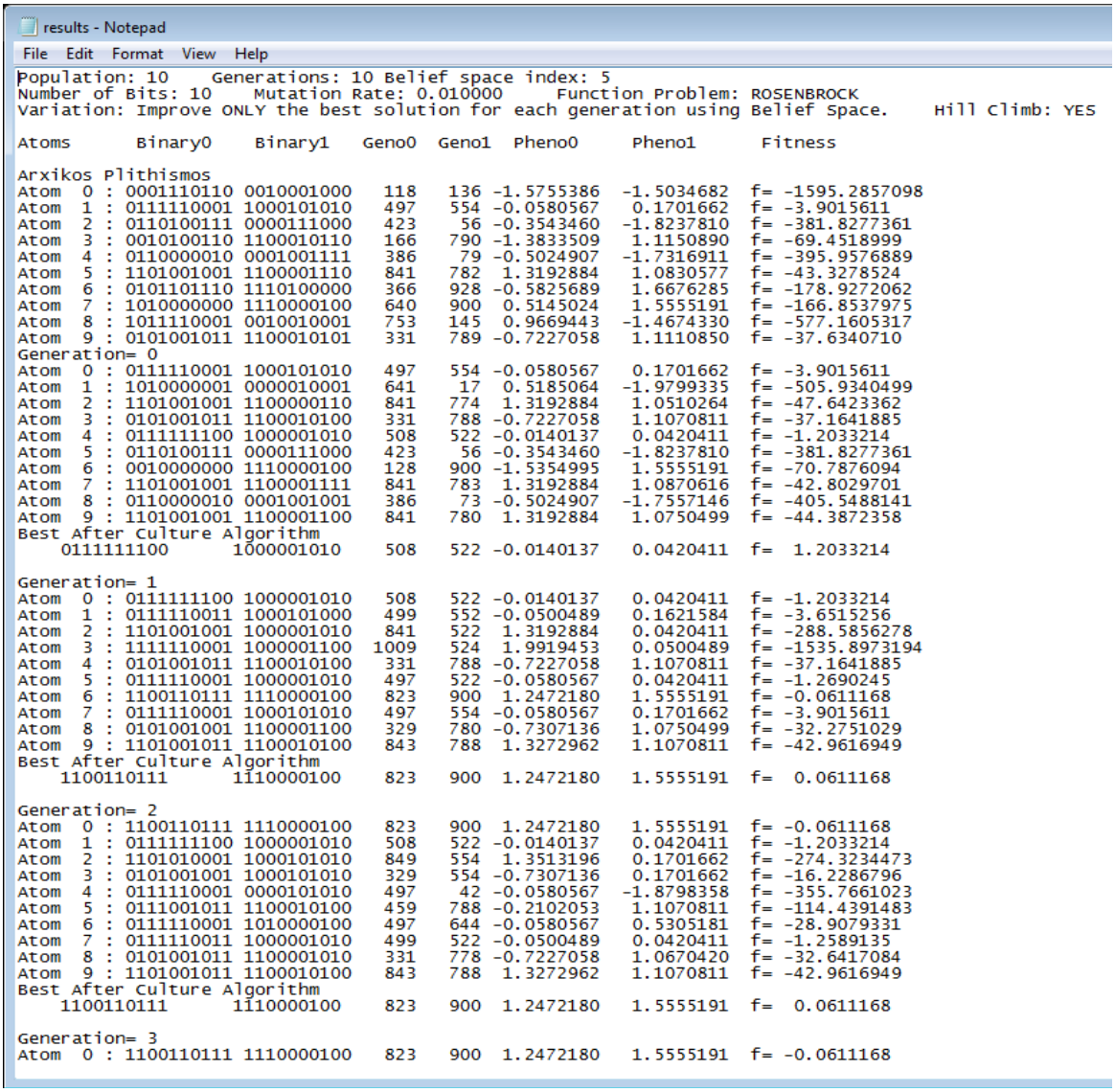
Συγκεκριμένα στην πρώτη φόρμα το μέγεθός του είναι 5 στοιχείων και αυτό, γιατί όπως φαίνεται στην Εικόνα 5 έχουμε δώσει αυτήν τιμή στην παράμετρο. Επίσης, παρατηρούμε πως όταν ο χώρος μνήμης συμπληρωθεί ξεκινάει η αντικατάσταση των πιο κακών ποιοτήτων από τις καλύτερες. Στο δεξιά κάτω μέρος της δεύτερης φόρμας βλέπουμε τρία κουμπιά. Το πρώτο από αριστερά ονομάζεται «Open File» και ανοίγει το αρχείο όπου επιλέξαμε να σώσουμε τα αποτελέσματα. Στην περίπτωση που δεν επιλέξαμε να σώσουμε τα αποτελέσματα δεν εμφανίζεται καθόλου. Το δεύτερο κουμπί «Back» μας πάει πίσω στο κύριο μενού σε περίπτωση που θέλουμε να τρέξουμε την εφαρμογή με διαφορετικές παραμέτρους. Το τρίτο κουμπί «Exit» είναι η έξοδος που οδηγεί στον τερματισμό της εφαρμογής.



Generatio	BinaryGeno0	BinaryGeno1	Geno0	Geno1	Pheno0	Pheno1	Fitness
0	0111111100	1000001010	508	522	-0.0140136852	0.0420410557	1.2033214142
1	1100110111	1110000100	823	900	1.2472179863	1.5555190616	0.0611168459
2	1100110111	1110000100	823	900	1.2472179863	1.5555190616	0.0611168459
3	1100110111	1110000100	823	900	1.2472179863	1.5555190616	0.0611168459
4	1100110111	1110000100	823	900	1.2472179863	1.5555190616	0.0611168459
5	1100110111	1110000100	823	900	1.2472179863	1.5555190616	0.0611168459
6	1100110111	1110000100	823	900	1.2472179863	1.5555190616	0.0611168459
7	1100110111	1110000100	823	900	1.2472179863	1.5555190616	0.0611168459
8	1100110111	1110000100	823	900	1.2472179863	1.5555190616	0.0611168459
9	1100110111	1110000100	823	900	1.2472179863	1.5555190616	0.0611168459

Εικόνα 6: Φόρμα Εμφάνισης των Αποτελεσμάτων

Στην Εικόνα 7 βλέπουμε το αρχείο στο οποίο εμφανίζουμε όλα τα αποτελέσματα του γενετικού αλγορίθμου. Δηλαδή, δε βλέπουμε μόνο την καλύτερη λύση κάθε γενιάς αλλά βλέπουμε όλες τις λύσεις του αλγορίθμου. Ακόμη βλέπουμε τη διαδικασία του Ελιτισμού καθώς όπως παρατηρούμε η καλύτερη λύση μιας γενιάς περνάει στην επόμενη γενιά. Στο πάνω μέρος όπως και στην φόρμα της εμφάνισης των αποτελεσμάτων βλέπουμε τις τιμές που δώσαμε στις παραμέτρους. Μετά από κάθε γενιά εμφανίζουμε και τον χώρο μνήμης. Σταδιακά μπορούμε να παρατηρήσουμε ότι οι καλύτερες λύσεις κάθε γενιάς εμφανίζονται στον χώρο μνήμης και ταξινομούνται κατά αύξουσα σειρά. Επιπροσθέτως, παρατηρούμε πως όταν ο χώρος μνήμης συμπληρωθεί ξεκινάει η αντικατάσταση των πιο κακών ποιοτήτων από τις καλύτερες.



results - Notepad

File Edit Format View Help

```
Population: 10    Generations: 10 Belief space index: 5
Number of Bits: 10    Mutation Rate: 0.010000     Function Problem: ROSENBROCK
Variation: Improve ONLY the best solution for each generation using Belief Space.    Hill Climb: YES

Atoms      Binary0    Binary1   Geno0  Geno1  Pheno0     Pheno1      Fitness

Arxikos Plithismos
Atom  0 : 0001110110 0010001000   118   136 -1.5755386  -1.5034682  f= -1595.2857098
Atom  1 : 0111110001 1000101010   497   554 -0.0580567   0.1701662  f= -3.9015611
Atom  2 : 0110100111 0000111000   423    56 -0.3543460  -1.8237810  f= -381.8277361
Atom  3 : 0010100110 1100010110   166   790 -1.3833509   1.1150890  f= -69.4518999
Atom  4 : 0110000010 0001001111   386    79 -0.5024907  -1.7316911  f= -395.9576889
Atom  5 : 1101001001 1100001110   841   782  1.3192884   1.0830577  f= -43.3278524
Atom  6 : 0101101110 1110100000   366   928 -0.5825689   1.6676285  f= -178.9272062
Atom  7 : 1010000000 1110000100   640   900  0.5145024   1.5555191  f= -166.8537975
Atom  8 : 1011110001 0010010001   753   145  0.9669443  -1.4674330  f= -577.1605317
Atom  9 : 0101001011 1100010101   331   789 -0.7227058   1.1110850  f= -37.6340710
Generation= 0
Atom  0 : 0111110001 1000101010   497   554 -0.0580567   0.1701662  f= -3.9015611
Atom  1 : 1010000001 0000010001   641    17  0.5185064  -1.9799335  f= -505.9340499
Atom  2 : 1101001001 1100000110   841   774  1.3192884   1.0510264  f= -47.6423362
Atom  3 : 0101001011 1100010100   331   788 -0.7227058   1.1070811  f= -37.1641885
Atom  4 : 0111111100 1000001010   508   522 -0.0140137   0.0420411  f= -1.2033214
Atom  5 : 0110100111 0000111000   423    56 -0.3543460  -1.8237810  f= -381.8277361
Atom  6 : 0010000000 1110000100   128   900 -1.5354995   1.5555191  f= -70.7876094
Atom  7 : 1101001001 1100001111   841   783  1.3192884   1.0870616  f= -42.8029701
Atom  8 : 0110000010 0001001001   386    73 -0.5024907  -1.7557146  f= -405.5488141
Atom  9 : 1101001001 1100001100   841   780  1.3192884   1.0750499  f= -44.3872358
Best After Culture Algorithm
    0111111100      1000001010    508   522 -0.0140137   0.0420411  f=  1.2033214

Generation= 1
Atom  0 : 0111111100 1000001010   508   522 -0.0140137   0.0420411  f= -1.2033214
Atom  1 : 0111110011 1000101000   499   552 -0.0500489   0.1621584  f= -3.6515256
Atom  2 : 1101001001 1000001010   841   522  1.3192884   0.0420411  f= -288.5856278
Atom  3 : 1111110001 1000001100  1009   524  1.9919453   0.0500489  f= -1535.8973194
Atom  4 : 0101001011 1100010100   331   788 -0.7227058   1.1070811  f= -37.1641885
Atom  5 : 0111110001 1000001010   497   522 -0.0580567   0.0420411  f= -1.2690245
Atom  6 : 1100110111 1110000100   823   900  1.2472180   1.5555191  f= -0.0611168
Atom  7 : 0111110001 1000101010   497   554 -0.0580567   0.1701662  f= -3.9015611
Atom  8 : 0101001001 1100001100   329   780 -0.7307136   1.0750499  f= -32.2751029
Atom  9 : 1101001011 1100010100   843   788  1.3272962   1.1070811  f= -42.9616949
Best After Culture Algorithm
    1100110111      1110000100    823   900  1.2472180   1.5555191  f=  0.0611168

Generation= 2
Atom  0 : 1100110111 1110000100   823   900  1.2472180   1.5555191  f= -0.0611168
Atom  1 : 0111111100 1000001010   508   522 -0.0140137   0.0420411  f= -1.2033214
Atom  2 : 1101010001 1000101010   849   554  1.3513196   0.1701662  f= -274.3234473
Atom  3 : 0101001001 1000101010   329   554 -0.7307136   0.1701662  f= -16.2286796
Atom  4 : 0111110001 0000101010   497    42 -0.0580567  -1.8798358  f= -355.7661023
Atom  5 : 0111001011 1100010100   459   788 -0.2102053   1.1070811  f= -114.4391483
Atom  6 : 0111110001 1010000100   497   644 -0.0580567   0.5305181  f= -28.9079331
Atom  7 : 0111110011 1000001010   499   522 -0.0500489   0.0420411  f= -1.2589135
Atom  8 : 0101001011 1100001010   331   778 -0.7227058   1.0670420  f= -32.6417084
Atom  9 : 1101001011 1100010100   843   788  1.3272962   1.1070811  f= -42.9616949
Best After Culture Algorithm
    1100110111      1110000100    823   900  1.2472180   1.5555191  f=  0.0611168

Generation= 3
Atom  0 : 1100110111 1110000100   823   900  1.2472180   1.5555191  f= -0.0611168
```

Εικόνα 7: Αρχείο Αποτελεσμάτων

2.3 Τεχνική Περιγραφή του κώδικα (technical reference)

Παρακάτω θα περιγραφεί η τεχνική υλοποίση της πτυχιακής. Όπως αναφέρθηκε η εφαρμογή απαρτίζεται από δύο φόρμες. Η πρώτη φόρμα είναι το κύριο μενού της εφαρμογής, όπου ο χρήστης μπορεί να επιλέξει ή να εισάγει τιμές για τις παραμέτρους του αλγορίθμου καθώς και να επιλέξει αν θέλει να σώσει τα αποτελέσματα της εφαρμογής σε αρχείο της προτίμησής του. Εφόσον επιλέξει τις τιμές των παραμέτρων, πατώντας στη συνεχεια το κουμπί τρέξιμο (Run), ξεκινάει το τρέξιμο του αλγορίθμου. Η δεύτερη φόρμα σχετίζεται με την εμφάνιση των παραμέτρων που ο χρήστης εισήγαγε καθώς και με τα αποτελέσματα του αλγορίθμου. Όσον αφορά τα αποτελέσματα ο χρήστης μπορεί να δεί ακριβώς τις καλύτερες λύσεις κάθε γενίας, όπως και των χώρο μνήμης όπου αποθηκεύτηκαν οι καλύτερες λύσεις.

Επιπλέον, μπορεί να ανοίξει το αρχείο που σώθηκαν όλα τα αποτελέσματα και τέλος, να επιστρέψει στο βασικό μενού της εφαρμογής ή να τερματίσει την εφαρμογή.

2.3.1 Βασική φόρμα ή κύριο μενού

Τα αρχεία της βασικής φόρμας είναι:

- Basic.cpp
- Basic.h

2.3.1.1 Basic.cpp

Συνάρτηση: TBasicForm

Περιγραφή:

Στη συνάρτηση TBasicForm γίνεται η αρχικοποίηση των αντικειμένων (Objects) που χρησιμοποιούνται στη Βασική Φόρμα της εφαρμογής. Τα συγκεκριμένα οπτικά αντικείμενα είναι τα ComboBox που είναι σύνθετα πλαίσια τα οποία επιτρέπουν στον χρήστη να εισάγει κάποια τιμή ή να επιλέξει μια τιμή από μία λίστα και τα CheckBox τα οποία είναι κουτάκια ελέγχου στα οποία ο χρήστης επιλέγει αν θέλει ή όχι μια ενέργεια. Ο πηγαίος κώδικας της TBasicForm παρατίθεται στη συνέχεια.

Πηγαίος Κώδικας:

```
__fastcall TBasicForm::TBasicForm(TComponent* Owner)
      : TForm(Owner)
{
   Population->ItemIndex=0; // Combobox για την τιμή του πληθυσμού
   Generations->ItemIndex=0; // Combobox για την τιμή των γενεών
   Beliefspace->ItemIndex=0; // Combobox για την τιμή του χώρου μνήμης
   Bits->ItemIndex=0; // Combobox για την τιμή των bits
   Problem->ItemIndex=0; // Combobox για το πρόβλημα που θα επιλυθεί
   Mutation->ItemIndex=2; // Combobox για την παράμετρο της μετάλλαξης
   Variation->ItemIndex=0; // Combobox για τν τρόπο επικοινωνίας μεταξύ Α.Κ και Γ.Κ
   Hillclimb->Checked=true; // CheckBox για τον τελεστή αναρρίχησης
   EditName->Visible=false; // EditBox για την εισαγωγή του ονόματος του αρχείου
   Label8->Visible=false; // Label για την εμφάνιση μηνύματος προς τον χρήστη
}
```

Συνάρτηση: RunClick

Περιγραφή:

Η συνάρτηση RunClick καλείται όταν κάνουμε κλικ το κουμπί Τρέξιμο (Run). Δεν επιστρέφει τίποτα και δέχεται σαν όρισμα τον δείκτη του control που έστειλε το message

δηλαδή που παρήγαγε το event. Σε αυτήν τη συνάρτηση μετράται ο χρόνος εκτέλεσης του προγράμματος και γίνονται οι βασικοί έλεγχοι για τις τιμές που πληκτρολογεί ο χρήστης στα πεδία ή όταν δεν υπάρχουν καθόλου τιμές στα πεδία. Επιπλέον, σε αυτήν την συνάρτηση καλείται ο βασικός αλγόριθμος, γίνεται η αρχικοποίηση των τιμών της δεύτερης φόρμας καθώς, και η εμφάνιση της. Στη συνέχεια παρατίθεται ο πηγαίος κώδικας της RunClick.

Πηγαίος Κώδικας:

```
void __fastcall TBasicForm::RunClick(TObject *Sender)
{
	float a;
	int b;
	int start_time, end_time;
	//Εναρξή της χρονομέτρησης του αλγορίθμου.
	start_time = GetTickCount();


// Η συνάρτηση TryStrToFloat μετατρέπει μια συμβολοσειρά σε πραγματικό αριθμό.
	if(!TryStrToFloat(Mutation->Text.c_str(),a))
	{ //return a float number
		ShowMessage("You must enter numerical characters");
		if(a==0)
			ShowMessage("You can't set 0 for Mutation Rate");
		Mutation->SetFocus();
		return;
	 }
	 else if(!TryStrToInt(Population->Text.c_str(),b) || b<10)
	 {
		ShowMessage("You must set only integer number in Population Field and bigger than 10");

		Mutation->SetFocus();
		return;
	 }
	 else if(!TryStrToInt(Generations->Text.c_str(),b) || b<10)
	 {
		ShowMessage("You must set only integer number in Generation Field and bigger than 10");
		Mutation->SetFocus();
		return;
	 }
	  else if(!TryStrToInt(Beliefspace->Text.c_str(),b) || b<5)
	 {
		ShowMessage("You must set only integer number in Belief Space Field and bigger than 5");
		Mutation->SetFocus();
```

```
            return;
        }
         else if(!TryStrToInt(Bits->Text.c_str(),b) || b<10)
        {
            ShowMessage("You must set only integer number in BITS Field and bigger
than 10");
            Mutation->SetFocus();
            return;
        }

        int B=BasicForm->Beliefspace->Text.ToInt();
                /* Αρχικοποιούμε το πάνελ στο οποίο βλέπουμε τις τιμές του
                χώρου μνήμης έτσι ώστε κάθε φορά που τρέχουμε το πρόγραμμα
                το πάνελ να είναι άδειο και να μην περιέχει σκουπίδια*/
        for(int i=0; i<B; i++)
            ViewForm->Panel2->DoubleBuffered=true;
            ViewForm->LabelBeliefS->Update();
        // Αρχικοποιήση των τιμών την αντικειμένων της δεύτερης φόρμας.
        initialize();
                // Εκτέλεση του Βασικού Αλγορίθμου.
        algorithm();

                // Εμφάνιση της View φόρμας.
        ViewForm->Show();
        BasicForm->Hide();
           // Έλεγχος για αν έχουμε επιλέξει να σώσουμε τα αποτελέσματα του αλγορίθμου.
        if(Savefile->Checked==false)
            ViewForm->Openfilebutton->Enabled=false;
        //Τέλος της χρονομέτρησης του αλγορίθμου.
        end_time = GetTickCount() - start_time;
        ViewForm->PanelTime->Caption = FloatToStr(end_time / 1000.0) + "sec";
    }
```

Συνάρτηση: ExitClick

Περιγραφή:

Η συνάρτηση ExitClick καλείται όταν κάνουμε κλικ στο κουμπί Έξοδος (Exit) και οδηγεί σε τερματισμό της εφαρμογής. Δεν επιστρέφει τίποτα και δέχεται σαν όρισμα τον δείκτη του control που έστειλε το message δηλαδή που παρήγαγε το event.

Πηγαίος Κώδικας:

```
void __fastcall TBasicForm::ExitClick(TObject *Sender)
```

```
{
    BasicForm->Close();
}
```

Συνάρτηση: SavefileClick

Περιγραφή:

Η συνάρτηση SavefileClick εκτελείται όταν κάνουμε κλικ το κουτάκι Save file, στην περίπτωση που θέλουμε να σώσουμε τα αποτελέσματα του αλγόριθμου σε αρχείο. Δεν επιστρέφει τίποτα και δέχεται σαν όρισμα τον δείκτη του control που έστειλε το message δηλαδή που παρήγαγε το event. Μας εμφανίζει το πλαίσιο για να ονοματίσουμε το αρχείο στο οποίο θέλουμε να σώσουμε τα αποτελέσματα ή παίρνει ένα προκαθορισμένο όνομα «result.txt». Στην περίπτωση που δεν το έχουμε επιλέξει δε μας εμφάνιζει κάποιο πλαίσιο. Στη συνέχεια παρουσιάζεται ο πηγαίος κώδικας της συνάρτησης SavefileClick.

