ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΓΡΑΦΙΚΗ ΔΙΑΠΡΟΣΩΠΕΙΑ ΓΙΑ ΤΟ ΛΟΓΙΣΜΙΚΟ OPTIMUS

Κωνσταντινόπουλος Δημήτριος

Επιβλέπων: Ιωάννης Τσούλος

Αναπληρωτής καθηγητής

Άρτα, Σεπτέμβρης, 2022

GRAPHICAL INTERFACE FOR THE OPTIMUS SOFTWARE

Εγκρίθηκε από τριμελή εξεταστική επιτροπή

Άρτα, 26/9/22

ΕΠΙΤΡΟΠΗ ΑΞΙΟΛΟΓΗΣΗΣ

1. Επιβλέπων καθηγητής

   Ιωάννης Τσούλος,

2. Μέλος επιτροπής

   Βασίλειος Χαριλόγης,

3. Μέλος επιτροπής

   Αλέξανδρος Τζάλλας

© Κωνσταντινόπουλος, Δημήτριος, 2022.
Με επιφύλαξη παντός δικαιώματος. Allrightsreserved

Δήλωσημηλογοκλοπής

Δηλώνω υπεύθυνα και γνωρίζοντας τις κυρώσεις του Ν. 2121/1993 περί Πνευματικής Ιδιοκτησίας, ότι η παρούσα μεταπτυχιακή εργασία είναι εξ ολοκλήρου αποτέλεσμα δικής μου ερευνητικής εργασίας, δεν αποτελεί προϊόν αντιγραφής ούτε προέρχεται από ανάθεση σε τρίτους. Όλες οι πηγές που χρησιμοποιήθηκαν (κάθε είδους, μορφής και προέλευσης) για τη συγγραφή της περιλαμβάνονται στη βιβλιογραφία.

Κωνσταντινόπουλος, Δημήτριος

Υπογραφή

ΕΥΧΑΡΙΣΤΙΕΣ

Ευχαριστώ πολύ την οικογένειά μου για την οικονομική και ψυχολογική υποστήριξη που μου παρείχαν συνεχώς, καθώς, και τους ελάχιστους καθηγητές/τριες που κατά τα σχολικά και πανεπιστημιακά μου έτη, υπήρξαν υπόδειγμα καθηγητή.

ΠΕΡΙΛΗΨΗ

Η πτυχιακή εργασία αποσκοπεί στην επεξήγηση του γενικού όρου της βελτιστοποίησης και σταδιακά η εισαγωγή του αναγνώστη σε κάποιο ορισμένο βάθος του παραπάνω όρου. Εκεί θα συναντήσει υποκατηγορίες και μεθόδους της βελτιστοποίησης, παρέχοντας τα αντίστοιχα παραδείγματα για περαιτέρω κατανόηση των όρων. Τέλος, παρουσιάζεται και το προγραμματιστικό κομμάτι της πτυχιακής που είναι μία εφαρμογή γραφικού περιβάλλοντος, όπου ο χρήστης μπορεί να παράξει τα δικά του αποτελέσματα μέσω ορισμένων μεθόδων βελτιστοποίησης που παρέχονται από την ίδια την εφαρμογή.

Λέξεις-κλειδιά: Βελτιστοποίηση, Μέθοδοι βελτιστοποίησης, Τοπική βελτιστοποίηση, Καθολική βελτιστοποίηση, Βελτιστοποίηση προβλημάτων.

ABSTRACT

This thesis aims at the explanation of the general term of optimization and gradually the insertion of the reader to a certain depth of the above term. This is where the reader will find subcategories and optimization methods, being provided with the respective paradigms for further understanding of terms. Lastly, the programming part of the thesis is presented, which is a graphical interface, where the users can produce their own results via certain optimization methods provided by the application.

Keywords: Optimization methods, Optimization problems, Local optimization, Global optimization, Optimization.

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

ΕΥΧΑΡΙΣΤΙΕΣ 6

ΠΕΡΙΛΗΨΗ 7

ABSTRACT 8

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ 9

1. Εισαγωγή 1

1.1 Παρούσα εργασία 1

1.2 Βελτιστοποίηση 1

1.2.1 Μεταβλητές σχεδιασμού 2

1.2.2 Αντικειμενική συνάρτηση 2

1.2.3 Συναρτήσεις περιορισμών 3

1.3 Τοπική βελτιστοποίηση 4

1.4 Καθολική βελτιστοποίηση 7

1.4.1 Χαρακτηριστικά μεθόδων καθολικής βελτιστοποίησης 8

1.4.2 Κατηγορίες μεθόδων καθολικής βελτιστοποίησης 10

1.5 Παραδείγματα προβλημάτων βελτιστοποίησης 23

1.5.1 Το πρόβλημα των 8 βασιλισσών 23

1.5.2 Το πρόβλημα του πλανόδιου πωλητή 24

2 Μέθοδοι Βελτιστοποίησης 24

2.1 Διαχωρισμός μεθόδων βελτιστοποίησης 24

2.1.1 Μαθηματικές μέθοδοι 25

2.1.2 Μεταευτερικές μέθοδοι 26

2.2 Η μέθοδος Gradient Descent 27

2.3 Η μέθοδος BFGS 30

2.4 Γενετικοί αλγόριθμοι 31

2.4.1 Επιλογή 32

2.4.2 Διασταύρωση 33

2.4.3 Μετάλλαξη 34

2.5 Βελτιστοποίηση σμήνους σωματιδίων (PSO) 36
3. Το λογισμικό OPTIMUS 39
3.1 Μεταγλώττιση 39
3.2 Κωδικοποίηση προβλημάτων 40
3.3 Παράμετροι προβλημάτων 41
3.4 Κωδικοποίηση μεθόδου βελτιστοποίησης 42
4. Η εφαρμογή 43
4.1 Εισαγωγή 43
4.2 Το μενού File 44
4.3 Παράδειγμα χρήσης της εφαρμογής 47
5. Συμπεράσματα 53
Παράρτημα: Κώδικας εφαρμογής 53
Βιβλιογραφία 57

ΠΙΝΑΚΑΣ ΕΙΚΟΝΩΝ

ΕΙΚΟΝΑ 1.1 ΣΧΗΜΑΤΙΚΗ ΑΝΑΠΑΡΑΣΤΑΣΗ ΜΙΑΣ "GRADIENT-BASED" ΜΕΘΟΔΟΥ 5
ΕΙΚΟΝΑ 1.2 ΓΡΑΦΙΚΗ ΑΝΑΠΑΡΑΣΤΑΣΗ ΑΝΑΖΗΤΗΣΗΣ HILL-CLIMBING 6
ΕΙΚΟΝΑ 1.3 ΓΡΑΦΙΚΗ ΑΝΑΠΑΡΑΣΤΑΣΗ ΤΕΧΝΙΚΗΣ BRANCH AND BOUND 8
ΕΙΚΟΝΑ 1.4-1.5 ΟΠΤΙΚΗ ΑΝΑΠΑΡΑΣΤΑΣΗ ΔΗΜΙΟΥΡΓΙΑΣ ΥΠΟΣΥΝΟΛΩΝ ΑΠΟ ΤΗΝ BRANCH AND BOUND 9
ΕΙΚΟΝΑ 1.6 ΟΠΤΙΚΗ ΑΠΕΙΚΟΝΙΣΗ ΤΗΣ DIRECT ΜΕΘΟΔΟΥ ΓΙΑ 3 ΕΠΑΝΑΛΗΨΕΙΣ 12
ΕΙΚΟΝΑ 1.7 Α) ΤΥΧΑΙΑ ΑΡΧΙΚΗ ΚΑΤΑΣΤΑΣΗ,**Β)** ΜΙΑ ΑΠΟ ΤΙΣ ΠΙΘΑΝΕΣ ΛΥΣΕΙΣ 24
ΕΙΚΟΝΑ 2.1 ΕΠΑΝΑΛΑΜΒΑΝΟΜΕΝΑ ΒΗΜΑΤΑ ΠΡΟΣ ΛΟΦΟ ΜΕ ΟΠΤΙΚΗ ΑΠΟ ΠΑΝΩ ΠΡΟΣ ΤΑ ΚΑΤΩ 29
ΕΙΚΟΝΑ 2.2 Α)ΑΝΑΖΗΤΗΣΗ ΜΕ ΜΕΓΑΛΟ LEARNING RATE, **Β)** ΑΝΑΖΗΤΗΣΗ ΜΕ ΜΙΚΡΟ LEARNING RATE 30
ΕΙΚΟΝΑ 2.3 ΤΕΧΝΙΚΗ ΑΝΑΖΗΤΗΣΗΣ ΤΗΣ ΜΕΘΟΔΟΥ ΒΣΣ ΣΕ ΠΟΛΥΔΙΑΣΤΑΣΤΟ ΧΩΡΟ ΑΝΑΖΗΤΗΣΗΣ 38
ΕΙΚΟΝΑ 4.1 ΑΡΧΙΚΟ ΠΑΡΑΘΥΡΟ ΤΗΣ ΕΦΑΡΜΟΓΗΣ ΚΑΤΑ ΤΗΝ ΕΚΤΕΛΕΣΗ ΤΗΣ 44
ΕΙΚΟΝΑ 4.2 ΕΜΦΑΝΙΣΗ ΕΠΙΛΟΓΩΝ ΤΟΥ ΜΕΝΟΥ FILE 45
ΕΙΚΟΝΑ 4.3 ΕΜΦΑΝΙΣΗ ΛΙΣΤΑΣ ΜΕΘΟΔΩΝ ΒΕΛΤΙΣΤΟΠΟΙΗΣΗΣ 46
ΕΙΚΟΝΑ 4.4 ΕΜΦΑΝΙΣΗ ΑΝΤΙΣΤΟΙΧΟΥ ΠΑΡΑΘΥΡΟΥ ΚΑΤΑ ΤΗΝ ΕΠΙΛΟΓΗ ΤΗΣ ΜΕΘΟΔΟΥ GENMIN ΑΠΟ ΤΗ ΛΙΣΤΑ ΜΕΘΟΔΩΝ 47
ΕΙΚΟΝΑ 4.5 ΕΙΔΟΠΟΙΗΣΗ ΤΟΥ ΧΡΗΣΤΗ ΓΙΑ ΤΗΝ ΕΠΙΤΡΕΠΤΗ ΕΠΙΛΟΓΗ ΑΡΧΕΙΟΥ ΠΡΟΒΛΗΜΑΤΟΣ 47
ΕΙΚΟΝΑ 4.6 ΕΜΦΑΝΙΣΗ ΠΛΗΡΟΦΟΡΙΩΝ ΓΙΑ ΤΟ ΕΠΙΛΕΓΜΕΝΟ ΠΡΟΒΛΗΜΑ ΒΕΛΤΙΣΤΟΠΟΙΗΣΗΣ 48
ΕΙΚΟΝΑ 4.7 ΔΙΑΛΟΓΟΣ ΕΝΗΜΕΡΩΣΗΣ ΓΙΑ ΕΠΙΛΟΓΗ ΠΡΟΒΛΗΜΑΤΟΣ ΒΕΛΤΙΣΤΟΠΟΙΗΣΗΣ 49
ΕΙΚΟΝΑ 4.8 ΔΙΑΛΟΓΟΣ ΠΡΟΕΙΔΟΠΟΙΗΣΗΣ ΚΑΤΑ ΤΟ ΚΛΕΙΣΙΜΟ ΠΑΡΑΘΥΡΟΥ ΠΑΡΑΜΕΤΡΩΝ 49
ΕΙΚΟΝΑ 4.9 ΔΙΑΛΟΓΟΣ ΕΝΗΜΕΡΩΣΗΣ ΤΟΥ ΧΡΗΣΤΗ ΚΑΤΑ ΤΗΝ ΕΛΕΙΨΗ ΣΥΜΠΛΗΡΩΣΗΣ ΟΛΩΝ ΤΩΝ ΠΕΔΙΩΝ ΤΗΣ ΜΕΘΟΔΟΥ 50
ΕΙΚΟΝΑ 4.10 ΜΗΝΥΜΑ ΕΝΗΜΕΡΩΣΗΣ ΓΙΑ ΤΗΝ ΕΠΙΤΥΧΗΜΕΝΗ ΕΝΗΜΕΡΩΣΗ ΤΩΝ ΠΑΡΑΜΕΤΡΩΝ 50
ΕΙΚΟΝΑ 4.11 ΜΗΝΥΜΑ ΕΝΗΜΕΡΩΣΗΣ ΓΙΑ ΤΗΝ ΕΛΕΙΨΗ ΣΥΜΠΛΗΡΩΣΗΣ ΠΕΔΙΩΝ 50
ΕΙΚΟΝΑ 4.12 ΑΠΟΤΕΛΕΣΜΑ ΕΚΤΕΛΕΣΗΣ ΠΡΟΒΛΗΜΑΤΟΣ ΜΕ ΤΗ ΜΕΘΟΔΟ BFGS 51
ΕΙΚΟΝΑ 4.13 ΕΜΦΑΝΙΣΗ ΠΕΡΙΟΡΙΣΜΕΝΩΝ ΕΠΙΛΟΓΩΝ ΜΕΤΑ ΤΗΝ ΕΚΤΕΛΕΣΗ ΤΟΥ ΠΡΟΓΡΑΜΜΑΤΟΣ 52