Πηγαίος Κώδικας:

```
void __fastcall TBasicForm::SavefileClick(TObject *Sender)
{
        if(Savefile->Checked==true)
        {
            EditName->Visible=true;
            Label8->Visible=true;
        }
        else
        {
          EditName->Visible=false;
          Label8->Visible=false;
        }
}
```

2.3.1.2 Basic.h

Περιγραφή:

Το συγκεκριμένο αρχείο δημιουργείται αυτόματα όταν δημιουργούμε μια νέα φόρμα. Έτσι βλέπουμε την κλάση της βασικής φόρμας (TBasicForm) η οποία κλειρονομεί όλες τις ιδιότητες από την βασική κλάση (TForm). Συνεπώς, όλα τα αντικείμενα, που χρησιμοποιούμε στην φόρμα μας, δηλώνονται αυτόματα εδώ καθώς και οι δηλώσεις των συναρτήσεων μελών που θα χρησιμοποιηθούν στο αρχείο. Το σώμα των συναρτήσεων βρίσκεται στο .cpp αρχείο. Πιο συγκεκριμένα, βλέπουμε ότι στην κλαση TBasicForm έχουν δηλώθεί όλα τα αντικείμενα που

τοποθετήσαμε για τις ανάγκες της εφαρμογής μας. Για παράδειγμα, το αντικείμενο Label1 είναι τύπου TLabel.

Στον παρακάτω πίνακα παρουσιάζονται οι Controlers που χρησιμοποιούνται στη βασική φόρμα και οι περιγραφές αυτών.

Controller	**Περιγραφή**
Label1	Ετικέτα τύπου (TLabel)
Label2	Ετικέτα τύπου (TLabel)
Label3	Ετικέτα τύπου (TLabel)
Label4	Ετικέτα τύπου (TLabel)
Label5	Ετικέτα τύπου (TLabel)
Label6	Ετικέτα τύπου (TLabel)
Label7	Ετικέτα τύπου (TLabel)
Generation	Σύνθετο πλαίσιο επιλογής τύπου (TCombobox). Περιέχει προεπιλεγμένες τιμές για την παράμετρο των γενεών.
BeliefSpace	Σύνθετο πλαίσιο επιλογής τύπου (TCombobox). Περιέχει προεπιλεγμένες τιμές για το μέγεθος του χώρου μνήμης.
Bits	Σύνθετο πλαίσιο επιλογής τύπου (TCombobox). Περιέχει προεπιλεγμένες τιμές για το μέγεθος της συμβολοσειράς σε bits.
Problem	Σύνθετο πλαίσιο επιλογής τύπου (TCombobox). Περιέχει τα προβλήματα που καλείται να επιλύση ο αλγόριθμος.
Mutation	Σύνθετο πλαίσιο επιλογής τύπου (TCombobox). Περιέχει προεπιλεγμένες τιμές για τον βαθμό της μετάλλαξης.
Variation	Σύνθετο πλαίσιο επιλογής τύπου (TCombobox). Περιέχει προεπιλεγμένες τιμές για τον τρόπο που αλληλεπιδρά ο χώρος μνήμης με τον Γ.Α.
Savefile	Πλαίσιο ελέγχου τύπου (TCheckBox). Επιτρέπει την αποθήκευση των αποτελεσμάτων ή όχι.
HillClimb	Πλαίσιο ελέγχου τύπου (TCheckBox). Επιτρέπει τη χρήση του τελεστή αναρρίχησης ή όχι.
Run	Κουμπί ενέργειας τύπου (TButton). Τρέχει τον αλγόριθμο και ανοίγει την φόρμα των αποτελεσμάτων.
Exit	Κουμπί ενέργειας τύπου (TButton). Κλείνει/τερματίζει την εφαρμογή.
Population	Σύνθετο πλαίσιο επιλογής τύπου (TCombobox). Περιέχει προεπιλεγμένες τιμές για το μέγεθος του πληθυσμού.
EditName	Πλαίσιο επεξεργασίας τύπου (TEdit). Δέχεται το όνομα του αρχείου που θα αποθηκευτούν τα αποτελέσματα.
Label8	Ετικέτα τυπου (TLabel)

Πίνακας 2: Περιγραφή των Controllers του Basic.h

Πηγαίος Κώδικας Αρχείου Κεφαλίδας:

class TBasicForm : public TForm

```
{
__published:   // IDE-managed Components
        TLabel *Label1;
        TLabel *Label2;
        TLabel *Label4;
        TLabel *Label5;
        TLabel *Label6;
        TLabel *Label7;
        TComboBox *Generations;
        TComboBox *Beliefspace;
        TComboBox *Bits;
        TComboBox *Problem;
        TComboBox *Mutation;
        TComboBox *Variation;
        TCheckBox *Savefile;
        TCheckBox *Hillclimb;
        TButton *Run;
        TButton *Exit;
        TLabel *Label3;
        TComboBox *Population;
        TEdit *EditName;
        TLabel *Label8;
        void __fastcall RunClick(TObject *Sender);
        void __fastcall ExitClick(TObject *Sender);
        void __fastcall SavefileClick(TObject *Sender);


private:        // User declarations
public:         // User declarations
        __fastcall TBasicForm(TComponent* Owner);
};
```

2.3.2 Φόρμα Εμφάνισης Αποτελεσμάτων

Τα αρχεία της φόρμας εμφάνισης αποτελεσμάτων είναι:

- View.cpp
- View.h

2.3.2.1 View.cpp

Συνάρτηση: BackbuttonClick

Περιγραφή:

Η συνάρτηση BackbuttonClick καλείται όταν πατήσουμε το κουμπί πίσω (Back). Καθαρίζουμε τον χώρο όπου εμφανίζεται ο χώρος μνήμης, κλείνουμε την φόρμα, και επιστρέφουμε στην βασική φόρμα. Δεν επιστρέφει τίποτα και δέχεται σαν όρισμα τον δείκτη του control που έστειλε το message δηλαδή που παρήγαγε το event. Ο πηγαίος κώδικας της συνάρτησης παρουσιάζεται στη συνέχεια.

Πηγαίος Κώδικας:

```
void __fastcall TViewForm::BackbuttonClick(TObject *Sender)
{
      int B=BasicForm->Beliefspace->Text.ToInt();
      for(int i=0; i<B; i++)
            ViewForm->Panel2->DoubleBuffered=true;
            ViewForm->LabelBeliefS->Update();
      ViewForm->Close();
      cleandata();
      BasicForm->Show();
}
```

Συνάρτηση: ExitbuttonClick

Περιγραφή:

Η συνάρτηση ExitbuttonClick καλείται όταν πατήσουμε το κουμπί έξοδος και οδηγεί στο οριστικό κλείσιμο της εφαρμογής. Δεν επιστρέφει τίποτα και δέχεται σαν όρισμα τον δείκτη του control που έστειλε το message δηλαδή που παρήγαγε το event. Ο πηγαίος κώδικας της συνάρτησης παρατίθεται στη συνέχεια.

Πηγαίος Κώδικας:

```
void __fastcall TViewForm::ExitbuttonClick(TObject *Sender)
{
      exit(0);
}
```

Συνάρτηση: OpenfilebuttonClick

Περιγραφή:

Η συνάρτηση OpenfilebuttonClick καλείται όταν πατήσουμε το κουμπί άνοιγμα του αρχείου. Δεν επιστρέφει τίποτα και δέχεται σαν όρισμα τον δείκτη του control που έστειλε το message δηλαδή που παρήγαγε το event. Με αυτή τη συνάρτηση ανοίγουμε το αρχείο που επιλέξαμε να σώσουμε τα αποτελέσματα της εκτέλεσης του αλγορίθμου. Επίσης, καλείται η συνάρτηση ShellExecute. Η συγκεκριμένη συνάρτηση καλείται με ορίσματα την λειτουργία, το

όνομα του αρχείου, τις παραμέτρους και τον τρόπο προβολής του αρχείου. Στη συνέχεια παρουσιάζεται ο πηγαίος κώδικας της συνάρτησης OpenfilebuttonClick.

Πηγαίος Κώδικας:

```
void __fastcall TViewForm::OpenfilebuttonClick(TObject *Sender)
{
        if(BasicForm->EditName->Text!="")
        {
          String name=BasicForm->EditName->Text + ".txt";
          ShellExecute(0, "open", name.c_str(), NULL, NULL,
SW_SHOWNORMAL); // Ανοίγει το αρχείο που ο χρήστης πληκτρολόγησε.
        }
        else
            ShellExecute(0, "open", "results.txt", NULL, NULL,
SW_SHOWNORMAL); //Ανοίγει το αρχείο με το προκαθορισμένο αρχείο.
}
```

2.3.2.2 View.h

Περιγραφή:

Το συγκεκριμένο αρχείο δημιουργείται αυτόματα όταν δημιουργούμε μια νέα φόρμα. Έτσι βλέπουμε την κλάση της φόρμας των αποτελεσμάτων (TViewForm) η οποία κλειρονομεί όλες τις ιδιότητες από την βασική κλάση (TForm). Συνεπώς, όλα τα αντικείμενα, που χρησιμοποιούμε στην φόρμα μας, δηλώνονται αυτόματα εδώ καθώς και οι δηλώσεις των συναρτήσεων μελών που θα χρησιμοποιηθούν στο αρχείο. Το σώμα των συναρτήσεων βρίσκεται στο .cpp αρχείο. Πιο συγκεκριμένα, βλέπουμε ότι στην κλαση TViewForm έχουν δηλώθεί όλα τα αντικείμενα που τοποθετήσαμε για τις ανάγκες της εφαρμογής μας. Για παράδειγμα, το αντικείμενο Panel1 είναι τύπου TPanel.

Στον παρακάτω πίνακα παρουσιάζονται οι Controlers που χρησιμοποιούνται στη φόρμα των αποτελεσμάτων και οι περιγραφές αυτών.

Controller	**Περιγραφή**
Panel1	Πλαίσιο εμφάνισης δεδομένων τύπου (TPanel). Περιέχει όλα τα πλαίσια τα οποία εμφανίζουν τις πληροφορίες για τις παραμέτρους που εισήγαγε ο χρήστης.
Panel2	Πλαίσιο εμφάνισης δεδομένων τύπου (TPanel). Περιέχει την ετικέτα όπου εμφανίζεται ο χώρος μνήμης
PanelPop	Πλαίσιο εμφάνισης δεδομένων τύπου (TPanel). Εμφανίζεται η παράμετρος του πληθυσμού που ο χρήστης εισήγαγε.
PanelGen	Πλαίσιο εμφάνισης δεδομένων τύπου (TPanel). Εμφανίζεται η παράμετρος των γενεών που ο χρήστης εισήγαγε.
PanelBS	Πλαίσιο εμφάνισης δεδομένων τύπου (TPanel). Εμφανίζεται η

	παράμετρος του χώρου μνήμης που ο χρήστης εισήγαγε.
PanelBITS	Πλαίσιο εμφάνισης δεδομένων τύπου (TPanel). Εμφανίζεται η παράμετρος των bits που ο χρήστης εισήγαγε.
Panelfunc	Πλαίσιο εμφάνισης δεδομένων τύπου (TPanel). Εμφανίζεται η παράμετρος του προβλήματος που ο χρήστης εισήγαγε.
Panelmrate	Πλαίσιο εμφάνισης δεδομένων τύπου (TPanel). Εμφανίζεται η παράμετρος της μετάλλαξης που ο χρήστης εισήγαγε.
PanelVar	Πλαίσιο εμφάνισης δεδομένων τύπου (TPanel). Εμφανίζεται η παράμετρος του τρόπου αλληλεπίδρασης μεταξύ του χώρου μνήμης και του Γ.Α που ο χρήστης εισήγαγε.
PanelSV	Πλαίσιο εμφάνισης δεδομένων τύπου (TPanel). Εμφανίζεται η παράμετρος για την αποθήκευση του αρχείου.
PanelTime	Πλαίσιο εμφάνισης δεδομένων τύπου (TPanel). Εμφανίζεται η παράμετρος του χρόνου που ο αλγόριθμος χρειάστηκε για να εκτελεστεί.
PanelHC	Πλαίσιο εμφάνισης δεδομένων τύπου (TPanel). Εμφανίζεται η παράμετρος του τελεστή αναρρίχησης.
Label1	Ετικέτα τυπου (TLabel)
Label2	Ετικέτα τυπου (TLabel)
Label3	Ετικέτα τυπου (TLabel)
LabelPop	Ετικέτα τυπου (TLabel)
LabelGen	Ετικέτα τυπου (TLabel)
LabelBS	Ετικέτα τυπου (TLabel)
Labelfunc	Ετικέτα τυπου (TLabel)
Labelmrate	Ετικέτα τυπου (TLabel)
LabelVar	Ετικέτα τυπου (TLabel)
LabelHC	Ετικέτα τυπου (TLabel)
LabelSV	Ετικέτα τυπου (TLabel)
LabelBeliefS	Ετικέτα τυπου (TLabel). Εμφανίζεται ο χώρος μνήμης δηλαδή οι καλύτερες τιμές του πολιτισμού.
LabelTime	Ετικέτα τυπου (TLabel)
Table	Πίνακας τύπου (TStringGrid). Εμφανίζει έναν πίνακα με την καλύτερη λύση κάθε γενίας.
Backbutton	Κουμπί ενέργειας τύπου (TButton). Επιστρέφει τον χρήστη στην βασική φόρμα της εφαρμογής.
Exitbutton	Κουμπί ενέργειας τύπου (TButton). Τερματίζει την εφαρμογή.
Openfilebutton	Κουμπί ενέργειας τύπου (TButton). Ανοίγει το αρχείο που σώθηκαν τα αποτελέσματα.

Πίνακας 3: Περιγραφή των Controllers του View.h

Πηγαίος Κώδικας Αρχείου Κεφαλίδας:

```
class TViewForm : public TForm
{
__published:   // IDE-managed Components
```

```
        TPanel *Panel1;
        TPanel *Panel2;
        TLabel *LabelPop;
        TLabel *LabelGen;
        TLabel *LabelBS;
        TPanel *PanelPop;
        TPanel *PanelGen;
        TPanel *PanelBS;
        TLabel *Label1;
        TLabel *Labelfunc;
        TLabel *Labelmrate;
        TPanel *PanelBITS;
        TPanel *Panelfunc;
        TPanel *Panelmrate;
        TLabel *LabelVar;
        TPanel *PanelVar;
        TLabel *LabelHC;
        TPanel *PanelHC;
        TLabel *LabelSV;
        TPanel *PanelSV;
        TLabel *LabelBeliefS;
        TStringGrid *Table;
        TLabel *Label2;
        TLabel *Label3;
        TButton *Backbutton;
        TButton *Exitbutton;
        TButton *Openfilebutton;
        TLabel *LabelTime;
        TPanel *PanelTime;
        void __fastcall BackbuttonClick(TObject *Sender);
        void __fastcall ExitbuttonClick(TObject *Sender);
        void __fastcall OpenfilebuttonClick(TObject *Sender);


    private:        // User declarations
    public:         // User declarations
        __fastcall TViewForm(TComponent* Owner);
    };
```

2.3.3 Αρχείο VisualFunction

Περιγραφή:

Τα αρχεία της VisualFunction είναι υπεύθυνα για την εμφάνιση των δεδομένων στα αντικείμενα της φόρμας των αποτελεσμάτων. Μεταφέρουν και εμφανίζουν τις τιμές των

παραμέτρων που έθεσε ο χρήστης. Επιπλέον, εμφανίζει τα αποτελέσματα του γενετικού αλγόριθμου όπως και τον χώρο μνήμης.

Συγκεκριμένα, τα αρχεία του VisualFunction είναι:

- VisualFunction.cpp
- VisualFunction.h

2.3.3.1 VisualFunction.cpp

Συνάρτηση: iniTable()

Περιγραφή:

Η συνάρτηση iniTable() καλείται για να αρχικοποιήσει τις στήλες που εμφανίζονται στον πίνακα της φόρμας των αποτελεσμάτων. Στη συνέχεια παρουσιάζεται ο πηγαίος κώδικας της συνάρτησης iniTable().

Πηγαίος Κώδικας:

```
void iniTable()
{
     // ViewForm->Table->ColCount=8;
      ViewForm->Table->RowCount=BasicForm->Generations->Text.ToInt()+1;
       ViewForm->Table->Cells[0][0]="Generations";
       ViewForm->Table->Cells[1][0]="BinaryGeno0";
       ViewForm->Table->Cells[2][0]="BinaryGeno1";
       ViewForm->Table->Cells[3][0]="Geno0";
       ViewForm->Table->Cells[4][0]="Geno1";
       ViewForm->Table->Cells[5][0]="Pheno0";
       ViewForm->Table->Cells[6][0]="Pheno1";
       ViewForm->Table->Cells[7][0]="Fitness";

       ViewForm->Table->ColWidths[0]=50;
       ViewForm->Table->ColWidths[1]=70;
       ViewForm->Table->ColWidths[2]=70;
       ViewForm->Table->ColWidths[3]=40;
       ViewForm->Table->ColWidths[4]=40;
       ViewForm->Table->ColWidths[5]=110;
       ViewForm->Table->ColWidths[6]=110;
       ViewForm->Table->ColWidths[7]=110;
}
```

Συνάρτηση: initialize()

Περιγραφή:

Η συνάρτηση initialize() καλείται για να αρχικοποιήσει τις τιμές που ο χρήστης εισήγαγε στη βασική φόρμα όσον αφορά τα πεδία των παραμέτρων του αλγορίθμου. Ο πηγαίος κώδικας της initialize() παρατίθεται στη συνέχεια.

Πηγαίος Κώδικας:

```
void initialize()
{

      ViewForm->PanelPop->Caption=BasicForm->Population->Text.ToInt();
      ViewForm->PanelGen->Caption=BasicForm->Generations->Text.ToInt();
      ViewForm->PanelBS->Caption=BasicForm->Beliefspace->Text.ToInt();
      ViewForm->PanelBITS->Caption=BasicForm->Bits->Text.ToInt();

      ViewForm->Panelfunc->Caption=BasicForm->Problem->Text;

     ViewForm->Panelmrate->Caption=BasicForm->Mutation->Text.ToDouble();

     ViewForm->PanelVar->Caption=BasicForm->Variation->Text;

     if(BasicForm->Hillclimb->Checked==true)
           ViewForm->PanelHC->Caption="true";
     else
           ViewForm->PanelHC->Caption="false";

    if(BasicForm->Savefile->Checked==true)
           ViewForm->PanelSV->Caption="true";
     else
           ViewForm->PanelSV->Caption="false";

    iniTable();
}
```

Συνάρτηση: cleandata()

Περιγραφή:

Η συνάρτηση cleandata() καλείται για να καθαρίσει τα δεδομένα του χώρου μνήμης. Κάνει ενεργό το κουμπί άνοιγμα αρχείου και καλεί τη συνάρτηση cleanFlagBS, η οποία είναι αυτή που αρχικοποιεί τη μεταβλητή flagBS. Ο πηγαίος κώδικας της συνάρτησης cleandata() φαίνεται παρακάτω.

Πηγαίος Κώδικας:

```
void cleandata()
{

	ViewForm->LabelBeliefS->Caption=" ";
	ViewForm->Openfilebutton->Enabled=true;
	cleanFlagBS();
}
```

2.3.3.3 VisualFunction.h

Περιγραφή:

Δήλωση των συναρτήσεων μελών που θα χρησιμοποιηθούν στο αρχείο.

Πηγαίος Κώδικας Αρχείου Κεφαλίδας:

```
void initialize();
void iniTable();
void cleandata();
```

2.3.4 Αρχεία Functions

Περιγραφή:

Σε αυτά τα αρχεία περιέχονται όλες οι συναρτήσεις για την εκτέλεση του αλγορίθμου. Τα αρχεία του Functions είναι:

- Functions.cpp
- Functions.h

2.3.4.1 Functions.cpp

Περιγραφή:

Δήλωση της δομής δεδομένων του ατόμου με χαρακτηριστικά το γονότυπο, το φαινότυπο και την ποιότητα του.

Πηγαίος Κώδικας:

```
struct atom{
	int geno[VARS];
	double pheno[VARS];
```

```
    double fitness;
};
```

Συνάρτηση: rnd()

Περιγραφή:

Δήλωση της συνάρτηση rnd() η οποία όταν καλείται μας επιστρέφει έναν τυχαίο αριθμό από 0 έως 1.

Πηγαίος Κώδικας:

```
double rnd(void)
{
    return (double)rand()/(RAND_MAX+0.1);
}
```

Συνάρτηση: Mutation()

Περιγραφή:

Δήλωση της συνάρτησης Mutation, η οποία είναι υπεύθυνη για την μετάλλαξη του ατόμου. Δέχεται το άτομο, την πιθανότητα της μετάλλαξης, και τα BITS που έχουμε θέσει. Μας επιστρέφει το μεταλλαγμένο άτομο.

Πηγαίος Κώδικας:

```
atom Mutation (atom x, double prob, int BITS)
{
    //ginetai praksi xor gia tin antistrofi tou bit
    double E,dek;
    int ak,mp,i,point,mask,Chrom,Chrompos;
    E=VARS*BITS*prob;        // E=ektimisi shmeiwn metallaksis
    ak=(int)E;
    dek=E-ak;
    mp=ak;
    if(rnd()<dek) mp++;
    //printf("BITS=%i prob=%lf E=%lf ak=%i dek=%lf mp=%i\n",BITS,prob,E,ak,dek,mp);
    for(i=0;i<mp;i++)
    {
        point=random(BITS*VARS); // 0 eos 39
        //printf("Before =%s\n",x.c_str());
        //printf("point=%i\n",point);
```

```
            Chrom = point/BITS; // 0,1,2,3...
            Chrompos = point%BITS;
            mask = 1<<(BITS-Chrompos-1) ;
            x.geno[Chrom]=x.geno[Chrom]^mask;
           //printf("After =%s\n",x.c_str());
        }
        //getch();
        return x;
}
```

Συνάρτηση: Crossover()

Περιγραφή:

Δήλωση της συνάρτησης Crossover, η οποία είναι υπεύθυνη για τον ανασυνδυασμό των γονέων για την παραγωγή του απογόνου. Δέχεται τους γονείς και τα BITS που έχουμε θέσει και επιστρέφει τον απόγονο.

Αρχικά βρίσκει έναν τυχαίο αριθμό point μεταξύ 0 και BITS*VARS και αποθηκεύει το αποτέλεσμα της διαίρεσης point/BITS στην μεταβλητή chrom. Άρα, το αποτέλεσμα της διαίρεσης θα ειναι 0 ή 1. Αποθηκεύει το υπόλοιπο της διαίρεσης στην μεταβλητή chrompos αν το αποτέλεσμα είναι 0 και εκτελείται η δεύτερη for όπου ο γονότυπος του παιδιού γίνεται ίσος με τον γονότυπο του δεύτερου γονέα. Αν το αποτέλεσμα είναι 1 εκτελείται η πρώτη for όπου ο γονότυπος του παιδιού γίνεται ίσος με τον γονότυπο του δεύτερου γονέα.

Στην συνέχεια βρίσκουμε τη μάσκα για να δημιουργήσουμε το γονότυπο που δεν αντιγράψαμε από κάποιον γονέα.

Πηγαίος Κώδικας:

```
atom Crossover (atom p1, atom p2, int BITS)
{
        //and &
        //or  |
        //xor ^
        //not ~

        int point, mask2, Chrom, Chrompos, j;
        atom child;
        point = random(VARS * BITS-1); //τυχαιός αριθμός απο 0 εώς BITS*VARS
        Chrom = point/BITS; // ο τυχαίος αριθμός διά των αριθμό των BITS
        Chrompos = point%BITS;  //από την θέση 0 έώς 7 το χρωμόσωμα όπου θα γίνει το
crossover
        for(j=0;j<Chrom;j++) {
            child.geno[j]=p1.geno[j];
```

```
        }
        for(j=Chrom+1;j<VARS;j++)
        {
              child.geno[j]=p2.geno[j];
        }
        mask2 = (1<<BITS-Chrompos-1)-1;

                    //ορισμός του χρωμοσώματος όπου θα γίνει το crossover
                    child.geno[Chrom]  =  (p1.geno[Chrom]&(~mask2))|(p2.geno[Chrom]&
mask2);

        return child;
  }
```

Συνάρτηση: Int2Str()

Περιγραφή:

Η συνάρτηση Int2Str δέχεται έναν ακέραιο αριθμό και τον αριθμό των BITS και μας επιστρέφει μια συμβολοσειρά από 0 και 1. Συνεπώς, μετατρέπει τους ακέραιους σε συμβολοσειρά από 0 και 1. Στη συνέχεια παρουσιάζεται ο πηγαίος κώδικας της Int2Str.

Πηγαίος Κώδικας:

```
String Int2Str(int x, int BITS)
{
      String s="";
      int mask,i;
      mask= 1<<(BITS-1);
      for(i=0;i<BITS;i++)
      {
            if(x&mask) s+='1';
      else
                  s+='0';
      mask=mask>>1;
      }
      return s;
}
```

Συνάρτηση: Str2Int()

Περιγραφή:

Η συνάρτηση Str2Int δέχεται μια συμβολοσειρά από 0 και 1και τον αριθμό των BITS και μας επιστρέφει έναν ακέραιο. Συνεπώς, μετατρέπει τη συμβολοσειρά από 0 και 1 σε ακέραιο αριθμό. Ο πηγαίος κώδικας της Str2Int φαίνεται παρακάτω.

Πηγαίος Κώδικας:

```
int Str2Int(String x, int BITS)
{
	int i,sum=0,p;

	p=1<<(BITS-1); //128
	for(i=1;i<=BITS;i++)
	  {
	    if(x[i] =='1')
	    sum+=p;
	    p=p/2; // p=p>>1;
	  }
	return sum;
}
```

Συνάρτηση: Geno2Pheno()

Περιγραφή:

Η συνάρτηση Geno2Pheno δέχεται ένα άτομο by reference, τον αριθμό των BITS και τη συνάρτηση για την οποία εκτελείται. Δεν επιστρέφει τίποτα. Αυτό που κάνει είναι να μετατρέπει τον γονότυπο του ατόμου σε φαινότυπο. Στη συνέχεια φαίνεται ο πηγαίος κώδικας της Geno2Pheno.

Πηγαίος Κώδικας:

```
void Geno2Pheno(atom & x, int BITS, int fun)
{
	int j;
	double vhma;
	//vhma=2*M_PI/(pow(2,BITS)-1);

	for(j=0;j<VARS;j++) {
		if(fun==0)
		{
			vhma=4.096/(pow(2,BITS)-1);
			x.pheno[j]=x.geno[j]*vhma-2.048;  //Για την συνάρτηση Rosenbrock
		}
		else if(fun==1)
		{
```

```
                vhma =10.24/(pow(2,BITS)-1);
                x.pheno[j]=x.geno[j]*vhma-5.12; //Για την συνάρτηση Rastrigin
            }
            else if(fun==2)
            {
                vhma=1000/(pow(2,BITS)-1);
                x.pheno[j]=x.geno[j]*vhma-500;  //Για την συνάρτηση Schwefel
            }
            else if(fun==3)
            {
                vhma=100/(pow(2,BITS)-1);
                x.pheno[j]=x.geno[j]*vhma+0.1;      //Για την συνάρτηση του
Παραλληλογράμμου
            }
        }
    }
```

Συνάρτηση: Fitness()

Περιγραφή:

Η συνάρτηση Fitness δέχεται ένα άτομο by reference, τον αριθμό των BITS και τη συνάρτηση για την οποία εκτελείται. Δεν επιστρέφει τίποτα. Αυτό που κάνει η συνάρτηση ποιότητας (Fitness) είναι να δημιουργεί την ποιότητα του ατόμου η οποία μας χρειάζεται για την αξιολόγησή του. Ο πηγαίος κώδικας της Fitness παρατίθεται στη συνέχεια.

Πηγαίος Κώδικας:

```
void Fitness(atom & x, int BITS, int fun)
{
    double x1,x2;
    Geno2Pheno(x, BITS, fun);
    x1=x.pheno[0];
    x2=x.pheno[1];
    if(fun==0)
        x.fitness= (100*pow(x1*x1-x2,2)+pow(1-x1,2));   //rosenbrock
    else if(fun==1)
        x.fitness=  20  +  (pow(x1,2)  -  10*cos(2*M_PI*x1))  +  (pow(x2,2)  -
10*cos(2*M_PI*x2));     //Rastrigin
    else if(fun==2)
        x.fitness=  (418.9829*2)  -  x1*sin(sqrt(fabs(x1)))  -  x2*sin(sqrt(fabs(x2)));
//schwefel
    else if(fun==3)
    {
        double z,ogkos=1000;
```

```
            //ο τύπος για τον όγκο του παραλληλογράμμου=x1*x2*z
            z=ogkos/(x1*x2);
            x.fitness= (2*(x1*z))+(2*(x2*z))+(2*(x1*x2));
                                    // Το εμβαδό της παράπλευρής επιφάνειας είναι το
άθροισμα των όγκων των 6 επιφανειών
        }
    }
```

Συνάρτηση: HillClimb()

Περιγραφή:

Η συνάρτηση HillClimb δέχεται ένα άτομο by reference, τον αριθμό των BITS και τη συνάρτηση για την οποία εκτελείται. Δεν επιστρέφει τίποτα. Η βασική ιδέα αυτής της συνάρτησης είναι να βοηθάει τον γενετικό αλγόριθμο να ξεπερνάει τα τοπικά βέλτιστα έτσι, ώστε να οδηγηθεί στο πραγματικό βέλτιστο. Στη συνέχεια παρατίθεται ο πηγαίος κώδικας της HillClimb.

Πηγαίος Κώδικας:

```
void HillClimb(atom & x, int BITS, int fun)
{
    int i,j;
    int steps[4]= {10,-10,1,-1};
    atom oldx;
    oldx = x;
    for (j=0;j<VARS;j++)
        for (i=0;i<4;i++)
            if((x.geno[j]+steps[i])>=0 && x.geno[j]+steps[i]<pow(2,BITS))
            {
                x.geno[j]=x.geno[j]+steps[i];
                Fitness(x, BITS, fun);
                if(x.fitness<oldx.fitness)
                    oldx=x;
                else
                    x=oldx;
            }
}
```

Συνάρτηση: Display()

Περιγραφή:

Η συνάρτηση Display δέχεται ένα δείκτη ατόμου, τον αριθμό των BITS, έναν ακέραιο αριθμό και έναν δείκτη αρχείου. Δεν επιστρέφει τίποτα. Η βασική ιδέα αυτής της συνάρτησης είναι να εμφανίζει τα αποτελέσματα του αλγορίθμου στο αρχείο που ο χρήστης επέλεξε να σωθούν. Ο κώδικας της συνάρτησης Display παρουσιάζεται παρακάτω.

Πηγαίος Κώδικας:

```
void Display(atom * x, int N, int BITS, FILE *f1)
{
      int i,j;
       for (i=0;i<N;i++) {
            fprintf(f1,"Atom %2i : ",i);
            for (j=0;j<VARS;j++) {
                  fprintf(f1, "%s ",Int2Str(x[i].geno[j], BITS));
            }
            for (j=0;j<VARS;j++) {
                  fprintf(f1,"%5i ",x[i].geno[j]);
            }
            for (j=0;j<VARS;j++) {
                  fprintf(f1,"%10.7lf  ",x[i].pheno[j]);
            }
            if(BasicForm->Problem->ItemIndex==0)
                  fprintf(f1,"f= %10.7lf \n",x[i].fitness*(-1));
            else
                  fprintf(f1,"f= %10.7lf \n",x[i].fitness);
      }
}
```

Συνάρτηση: Roulette()

Περιγραφή:

Η συνάρτηση Roulette δέχεται ένα δείκτη ατόμου και έναν ακέραιο αριθμό. Επιστρέφει έναν ακέραιο αριθμό. Η βασική ιδέα αυτής της συνάρτησης είναι ο τροχός της ρουλέτας και αποτελεί τη μέθοδο της επιλογής των γονέων που χρησιμοποιούμε στον αλγόριθμο. Στη συνέχεια παρουσιάζεται ο πηγαίος κώδικας της συνάρτησης Roulette.

Πηγαίος Κώδικας:

```
int Roulette(atom * x, int N)
{
      double sum,sum2=0,sumrnd,max_fs,min_fs,s;
      int i;
      sum=0;
      max_fs=x[0].fitness;
```

```
s=0;
//Δυναμική δήλωση ενός δείκτη απο N πραφματικούς αριθμούς ??
        double *rev= new double[N];


//Εύρεση και αποθήκευση της μεγαλύτερης ποιότητας των υποψηφίων γονιών
        for (i=1;i<N;i++)
    if(max_fs<x[i].fitness)
        max_fs=x[i].fitness;


//Για να αποκλείσουμε την περίπτωση όπου το max_fs θα ειναι 0 προσθέτουμε το 0.1
        s=max_fs+0.1;


        //Αποθήκευση των νέων ποιοτήτων στον πίνακα rev
for (i=0;i<N;i++)
    rev[i]=s-x[i].fitness;


        //Εύρεση του αθροίσματος των νέων ποιοτήτων
for (i=0;i<N;i++)
    sum+=rev[i];

        //Εύρεση ενός τυχαίου αριθμού απο 0 εώς sum
sumrnd=rnd()*sum;

for (i=0;i<N;i++)
{
    sum2+=rev[i];    //Αποθήκευση του αθροίσματος των νέων ποιοτήτων
    if(sum2>sumrnd) break; //Αν το νέο άθροισμα ειναι μεγαλύτερο απο τον τυχαίο αριθμό επιστρέφει το συγκεκριμένο άθροισμα.
}

if (i==N)
{
    AnsiString st,st2;
    st.printf("sum=%lf sumrnd=%lf sum2=%lf max=%lf min=%lf s=%lf\n",sum,sumrnd,sum2,max_fs,min_fs,s);
    for (int j=0 ; j<N ; j++) {st2.printf("%lf ",x[j].fitness); st=st+st2; }
    ShowMessage(st);
}

//Αποδέσμευση του δείκτη για αποφυγή προβλημάτων στην μνήμη.
```

```
        delete [] rev;

        return i;
}
```

Συνάρτηση: FindBest()

Περιγραφή:

Η συνάρτηση FindBest δέχεται ένα δείκτη ατόμου και έναν ακέραιο αριθμό. Επιστρέφει έναν ακέραιο αριθμό. Τη χρησιμοποιούμε για να βρούμε την καλύτερη λύση κάθε γενιάς. Ο πηγαίος κώδικας αυτής της συνάρτησης παρατίθεται στη συνέχεια.

Πηγαίος Κώδικας:

```
int FindBest(atom *x, int N)
{
        int i,Best=0;
        double BestFit=x[0].fitness;
        for (i=1;i<N;i++)
                if (x[i].fitness<BestFit)
        {
                BestFit=x[i].fitness;
                Best=i;
        }
        return Best;
}
```

Συνάρτηση: UpdateBeliefSpace()

Περιγραφή:

Η συνάρτηση UpdateBeliefSpace δέχεται ένα αντικείμενο τύπου άτομο, έναν πίνακα τύπου άτομο και έναν ακέραιο αριθμό. Δεν επιστρέφει τίποτα. Η συγκεκριμένη συνάρτηση καλείται για την εισαγωγή και ταξινόμηση της καλύτερης λύσης στον χώρο μνήμης. Στην περίπτωση που έχει γεμίσει ο χώρος μνήμης διαγράφει τις κακές λύσεις για να εισάγει τις καλύτερες. Παρακάτω βλέπουμε τον πηγαίο κώδικα της UpdateBeliefSpace.

Πηγαίος Κώδικας:

```
void UpdateBeliefSpace(atom best, atom BeliefSpace[], int B)
{
        bool flag= false;
        int sum=0;
```

```
                    //Ελεγχος αν οι γονότυποι του Best υπάρχουν ήδη σε κάποιο άτομο του χώρου μνήμης.
            for(int j=0; j<flagBS; j++)
            {
                  for(int k=0; k<VARS; k++)
                        if(best.geno[k] == BeliefSpace[j].geno[k])
                              sum++;

                  if(VARS == sum)
                        flag = true;

                  sum=0;
            }

            if(flagBS<B & flag==false)    //Γεμίζει και ταξινομεί τον πίνακα
            {
                  BeliefSpace[flagBS]=best;
                  for(int i = 0; i <flagBS; i++)
                        for(int j =0; j<flagBS-i; j++)
                              if(BeliefSpace[j].fitness>BeliefSpace[j+1].fitness)
                              {
                                    atom temp=BeliefSpace[j];
                                    BeliefSpace[j]=BeliefSpace[j+1];
                                    BeliefSpace[j+1]=temp;
                              }
                        flagBS++;

            }
            else if(flagBS>=B & flag==false)  //Όταν γεμίσει ο πίνακας κάνει αντικατάσταση
            {
                  int i=B-1;
                  while(best.fitness < BeliefSpace[i].fitness && i>=0)
                  {
                              if(i>0)
                                    BeliefSpace[i]=BeliefSpace[i-1];
                              i--;

                  }
                  if(i!=B-1)
                        BeliefSpace[i+1]=best;
            }
      }
```

Συνάρτηση: HammDist()

Περιγραφή:

Η συνάρτηση HammDist δέχεται δύο ακέραιους αριθμούς. Επιστρέφει μία ακέραιη τιμή η οποία είναι η απόσταση (διαφορά) των δύο γονοτύπων. Ο τελεστής &= Bitwise κάνει τη σύγκριση bit pros bit της ακολουθίας που παράχθηκε από την XOR των δύο γονοτύπων. Επιστρέφει true μόνο αν υπάρχει μονάδα στην ίδια θέση και στις δύο ακολουθίες, αλλιώς επιστρέφει false. Ο πηγαίος κώδικας της HammDist παρουσιάζεται στη συνέχεια.

Πηγαίος Κώδικας:

```
int HammDist(int gen1, int gen2)
{
	int dist=0;
	int val=gen1^gen2;
	while(val)
	{
		++dist;
		val &=val-1;
	}
	return dist;
}
```

Συνάρτηση: FindDistance()

Περιγραφή:

Η συνάρτηση FindDistance δέχεται ένα αντικείμενο τύπου άτομο, δύο ακέραιους αριθμούς και τον χώρο μνήμης. Επιστρέφει μια ακέραιη τιμή. Την χρησιμοποιούμε για να υπολογίσουμε την πιο κοντινή λύση που υπάρχει στον χώρο μνήμης σε σύγκριση με το άτομο που δέχεται ως όρισμα. Ο πηγαίος κώδικας της FindDistance παρατίθεται αμέσως μετά.

Πηγαίος Κώδικας:

```
int FindDistance(atom best,int point, atom BeliefSpace[], int B)
{
	int sum =0,min =0,position=-1;
	int endpoint;
	if(point<B)
		endpoint=point;
	else
		endpoint=B;

	for(int i=0; i<endpoint; i++)
```

```
        {
                        //Άθροισμα των διαφορών των γονοτύπων ανάμεσα στην καλύτερη λύση και της αντίστοιχης του χώρου μνήμης
            for(int j=0; j<VARS; j++)
                sum=sum+HammDist(best.geno[j],BeliefSpace[i].geno[j]);
            if(i==0 || sum<min)
            {
                min=sum;
                position=i;
            }
            sum=0;
        }

        return position;
    }
```

Συνάρτηση: Compare()

Περιγραφή:

Η συνάρτηση Compare δέχεται ένα αντικείμενο τύπου άτομο, τρείς ακέραιους αριθμούς και τον χώρο μνήμης. Τη χρησιμοποιούμε για να ελέγξουμε αν η υπάρχουσα καλύτερη λύση είναι καλύτερη από την καλύτερη λύση που θα δημιουργήσουμε μετά τον ανασυνδυασμό αυτής με το κατάλληλο άτομο του χώρου μνήμης και επιστρέφει αυτήν τη λύση. Στη συνέχεια παρουσιάζεται ο πηγαίος κώδικας της Compare.

Πηγαίος Κώδικας:

```
atom Compare(atom best,int position, int BITS, int fun,atom BeliefSpace[])
{
    atom temp;

    temp=Crossover(best,BeliefSpace[position],BITS);
    Fitness(temp, BITS, fun);

    if(best.fitness<temp.fitness)
        return best;
    else
        return temp;
}
```

Συνάρτηση: cleanFlagBS()

Περιγραφή:

Η συνάρτηση cleanFlagBS δεν δέχεται τίποτα σαν όρισμα και δεν επιστρέφει τίποτα. Την χρησιμοποιούμε για να αρχικοποιήσουμε με μηδέν την καθολική μεταβλητή που χρησιμοποιούμε για το χώρο μνήμης. Ο πηγαίος κώδικας της cleanFlagBS φαίνεται στη συνέχεια.

Πηγαίος Κώδικας:

```
void cleanFlagBS()
{
      flagBS=0;
}
```

Συνάρτηση: algorithm()

Περιγραφή:

Η συνάρτηση algorithm δεν δέχεται ορίσματα και δεν επιστρέφει τίποτα. Είναι η βασική συνάρτηση της εφαρμογής, καθώς εκτελεί τον αλγόριθμο. Γίνεται η αρχικοποίηση των τιμών που επέλεξε ο χρήστης σε μεταβλητές, γίνεται η δήλωση των πινάκων τύπου Struct ατόμου για τους γονείς, τους απογόνους και το χώρο μνήμης, καθώς και άλλων μεταβλητών. Στην συνέχεια, γίνεται η αρχικοποίηση του αρχικού πληθυσμού και η αξιολόγησή του χρησιμοποιώντας τη συνάρτηση ποιότητας. Αμέσως μετά, γίνεται η δημιουργία του αρχείου για τα αποτελέσματα του αλγορίθμου.