1. Εισαγωγή

1.1 Παρούσα εργασία

Σκοπός της παρούσας εργασίας είναι η κατανόηση της ιδέας πίσω από τον γενικό όρο της βελτιστοποίησης, συμπεριλαμβανομένων κάποιων κλάδων της, καθώς και η ανάλυση ορισμένων μεθόδων που ανήκουν στους κλάδους αυτούς. Μετά την απαραίτητη πληροφόρηση, τον ορισμό και τον τρόπο λειτουργίας και φιλοσοφίας των παραπάνω, η εργασία προχωρά στο κομμάτι του κύριου αντικειμένου της, με την παρουσίαση μίας εφαρμογής διαπροσωπείας, μέσω της οποίας ο χρήστης μπορεί να χειριστεί αρκετές μεθόδους βελτιστοποίησης, για την επίλυση διαφόρων προβλημάτων. Ο κάθε χρήστης θα μπορεί εύκολα να επιλέξει τη μέθοδο και το πρόβλημα βελτιστοποίησης που επιθυμεί, έχοντας την επιπρόσθετη ευκολία εισαγωγής των απαραίτητων παραμέτρων για κάθε κατάσταση ξεχωριστά. Η εφαρμογή μπορεί να δεχθεί και έγκυρα προβλήματα προς λύση που επιθυμεί να εισάγει ο χρήστης, ενώ κατά την ολοκλήρωση, η εφαρμογή παρέχει αναλυτικά τα αποτελέσματα της διαδικασίας βελτιστοποίησης.

1.2 Βελτιστοποίηση

Βελτιστοποίηση, ως γενική έννοια, ονομάζεται η τεχνική εύρεσης καλύτερων (βέλτιστων) αποτελεσμάτων για προβλήματα βελτιστοποίησης που μπορούν να βρεθούν σε πολλά επιστημονικά πεδία όπως είναι η φυσική, η βιολογία, η οικονομία κτλ [1]. Πιο συγκεκριμένα, η βελτιστοποίηση είναι η τεχνική εύρεσης του σημείου x στο οποίο η συνάρτηση f(x) παίρνει την βέλτιστη τιμή της (ελάχιστη για προβλήματα ελαχιστοποίησης ή μέγιστη για προβλήματα μεγιστοποίησης αντίστοιχα), καθώς και την τιμή αυτή. Το σημείο x είναι ένα n-διάστατο διάνυσμα(π.χ. $x = [x_1, x_2, \ldots, x_n]^T \in R^n$) στις n-διαστάσεις $\Omega \subseteq R^n$, με R^n να είναι ο n-διάστατοςΕυκλείδειος χώρος. Η συνάρτηση f(x) της οποίας ψάχνουμε το ελάχιστο ή μέγιστο ονομάζεται αντικειμενική συνάρτηση. Το πρόβλημα της βελτιστοποίησης θα μπορούσε να κατηγοριοποιηθεί ανάλογα με τα χαρακτηριστικά του, βάσει της συνάρτησης ως εξής:

- Αν είναι διαφορίσιμη ή μη διαφορίσιμη.
- Αν είναι γραμμική ή μη γραμμική.
- Αν το πρόβλημα προς βελτιστοποίηση έχει ή όχι περιορισμούς (άρα δηλαδή όταν το σύνολο $\Omega = R^n$ ή $\Omega \subset R^n$ αντίστοιχα).
- Αν ο σκοπός εύρεσης είναι για τοπικό ή ολικό ελάχιστο/μέγιστο.

Έτσι, ανάλογα με το πρόβλημα που προκύπτει χρησιμοποιείται και ο αντίστοιχα κατάλληλος αλγόριθμος. Να σημειωθεί ότι δεν υπάρχει αλγόριθμος που να λύνει όλα τα προβλήματα βελτιστοποίησης αποτελεσματικά. Με σκοπό λοιπόν της μεθόδου βελτιστοποίησης να είναι η εύρεση του ολικού ελαχίστου/μεγίστου, για αντικειμενικές συναρτήσεις που είναι είτε γραμμικές είτε μη γραμμικές, αλλά κυρτές, η εύρεσή του είναι γρήγορη και σίγουρη (όταν αυτό υπάρχει). Όταν όμως η αντικειμενική συνάρτηση

είναι μη κυρτή, εμφανίζεται ένας αριθμός τοπικών ελαχίστων/μεγίστων εκ των οποίων ένα είναι το ολικό. Στην τελευταία περίπτωση υπάρχει κίνδυνος οι μέθοδοι βελτιστοποίησης να εγκλωβιστούν σε περιοχή τοπικού ακροτάτου ή να ξοδέψουν χρόνο και υπολογιστικούς πόρους έως ότου καταφέρουν να απεγκλωβιστούν μετά από έναν μεγάλο αριθμό άσκοπων επαναλήψεων. [18]

1.2.1 Μεταβλητές σχεδιασμού

Μεταβλητές σχεδιασμού ονομάζονται οι παράμετροι που αποτελούν έναν σχεδιασμό. Ο σχεδιασμός αυτός μπορεί να θεωρηθεί εφικτός (feasible design), αν πληροί τις προϋποθέσεις των περιορισμών του προβλήματος ή ανέφικτος (unfeasible design), αν όχι. Κάθε εφικτός προγραμματισμός αποτελεί την κάθε δυνατή λύση του προβλήματος προς βελτιστοποίηση, με τον καλύτερο σχεδιασμό να είναι αυτός που βελτιστοποιεί (ελαχιστοποιεί ή μεγιστοποιεί) τη συνάρτηση. Οι μεταβλητές σχεδιασμού μπορεί να είναι συνεχείς, ακέραιες ή διακριτές.