Στο επόμενο στάδιο κατασκευάζονται οι δύο βασικές επαναλήψεις του αλγόριθμου. Η πρώτη είναι η επανάληψη των γενεών και εσωτερικά αυτής δημιουργείται η επανάληψη των απογόνων, όπου γίνεται η χρήση του Ελιτισμού, έτσι ώστε η καλύτερη λύση να περνάει πρώτη στην επόμενη γενιά. Μέσα στην επανάληψη των απογόνων γίνεται η επιλογή των γονέων με την χρήση της συνάρτησης ρουλέτας, γίνεται ο ανασυνδυασμός των γονέων για την αναπαραγωγή του απογόνου, καθώς και η μετάλλαξη του. Τέλος, γίνεται η αξιολόγηση του.

Στην επανάληψη της γενιάς βρίσκουμε τη θέση του καλύτερου απογόνου, χρησιμοποιώντας τον τελεστή αναρρίχησης εφόσον έχει επιλεγεί από το χρήστη. Στην συνέχεια, χρησιμοποιείται η συνάρτηση για την εισαγωγή της καλύτερης λύσης στο χώρο μνήμης ή γίνεται αντικατάσταση αυτού σε περίπτωση που είναι γεμάτος. Έπειτα εμφανίζονται τα αποτελέσματα στην φόρμα εμφάνισης των αποτελεσμάτων και στο αρχείο εφόσον το έχει επιλέξει ο χρήστης. Τέλος, κάνει αντικατάσταση των γονέων από τους απογόνους και εμφανίζει τον τελικό χώρο μνήμης στην φόρμα και στο αρχείο εφόσον έχει επιλεγεί. Ο πηγαίος κώδικας της algorithm παρουσιάζεται στη συνέχεια.

Πηγαίος Κώδικας:

```
void algorithm()
```

```
{
    //Αρχικοποίηση μεταβλητών
     int N=BasicForm->Population->Text.ToInt();
     int g=BasicForm->Generations->Text.ToInt();
     int B=BasicForm->Beliefspace->Text.ToInt();
     int BITS=BasicForm->Bits->Text.ToInt();
     int fun=BasicForm->Problem->ItemIndex;
     double mutation=BasicForm->Mutation->Text.ToDouble();
     int paralagi = BasicForm->Variation->ItemIndex;

     //Δήλωση μεταβλητών
     atom* BeliefSpace= new atom[B];
     atom *Par= new atom[N];
     atom *Pop= new atom[N];

     int i,j,Best,child,gen,position,p1,p2;
     FILE *f1;

     srand((unsigned)time(NULL));

     //Αρχικοποίηση του αρχικού πληθυσμού των γονέων
     for (i=0;i<N;i++)
          for(j=0;j<VARS;j++)
               Par[i].geno[j]=random(pow(2,BITS));
     //Αξιολόγηση των γονέων
     for (i=0;i<N;i++)
          Fitness(Par[i], BITS, fun);

     //Δημιουργία του αρχείου για τα αποτελέσματα του αλγορίθμου
     if(BasicForm->Savefile->Checked)
     {
          if(BasicForm->EditName->Text!="")
          {
             String name=BasicForm->EditName->Text + ".txt";
             f1=fopen(name.c_str(),"w");
          }
          else
               f1=fopen("results.txt","w");

          fprintf(f1,"Population: %i     Generations: %i Belief space index: %i \n", N, g,
B);
          fprintf(f1,"Number of Bits: %i     Mutation Rate: %f      Function Problem: %s
\n", BITS, mutation, BasicForm->Problem->Text);
          fprintf(f1,"Variation: %s    ", BasicForm->Variation->Text);

          if(BasicForm->Hillclimb->Checked)
```

```
                    fprintf(f1,"Hill Climb: YES\n\n");
                else
                    fprintf(f1,"Hill Climb: NO\n\n");

                fprintf(f1,"Atoms  Binary0  Binary1  Geno0  Geno1  Pheno0  Pheno1
Fitness \n\n");
                fprintf(f1,"Arxikos Plithismos\n");
                Display(Par,N,BITS,f1);
            }

                    //Η κύρια επανάληψη των γενεών
             for (gen=0;gen<g;gen++)
             {
                if(BasicForm->Savefile->Checked)
                    fprintf(f1,"Generation= %i \n",gen);
        // Χρήση του Ελιτισμού έτσι ώστε η καλύτερη λύση να περνάει πρώτη στην επόμενη
γενιά.
                Best=FindBest(Par, N);
                Pop[0]=Par[Best];
        //Η επανάληψη για την δημιουργία των απογόνων
                for (child=1; child<N; child++)
                {

                    //Επιλογή γονέων με την χρήση της Ρουλέτας
                    p1=Roulette(Par, N);
                    p2=Roulette(Par, N);

                    //Ανασυνδυασμός των γονέων
                    Pop[child]= Crossover(Par[p1],Par[p2], BITS);

                    //Χρήση του τελεστή της Μετάλλαξης στον απόγονο
                    Pop[child]= Mutation (Pop[child], mutation, BITS);

                    //Περιπτώσεις 2 και 3 σε περίπτωση επιλογής απο τον χρήστη
                    if(paralagi==1)
                    {
                        int position=FindDistance(Pop[child],gen,BeliefSpace,B);
                        if (position!=-1)
                            Pop[child]=Compare(Pop[child],position,  BITS,  fun,
BeliefSpace);

                    }
                    else if(paralagi==2)
                    {
                        position=FindDistance(Pop[child],gen,BeliefSpace,B);
                        if (position!=-1)
```

```
                    Pop[child]=Crossover(Pop[child],BeliefSpace[position], BITS);
                }

                //Αξιολόγηση του απόγονου
                Fitness(Pop[child], BITS, fun);
            }

        //Βρίσκουμε τη θέση του καλύτερου απόγονου
            Best=FindBest(Pop, N);
        //Χρήση του τελεστή αναρήχησης αν έχει επιλεγεί απο τον χρήστη
            if(BasicForm->Hillclimb->Checked)
                HillClimb(Pop[Best], BITS, fun);

        // Εύρεση του στοιχείου του χώρου μνήμης που ειναι κοντά με την καλύτερη λύση της γενιάς.
            position=FindDistance(Pop[Best],gen,BeliefSpace,B);
            if(position!=-1)
                Pop[Best]=Compare(Pop[Best],position, BITS, fun, BeliefSpace);

        //Γέμισμα ή ενημέρωση του χώρου μνήμης
            UpdateBeliefSpace(Pop[Best],BeliefSpace,B);

        //Εμφάνιση των αποτελεσμάτων στη δεύτερη φόρμα και στο αρχείο
            if(BasicForm->Savefile->Checked)
            {
                Display (Pop, N, BITS, f1);
                fprintf(f1,"Best After Culture Algorithm \n");
            }

            ViewForm->Table->Cells[0][gen+1]=gen;
            for (j=0;j<VARS;j++) {
                if(BasicForm->Savefile->Checked)
                    fprintf(f1,"    %s  ",Int2Str(Pop[Best].geno[j],BITS));
                ViewForm->Table->Cells[j+1][gen+1]=Int2Str(Pop[Best].geno[j], BITS);
            }
            for (j=0;j<VARS;j++)
            {
                if(BasicForm->Savefile->Checked)
                    fprintf(f1,"%5i ",Pop[Best].geno[j]);
                ViewForm->Table->Cells[3+j][gen+1]=Pop[Best].geno[j];
            }

            for (j=0; j<VARS; j++)
            {
                if(BasicForm->Savefile->Checked)
                    fprintf(f1,"%10.7lf  ",Pop[Best].pheno[j]);
```

```
                    std::stringstream pheno;
                    pheno<<std::fixed<<std::setprecision(10)<< Pop[Best].pheno[j];
                    ViewForm->Table->Cells[5+j][gen+1]=pheno.str().c_str();
                }

                if(BasicForm->Savefile->Checked)
                    fprintf(f1,"f= %10.7lf \n\n",Pop[Best].fitness);


                std::stringstream fitness;
                fitness<<std::fixed<<std::setprecision(10)<< Pop[Best].fitness;
                ViewForm->Table->Cells[7][gen+1]=fitness.str().c_str();
                //ViewForm->Table->Cells[7][gen+1]=  Pop[Best].fitness;

                //Αντικατάσταση των γονέων απο τους απογόνους
                for (i=0;i<N;i++)
                  Par[i]=Pop[i];

            }

            //Εμφάνιση του τελικού χώρου μνήμης στη δεύτερη φόρμα
            for(i=0; i<flagBS; i++)
                if(BasicForm->Problem->ItemIndex==0)
                {
                    std::stringstream bf1;
                    bf1<<std::fixed<<std::setprecision(10)<<
Double(BeliefSpace[i].fitness)*(1);
                    ViewForm->LabelBeliefS->Caption=ViewForm->LabelBeliefS->Caption
+ "\n" + bf1.str().c_str();
                }
                else
                {
                    std::stringstream bf;
                    bf<<std::fixed<<std::setprecision(10)<< Double(BeliefSpace[i].fitness);
                    ViewForm->LabelBeliefS->Caption=ViewForm->LabelBeliefS->Caption
+ "\n" + bf.str().c_str();
                }
                    //Εμφάνιση του τελικού χώρου μνήμης στο αρχείο
            if(BasicForm->Savefile->Checked)
            {
                fprintf(f1,"Belief Space \n");
                for(i=0; i<flagBS; i++)
                     if(BasicForm->Problem->ItemIndex==0)
                         fprintf(f1,"%10.7lf \n", BeliefSpace[i].fitness*(1));
                     else
                         fprintf(f1,"%10.7lf \n", BeliefSpace[i].fitness);
```

```
            fclose(f1);
        }

        //Αποδέσμευση της μνήμης απο τα αντικείμενα
        delete Par;
        delete Pop;
        delete BeliefSpace;
    }
```

2.3.4.2 Functions.h

Περιγραφή:

Αρχικά δηλώνουμε τις βιβλιοθήκες τις οποίες χρειαζόμαστε για τη δημιουργία της εφαρμογής. Στη συνέχεια γίνεται η δήλωση δύο καθολικών μεταβλήτων και η δήλωση της δομής δεδομένων του ατόμου με χαρακτηριστικά το γονότυπο, το φαινότυπο και την ποιότητα του. Έπειτα, γίνεται η δήλωση των συναρτήσεων μελών που θα χρησιμοποιηθούν στο αρχείο. Το σώμα τους βρίσκεται στο .cpp αρχείο. Ο πηγαίος κώδικας του αρχείου Functions.h παρατίθεται αμέσως μετά.

Πηγαίος Κώδικας Αρχείου Κεφαλίδας:

```
#include <vcl.h>
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <math.h>
#define VARS 2
#define OFFSET 4000 // gai tn Rastrigin
struct atom{
    int geno[VARS];
    double pheno[VARS];
    double fitness;
};

double rnd(void);
atom Mutation (atom x, double prob, int BITS);
atom Crossover (atom p1, atom p2, int BITS);
String Int2Str(int x, int BITS);
int Str2Int(String x, int BITS);
void Geno2Pheno(atom & x, int BITS, int fun);
void Fitness(atom & x, int BITS, int fun);
void HillClimb(atom & x, int BITS, int fun);
void Display(atom * x,int N, int BITS, FILE *f1);
int Roulette(atom * x, int N);
```

```
int FindBest(atom *x, int N);
void UpdateBeliefSpace(atom best, atom BeliefSpace[], int B);
int HammDist(int gen1, int gen2);
int FindDistance(atom best,int point, atom BeliefSpace[], int B);
atom Compare(atom best,int position, int BITS, int fun, atom BeliefSpace[]);
void cleanFlagBS();
void algorithm();
```

2.4 Προβλήματα επίλυσης

Στην παρούσα πτυχιακή εργασία αντιμετωπίστηκαν τέσσερα προβλήματα. Αυτά είναι:

- Η συνάρτηση Rosenbrock.
- Η συνάρτηση Rastrigin.
- Η συνάρτηση Schwefel.
- Ένα ρεαλιστικό πρόβλημα το οποίο είναι ένα Παραλληλόγραμμο σταθερού όγκου και ελάχιστης επιφάνειας.

2.4.1 Περιγραφή της συνάρτησης Rosenbrock

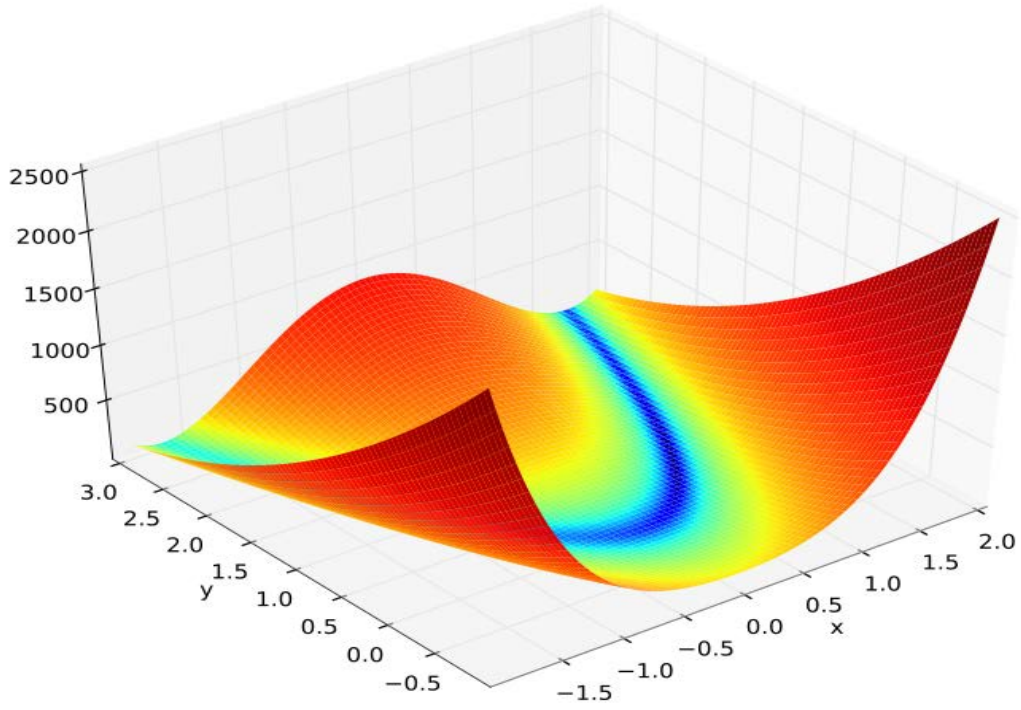
Η συνάρτηση Rosenbrock διατυπώθηκε το 1960 από τον Howard H. Rosenbrock. Χρησιμοποιήθηκε ως πρόβλημα για την δοκιμή της απόδοσης των αλγορίθμων βελτιστοποίησης. Συχνά αναφέρεται και ως συνάρτηση μπανάνα, διότι το γράφημα που δημιουργεί παραπέμπει σε σχήμα μπανάνας.

Η γενικευμένη μορφή της συνάρτησης Rosenbrock είναι:

$$f(\mathbf{x}) = \sum_{i=1}^{N-1} 100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2 \quad \text{where} \quad \mathbf{x} = [x_1, \ldots, x_N] \in \mathbb{R}^N.$$

Όπου x_i παίρνει τιμές στο διάστημα $-2048 \leq x_i \leq 2048$.

Η γραφική παράσταση της συνάρτησης Rosenbrock φαίνεται στην Εικόνα 8.



Εικόνα 8: Γραφική παράσταση της συνάρτησης Rosenbrock

2.4.2 Περιγραφή της συνάρτησης Rastrigin

Η συνάρτηση Rastrigin είναι μια μη κυρτή συνάρτηση. Χρησιμοποιήθηκε ως πρόβλημα για τη δοκιμή της απόδοσης των αλγορίθμων βελτιστοποίησης. Για πρώτη φορά προτάθηκε από τον Rastrigin ως μια δισδιάστατη συνάρτηση και γενικεύτηκε από τους Mühlenbein et al. Η εύρεση του βέλτιστου αυτής της συνάρτησης είναι ένα αρκετά δύσκολο πρόβλημα λόγω του μεγάλου χώρου αναζήτησης και του μεγάλου αριθμού τοπικών ελάχιστων.

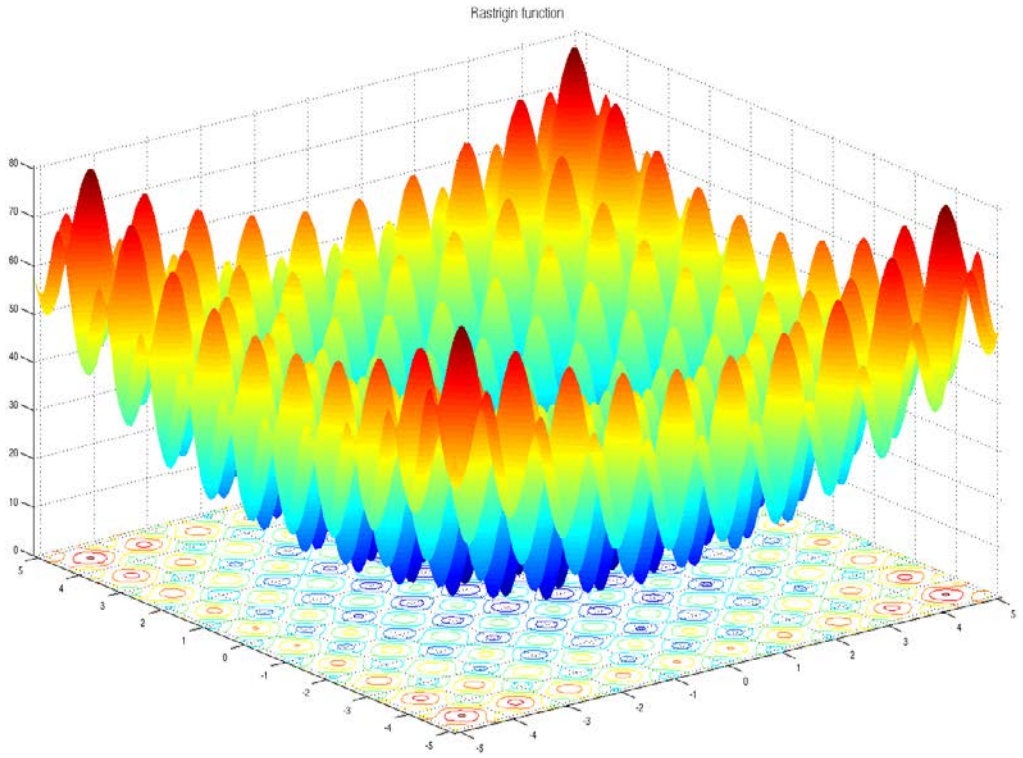
Η γενικευμένη μορφή της συνάρτησης Rastrigin είναι:

$$f(\mathbf{x}) = An + \sum_{i=1}^{n} \left[x_i^2 - A\cos(2\pi x_i) \right]$$

Όπου x_i παίρνει τιμές στο διάστημα $-5.12 \leq x_i \leq 5.12$.

Όπου A ισούται με 10 και n ισούται με τον αριθμό των μεταβλητών. Στην περίπτωσή μας n = 2.

Η γραφική παράσταση της συνάρτησης Rastrigin παρουσιάζεται στην Εικόνα 9.