- Συνεχείς μεταβλητές ονομάζονται οι πραγματικές. Το σύνολο τιμών X_i που δέχεται συνεχείς μεταβλητές x_i χαρακτηρίζεται ως διάστημα πραγματικών αριθμών, και έχει τη μορφή, $L_i \leq x_i \leq U_i$, όπου L_i και U_i είναι το κάτω (Lower) και άνω (Upper) όριο αντίστοιχα.
- Οι ακέραιες μεταβλητές χρησιμοποιούνται κυρίως στον ακέραιο προγραμματισμό. Ειδική κατηγορία των ακέραιων μεταβλητών, είναι όταν ονομάζονται δυαδικές, καθώς μπορούν να λάβουν μόνο τις τιμές 0 και 1.
- Οι διακριτές μεταβλητές λαμβάνουν τιμές από πεπερασμένα σύνολα X_i, είτε πραγματικά είτε ακέραια.

Ένα πρόβλημα, μπορεί βέβαια να χρησιμοποιεί διαφορετικά ήδη μεταβλητών. Συνήθως τέτοια προβλήματα είναι μεικτά και μετατρέπονται σε συνεχή πριν την επίλυσή τους. Η ορθή επιλογή του σωστού είδους μεταβλητών είναι πολύ σημαντική, εφόσον επηρεάζει την αποδοτικότητα της βελτιστοποίησης, όπως με την αύξηση του υπολογιστικού κόστους. Για την επίτευξη εύκολης ανάλυσης ευαισθησίας των μεταβλητών, γνωστή τεχνική είναι η ομαδοποίηση των μεταβλητών σχεδιασμού. Έτσι μπορεί να μειωθεί σημαντικά το υπολογιστικό κόστος και ο απαιτούμενος χρόνος επίλυσης. [18]

1.2.2 Αντικειμενική συνάρτηση

Η αντικειμενική συνάρτηση είναι το μέσο με το οποίο μπορεί κανείς να αξιολογήσει τον κάθε εφικτό σχεδιασμό ώστε να καταλήξει στον βέλτιστο. Σκοπός λοιπόν του κάθε αλγορίθμου βελτιστοποίησης είναι να λαμβάνει την τιμή της αντικειμενικής συνάρτησης επιλύοντάς την με κάθε μεταβλητή σχεδιασμού. Για προβλήματα ελαχιστοποίησης, δεκτή γίνεται η αντικειμενική συνάρτηση F(x) με τη μικρότερη δυνατή τιμή (και κατ' επέκταση η μεταβλητή σχεδιασμού που χρησιμοποιήθηκε), ενώ για προβλήματα μεγιστοποίησης, δεκτή γίνεται η αντικειμενική συνάρτηση με τη μεγαλύτερη τιμή, δηλαδή η –F(x). Σε ένα πρόβλημα μπορεί να χρησιμοποιείται μία, ή παραπάνω από μία

αντικειμενικές συναρτήσεις για την εύρεση της βέλτιστης λύσης. Στην πρώτη περίπτωση το πρόβλημα αυτό ανήκει στα προβλήματα βελτιστοποίησης μίας αντικειμενικής συνάρτησης (Single Objective Optimization), ενώ στη δεύτερη, το πρόβλημα ανήκει στα προβλήματα βελτιστοποίησης πολλών αντικειμενικών συναρτήσεων (Multi Objective Optimization), και εφόσον η κάθε τιμή της κάθε συνάρτησης μπορεί να επηρεάζει την τιμή την άλλης, ζητούμενο είναι να βρεθεί ένα σύνολο βέλτιστων λύσεων, που συνολικά παρουσιάζει την καλύτερη συμπεριφορά ως προς όλες τις αντικειμενικές συναρτήσεις. [18]

1.2.3 Συναρτήσεις περιορισμών

Κάθε μεταβλητή ενός προβλήματος μπορεί να πάρει ορισμένες τιμές, μεσώ του πεδίου τιμών που ορίζεται για το συγκεκριμένο αυτό πρόβλημα. Παρόλα αυτά, ακόμη και με τις επιτρεπόμενες τιμές, υπάρχει η περίπτωση κάποιοι συνδυασμοί μεταβλητών να οδηγούν είτε σε μη λογικές, είτε σε μη επιτρεπόμενες λύσεις. Για να αποφευχθεί κάτι τέτοιο, είναι απαραίτητο να εφαρμοστούν κάποιοι περιορισμοί στο πρόβλημα βελτιστοποίησης. Οι περιορισμοί αυτοί μπορεί να είναι είτε ανισοτικοί ($g_j(x) \leq 0$) ή ισοτικοί ($h_j(x) = 0$). Ένας σχεδιασμός που υπακούει σε όλους τους περιορισμούς ονομάζεται εφικτός (feasible design) και αποτελεί πλέον μία υλοποιήσιμη λύση του προβλήματος, χωρίς να είναι απαραίτητα η βέλτιστη. Αντιθέτως, όταν ένας σχεδιασμός δεν υπακούει σε όλους τους περιορισμούς, ονομάζεται ανέφικτος σχεδιασμός (infeasible design) και απορρίπτεται από τον αλγόριθμο. Σημαντικό είναι οι συναρτήσεις περιορισμών που θα δημιουργηθούν να είναι αυστηρά οι απαραίτητες, καθώς αυξάνεται αρκετά το υπολογιστικό κόστος. Επίσης, αν οι συναρτήσεις περιορισμών είναι αρκετά αυστηρές με πολύ στενά περιθώρια ικανοποίησής τους, ή λόγω πλήθους, υπάρχουν περιορισμοί που αλληλοσυγκρούονται, όπου η ικανοποίηση του ενός περιορισμού οδηγεί στην παραβίαση του άλλου, τότε το πρόβλημα προς βελτιστοποίηση καταντά άλυτο. Η μείωση του περιττού κόστους, η προσθήκη μεγαλύτερης ελαστικότητας και η αναθεώρηση των αλληλοσυγκρουόμενων περιορισμών, κρίνεται απαραίτητη πριν την εφαρμογή της μεθόδου βελτιστοποίησης.