Εικόνα 9: Γραφική Παράσταση της Συνάρτησης Rastrigin

2.4.3 Περιγραφή της συνάρτησης Schwefel

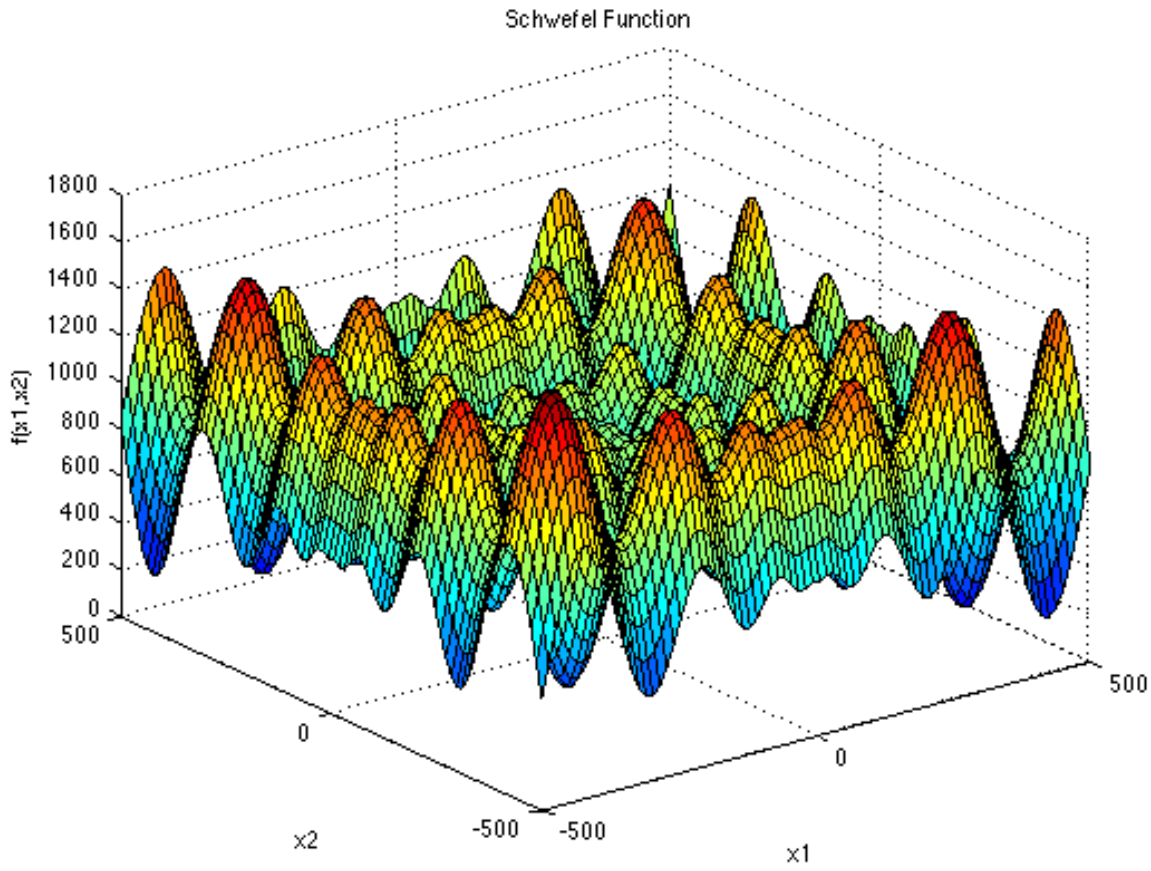
Η συνάρτηση Schwefel είναι περίπλοκη, διότι έχει πολλά τοπικά ελάχιστα. Χρησιμοποιήθηκε ως πρόβλημα για τη δοκιμή της απόδοσης των αλγορίθμων βελτιστοποίησης. Όπως θα δούμε και στη γραφική παράσταση παρουσιάζεται η δισδιάστατη μορφή της συνάρτησης.

Η γενικευμένη μορφή της συνάρτησης Schwefel είναι:

$$f_7(x) = \sum_{i=1}^{n} -x_i \cdot \sin\left(\sqrt{|x_i|}\right) \qquad -500 \le x_i \le 500$$

Όπου x_i παίρνει τιμές στο διάστημα $-418.9829 \le x_i \le 418.9829$.

Η γραφική παράσταση της συνάρτησης Schwefel φαίνεται στην Εικόνα 10.



Εικόνα 10: Γραφική Παράσταση της Συνάρτησης Schwefel

2.4.4 Περιγραφή του ρεαλιστικού προβλήματος.

Το τελευταίο πρόβλημα είναι ένα παραλληλόγραμμο με διαστάσεις ορθογωνίου το οποίο έχει σταθερό όγκο και ελάχιστη επιφάνεια. Το χρησιμοποιήσαμε ως πρόβλημα για τη δοκιμή της απόδοσης του αλγόριθμου. Συγκεκριμένα, μας δίνεται ένα ορθογώνιο παραλληλόγραμμο σταθερού όγκου 1000 ml.

Αρχικά υπολογίζουμε τον όγκο του παραλληλόγραμμου ο οποίος δίνεται από το πηλίκο όγκος = x1*x2*z. Το z υπολογίζεται με βάση τον σταθερό όγκο ως z= όγκος /(x1*x2). Στη συνέχεια υπολογίζουμε το εμβαδόν της παράπλευρης επιφάνειας το οποίο είναι το άθροισμα των όγκων των 6 επιφανειών του παραλληλογράμμου. Συνεπώς, η γενικευμένη μορφή αυτού του προβλήματος θα είναι το αποτέλεσμα της παράπλευρης επιφάνειας του παραλληλογράμμου.:

f = (2*(x1*z))+(2*(x2*z))+(2*(x1*x2))

Όπου z θα είναι z = όγκος /(x1*x2) και ο όγκος θα ισούται με 1000.

2.5 Περιγραφή των πειραμάτων

Σε αυτήν την ενότητα θα περιγράψουμε τα πειράματα που αποφασίσαμε να διεξαχθούν για τον έλεγχο της απόδοσης του αλγόριθμου και της επίδρασης των παραμέτρων σε αυτόν. Συνεπώς όπως αναφέραμε πιο πάνω θα δοκιμάσουμε τις επιδόσεις του αλγορίθμου με την βοήθεια τεσσάρων προβλημάτων (benchmark problems). Κάθε ένα από τα παραπάνω προβλήματα είναι μοναδικό και έχει διαφορετικό πεδίο ορισμού. Συνεπώς οι παράμετροι του αλγορίθμου που θα αποδώσουν καλύτερα αποτελέσματα σε κάθε πρόβλημα θα είναι διαφορετικοί. Οι παράμετροι που θα ληφθούν υπόψη στα πειράματα που θα πραγματοποιήσουμε είναι οι εξής:

- Ο αριθμός των γενεών που θα τρέξει ο αλγόριθμος (Generations)
- Ο αριθμός του πληθυσμού του αλγορίθμου (Population)
- Το μέγεθος του χώρου μνήμης (Belief Space)
- Ο αριθμός των bits της συμβολοσειράς που θα απαρτίζει τον γενότυπο των ατόμων (Bits)
- Η πιθανότητα της μετάλλαξης των απογόνων (Mutation Rate)
- Ο τρόπος που ο χώρος μνήμης αλληλοεπιδρά με τον αλγόριθμο (Variation Solution)
- Αν θα χρησιμοποιήσουμε τον τελεστή αναρρίχησης ή όχι. (Hill Climb)

Συνεπώς για την πρώτη παράμετρο (Generations) οι τιμές με τις οποίες θα πειραματιστούμε θα είναι 200, 500, 1000, 5000 γενιές. Για την δεύτερη παράμετρο που είναι ο πληθυσμός (Population) οι τιμές με τις οποίες θα πειραματιστούμε θα είναι 10, 20, 50, 100 άτομα. Για την τρίτη παράμετρο δηλαδή τον χώρο μνήμης (Belief Space) οι τιμές θα είναι 10%, 20% και 50% του πληθυσμού. Για την τέταρτη παράμετρο οι τιμές της συμβολοσειράς (Bits) θα είναι 10, 12, 14, 16 . Για την πέμπτή παράμετρο οι τιμές τις πιθανότητας της μετάλλαξης (Mutation Rate) θα είναι 0.005 ,0.01, 0.02, 0.05 . Για την έκτη παράμετρο θα πειραματιστούμε με δύο τρόπους όσον αναφορά την αλληλεπίδραση του αλγορίθμου με τον χώρο μνήμης (Variation Solution) , αυτοί είναι αρχικά να βελτιώνεται μόνο η καλύτερη λύση της κάθε γενιάς με την βοήθεια του χώρου μνήμης και δεύτερόν όλες οι λύσεις της κάθε γενιάς με την βοήθεια του χώρου μνήμης. Για την τελευταία παράμετρο που είναι ο τελεστής αναρρίχησης (Hill Climb) θα πειραματιστούμε με βάση την χρήση αυτού του τελεστή ή όχι.

Συνεπώς όπως αντιλαμβανόμαστε είναι αρκετά δύσκολο και ασαφές να πειραματιστούμε ταυτόχρονα με όλους αυτούς τους παραμέτρους. Έτσι θα ξεκινήσουμε τα πειράματά μας αρχικά με μια παράμετρο από τις παραπάνω κρατώντας την καλύτερη απόδοση του αλγορίθμου, στην συνέχεια θα προχωρήσουμε στην επόμενη παράμετρο μέχρι να δοκιμάσουμε τον αλγόριθμο για όλες τις παραμέτρους. Αξίζει να σημειωθεί ότι η καλύτερή επίδοση του αλγορίθμου υπολογίζεται από την καλύτερη ποιότητα που πέτυχε σε συνδυασμό με την γενιά στην οποία βρέθηκε και τον χρόνο που χρειάστηκε για να την επιτύχει.

Όμως επειδή ο αλγόριθμος είναι στοχαστικός θα πρέπει για κάθε πείραμα που αναφέραμε παραπάνω να τον τρέξουμε ας πούμε 10 φορές και να λάβουμε υπόψιν μας την μέση τιμή της καλύτερης λύσης, της γενιάς στην οποία βρέθηκε και του χρόνου.

Συγκεκριμένα ξεκινήσαμε τα πειράματα αλλάζοντας μια προς μια της παραμέτρους. Αρχικά ξεκινήσαμε με τον πληθυσμό των ατόμων του αλγορίθμου, κρατήσαμε την καλύτερη απόδοση και την χρησιμοποιήσαμε ως σταθερά για την επόμενη παράμετρο. Στις τρείς

τελευταίες στήλες βλέπουμε την απόδοση του αλγόριθμου καθώς βλέπουμε την καλύτερη ποιότητα που επιτεύχθηκε σε πόσες γενιές συνέβη καθώς και πόσο χρόνο χρειάστηκε ο αλγόριθμος για να φτάσει σε αυτό το αποτέλεσμα. Οι τιμές των τελευταίων στηλών ληφθήκαν υπόψιν μετά από 10 επαναλήψεις, δηλαδή είναι οι μέσος όρος των αποτελεσμάτων για κάθε παράμετρο, αυτός είναι ο λόγος που μπορεί να δούμε για παράδειγμα ότι η καλύτερη ποιότητα επιτεύχθηκε σε 7,2 γενιές. Οι γενιές είναι ακέραιος αριθμός και όχι πραγματικός.

Κεφάλαιο 3 Αποτελέσματα και συμπεράσματα

3.1 Αποτελέσματα πειραμάτων

3.1.1 Αποτελέσματα πειραμάτων για τη συνάρτηση Rosenbrock

Στον Πίνακα 4 παρουσιάζονται τα αποτελέσματα του αλγόριθμου κουλτούρας για τη συνάρτηση Rosenbrock.

Find the best combination for the parameters in the optimization of Rosenbrock function											Best values
Problem	Population	Generations	BITS	Belief Space	Mutation	HillClimb	Solution	Best Fitness	Gen of Best Fit	Time	
Rosenbrock	10	10	10	5	0.01	yes	best only	0.43568005	6.8	0.0188	
Rosenbrock	20	10	10	5	0.01	yes	best only	0.14258756	5.2	0.022	
Rosenbrock	50	10	10	5	0.01	yes	best only	0.03547668	3.7	0.0297	
Rosenbrock	100	10	10	5	0.01	yes	best only	0.04571495	4.5	0.0533	pop
Rosenbrock	100	10	12	5	0.01	yes	best only	0.0101554	3	0.047	
Rosenbrock	100	10	14	5	0.01	yes	best only	0.0075811	10	0.047	
Rosenbrock	100	10	16	5	0.01	yes	best only	0.0042115	10	0.063	bits
Rosenbrock	100	10	16	10	0.01	yes	best only	0.0061662	10	0.047	
Rosenbrock	100	10	16	20	0.01	yes	best only	0.0000071	8	0.047	bs
Rosenbrock	100	10	16	50	0.01	yes	best only	0.0032783	10	0.047	
Rosenbrock	100	10	16	20	0.01	yes	best only	0.0184923	10	0.063	
Rosenbrock	100	10	16	20	0.02	yes	best only	0.0189386	10	0.047	
Rosenbrock	100	10	16	20	0.05	yes	best only	0.0001451	10	0.047	mutation
Rosenbrock	100	10	16	20	0.005	yes	best only	0.0134146	10	0.046	
Rosenbrock	100	10	16	20	0.05	yes	best only	0.0001451	10	0.047	hillclimb
Rosenbrock	100	10	16	20	0.05	no	best only	0.0009072	9	0.047	
Rosenbrock	100	10	16	20	0.05	yes	best only	0.0381617	10	0.047	
Rosenbrock	100	10	16	20	0.05	yes	all	0.004104	10	0.047	solution
Rosenbrock	100	200	16	20	0.05	yes	all	0.0002515	59	0.719	
Rosenbrock	100	500	16	20	0.05	yes	all	0.0000847	314	1.719	
Rosenbrock	100	1000	16	20	0.05	yes	all	0.0000003	33	3.406	
Rosenbrock	100	5000	16	20	0.05	yes	all	0	4901	16.563	generations

Πίνακας 4: Αποτελέσματα του αλγόριθμου κουλτούρας για τη συνάρτηση Rosenbrock

3.1.2 Αποτελέσματα πειραμάτων για τη συνάρτηση Rastrigin

Στον Πίνακα 5 παρουσιάζονται τα αποτελέσματα του αλγόριθμου κουλτούρας για τη συνάρτηση Rastrigin.

Find the best combination for the parameters in the optimization of Rastrigin function											Best values
Problem	Population	Generations	BITS	Belief Space	Mutation	HillClimb	Solution	Best Fitness	Gen of Best Fit	Time	
Rastrigin	10	10	10	5	0.01	yes	best only	3.98499554	4.6	0.0188	
Rastrigin	20	10	10	5	0.01	yes	best only	2.79198174	4	0.0282	
Rastrigin	50	10	10	5	0.01	yes	best only	0.80209324	4.6	0.0346	
Rastrigin	100	10	10	5	0.01	yes	best only	0.0099382	2.75	0.0535	pop
Rastrigin	100	10	12	5	0.01	yes	best only	0.04172902	7.4	0.0566	bits
Rastrigin	100	10	14	5	0.01	yes	best only	0.73114084	8.8	0.05	
Rastrigin	100	10	16	5	0.01	yes	best only	0.36169594	10	0.0564	
Rastrigin	100	10	12	10	0.01	yes	best only	0.49803725	5.25	0.047	
Rastrigin	100	10	12	20	0.01	yes	best only	0.49803725	4.75	0.04275	bs
Rastrigin	100	10	12	50	0.01	yes	best only	0.60148958	8.4	0.0468	
Rastrigin	100	10	12	20	0.01	yes	best only	0.49803725	4.75	4.75	
Rastrigin	100	10	12	20	0.02	yes	best only	0.19958708	5.4	0.0468	mutation
Rastrigin	100	10	12	20	0.05	yes	best only	1.01478018	5.8	0.0406	
Rastrigin	100	10	12	20	0.005	yes	best only	0.40500246	8.6	0.047	
Rastrigin	100	10	12	20	0.02	yes	best only	0.19958708	5.4	0.0468	hillclimb
Rastrigin	100	10	12	20	0.02	no	best only	0.63298032	7.8	0.0438	
Rastrigin	100	10	12	20	0.02	yes	best only	0.19958708	5.4	0.0468	solution
Rastrigin	100	10	12	20	0.02	yes	all	0.19958708	7.6	0.047	
Rastrigin	100	200	12	20	0.02	yes	best only	0.0006203	28.6	0.5748	
Rastrigin	100	500	12	20	0.02	yes	best only	0.0006203	10	1.3338	generations
Rastrigin	100	1000	12	20	0.02	yes	best only	0.0006203	12.6	2.7624	
Rastrigin	100	5000	12	20	0.02	yes	best only	0.0006203	17.8	14.3842	

Πίνακας 5: Αποτελέσματα του αλγόριθμου κουλτούρας για τη συνάρτηση Rastrigin

3.1.3 Αποτελέσματα πειραμάτων για τη συνάρτηση Schwefel

Στον Πίνακα 6 παρουσιάζονται τα αποτελέσματα του αλγόριθμου κουλτούρας για τη συνάρτηση Schwefel.

Find the best combination for the parameters in the optimization of Schwefel function											Best values
Problem	Population	Generations	BITS	Belief Space	Mutation	HillClimb	Solution	Best Fitness	Gen of Best Fit	Time	
Schwefel	10	10	10	5	0.01	yes	best only	79.6907199	6.2	0.0188	
Schwefel	20	10	10	5	0.01	yes	best only	90.8071573	5.2	0.0188	
Schwefel	50	10	10	5	0.01	yes	best only	43.43290948	7	0.025	
Schwefel	100	10	10	5	0.01	yes	best only	0.0055255	3.8	0.047	pop and bits
Schwefel	100	10	12	5	0.01	yes	best only	0.00586364	8.4	0.0138	
Schwefel	100	10	14	5	0.01	yes	best only	1.11419572	10	0.0436	
Schwefel	100	10	16	5	0.01	yes	best only	2.83661752	9.8	0.047	
Schwefel	100	10	10	10	0.01	yes	best only	0.0055255	5	0.0404	
Schwefel	100	10	10	20	0.01	yes	best only	0.0055255	5.4	0.0314	
Schwefel	100	10	10	50	0.01	yes	best only	0.0055255	4.8	0.0438	bs
Schwefel	100	10	10	50	0.01	yes	best only	0.0055255	4.8	0.0438	mutation
Schwefel	100	10	10	50	0.02	yes	best only	0.0055255	5	0.408	
Schwefel	100	10	10	50	0.05	yes	best only	0.0055255	5	0.041	
Schwefel	100	10	10	50	0.005	yes	best only	0.0055255	5	0.0408	
Schwefel	100	10	10	50	0.01	yes	best only	0.0055255	4.8	0.0438	hillclimb
Schwefel	100	10	10	50	0.01	no	best only	1.76403232	8	0.0438	
Schwefel	100	10	10	50	0.01	yes	best only	0.0055255	4.8	0.0438	
Schwefel	100	10	10	50	0.01	yes	all	0.0055255	4.8	0.0434	solution
Schwefel	100	200	10	50	0.01	yes	all	0.0055255	5.4	0.5938	
Schwefel	100	500	10	50	0.01	yes	all	0.0055255	5	1.5188	generations
Schwefel	100	1000	10	50	0.01	yes	all	0.0055255	5.4	2.7654	
Schwefel	100	5000	10	50	0.01	yes	all	0.0055255	6.8	14.2624	

Πίνακας 6: Αποτελέσματα του αλγόριθμου κουλτούρας για τη συνάρτηση Schwefel

3.1.4 Αποτελέσματα πειραμάτων για τη συνάρτηση του Παραλληλόγραμμου

Στον Πίνακα 7 παρουσιάζονται τα αποτελέσματα του αλγόριθμου κουλτούρας για τη συνάρτηση του Παραλληλόγραμμου.

Find the best combination for the parameters in the optimization of Parallilogram function											Best values
Problem	Population	Generations	BITS	Belief Space	Mutation	HillClimb	Solution	Best Fitness	Gen of Best Fit	Time	
Parallilogram	10	10	10	5	0.01	yes	best only	601.032663	9.6	0.0188	
Parallilogram	20	10	10	5	0.01	yes	best only	600.004407	8.8	0.0158	pop and bits
Parallilogram	50	10	10	5	0.01	yes	best only	600.0361382	9	0.031	
Parallilogram	100	10	10	5	0.01	yes	best only	600.0050381	7.8	0.0408	
Parallilogram	20	10	12	5	0.01	yes	best only	603.142763	9	0.0218	
Parallilogram	20	10	14	5	0.01	yes	best only	624.8034755	10	0.0216	
Parallilogram	20	10	16	5	0.01	yes	best only	614.8269384	10	0.0188	
Parallilogram	20	10	10	10	0.01	yes	best only	600.0050381	7.4	0.0222	bs
Parallilogram	20	10	10	20	0.01	yes	best only	600.0063004	9	0.0216	
Parallilogram	20	10	10	50	0.01	yes	best only	600.0199572	7	0.0252	
Parallilogram	20	10	10	10	0.01	yes	best only	600.0050381	7.4	0.0222	
Parallilogram	20	10	10	10	0.02	yes	best only	600.0095546	7.4	0.0218	
Parallilogram	20	10	10	10	0.05	yes	best only	600.004407	7	0.0188	mutation
Parallilogram	20	10	10	10	0.005	yes	best only	600.1413126	8.6	0.0282	
Parallilogram	20	10	10	10	0.05	yes	best only	600.004407	7	0.0188	hillclimb
Parallilogram	20	10	10	10	0.05	no	best only	607.7976322	9	0.019	
Parallilogram	20	10	10	10	0.05	yes	best only	600.004407	7	0.0188	solution
Parallilogram	20	10	10	10	0.05	yes	all	600.0595856	8.6	0.0218	
Parallilogram	20	200	10	10	0.05	yes	best only	600.004407	11.4	0.1158	
Parallilogram	20	500	10	10	0.05	yes	best only	600.004407	7.4	0.2564	generations
Parallilogram	20	1000	10	10	0.05	yes	best only	600.004407	7.4	0.4998	
Parallilogram	20	5000	10	10	0.05	yes	best only	600.004407	7.8	2.418	

Πίνακας 7: Αποτελέσματα του αλγόριθμου κουλτούρας για τη συνάρτηση του Παραλληλόγραμμου

3.2 Εξαγωγή συμπερασμάτων για την απόδοση του αλγόριθμου και την επίδραση των παραμέτρων

3.2.1 Σχολιασμός των πειραμάτων για την συνάρτηση Rosenbrock

Παραπάνω βλέπουμε τα αποτελέσματα των πειραμάτων για τη συνάρτηση Rosenbrock. Θα αναλύσουμε εκτενέστερα την χρήση των παραμέτρων και τα αποτελέσματα του αλγορίθμου για κάθε τιμή που θέσαμε σε αυτούς. Συνεπώς, σχετικά με τις τιμές του πληθυσμού όπως βλέπουμε με το πράσινο χρώμα ο αλγόριθμος πέτυχε καλύτερη απόδοση χρησιμοποιώντας την τιμή 100 για την παράμετρο του πληθυσμού. Στην συνέχεια κρατάμε αυτήν τιμή για τον πληθυσμό και πάμε να πειραματιστούμε με τις τιμές της παραμέτρου bits, όπου βλέπουμε ότι ο αλγόριθμος αποδίδει καλύτερα όταν χρησιμοποιούμε 16 bits συμβολοσειρά. Έχοντας τις τιμές αυτές ως σταθερές πάμε να πειραματιστούμε με τις τιμές του χώρου μνήμης όπου βλέπουμε ότι η απόδοση είναι καλύτερη με την αποθήκευση των 20 καλύτερων λύσεων στον χώρο μνήμης και όχι με λιγότερες ή περισσότερες. Συνεχίζουμε με τα πειράματα σχετικά με την πιθανότητα της μετάλλαξης όπου βλέπουμε ότι χρησιμοποιώντας πιθανότητα 0,05 ο αλγόριθμος πετυχαίνει καλύτερη απόδοση, συγκεκριμένα βλέπουμε ότι βρίσκει πολύ καλή λύση (βέλτιστη λύση) στον ίδιο χρόνο και στον αριθμό γενιάς σε σύγκριση με τις άλλες τιμές που θέσαμε στην συγκεκριμένη παράμετρο. Αμέσως μετά πειραματιζόμαστε με την χρήση του τελεστή αναρρίχησης που παρατηρούμε ότι βρίσκουμε πολύ καλύτερη λύση με την χρήση του όμως σε

περισσότερες γενιές. Στην συνέχεια πειραματιζόμαστε με την παράμετρο του τρόπου με τον οποίο ο χώρος μνήμης αλληλοεπιδρά με τον αλγόριθμο, βλέπουμε ότι ο αλγόριθμος αποδίδει πολύ καλύτερα με την βελτίωση όλων των ατόμων της γενιάς και όχι μόνο με την καλύτερη λύση κάθε γενιάς. Τέλος πειραματιζόμαστε με τον αριθμό των γενεών όπου βλέπουμε ότι με την χρήση της τιμής 5000 γενεών ο αλγόριθμος βρίσκει την βέλτιστη λύση στο 0, οπού συνειδητοποιούμε ότι η αλγόριθμος με την χρήση της συνάρτησης Rosenbrock μπορεί να βρει την βέλτιστη τιμή.

3.2.2 Σχολιασμός των πειραμάτων για την συνάρτηση Rastrigin

Για την συνάρτηση Rastrigin εργαστήκαμε ακριβώς όπως και για τη συνάρτηση Rosenbrock. Συνεπώς τα πειράματα είναι ακριβώς τα ίδια, αυτό που αλλάζει όμως είναι ότι η συνάρτηση αρχικά είναι διαφορετική και οι τιμές των παραμέτρων που παράγουν την καλύτερη απόδοση του αλγορίθμου έναντι της συνάρτησης είναι διαφορετικοί. Άρα παρατηρούμε ότι ο αλγόριθμος αποδίδει καλύτερα με 100 άτομα πληθυσμό και σε αυτήν την συνάρτηση. Τα bits που αποδίδουν καλύτερα αυτήν την φορά είναι 12, όμως το μέγεθος του χώρου μνήμης που θα έχει καλύτερη αποδοτικότητα για τον αλγόριθμο θα είναι 20 θέσεων. Η τιμή της παραμέτρου της μετάλλαξης θα είναι 0,02 καθώς όπως προκύπτει από τα πειράματα έχει την καλύτερη απόδοση. Όταν χρησιμοποιείται ο τελεστής αναρρίχησης βλέπουμε αισθητή διαφορά στην αναζήτηση της καλύτερης λύσης και σε αυτήν την συνάρτηση ο αλγόριθμος βρίσκει καλύτερη απόδοση βελτιώνοντας μόνο την καλύτερη λύση της γενιάς και όχι όλες. Τέλος βλέπουμε ότι ο αλγόριθμος αποδίδει καλύτερα με 500 γενιές και όχι με παραπάνω διότι δεν μπορεί να βρει καλύτερη λύση όσον αναφορά τις γενιές και σε λιγότερο χρόνο εκτέλεσης.

3.2.3 Σχολιασμός των πειραμάτων για την συνάρτηση Schwefel

Όσον αναφορά την συνάρτηση Schwefel, ο τρόπος που εργαστήκαμε για τα πειράματά μας είναι σχεδόν ο ίδιος με μόνη εξαίρεση ότι αλγόριθμος έχει καλύτερη απόδοση σε διαφορετικές τιμές που είναι απολύτως λογικό διότι είναι τελείως διαφορετική συνάρτηση. Έτσι παρατηρούμε ότι και σε αυτήν την συνάρτηση ο αλγόριθμος έχει καλύτερη απόδοση με την τιμή 100 για την παράμετρο του πληθυσμού. Ακόμη παρατηρούμε ότι ο αριθμός των bits που κάνουν τον αλγόριθμο αποδοτικότερο είναι τα 10 και όχι περισσότερα όπως είδαμε στις παραπάνω συναρτήσεις. Σχετικά με το μέγεθος του χώρου μνήμης ο αλγόριθμος αποδίδει καλύτερα με την τιμή 50 δηλαδή με την αποθήκευση των 50 καλύτερων λύσεων και αυτό το βλέπουμε στο πόσο γρήγορα τείνει στην καλύτερη λύση. Η παράμετρος της μετάλλαξης που αποδίδει καλύτερη απόδοση στον αλγόριθμο είναι η τιμή 0,01 και όπως βλέπουμε έχει πολύ μικρές διαφορές με τις υπόλοιπες τιμές, αυτές τις διαφορές τις βλέπουμε στον χρόνο και στο πόσο γρήγορα τείνει ο αλγόριθμος στην καλύτερη λύση. Όσον αναφορά τον τελεστή αναρρίχησης παρατηρούμε ότι με την χρήση του βλέπουμε αισθητή διαφορά στην αναζήτηση της καλύτερης λύσης. Ο τρόπος αλληλεπίδρασης του χώρου μνήμης με τον αλγόριθμο είναι αποδοτικότερος εάν όλες οι λύσεις της γενιάς βελτιωθούν, αν και η διαφορά είναι πολύ μικρή και έχει να κάνει με τον χρόνο εκτέλεσης του αλγορίθμου. Τέλος παρατηρούμε ότι ο αλγόριθμος είναι περισσότερο αποδοτικός με 500 γενιές και όχι περισσότερες όσον αναφορά τον χρόνο εκτέλεσης και την ταχύτητα του να βρει την καλύτερη λύση.

3.2.4 Σχολιασμός των πειραμάτων για το Πρόβλημα παραλληλογράμμου.

Για τα πειράματα του παραλληλογράμμου εργαστήκαμε ακριβώς όπως και στα υπόλοιπα πειράματα των παραπάνω συναρτήσεων. Συγκεκριμένα, παρατηρούμε ότι ο αλγόριθμος αποδίδει καλύτερα στην περίπτωση που ο πληθυσμός του αλγορίθμου είναι 20 άτομα. Ακόμη βλέπουμε ότι η καλύτερη λύση επιτυγχάνεται με την χρήση των 10 bits και όχι περισσοτέρων διότι ο αλγόριθμος δεν προσεγγίζει την καλύτερη λύση. Όσον αναφορά το μέγεθος του χώρου μνήμης η καλύτερη τιμή είναι 10 διότι αν αποθηκεύσουμε περισσότερες λύσεις στον χώρο μνήμης αυξάνεται ο χρόνος εκτέλεσης του αλγορίθμου καθώς και ότι ο αλγόριθμος δεν προσεγγίζει την καλύτερη λύση. Σχετικά με τις τιμές της μετάλλαξης παρατηρούμε ότι ο αλγόριθμος έχει καλύτερη απόδοση όταν χρησιμοποιεί την τιμή 0,05 ως προς την αναζήτηση της καλύτερης λύσης καθώς και στην ταχύτητα που την βρίσκει. Ο τελεστής αναρρίχησης όπως βλέπουμε όταν χρησιμοποιείται βοηθάει αισθητά στην αναζήτηση της καλύτερης λύσης. Με την βελτίωση μόνο της καλύτερης λύσης κάθε γενιάς βλέπουμε μεγάλη διαφορά στην αναζήτηση της καλύτερης λύσης από την βελτίωση όλων των λύσεων. Τέλος ο αλγόριθμος έχει καλύτερη απόδοση ως προς τον χρόνο εκτέλεσης χρησιμοποιώντας 500 γενιές και όχι περισσότερες διότι σε αυτήν την περίπτωση παρατηρούμε ότι η αναζήτηση της καλύτερης λύσης παραμένει σταθερή αλλά ο χρόνος εκτέλεσης του αλγόριθμου αυξάνεται αισθητά.

3.2.5 Παρουσίαση των καλύτερων παραμέτρων για κάθε πρόλημα.

Στον Πίνακα 8 παρουσιάζονται τα προβλήματα που επιλύθηκαν παραπάνω με τις βέλτιστες παραμέτρους, οι οποίοι μας οδήγησαν στα παραπάνω αποτελέσματα.

Best Parameters For Each Problem				
Params	Rosenbrock	Rastrigin	Schwefel	Parallilogram
Population	100	100	100	20
Generations	5000	500	500	500
BITS	16	12	10	10
Belief Space	20	20	50	10
Mutation	0.05	0.02	0.01	0.05
Hill Climb	Yes	Yes	Yes	Yes
Solution	All	Best Only	All	Best Only

Πίνακας 8: Βέλτιστες τιμές παραμέτρων για κάθε πρόβλημα

Κεφάλαιο 4 Συμπεράσματα από την πραγματοποίηση της πτυχιακής, προτάσεις για μελλοντικές επεκτάσεις

4.1 Συμπεράσματα χρήσης των Αλγορίθμων Κουλτούρας

‘Οπως είδαμε η πτυχιακή μας ήταν η δημιουργία Αλγορίθμων Κουλτούρας που ήταν επέκταση των γενετικών αλγόριθμών. Οι Αλγόριθμοι Κουλτούρας συνετέλεσαν στην πιο γρήγορη εύρεση της βέλτιστης λύσης σε σύγκριση με τους Γενετικούς Αλγορίθμους. Αυτό συνέβη διότι όπως είδαμε οι καλύτερες λύσεις κάθε γενιάς αποθηκεύονται σε έναν χώρο μνήμης που ανάλογα με το μέγεθός του οι κακές λύσεις αντικαθίστανται από καλύτερες και αυτές οι λύσεις αλληλοεπιδράνε με τον Γενετικό Αλγόριθμο, έχοντας έτσι πάντα μια εξελικτική διαδικασία και καλύτερες επιδόσεις. Στην συνέχεια παρουσιάζονται οι συγκριτικοί πίνακες για κάθε πρόβλημα, στους οποίους βλέπούμε τα αποτελέσματα των δύο αλγορίθμων. Παρατηρείται ότι ο Αλγόριθμος Κουλτούρας μπορεί να βρεί την βέλτιστη λύση πιο γρήγορα καθώς και πολλές φορές να συγκλίνει προς την βέλτιστη λύση καλύτερα από ότι ο Γενετικός Αλγόριθμος. Τα αποτελέσματα για τον Γενετικό Αλγόριθμο υπολογίστηκαν με τον ίδιο τρόπο που υπολογίστηκαν για τον Αλγόριθμο Κουλτούρας, δηλαδή με τις ίδιες παραμέτρους και με δέκα επαναλήψεις ανα συνδυασμό διότι και ο Γενετικός Αλγόριθμος είναι στοχαστικός.

4.1.1 Συγκριτικός πίνακας των αποτελεσμάτων για τη συνάρτηση Rosenbrock

Στον Πίνακα 9 παρουσιάζονται τα αποτελέσματα των παραμέτρων της συνάρτησης Rosenbrock για τον αλγόριθμο κουλτούρας και τον γενετικό αλγόριθμο.

Problem	Population	Generations	BITS	Belief Space	Mutation	HillClimb	Solution	Best Fitness	Gen of Best Fit	Time		GA Best Fit	GA Gen Best Fit	GA Time
Rosenbrock	10	10	10	5	0.01	yes	best only	0.43568005	6.8	0.0188		0.9042897	6.2	0.0188
Rosenbrock	20	10	10	5	0.01	yes	best only	0.14258756	5.2	0.022		0.01189856	4.6	0.0186
Rosenbrock	50	10	10	5	0.01	yes	best only	0.03547668	3.7	0.0297		0.01263446	2	0.022
Rosenbrock	100	10	10	5	0.01	yes	best only	0.04571495	4.5	0.0533	pop	0.01501942	5.6	0.0438
Rosenbrock	100	10	12	5	0.01	yes	best only	0.0101554	3	0.047		0.031601	6.6	0.066
Rosenbrock	100	10	14	5	0.01	yes	best only	0.0075811	10	0.047		0.02333084	8.8	0.0782
Rosenbrock	100	10	16	5	0.01	yes	best only	0.0042115	10	0.063	bits	0.02812372	10	0.078
Rosenbrock	100	10	16	10	0.01	yes	best only	0.0061662	10	0.047		/	/	/
Rosenbrock	100	10	16	20	0.01	yes	best only	0.0000071	8	0.047	bs	/	/	/
Rosenbrock	100	10	16	50	0.01	yes	best only	0.0032783	10	0.047		/	/	/
Rosenbrock	100	10	16	20	0.01	yes	best only	0.0184923	10	0.063		0.02812372	10	0.078
Rosenbrock	100	10	16	20	0.02	yes	best only	0.0189386	10	0.047		0.0367294	10	0.0716
Rosenbrock	100	10	16	20	0.05	yes	best only	0.0001451	10	0.047	mutation	0.00941762	10	0.078
Rosenbrock	100	10	16	20	0.005	yes	best only	0.0134146	10	0.046		0.03893846	9.2	0.0718
Rosenbrock	100	10	16	20	0.05	yes	best only	0.0001451	10	0.047	hillclimb	0.00941762	10	0.078
Rosenbrock	100	10	16	20	0.05	no	best only	0.0009072	9	0.047		0.01375436	7.2	0.0782
Rosenbrock	100	10	16	20	0.05	yes	best only	0.0381617	10	0.047		/	/	/
Rosenbrock	100	10	16	20	0.05	yes	all	0.004104	10	0.047	solution	0.02307406	10	0.0782
Rosenbrock	100	200	16	20	0.05	yes	all	0.0002515	59	0.719		0.00005748	88.2	0.9548
Rosenbrock	100	500	16	20	0.05	yes	all	0.0000847	314	1.719		0.00007514	275	2.1278
Rosenbrock	100	1000	16	20	0.05	yes	all	0.0000003	33	3.406		0.00006574	710.6	4.6
Rosenbrock	100	5000	16	20	0.05	yes	all	0	4901	16.563	generations	0	2956.8	21.3062

Πίνακας 9: Συγκριτικός πίνακας των αποτελεσμάτων για τη συνάρτηση Rosenbrock

4.1.2 Συγκριτικός πίνακας των αποτελεσμάτων για τη συνάρτηση Rastrigin

Στον Πίνακα 10 παρουσιάζονται τα αποτελέσματα των παραμέτρων της συνάρτησης Rastrigin για τον αλγόριθμο κουλτούρας και τον γενετικό αλγόριθμο.

Find the best combination for the parameters in the optimization of Rastrigin function											Best values	GA Results		
Problem	Population	Generations	BITS	Belief Space	Mutation	HillClimb	Solution	Best Fitness	Gen of Best Fit	Time		GA Best Fit	GA Gen Best Fit	GA Time
Rastrigin	10	10	10	5	0.01	yes	best only	3.98499554	4.6	0.0188		1.59741528	4.4	0.0218
Rastrigin	20	10	10	5	0.01	yes	best only	2.79198174	4	0.0282		1.41045904	6.8	0.0218
Rastrigin	50	10	10	5	0.01	yes	best only	0.80209324	4.6	0.0346		0.60405448	5.8	0.0316
Rastrigin	100	10	10	5	0.01	yes	best only	0.0099382	2.75	0.0535	pop	0.60405448	7.6	0.0438
Rastrigin	100	10	12	5	0.01	yes	best only	0.04172902	7.4	0.0566	bits	0.39855386	8	0.0436
Rastrigin	100	10	14	5	0.01	yes	best only	0.73114084	8.8	0.05		0.44845834	9.2	0.047
Rastrigin	100	10	16	5	0.01	yes	best only	0.36169594	10	0.0564		0.52916244	10	0.0534
Rastrigin	100	10	12	10	0.01	yes	best only	0.49803725	5.25	0.047		/	/	/
Rastrigin	100	10	12	20	0.01	yes	best only	0.49803725	4.75	0.04275	bs	/	/	/
Rastrigin	100	10	12	50	0.01	yes	best only	0.60148958	8.4	0.0468		/	/	/
Rastrigin	100	10	12	20	0.01	yes	best only	0.49803725	4.75	0.04275		/	/	/
Rastrigin	100	10	12	20	0.02	yes	best only	0.19958708	5.4	0.0468	mutation	0.59742188	7.2	0.0438
Rastrigin	100	10	12	20	0.05	yes	best only	1.01478018	5.8	0.0406		0.39855386	7.4	0.0438
Rastrigin	100	10	12	20	0.005	yes	best only	0.40500246	8.6	0.047		0.39905006	6.6	0.047
Rastrigin	100	10	12	20	0.02	yes	best only	0.19958708	5.4	0.0468	hillclimb	0.59742188	7.2	0.0438
Rastrigin	100	10	12	20	0.02	no	best only	0.63298032	7.8	0.0438		0.64989134	8.6	0.0468
Rastrigin	100	10	12	20	0.02	yes	best only	0.19958708	5.4	0.0468	solution	0.59742188	7.2	0.0438
Rastrigin	100	10	12	20	0.02	yes	all	0.19958708	7.6	0.047		0.19958708	7.2	0.047
Rastrigin	100	200	12	20	0.02	yes	best only	0.0006203	28.6	0.5748		0.0006203	7.8	0.5784
Rastrigin	100	500	12	20	0.02	yes	best only	0.0006203	10	1.3338	generations	0.0006203	8.6	1.3312
Rastrigin	100	1000	12	20	0.02	yes	best only	0.0006203	12.6	2.7624		0.0006203	19.2	2.7844
Rastrigin	100	5000	12	20	0.02	yes	best only	0.0006203	17.8	14.3842		0.0006203	29	13.0218

Πίνακας 10: Συγκριτικός πίνακας των αποτελεσμάτων για τη συνάρτηση Rastrigin

4.1.3 Συγκριτικός πίνακας των αποτελεσμάτων για τη συνάρτηση Schwefel

Στον Πίνακα 11 παρουσιάζονται τα αποτελέσματα των παραμέτρων της συνάρτησης Schwefel για τον αλγόριθμο κουλτούρας και τον γενετικό αλγόριθμο.