Οι ανισοτικές συναρτήσεις περιορισμών, μπορεί να είναι ενεργές, ανενεργές ή να παραβιάζονται. Αν δηλαδή ισχύει, $g_j(x) = 0$, η συνάρτηση θεωρείται ενεργή, αν ισχύει $g_j(x) < 0$, η συνάρτηση θεωρείται ανενεργή, ενώ αν δεν ισχύει καν η αρχική ανισότητα, δηλαδή $g_j(x) > 0$, τότε η συνάρτηση θεωρείται ότι παραβιάζεται. Με παρόμοια λογική, οι ισοτικές συναρτήσεις περιορισμών, μπορεί να είναι ενεργές αν ισχύει $h_j(x) = 0$, ή να παραβιάζονται αν δεν ισχύει η ισότητα, δηλαδή $h_j(x) \neq 0$. Οι ανισοτικές συναρτήσεις περιορισμών, σε αντίθεση με τις ισοτικές, μπορούν ανά βήμα βελτιστοποίησης, να ενεργοποιούνται και να απενεργοποιούνται. Για να μειωθεί το υπολογιστικό κόστος για τον υπολογισμό των τιμών όλων των περιορισμών σε κάθε βήμα, αρκετοί αλγόριθμοι βελτιστοποίησης ελέγχουν κάθε φορά ποιοι αλγόριθμοι είναι ανενεργοί. Αγνοώντας τους, καταλήγουν να υπολογίζουν μόνο τους περιορισμούς που είναι είτε ενεργοί, είτε παραβιάζονται, αυξάνοντας έτσι την αποδοτικότητά τους. Ένα παράδειγμα ανεκτής

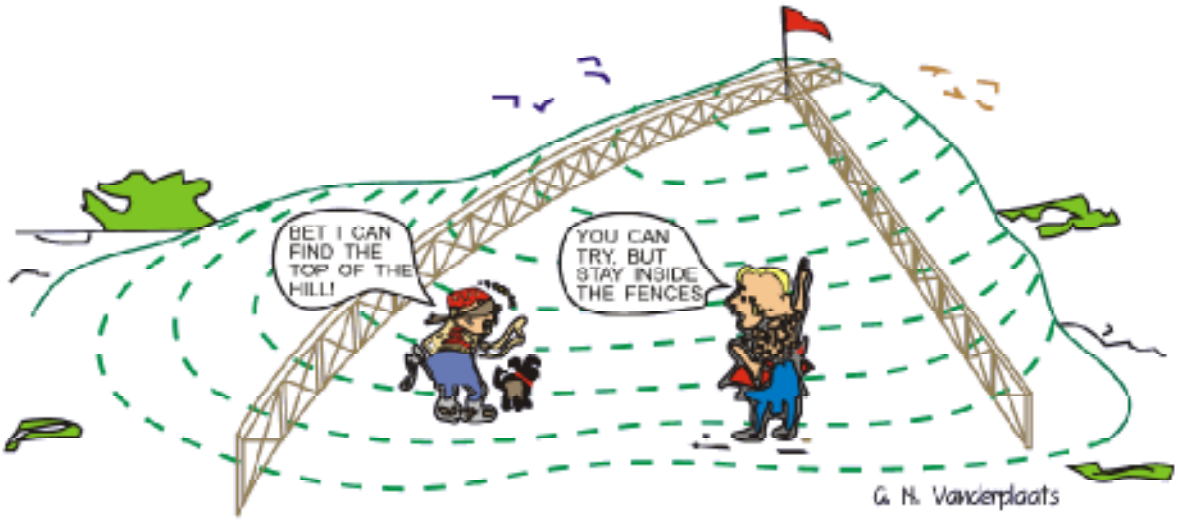
ελαστικότητας, είναι οι ανισοτικές και οι ισοτικές συναρτήσεις περιορισμών, εφόσον είναι πρακτικά σπάνιο να ισούνται με το μηδέν, να θεωρούνται ενεργές όταν η τιμή τους είναι περίπου μηδέν ($g_j(x) \approx 0$ και $h_j(x) \approx 0$ αντίστοιχα). [18]

1.3 Τοπική βελτιστοποίηση

Το πρόβλημα της τοπικής βελτιστοποίησης είναι η εύρεση εκείνου του σημείου (ονομαζόμενο και τοπικός ελαχιστοποιητής/μεγιστοποιητής), έστω το διάνυσμα x* που παίρνει τιμές στον χώρο R^n, τέτοιο ώστε, για κάθε άλλο διάνυσμα x που βρίσκεται εντός μίας περιοχής γύρω από το x*, έστω S, όπου $S \subset R^n$, να ισχύει:

$$f(x^*) \leq f(x) \; \forall \; x \in S$$

Οι πιο πολλές μέθοδοι τοπικής βελτιστοποίησης βασίζονται στη λογική της κλίσης (gradient-based), όπου όπως αναφέρει και το όνομα, οι τεχνικές αυτές εκμεταλλεύονται τις πληροφορίες κλίσης για να βρουν τη βέλτιστη λύση. Αυτές οι τεχνικές χρησιμοποιούνται ευρέως, καθώς είναι αποδοτικές όσον αφορά τις συναρτήσεις που πρέπει να κληθούν μέχρι να βρεθεί το ακρότατο, μπορούν να λύσουν προβλήματα με μεγάλο πλήθος μεταβλητών σχεδιασμού και απαιτούν μικρή ρύθμιση των παραμέτρων του προβλήματος. Παρόλα αυτά, οι μέθοδοι αυτές παρουσιάζουν και κάποια αρνητικά, όπως, του ότι μπορούν να εντοπίσουν μόνο τοπικό ακρότατο, δυσκολεύονται στο να λύσουν διακριτά προβλήματα βελτιστοποίησης, είναι αρκετά περίπλοκοι αλγόριθμοι, κάτι που τους κάνει δύσκολο να εφαρμοστούν αποδοτικά, και κάποιοι από αυτούς μπορεί να είναι επιρρεπείς σε αριθμητικό θόρυβο. Η γενική ιδέα είναι πως μία μέθοδος τοπικής βελτιστοποίησης παράγει αποτελέσματα που εξαρτώνται από ένα αρχικό σημείο. Οι μέθοδοι αυτές δηλαδή, έχουν άμεση εξάρτηση από την ύπαρξη μίας τουλάχιστον αρχικής παραμέτρου. Παράδειγμα είναι η gradient descent, που θα αναλυθεί σε επόμενο κεφάλαιο, και η οποία χρειάζεται μία παράγωγο σε μορφή κλίσης. Οι τεχνικές κλίσης είναι κατάλληλες για συνεχείς αντικειμενικές συναρτήσεις πράγμα που σημαίνει ότι γίνονται ανεπαρκείς όταν υπάρχουν ασυνέχειες στην επιφάνεια στην οποία η κλίση υπολογίζεται. Οι "gradient-based" μέθοδοι ουσιαστικά, εφαρμόζουν μία διαδικασία δύο βημάτων για να βρουν το ακρότατο. Αυτή η διαδικασία δύο βημάτων μπορεί να φανεί στην εικόνα 1.1 όπου ένα παιδί με κλειστά μάτια βρίσκεται σε έναν λόφο. Σκοπός του παιδιού είναι να φτάσει το ψηλότερο σημείο του λόφου, όσο μένει μέσα στα όρια του φράκτη. Οι μεταβλητές σχεδιασμού είναι οι x και y συντεταγμένες για το παιδί. Να σημειωθεί ότι το παιδί μπορεί να ξεκινήσει και έξω από τους φράκτες.