Find the best combination for the parameters in the optimization of Schwefel function											Best values	GA Results		
Problem	Population	Generations	BITS	Belief Space	Mutation	HillClimb	Solution	Best Fitness	Gen of Best Fit	Time		GA Best Fit	GA Gen Best Fit	GA Time
Schwefel	10	10	10	5	0.01	yes	best only	79.6907199	6.2	0.0188		118.8584104	7.4	0.0188
Schwefel	20	10	10	5	0.01	yes	best only	90.8071573	5.2	0.0188		23.6926494	6.4	0.0154
Schwefel	50	10	10	5	0.01	yes	best only	43.43290948	7	0.025		0.0055255	6.4	0.0286
Schwefel	100	10	10	5	0.01	yes	best only	0.0055255	3.8	0.047	pop and bits	0.0055255	6	0.0468
Schwefel	100	10	12	5	0.01	yes	best only	0.00586364	8.4	0.0138		0.00586364	7.2	0.047
Schwefel	100	10	14	5	0.01	yes	best only	1.11419572	10	0.0436		1.04255794	9.8	0.047
Schwefel	100	10	16	5	0.01	yes	best only	2.83661752	9.8	0.047		9.80359484	10	0.0498
Schwefel	100	10	10	10	0.01	yes	best only	0.0055255	5	0.0404		/	/	/
Schwefel	100	10	10	20	0.01	yes	best only	0.0055255	5.4	0.0314		/	/	/
Schwefel	100	10	10	50	0.01	yes	best only	0.0055255	4.8	0.0438	bs	/	/	/
Schwefel	100	10	10	50	0.01	yes	best only	0.0055255	4.8	0.0438	mutation	0.00586364	6	0.0468
Schwefel	100	10	10	50	0.02	yes	best only	0.0055255	5	0.408		0.0055255	6.2	0.0408
Schwefel	100	10	10	50	0.05	yes	best only	0.0055255	5	0.041		0.0055255	5.4	0.0372
Schwefel	100	10	10	50	0.005	yes	best only	0.0055255	5	0.0408		0.0055255	4.2	0.0438
Schwefel	100	10	10	50	0.01	yes	best only	0.0055255	4.8	0.0438	hillclimb	0.0055255	6	0.0468
Schwefel	100	10	10	50	0.01	no	best only	1.76403232	8	0.0438		0.94327624	8.8	0.0378
Schwefel	100	10	10	50	0.01	yes	best only	0.0055255	4.8	0.0438		0.0055255	6	0.0468
Schwefel	100	10	10	50	0.01	yes	all	0.0055255	4.8	0.0434	solution	0.0055255	5.6	0.0438
Schwefel	100	200	10	50	0.01	yes	all	0.0055255	5.4	0.5938		0.0055255	5.2	0.606
Schwefel	100	500	10	50	0.01	yes	all	0.0055255	5	1.5188	generations	0.0055255	5.6	1.4406
Schwefel	100	1000	10	50	0.01	yes	all	0.0055255	5.4	2.7654		0.0055255	5.4	3.083
Schwefel	100	5000	10	50	0.01	yes	all	0.0055255	6.8	14.2624		0.0055255	5.4	13.5468

Πίνακας 11: Συγκριτικός πίνακας των αποτελεσμάτων για τη συνάρτηση Schwefel

4.1.4 Συγκριτικός πίνακας των αποτελεσμάτων για τη συνάρτηση του Παραλληλόγραμμου

Στον Πίνακα 12 παρουσιάζονται τα αποτελέσματα των παραμέτρων της συνάρτησης του Παραλληλόγραμμου για τον αλγόριθμο κουλτούρας και τον γενετικό αλγόριθμο.

Find the best combination for the parameters in the optimization of Parallilogram function											Best values	GA Results		
Problem	Population	Generations	BITS	Belief Space	Mutation	HillClimb	Solution	Best Fitness	Gen of Best Fit	Time		GA Best Fit	GA Gen Best Fit	GA Time
Parallilogram	10	10	10	5	0.01	yes	best only	601.032663	9.6	0.0188		607.4685622	9.2	0.016
Parallilogram	20	10	10	5	0.01	yes	best only	600.004407	8.8	0.0158	pop and bits	600.0050381	9.8	0.022
Parallilogram	50	10	10	5	0.01	yes	best only	600.0361382	9	0.031		600.0125865	7.4	0.0312
Parallilogram	100	10	10	5	0.01	yes	best only	600.0050381	7.8	0.0408		600.0504153	6.4	0.0468
Parallilogram	20	10	12	5	0.01	yes	best only	603.142763	9	0.0218		608.5312818	10	0.025
Parallilogram	20	10	14	5	0.01	yes	best only	624.8034755	10	0.0216		610.682337	10	0.022
Parallilogram	20	10	16	5	0.01	yes	best only	614.8269384	10	0.0188		607.3772488	10	0.0188
Parallilogram	20	10	10	10	0.01	yes	best only	600.0050381	7.4	0.0222	bs	/	/	/
Parallilogram	20	10	10	20	0.01	yes	best only	600.0063004	9	0.0216		/	/	/
Parallilogram	20	10	10	50	0.01	yes	best only	600.0199572	7	0.0252		/	/	/
Parallilogram	20	10	10	10	0.01	yes	best only	600.0050381	7.4	0.0222		600.0504153	6.4	0.0468
Parallilogram	20	10	10	10	0.02	yes	best only	600.0095546	7.4	0.0218		600.1188727	6.2	0.0216
Parallilogram	20	10	10	10	0.05	yes	best only	600.004407	7	0.0188	mutation	600.1689364	9.2	0.0248
Parallilogram	20	10	10	10	0.005	yes	best only	600.1413126	8.6	0.0282		601.1011835	8.8	0.0252
Parallilogram	20	10	10	10	0.05	yes	best only	600.004407	7	0.0188	hillclimb	600.1689364	9.2	0.0248
Parallilogram	20	10	10	10	0.05	no	best only	607.7976322	9	0.019		616.476954	7.6	0.0188
Parallilogram	20	10	10	10	0.05	yes	best only	600.004407	7	0.0188	solution	600.1689364	9.2	0.0248
Parallilogram	20	10	10	10	0.05	yes	all	600.0595856	8.6	0.0218		600.4442411	9	0.022
Parallilogram	20	200	10	10	0.05	yes	best only	600.004407	11.4	0.1158		600.004407	7.8	0.1218
Parallilogram	20	500	10	10	0.05	yes	best only	600.004407	7.4	0.2564	generations	600.004407	8.8	0.2532
Parallilogram	20	1000	10	10	0.05	yes	best only	600.004407	7.4	0.4998		600.004407	8.8	0.4904
Parallilogram	20	5000	10	10	0.05	yes	best only	600.004407	7.8	2.418		600.004407	9.6	2.3466

Πίνακας 12: Συγκριτικός πίνακας των αποτελεσμάτων για τη συνάρτηση του Παραλληλόγραμμου

4.2 Μελλοντικές Επεκτάσεις

Όσον αφορά τις μελλοντικές επεκτάσεις της πτυχιακής, μία από αυτές θα μπορούσε να είναι η προσθήκη περισσοτέρων συναρτήσεων βελτιστοποίησης ή προβλημάτων. Επιπλέον, θα ήταν καλό η παραπάνω πτυχιακή να αναπτυχθεί σε κάποιο άλλο πιο μοντέρνο περιβάλλον (IDE), καθώς στις μέρες μας υπάρχουν πολύ χρήσιμα και πιο εύκολα εργαλεία ανάπτυξης κώδικα όσον αφορά τη λειτουργικότητα και την παραγωγικότητα του προγραμματιστή. Μία ακόμη πρόταση είναι η δημιουργία γραφημάτων μετά το τέλος της εκτέλεσης του αλγορίθμου έτσι, ώστε ο χρήστης να έχει και οπτική εικόνα των αποτελεσμάτων.

Βιβλιογραφία

Έντυπη:

1. Καζαρλής Σπυρίδων. Σημειώσεις του μαθήματος Εξελικτική Υπολογιστική
2. Καζαρλής Σπυρίδων. Ασκήσεις εργαστηρίων του μαθήματος Εξελικτική Υπολογιστική

Ηλεκτρονική:

1. http://www.cleveralgorithms.com/nature-inspired/physical/cultural_algorithm.html
2. http://eprints.uwe.ac.uk/25649/1/From%20Evolutionary%20Computation%20to%20the%20Evolution%20of%20Things.pdf
3. http://www.ra.cs.uni tuebingen.de/mitarb/streiche/publications/Introduction_to_Evolutionary_Algorithms.pdf
4. https://stackoverflow.com/questions/22479584/how-to-count-the-hamming-distance-of-two-short-int
5. https://stackoverflow.com/questions/8760473/roulette-wheel-selection-for-function-minimization
6. http://www.geatbx.com/docu/fcnindex-01.html
7. https://stackoverflow.com/questions/9878965/rand-between-0-and-1
8. http://www.cplusplus.com/forum/general/65862/
9. https://docs.microsoft.com/en-us/windows/desktop/api/shellapi/nf-shellapi-shellexecutea
10. http://docs.embarcadero.com/products/rad_studio/delphiAndcpp2009/HelpUpdate2/EN/html/delphivclwin32/SysUtils_TryStrToFloat@string@Double.html
11. http://docs.embarcadero.com/products/rad_studio/delphiAndcpp2009/HelpUpdate2/EN/html/delphivclwin32/SysUtils_TryStrToInt.html
12. https://docs.microsoft.com/en-us/windows/desktop/api/sysinfoapi/nf-sysinfoapi-gettickcount

Παράθεση κώδικα

Αρχείο Basic.cpp

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Basic.h"
#include "Functions.h"
#include "View.h"
#include "VisualFunctions.h"

//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TBasicForm *BasicForm;
//---------------------------------------------------------------------------

__fastcall TBasicForm::TBasicForm(TComponent* Owner)
        : TForm(Owner)
{

   Population->ItemIndex=0;
   Generations->ItemIndex=0;
   Beliefspace->ItemIndex=0;
   Bits->ItemIndex=0;
   Problem->ItemIndex=0;
   Mutation->ItemIndex=2;
   Variation->ItemIndex=0;
   Hillclimb->Checked=true;
   EditName->Visible=false;
   Label8->Visible=false;

}
//---------------------------------------------------------------------------

//---------------------------------------------------------------------------


//---------------------------------------------------------------------------

void __fastcall TBasicForm::RunClick(TObject *Sender)
```

```
{
	float a;
	int b;
	int start_time, end_time;
	//Εναρξή της χρονομέτρησης του αλγορίθμου.
	start_time = GetTickCount();

		// Η συνάρτηση TryStrToFloat μετατρέπει μια συμβολοσειρά σε πραγματικό αριθμό.
	if(!TryStrToFloat(Mutation->Text.c_str(),a))
	{
		ShowMessage("You must enter numerical characters");
		if(a==0)
			ShowMessage("You can't set 0 for Mutation Rate");
		Mutation->SetFocus();
		return;
	 }
	 else if(!TryStrToInt(Population->Text.c_str(),b) || b<10)
	 {
		ShowMessage("You must set only integer number in Population Field and bigger than 10");

		Mutation->SetFocus();
		return;
	 }
	 else if(!TryStrToInt(Generations->Text.c_str(),b) || b<10)
	 {
		ShowMessage("You must set only integer number in Generation Field and bigger than 10");
		Mutation->SetFocus();
		return;
	 }
	  else if(!TryStrToInt(Beliefspace->Text.c_str(),b) || b<5)
	 {
		ShowMessage("You must set only integer number in Belief Space Field and bigger than 5");
		Mutation->SetFocus();
		return;
	 }
	  else if(!TryStrToInt(Bits->Text.c_str(),b) || b<10)
	 {
		ShowMessage("You must set only integer number in BITS Field and bigger than 10");
		Mutation->SetFocus();
		return;
	 }
```

```
        int B=BasicForm->Beliefspace->Text.ToInt();
                /* Αρχικοποιούμε το πάνελ στο οποίο βλέπουμε τις τιμές του
                χώρου μνήμης έτσι ώστε κάθε φορά που τρέχουμε το πρόγραμμα
                το πάνελ να είναι άδειο και να μην περιέχει σκουπίδια*/
        for(int i=0; i<B; i++)
            ViewForm->Panel2->DoubleBuffered=true;
            ViewForm->LabelBeliefS->Update();
        // Αρχικοποιήση των τιμών την αντικειμένων της δεύτερης φόρμας.
        initialize();
                // Εκτέλεση του Βασικού Αλγορίθμου.
        algorithm();

                // Εμφάνιση της View φόρμας.
        ViewForm->Show();
        BasicForm->Hide();
                // Έλεγχος για αν έχουμε επιλέξει να σώσουμε τα αποτελέσματα του αλγορίθμου.
        if(Savefile->Checked==false)
            ViewForm->Openfilebutton->Enabled=false;
        //Τέλος της χρονομέτρησης του αλγορίθμου.
        end_time = GetTickCount() - start_time;
        ViewForm->PanelTime->Caption = FloatToStr(end_time / 1000.0)+ " sec";

}
//---------------------------------------------------------------------------


void __fastcall TBasicForm::ExitClick(TObject *Sender)
{
        BasicForm->Close();
}
//---------------------------------------------------------------------------

void __fastcall TBasicForm::SavefileClick(TObject *Sender)
{
                if(Savefile->Checked==true)
                {
                    EditName->Visible=true;
                    Label8->Visible=true;
                }
                else
                {
                  EditName->Visible=false;
                  Label8->Visible=false;
                }

}
```

```
//---------------------------------------------------------------------------
```

Αρχείο Basic.h

```
//---------------------------------------------------------------------------

#ifndef BasicH
#define BasicH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
//---------------------------------------------------------------------------
class TBasicForm : public TForm
{
__published:    // IDE-managed Components
        TLabel *Label1;
        TLabel *Label2;
        TLabel *Label4;
        TLabel *Label5;
        TLabel *Label6;
        TLabel *Label7;
        TComboBox *Generations;
        TComboBox *Beliefspace;
        TComboBox *Bits;
        TComboBox *Problem;
        TComboBox *Mutation;
        TComboBox *Variation;
        TCheckBox *Savefile;
        TCheckBox *Hillclimb;
        TButton *Run;
        TButton *Exit;
        TLabel *Label3;
        TComboBox *Population;
        TEdit *EditName;
        TLabel *Label8;
        void __fastcall RunClick(TObject *Sender);
        void __fastcall ExitClick(TObject *Sender);
        void __fastcall SavefileClick(TObject *Sender);
```

```
private:        // User declarations
public:         // User declarations
        __fastcall TBasicForm(TComponent* Owner);

};
//---------------------------------------------------------------------------
extern PACKAGE TBasicForm *BasicForm;
//---------------------------------------------------------------------------
#endif
```

Αρχείο View.cpp

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "View.h"
#include "Functions.h"
#include "VisualFunctions.h"
#include "Basic.h"

//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TViewForm *ViewForm;
//---------------------------------------------------------------------------
__fastcall TViewForm::TViewForm(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------




//---------------------------------------------------------------------------


//---------------------------------------------------------------------------

void __fastcall TViewForm::BackbuttonClick(TObject *Sender)
{
```

```
	int B=BasicForm->Beliefspace->Text.ToInt();
	for(int i=0; i<B; i++)
		ViewForm->Panel2->DoubleBuffered=true;
		ViewForm->LabelBeliefS->Update();
	ViewForm->Close();
	cleandata();
	BasicForm->Show();

}
//---------------------------------------------------------------------------


void __fastcall TViewForm::ExitbuttonClick(TObject *Sender)
{
	exit(0);
}
//---------------------------------------------------------------------------

void __fastcall TViewForm::OpenfilebuttonClick(TObject *Sender)
{
		if(BasicForm->EditName->Text!="")
		{
		   String name=BasicForm->EditName->Text + ".txt";
		   ShellExecute(0, "open", name.c_str(), NULL, NULL, SW_SHOWNORMAL);
		}
		else
			ShellExecute(0, "open", "results.txt", NULL, NULL, SW_SHOWNORMAL);
}
//---------------------------------------------------------------------------
```

Αρχείο View.h

```
//---------------------------------------------------------------------------

#ifndef ViewH
#define ViewH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ExtCtrls.hpp>
#include <Grids.hpp>
//---------------------------------------------------------------------------
```

```
class TViewForm : public TForm
{
__published:   // IDE-managed Components
        TPanel *Panel1;
        TPanel *Panel2;
        TLabel *LabelPop;
        TLabel *LabelGen;
        TLabel *LabelBS;
        TPanel *PanelPop;
        TPanel *PanelGen;
        TPanel *PanelBS;
        TLabel *Label1;
        TLabel *Labelfunc;
        TLabel *Labelmrate;
        TPanel *PanelBITS;
        TPanel *Panelfunc;
        TPanel *Panelmrate;
        TLabel *LabelVar;
        TPanel *PanelVar;
        TLabel *LabelHC;
        TPanel *PanelHC;
        TLabel *LabelSV;
        TPanel *PanelSV;
        TLabel *LabelBeliefS;
        TStringGrid *Table;
        TLabel *Label2;
        TLabel *Label3;
        TButton *Backbutton;
        TButton *Exitbutton;
        TButton *Openfilebutton;
        TLabel *LabelBel;
        TLabel *LabelTime;
        TPanel *PanelTime;
        void __fastcall BackbuttonClick(TObject *Sender);
        void __fastcall ExitbuttonClick(TObject *Sender);
        void __fastcall OpenfilebuttonClick(TObject *Sender);


private:        // User declarations
public:         // User declarations
        __fastcall TViewForm(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TViewForm *ViewForm;
//---------------------------------------------------------------------------
#endif
```

Αρχείο VisualFunctions.cpp

```
#include <vcl.h>
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <math.h>
#include "Basic.h"
#include "View.h"
#include "Functions.h"


void iniTable()
{
     // ViewForm->Table->ColCount=8;
      ViewForm->Table->RowCount=BasicForm->Generations->Text.ToInt()+1;
       ViewForm->Table->Cells[0][0]="Generations";
       ViewForm->Table->Cells[1][0]="BinaryGeno0";
       ViewForm->Table->Cells[2][0]="BinaryGeno1";
       ViewForm->Table->Cells[3][0]="Geno0";
       ViewForm->Table->Cells[4][0]="Geno1";
       ViewForm->Table->Cells[5][0]="Pheno0";
       ViewForm->Table->Cells[6][0]="Pheno1";
       ViewForm->Table->Cells[7][0]="Fitness";

       ViewForm->Table->ColWidths[0]=50;
       ViewForm->Table->ColWidths[1]=70;
       ViewForm->Table->ColWidths[2]=70;
       ViewForm->Table->ColWidths[3]=40;
       ViewForm->Table->ColWidths[4]=40;
       ViewForm->Table->ColWidths[5]=110;
       ViewForm->Table->ColWidths[6]=110;
       ViewForm->Table->ColWidths[7]=110;


}



void initialize()
{

      ViewForm->PanelPop->Caption=BasicForm->Population->Text.ToInt();
      ViewForm->PanelGen->Caption=BasicForm->Generations->Text.ToInt();
      ViewForm->PanelBS->Caption=BasicForm->Beliefspace->Text.ToInt();
```

```
        ViewForm->PanelBITS->Caption=BasicForm->Bits->Text.ToInt();

        ViewForm->Panelfunc->Caption=BasicForm->Problem->Text;

       ViewForm->Panelmrate->Caption=BasicForm->Mutation->Text.ToDouble();

       ViewForm->PanelVar->Caption=BasicForm->Variation->Text;

       if(BasicForm->Hillclimb->Checked==true)
              ViewForm->PanelHC->Caption="true";
       else
              ViewForm->PanelHC->Caption="false";

      if(BasicForm->Savefile->Checked==true)
              ViewForm->PanelSV->Caption="true";
       else
              ViewForm->PanelSV->Caption="false";

      iniTable();

}


void cleandata()
{

        ViewForm->LabelBeliefS->Caption=" ";
        ViewForm->Openfilebutton->Enabled=true;
        cleanFlagBS();
}
```

Αρχείο VisualFunctions.h

```
#include <vcl.h>
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <math.h>

void initialize();
void iniTable();
void cleandata();
```