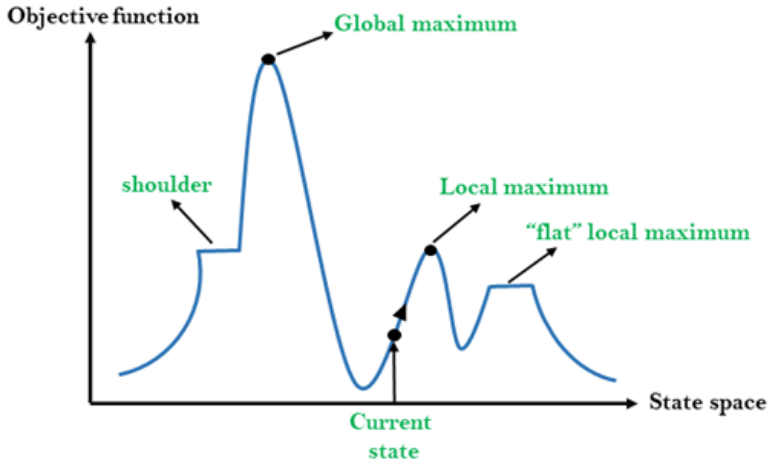
Εικόνα 1.1 Σχηματική αναπαράσταση μίας "gradient-based" μεθόδου ([28])

Η αναλογία του παιδιού στην παραπάνω εικόνα με την "gradient-based" βελτιστοποίηση, είναι πως το παιδί μπορεί να φτάσει την κορυφή του λόφου με το να κάνει ένα βήμα στον άξονα x και ένα βήμα στον άξονα y κάθε φορά. Τότε, βασιζόμενο στις πληροφορίες που σύλλεξε από αυτά τα δύο βήματα, μπορεί να θεωρήσει ποια κατεύθυνση θα μπορούσε να το οδηγήσει στην κορυφή. Τότε το παιδί αρχίζει να κινείται προς αυτή την κατεύθυνση μέχρι να φτάσει σε ένα σημείο που, είτε όσο κινείται η πρόοδος του μένει στάσιμη, είτε φτάσει σε όριο του φράκτη. Εκεί κάνει άλλα δύο μικρά βήματα και για να συλλέξει πάλι πληροφορίες κλίσης και να μπορεί πάλι να θεωρήσει μία νέα κατεύθυνση για να κινηθεί προς αυτήν. Συνεχίζει αυτή τη διαδικασία μέσα στα όρια των φρακτών μέχρι να βρει την κορυφή του λόφου. Η παραπάνω μέθοδος δύο βημάτων μπορεί να εκφραστεί μαθηματικά παρακάτω:

$$x^q = x^{q-1} + a * S^q,$$

όπου το πρώτο βήμα είναι για να χρησιμοποιηθούν οι πληροφορίες κλίσης για να βρεθεί η κατεύθυνση αναζήτησης S, και το δεύτερο βήμα είναι να συνεχιστεί η αναζήτηση προς αυτή την κατεύθυνση μέχρι να μην υπάρχει άλλη βελτίωση. Το δεύτερο βήμα είναι γνωστό και ως μονοδιάστατη αναζήτηση ή αναζήτηση γραμμής, και προσφέρει το βέλτιστο μέγεθος βήματος (step size), a. Να σημειωθεί βέβαια, πως υπάρχουν και "gradient-based" μέθοδοι που δε στηρίζονται στη μονοδιάστατη αναζήτηση. [1][4][18][28] Παρακάτω αναλύονται κάποιες μέθοδοι τοπικής αναζήτησης.

- *Μέθοδος Αναρρίχησης λόφου (Hill-Climbing)*:
  Η αναρρίχηση λόφου είναι μία μέθοδος τοπικής αναζήτησης, κατάλληλη για βελτιστοποίηση μαθηματικών προβλημάτων, και σκοπός της είναι η εύρεση της κορυφής του λόφου (είτε τοπική, είτε ολική βέλτιστη λύση), ξεκινώντας με μία τυχαία αρχική λύση και ακολουθώντας κάθε φορά ένα καλύτερο σημείο. Χαρακτηρίζεται και ως μία από τις άπληστες μεθόδους τοπικής αναζήτησης, καθώς η λογική που ακολουθεί είναι απλά ένας βρόγχος που μετακινείται συνεχώς προς την κατεύθυνση της αυξανόμενης τιμής που περιέχουν γειτονικά σημεία, ελπίζοντας ότι θα οδηγηθεί στην κορυφή, όπως φαίνεται και στην παρακάτω εικόνα 1.2, όπου ο άπληστος αλγόριθμος ακολουθεί τυφλά την γειτονική καλύτερη τιμή με αποτέλεσμα να βρει το τοπικό και όχι το ολικό ακρότατο. Ο αλγόριθμος τερματίζει όταν βρεθεί σε ένα σημείο που κανένα γειτονικό του σημείο δεν έχει καλύτερη τιμή.