Αρχείο Functions.cpp

```
//---------------------------------------------------------------------------

#include <vcl.h>
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <math.h>
#include <sstream>
#include <iomanip>
#include "Basic.h"
#include "View.h"
#include "VisualFunctions.h"
#define VARS 2

static flagBS;

/* Δήλωση της δόμης δεδόμενων του ατόμου
με χαρακτηριστικά τον γονότυπο,
τον φενότυπο και την ποιότητα του. */
struct atom{
	int geno[VARS];
	double pheno[VARS];
	double fitness;
};

/*Δήλωση της συνάρτης rnd() η οποία όταν καλείται μας επιστρέφει
έναν τυχαίο αριθμό απο 0 εώς 1. */
double rnd(void)
{
	/*Δήλωση της συνάρτης rnd() η οποία όταν καλείται μας επιστρέφει
	έναν τυχαίο αριθμό απο 0 εώς 1. */

	return (double)rand()/(RAND_MAX+0.1);   //!!

}




atom Mutation (atom x, double prob, int BITS)
{
	//ginetai praksi xor gia tin antistrofi tou bit
	double E,dek;
```

```
	int ak,mp,i,point,mask,Chrom,Chrompos;
	E=VARS*BITS*prob;		// E=ektimisi shmeiwn metallaksis
	ak=(int)E;
	dek=E-ak;
	mp=ak;
	if(rnd()<dek) mp++;
	//printf("BITS=%i prob=%lf E=%lf ak=%i dek=%lf mp=%i\n",BITS,prob,E,ak,dek,mp);
	for(i=0;i<mp;i++)
	{
		point=random(BITS*VARS); // 0 eos 39
		//printf("Before =%s\n",x.c_str());
		//printf("point=%i\n",point);
		Chrom = point/BITS; // 0,1,2,3...
		Chrompos = point%BITS;
		mask = 1<<(BITS-Chrompos-1) ;
		x.geno[Chrom]=x.geno[Chrom]^mask;
		//printf("After =%s\n",x.c_str());
	}
	//getch();
	return x;
}



atom Crossover (atom p1, atom p2, int BITS)
{
	//and &
	//or  |
	//xor ^
	//not ~

	int point, mask2, Chrom, Chrompos, j;
	atom child;
	point = random(VARS * BITS-1); //τυχαιός αριθμός απο 0 εώς BITS*VARS
	Chrom = point/BITS; // ο τυχαίος αριθμός διά των αριθμό των BITS
	Chrompos = point%BITS;  //από την θέση 0 έώς 7 το χρωμόσωμα όπου θα γίνει το crossover
	for(j=0;j<Chrom;j++) {
		child.geno[j]=p1.geno[j];
	}
	for(j=Chrom+1;j<VARS;j++)
	{
		child.geno[j]=p2.geno[j];
	}
	mask2 = (1<<BITS-Chrompos-1)-1;
```

```
            //ορισμός του χρωμοσώματος όπου θα γίνει το crossover
            child.geno[Chrom] = (p1.geno[Chrom]&(~mask2))|(p2.geno[Chrom]& mask2);
    /* θα πάρουμε απο το p1 τόσο όσο ορίζει η mask1
    και απο το p2 τόσο όσο ορίζει η mask2 */
    //enonoume me tin praksi or
    // ~ NOT operator

    /*Αρχικά βρίσκει έναν τυχαίο αριθμό point μεταξύ 0 και BITS*VARS
    αποθηκεύει το αποτέλεσμα της διαίρεσης point/BITS στην μεταβλητή chrom
    άρα το αποτέλεσμα της διαίρεσης θα ειναι 0 ή 1
            αποθηκεύει το υπόλοιπο της διαίρεσης στην μεταβλητή chrompos
            αν το αποτέλεσμα ειναι 0 εκετελείται η δεύτερη for
            όπου ο γονότυπος του παιδιού γίνεται ίσος με τον γονότυπο του δεύτερου γονέα.
    αν το αποτέσμα ειναι 1 εκτελείται η πρώτη for
    όπου ο γονότυπος του παιδιού γίνεται ίσος με τον γονότυπο του δεύτερου γονέα.
            στην συνέχεια βρίσκουμε την μάσκα για να δημιουργήσουμε το γενότυπο που
            δεν αντιγράψαμε απο κάποιον γονέα. */


    return child;
}



String Int2Str(int x, int BITS)
{
    String s="";
    int mask,i;
    mask= 1<<(BITS-1);
    for(i=0;i<BITS;i++)
    {
        if(x&mask) s+='1';
    else
            s+='0';
    mask=mask>>1;
    }
    return s;
}




int Str2Int(String x, int BITS)
{
    int i,sum=0,p;
```

```
	p=1<<(BITS-1); //128
	for(i=1;i<=BITS;i++)
	  {
	    if(x[i] =='1')
	    sum+=p;
	    p=p/2; // p=p>>1;
	  }
	return sum;

}



void Geno2Pheno(atom & x, int BITS, int fun)
{
	int j;
	double vhma;
	//vhma=2*M_PI/(pow(2,BITS)-1);

	for(j=0;j<VARS;j++) {
		if(fun==0)
		{
			vhma=4.096/(pow(2,BITS)-1);
			x.pheno[j]=x.geno[j]*vhma-2.048;  //Για την συνάρτηση Rosenbrock
		}
		else if(fun==1)
		{
			vhma =10.24/(pow(2,BITS)-1);
			x.pheno[j]=x.geno[j]*vhma-5.12; //Για την συνάρτηση Rastrigin
		}
		else if(fun==2)
		{
			vhma=1000/(pow(2,BITS)-1);
			x.pheno[j]=x.geno[j]*vhma-500;  //Για την συνάρτηση Schwefel
		}
		else if(fun==3)
		{
			vhma=100/(pow(2,BITS)-1);
			x.pheno[j]=x.geno[j]*vhma+0.1; //Για την συνάρτηση του Παραλληλογράμμου
		}
	}
}
```

```
void Fitness(atom & x, int BITS, int fun)
{
	double x1,x2;
	Geno2Pheno(x, BITS, fun);
	x1=x.pheno[0];
	x2=x.pheno[1];
	if(fun==0)
		x.fitness= (100*pow(x1*x1-x2,2)+pow(1-x1,2));   //rosenbrock
	else if(fun==1)
		x.fitness=  20  +  (pow(x1,2)  -  10*cos(2*M_PI*x1))  +  (pow(x2,2)  -
10*cos(2*M_PI*x2));     //Rastrigin
	else if(fun==2)
		x.fitness= (418.9829*2) - x1*sin(sqrt(fabs(x1))) - x2*sin(sqrt(fabs(x2))); //schwefel
	else if(fun==3)
	{
		double z,ogkos=1000;
		//ο τύπος για τον όγκο του παραλληλογράμμου=x1*x2*z
		z=ogkos/(x1*x2);
		x.fitness= (2*(x1*z))+(2*(x2*z))+(2*(x1*x2));
				// Το εμβαδό της παράπλευρής επιφάνειας ειναι το άθροισμα των
όγκων των 6 επιφανειών
	}
}



void HillClimb(atom & x, int BITS, int fun)
{

	int i,j;
	int steps[4]= {10,-10,1,-1};
	atom oldx;
	oldx = x;
	for (j=0;j<VARS;j++)
		for (i=0;i<4;i++)
			if((x.geno[j]+steps[i])>=0 && x.geno[j]+steps[i]<pow(2,BITS))
			{
				x.geno[j]=x.geno[j]+steps[i];
				Fitness(x, BITS, fun);
				if(x.fitness<oldx.fitness)
					oldx=x;
				else
					x=oldx;
			}

}
```

```
void Display(atom * x, int N, int BITS, FILE *f1)
{
      int i,j;
       for (i=0;i<N;i++) {
            fprintf(f1,"Atom %2i : ",i);
            for (j=0;j<VARS;j++) {
                  fprintf(f1, "%s ",Int2Str(x[i].geno[j], BITS));
            }
            for (j=0;j<VARS;j++) {
                  fprintf(f1,"%5i ",x[i].geno[j]);
            }
            for (j=0;j<VARS;j++) {
                  fprintf(f1,"%10.7lf  ",x[i].pheno[j]);
            }
            if(BasicForm->Problem->ItemIndex==0)
                  fprintf(f1,"f= %10.7lf \n",x[i].fitness*(-1));
            else
                  fprintf(f1,"f= %10.7lf \n",x[i].fitness);
       }
}




int Roulette(atom * x, int N)
{
       double sum,sum2=0,sumrnd,max_fs,min_fs,s;
       int i;
       sum=0;
       max_fs=x[0].fitness;

       s=0;
       //Δυναμική δήλωση ενός δείκτη απο Ν πραφματικούς αριθμούς ??
                  double *rev= new double[N];


       //Εύρεση και αποθήκευση της μεγαλύτερης ποιότητας των υποψηφίων γονιών
                  for (i=1;i<N;i++)
            if(max_fs<x[i].fitness)
                  max_fs=x[i].fitness;
```

```
	//Για να αποκλείσουμε την περίπτωση όπου το max_fs θα ειναι 0 προσθέτουμε το 0.1
		s=max_fs+0.1;


		//Αποθήκευση των νέων ποιοτήτων στον πίνακα rev
	for (i=0;i<N;i++)
		rev[i]=s-x[i].fitness;


		//Εύρεση του αθροίσματος των νέων ποιοτήτων
	for (i=0;i<N;i++)
		sum+=rev[i];

		//Εύρεση ενός τυχαίου αριθμού απο 0 εώς sum
	sumrnd=rnd()*sum;

	for (i=0;i<N;i++)
	{
		sum2+=rev[i];	//Αποθήκευση του αθροίσματος των νέων ποιοτήτων
		if(sum2>sumrnd) break; //Αν το νέο άθροισμα ειναι μεγαλύτερο απο τον τυχαίο αριθμό επιστρέφει το συγκεκριμένο άθροισμα.
	}

	if (i==N)
	{
		AnsiString st,st2;
		st.printf("sum=%lf sumrnd=%lf sum2=%lf max=%lf min=%lf s=%lf\n",sum,sumrnd,sum2,max_fs,min_fs,s);
		for (int j=0 ; j<N ; j++) {st2.printf("%lf ",x[j].fitness); st=st+st2; }
		ShowMessage(st);
	}

	//Αποδέσμευση του δείκτη για αποφυγή προβλημάτων στην μνήμη.
	delete [] rev;

	return i;
}




int FindBest(atom *x, int N)
{
	int i,Best=0;
	double BestFit=x[0].fitness;
	for (i=1;i<N;i++)
```

```
        if (x[i].fitness<BestFit)
    {
        BestFit=x[i].fitness;
        Best=i;
    }
    return Best;
}



void UpdateBeliefSpace(atom best, atom BeliefSpace[], int B)
{
    bool flag= false;
    int sum=0;
            //Ελεγχος αν τα γενότυπα του Best υπάρχουν ήδη σε κάποιο άτομο του χώρου μνήμης.
    for(int j=0; j<flagBS; j++)
    {
        for(int k=0; k<VARS; k++)
            if(best.geno[k] == BeliefSpace[j].geno[k])
                sum++;

         if(VARS == sum)
            flag = true;

         sum=0;
    }

   if(flagBS<B & flag==false)    //Γεμίζει και ταξινομεί τον πίνακα
   {
        BeliefSpace[flagBS]=best;
        for(int i = 0; i <flagBS; i++)
            for(int j =0; j<flagBS-i; j++)
                if(BeliefSpace[j].fitness>BeliefSpace[j+1].fitness)
                {
                    atom temp=BeliefSpace[j];
                    BeliefSpace[j]=BeliefSpace[j+1];
                    BeliefSpace[j+1]=temp;
                }
       flagBS++;

   }
   else if(flagBS>=B & flag==false)  //Όταν γεμίσει ο πίνακας κάνει αντικατάσταση
   {
        int i=B-1;
```

```
		while(best.fitness < BeliefSpace[i].fitness && i>=0)
		{
			if(i>0)
				BeliefSpace[i]=BeliefSpace[i-1];
			i--;

		}
		if(i!=B-1)
			BeliefSpace[i+1]=best;
	}




}



int HammDist(int gen1, int gen2)
{
	int dist=0;
	int val=gen1^gen2;
	while(val)
	{
		++dist;
		val &=val-1;
	}
	return dist;
}


int FindDistance(atom best,int point, atom BeliefSpace[], int B)
{
	 int sum =0,min =0,position=-1;
	 int endpoint;
	 if(point<B)
		endpoint=point;
	 else
		endpoint=B;

	 for(int i=0; i<endpoint; i++)
	 {
				//Άθροισμα των διαφορών των γονοτύπων ανάμεσα στην καλύτερη λύση και της αντίστηχης του χώρου μνήμης
		for(int j=0; j<VARS; j++)
			sum=sum+HammDist(best.geno[j],BeliefSpace[i].geno[j]);
```

```
            if(i==0 || sum<min)
            {
                  min=sum;
                  position=i;

            }
            sum=0;
      }

      return position;
}


atom Compare(atom best,int position, int BITS, int fun,atom BeliefSpace[])
{
      atom temp;

      temp=Crossover(best,BeliefSpace[position],BITS);
      Fitness(temp, BITS, fun);

      if(best.fitness<temp.fitness)
            return best;
      else
            return temp;
}


void cleanFlagBS()
{
      flagBS=0;
}



void algorithm()
{


     //Αρχικοποίηση μεταβλητών
      int N=BasicForm->Population->Text.ToInt();
      int g=BasicForm->Generations->Text.ToInt();
      int B=BasicForm->Beliefspace->Text.ToInt();
      int BITS=BasicForm->Bits->Text.ToInt();
      int fun=BasicForm->Problem->ItemIndex;
      double mutation=BasicForm->Mutation->Text.ToDouble();
      int paralagi = BasicForm->Variation->ItemIndex;
```

```
//Δήλωση μεταβλητών
atom* BeliefSpace= new atom[B];
atom *Par= new atom[N];
atom *Pop= new atom[N];

int i,j,Best,child,gen,position,p1,p2;
FILE *f1;

srand((unsigned)time(NULL));

//Αρχικοποίηση του αρχικού πληθυσμού των γονέων
for (i=0;i<N;i++)
      for(j=0;j<VARS;j++)
            Par[i].geno[j]=random(pow(2,BITS));
//Αξιολόγηση των γονέων
for (i=0;i<N;i++)
      Fitness(Par[i], BITS, fun);

//Δημιουργία του αρχείου για τα αποτελέσματα του αλγορίθμου
if(BasicForm->Savefile->Checked)
{
      if(BasicForm->EditName->Text!="")
      {
         String name=BasicForm->EditName->Text + ".txt";
         f1=fopen(name.c_str(),"w");
      }
      else
            f1=fopen("results.txt","w");

      fprintf(f1,"Population: %i    Generations: %i Belief space index: %i \n", N, g, B);
      fprintf(f1,"Number of Bits: %i     Mutation Rate: %f      Function Problem: %s  \n",
BITS, mutation, BasicForm->Problem->Text);
      fprintf(f1,"Variation: %s    ", BasicForm->Variation->Text);

      if(BasicForm->Hillclimb->Checked)
            fprintf(f1,"Hill Climb: YES\n\n");
      else
           fprintf(f1,"Hill Climb: NO\n\n");

      fprintf(f1,"Atoms     Binary0    Binary1    Geno0  Geno1  Pheno0     Pheno1     Fitness
\n\n");
      fprintf(f1,"Arxikos Plithismos\n");
      Display(Par,N,BITS,f1);
}
```

```
        //Η κύρια επανάληψη των γενεών
for (gen=0;gen<g;gen++)
{
    if(BasicForm->Savefile->Checked)
        fprintf(f1,"Generation= %i \n",gen);

    Best=FindBest(Par, N);
    Pop[0]=Par[Best];
                    //Η επανάληψη για την δημιουργία των απογόνων
    for (child=1; child<N; child++)
    {

        //Επιλογή γονέων με την χρήση της Ρουλέτας
        p1=Roulette(Par, N);
        p2=Roulette(Par, N);

        //Ανασυνδυασμός των γονέων
        Pop[child]= Crossover(Par[p1],Par[p2], BITS);

        //Χρήση του τελεστή της Μετάλλαξης στον απόγονο
        Pop[child]= Mutation (Pop[child], mutation, BITS);

        //Περιπτώσεις 2 και 3 σε περίπτωση επιλογής απο τον χρήστη
        if(paralagi==1)
        {
            int position=FindDistance(Pop[child],gen,BeliefSpace,B);
            if (position!=-1)
                Pop[child]=Compare(Pop[child],position, BITS, fun, BeliefSpace);

        }
        else if(paralagi==2)
        {
            position=FindDistance(Pop[child],gen,BeliefSpace,B);
            if (position!=-1)
                Pop[child]=Crossover(Pop[child],BeliefSpace[position], BITS);
        }

        //Αξιολόγηση του απογόνου
        Fitness(Pop[child], BITS, fun);
    }

                    //Βρίσκουμε την θέση του καλύτερου απογόνου
    Best=FindBest(Pop, N);
                    //Χρήση του τελεστή αναρήχησης αν επιλεγεί απο τον χρήστη
    if(BasicForm->Hillclimb->Checked)
        HillClimb(Pop[Best], BITS, fun);
```

```
                    //Εύρεση της θέσης .....
position=FindDistance(Pop[Best],gen,BeliefSpace,B);
if(position!=-1)
      Pop[Best]=Compare(Pop[Best],position, BITS, fun, BeliefSpace);

                    //Γέμισμα ή ενημέρωσει του χώρου μνήμης
UpdateBeliefSpace(Pop[Best],BeliefSpace,B);

                    //Εμφάνιση των αποτελεσμάτων στην δεύτερη φόρμα και στο
αρχείο
if(BasicForm->Savefile->Checked)
{
      Display (Pop, N, BITS, f1);
      fprintf(f1,"Best After Culture Algorithm \n");
}

ViewForm->Table->Cells[0][gen+1]=gen;
for (j=0;j<VARS;j++) {
      if(BasicForm->Savefile->Checked)
            fprintf(f1,"    %s  ",Int2Str(Pop[Best].geno[j],BITS));
      ViewForm->Table->Cells[j+1][gen+1]=Int2Str(Pop[Best].geno[j], BITS);
}
for (j=0;j<VARS;j++)
{
      if(BasicForm->Savefile->Checked)
            fprintf(f1,"%5i ",Pop[Best].geno[j]);
      ViewForm->Table->Cells[3+j][gen+1]=Pop[Best].geno[j];
}

for (j=0; j<VARS; j++)
{
      if(BasicForm->Savefile->Checked)
            fprintf(f1,"%10.7lf  ",Pop[Best].pheno[j]);
      std::stringstream pheno;
      pheno<<std::fixed<<std::setprecision(10)<< Pop[Best].pheno[j];
      ViewForm->Table->Cells[5+j][gen+1]=pheno.str().c_str();
}

if(BasicForm->Savefile->Checked)
      fprintf(f1,"f= %10.7lf \n\n",Pop[Best].fitness);


std::stringstream fitness;
fitness<<std::fixed<<std::setprecision(10)<< Pop[Best].fitness;
ViewForm->Table->Cells[7][gen+1]=fitness.str().c_str();
```

```
            //ViewForm->Table->Cells[7][gen+1]=  Pop[Best].fitness;

            //Αντικατάσταση των γονέων απο τους απογόνους
            for (i=0;i<N;i++)
              Par[i]=Pop[i];

      }

      //Εμφάνιση του τελικού χώρου μνήμης στην δεύτερη φόρμα
      for(i=0; i<flagBS; i++)
            if(BasicForm->Problem->ItemIndex==0)
            {
                  std::stringstream bf1;
                  bf1<<std::fixed<<std::setprecision(10)<< Double(BeliefSpace[i].fitness)*(1);
                  ViewForm->LabelBeliefS->Caption=ViewForm->LabelBeliefS->Caption  +  "\n"
+ bf1.str().c_str();
            }
            else
            {
                  std::stringstream bf;
                  bf<<std::fixed<<std::setprecision(10)<< Double(BeliefSpace[i].fitness);
                  ViewForm->LabelBeliefS->Caption=ViewForm->LabelBeliefS->Caption  +  "\n"
+ bf.str().c_str();
            }
                  //Εμφάνιση του τελικού χώρου μνήμης στο αρχείο
      if(BasicForm->Savefile->Checked)
      {
            fprintf(f1,"Belief Space \n");
            for(i=0; i<flagBS; i++)
                  if(BasicForm->Problem->ItemIndex==0)
                        fprintf(f1,"%10.7lf \n", BeliefSpace[i].fitness*(1));
                  else
                        fprintf(f1,"%10.7lf \n", BeliefSpace[i].fitness);

            fclose(f1);
      }


      //Αποδέσμευση της μνήμης απο τα αντικείμενα
      delete Par;
      delete Pop;
      delete BeliefSpace;
}


//---------------------------------------------------------------------------
```

Αρχείο Functions.h

```
#include <vcl.h>
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <math.h>
#define VARS 2
#define OFFSET 4000
struct atom{
      int geno[VARS];
      double pheno[VARS];
      double fitness;
};


double rnd(void);
atom Mutation (atom x, double prob, int BITS);
atom Crossover (atom p1, atom p2, int BITS);
String Int2Str(int x, int BITS);
int Str2Int(String x, int BITS);
void Geno2Pheno(atom & x, int BITS, int fun);
void Fitness(atom & x, int BITS, int fun);
void HillClimb(atom & x, int BITS, int fun);
void Display(atom * x,int N, int BITS, FILE *f1);
int Roulette(atom * x, int N);
int FindBest(atom *x, int N);
void UpdateBeliefSpace(atom best, atom BeliefSpace[], int B);
int HammDist(int gen1, int gen2);
int FindDistance(atom best,int point, atom BeliefSpace[], int B);
atom Compare(atom best,int position, int BITS, int fun, atom BeliefSpace[]);
void cleanFlagBS();
void algorithm();
```

Αρχείο Algorithm.cpp

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop
//---------------------------------------------------------------------------
```

```
USEFORM("Basic.cpp", BasicForm);
USEFORM("View.cpp", ViewForm);
//---------------------------------------------------------------------------
WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int)
{
	try
	{
		Application->Initialize();
		Application->CreateForm(__classid(TBasicForm), &BasicForm);
		Application->CreateForm(__classid(TViewForm), &ViewForm);
		Application->Run();
	}
	catch (Exception &exception)
	{
		Application->ShowException(&exception);
	}
	catch (...)
	{
		try
		{
			throw Exception("");
		}
		catch (Exception &exception)
		{
			Application->ShowException(&exception);
		}
	}
	return 0;
}
//---------------------------------------------------------------------------
```