Εικόνα1.2 Γραφική αναπαράσταση αναζήτησης Hill-Climbing ([37])

Παρακάτω φαίνεται η μεθοδολογία του αλγορίθμου της μέθοδος αναρρίχησης λόφου:

1. Εκτίμηση της αρχικής κατάστασης, αν η κατάσταση είναι και η επιθυμητή, επιστροφή αποτελέσματος και τέλος αλγορίθμου.
2. Επανάληψη μέχρι να βρεθεί η βέλτιστη λύση, ή να μην υπάρχουν άλλοι διαθέσιμοι τελεστές για να εφαρμοστούν.
3. Επιλογή και εφαρμογή καινούριου τελεστή στην τρέχουσα κατάσταση
4. Έλεγχος νέας κατάστασης
   I. Αν η νέα κατάσταση είναι και η επιθυμητή, επιστροφή αποτελέσματος και τέλος αλγορίθμου
   II. Αλλιώς, αν η νέα κατάσταση είναι καλύτερη από την προηγούμενη, τότε ορισμός της νέας κατάστασης ως την τρέχουσα κατάσταση

III. Αλλιώς, αν η νέα κατάσταση είναι χειρότερη από την προηγούμενη, τότε επιστροφή στο βήμα 2.

5. Έξοδος. [37][38]

- *Μέθοδος τοπικής ακτινικής αναζήτησης*

  Η τοπική ακτινική αναζήτηση διαφέρει αρκετά από τη μέθοδο αναρρίχησης λόφου, που αναπτύχθηκε παραπάνω. Αρχικά, αυτό συμβαίνει γιατί η μέθοδος αυτή χειρίζεται k καταστάσεις αντί για μία, τις οποίες κιόλας κρατά στη μνήμη της, επιτρέποντάς της τη δυνατότητα οπισθοδρόμησης (backtracking). Ο αλγόριθμος ξεκινά παράγοντας k τυχαίες καταστάσεις με τους διαδόχους τους, ελέγχοντας κάθε φορά αν κάποια από τις καταστάσεις αυτές είναι και ο επιθυμητός στόχος. Αν ναι, τότε ο αλγόριθμος σταματά. Διαφορετικά, επιλέγονται οι k καλύτεροι διάδοχοι, από τους οποίους παράγονται οι δικοί τους διάδοχοι, ενώ πριν την παραγωγή των νεών διαδόχων, οι γείτονες κόμβοι επικοινωνούν μεταξύ τους σε κάθε επίπεδο που βρίσκεται ο αλγόριθμος, για να ανταλλάξουν σημαντικές πληροφορίες που θα στρέψουν την αναζήτηση εκεί που υπάρχει μεγαλύτερη πρόοδος. Η επαναληπτική αυτή διαδικασία συνεχίζει, είτε μέχρι να ικανοποιηθεί ο επιθυμητός στόχος, είτε μέχρι να ικανοποιηθεί κάποιος περιορισμός τερματισμού. [38][39]

1.4 Καθολική βελτιστοποίηση

Το πρόβλημα της καθολικής βελτιστοποίησης περιλαμβάνει παρόμοια λογική με την τοπική βελτιστοποίηση. Η μόνη διαφορά κατά την εύρεση εκείνου του σημείου (ονομαζόμενο και καθολικός ελαχιστοποιητής/μεγιστοποιητής), έστω το διάνυσμα x*, αφορά την αντικειμενική συνάρτηση να παρουσιάζει ολικό ελάχιστο/μέγιστο και όχι τοπικό. Δυστυχώς όμως, λίγες μέθοδοι έχουν μελετηθεί για την εύρεση ολικού ελαχίστου σε σύγκριση με την πληθώρα μεθόδων που υπάρχουν για την εύρεση τοπικού ελαχίστου. Στόχος λοιπόν της καθολικής βελτιστοποίησης είναι η εύρεση αυτού του διανύσματος x* που παίρνει τιμές στο χώρο R^n, ώστε για κάθε άλλο διάνυσμα x, να ισχύει:

$$f(x^*) \leq f(x) \; \forall \; x \in R^n$$

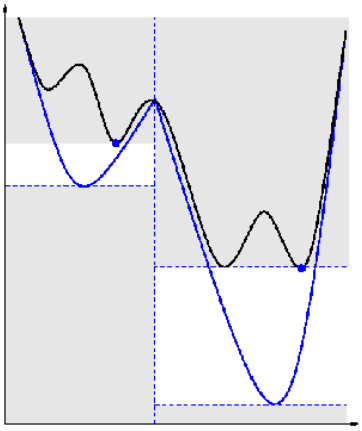
Μία μέθοδος καθολικής βελτιστοποίησης παράγει αποτελέσματα που εξαρτώνται από τις αρχικές συνθήκες. Εξαρτώνται δηλαδή κυρίως από τους φραγμούς στο χώρο των λύσεων. Αν δεν υπάρχουν φραγμοί οι μέθοδοι αυτές γίνονται αποτελεσματικές καθώς αντιμετωπίζουν έναν πολύ διευρυμένο χώρο. Είναι ικανοί να αντιμετωπίσουν χώρους με ασυνέχειες, προβλήματα με φραγμένες παραμέτρους και χώρους με πολλά τοπικά μέγιστα ή ελάχιστα, αλλά έχουν αργή σύγκλιση προς τη λύση. [4][5][18]

1.4.1 Χαρακτηριστικά μεθόδων καθολικής βελτιστοποίησης

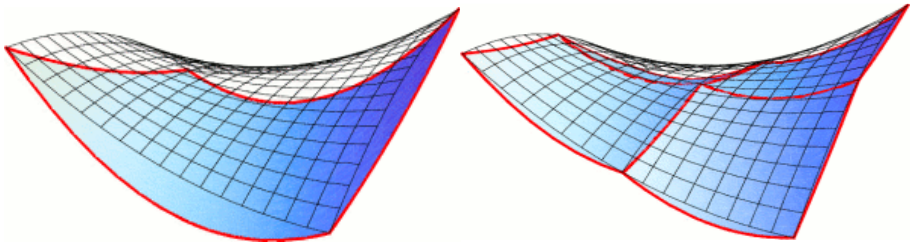
Οι μέθοδοι καθολικής βελτιστοποίησης εφόσον έχουν σκοπό να εντοπίσουν το ολικό ελάχιστο/μέγιστο οφείλουν να εξετάσουν όλα τα ακρότατα που εμφανίζει η αντικειμενική συνάρτηση. Μία χρήσιμη ικανότητά τους, είναι ότι μπορούν να χρησιμοποιούν μεθόδους τοπικής βελτιστοποίησης, καθώς διαθέτουν μεγάλη ακρίβεια στον προσδιορισμό της τιμής ενός τοπικού ακροτάτου σε μικρές περιοχές ενδιαφέροντος. Συνδυαστικά, επίσης χρήσιμη ικανότητα είναι ότι μπορούν να απεγκλωβιστούν από τοπικά ακρότατα με διάφορους τόπους (όπως θα δούμε και παρακάτω) [5]. Τα κύρια χαρακτηριστικά λοιπόν που συνθέτουν έναν αποδοτικό αλγόριθμο καθολικής βελτιστοποίησης εμφανίζονται παρακάτω:

1. *Αναπαράσταση του πεδίου αναζήτησης*:
   Το πεδίο αναζήτησης εξαρτάται από το πεδίο ορισμού των παραμέτρων του προβλήματος. Μπορεί να είναι περιορισμένο ή άπειρο, αναλόγως με το αν υπάρχουν συνάρτησης περιορισμών ή όχι αντίστοιχα. Κλασσικό παράδειγμα είναι το γνωστό πρόβλημα του πλανόδιου πωλητή που θα αναφερθεί και αργότερα στο οποίο το πεδίο ορισμού αναζήτησης είναι ένα σύνολο τιμών που εκφράζουν την κάθε πόλη μέσα στο πρόβλημα. [5]

2. *Στρατηγικές διαίρεσης*
   Ενώ άλλες μέθοδοι βελτιστοποίησης ερευνούν όλο το πεδίο ορισμού για τον εντοπισμό της βέλτιστης λύσης, κάποιες άλλες χωρίζουν το χώρο σχεδιασμού σε τμήματα εξετάζοντας το κάθε υποσύνολο ξεχωριστά. Τέτοιας φιλοσοφίας είναι και η τεχνική διακλάδωσης και φράγματος (branch and bound), η οποία ανήκει στις αιτιοκρατικές καθολικές μέθοδος, και στόχος της είναι να απορρίπτει ορισμένα υποσύνολα ή να υποδιαιρεί κάποια άλλα, τα οποία ενδεχομένως να περιέχουν τη βέλτιστη λύση. Χρησιμοποιείται για τη γρήγορη επίλυση εκθετικής χρονικής πολυπλοκότητας συνδυαστικών προβλημάτων βελτιστοποίησης, καθώς και έχει αποτελέσει βάση για την ανάπτυξη άλλων αλγορίθμων. Στις εικόνες 1.3, 1.4 και 1.5παρακάτω, φαίνεται οπτικά η τεχνική branch and bound. [5][20]



Εικόνα 1.3 Γραφική αναπαράσταση τεχνικής branch and bound ([21])



Εικόνα 1.4 - 1.5 Οπτική αναπαράσταση δημιουργίας υποσυνόλων από την branch and bound ([21])

3. *Πρόβλεψη της κατεύθυνσης διερεύνησης στο πεδίο ορισμού των παραμέτρων*
   Η επιλογή της κατάλληλης κατεύθυνσης αναζήτησης για τον εντοπισμό του ολικού ελαχίστου/μεγίστου φέρει καίριας σημασίας, καθώς η σωστή επιλογή θα σώσει αρκετό χρόνο και υπολογιστικούς πόρους. Αρκετές μέθοδοι υπολογίζουν το σύνολο τιμών της συνάρτησης σε ένα επιθυμητό πεδίο ορισμού και επιλέγουν να εξετάσουν τις περιοχές που το ακρότατο είναι μικρότερο. Αργότερα το διάστημα που περιέχει το ακρότατο αυτό υποδιαιρείται και ελέγχεται πάλι. Η διαδικασία αυτή επαναλαμβάνεται έως ότου βρεθεί το ολικό ακρότατο. Σε άλλες μεθόδους σημαντικό ρόλο παίζει και η τιμή συνάρτησης του γειτονικού σημείου, γιατί παρέχει πληροφορία για την κατεύθυνση αναζήτησης. Πέρα από τις παραδοσιακές μεθόδους βέβαια, διαφορετική προσέγγιση αποτελούν οι γενετικοί αλγόριθμοι οι οποίοι όπως θα αναφερθεί και αργότερα, η κατεύθυνση που θα επιλέξουν για την αναζήτησή τους θα είναι προς εκείνα τα μέλη του πληθυσμού που εμφανίζουν μεγαλύτερη τιμή καταλληλότητας. [5]

4. *Μηχανισμοί απεγκλωβισμού από τοπικά ακρότατα*
   Για να μπορέσει ένας αλγόριθμος να εξετάσει όλα τα ακρότατα ώστε να βρει το ολικό, απαραίτητο είναι να μπορεί να ξεφεύγει από τα τοπικά. Γνωστή μέθοδος είναι η προσομοιωμένη ανόπτηση (simulated annealing), η οποία ανήκει στις στοχαστικές καθολικές μεθόδους και κατέχει τη λογική της διαδικασίας κατεργασίας μετάλλων, όπου στην αρχή που η θερμοκρασία είναι υψηλή, τα άτομα του μετάλλου παρουσιάζουν μεγάλη κινητικότητα, ενώ όσο η θερμοκρασία πέφτει, αντιστοίχως πέφτει και η κινητικότητα των ατόμων. Η παραπάνω λογική μεταφρασμένη σε γλώσσα βελτιστοποίησης αντιστοιχεί στην επαναληπτική αναζήτηση στο προκαθορισμένο πεδίο ορισμού της συνάρτησης, όπου αρχικά υπάρχει υψηλή πιθανότητα να γίνουν αποδεκτές ακόμη και λύσεις χειρότερες ως αντικατάσταση της τωρινής, προοδευτικά μειούμενη βέβαια καθώς η μέθοδος συγκλίνει προς τη βέλτιστη λύση. Παρόμοιας αποτελεσματικότητας είναι και η τεχνική των γενετικών αλγορίθμων, όπου πιθανοτικά μπορούν να μεταφερθούν σε επόμενο σημείο, η τιμή του οποίου δεν έχει καμία σχέση με του προηγούμενου. Έτσι δημιουργείται μία ποικιλία στα διαθέσιμα σημεία που μπορεί να μεταφερθεί ο κάθε αλγόριθμος παρόμοιας λογικής, πιθανοτικά ικανός

να εξετάσει όλο το επιθυμητό πεδίο ορισμού, χωρίς να εγκλωβιστεί σε κάποιο τοπικό ακρότατο. [5][22]

5. *Κριτήρια τερματισμού*
   Δεδομένου του ότι ένας αλγόριθμος βελτιστοποίησης θα πρέπει να εκτελείται μέσω ενός καθορισμένου ορίου, θα πρέπει να υπάρχουν κάποια κριτήρια κατά τα οποία ο αλγόριθμος να τερματίζει, είτε έχει βρει τη βέλτιστη λύση είτε όχι. Επιτακτική ανάγκη είναι ο αλγόριθμος να τερματίζει μετά από κάποιο χρονικό διάστημα, έχοντας επιστρέψει ικανοποιητικής προσέγγισης λύσεις· γι' αυτό επιζητείται η χρυσή τομή μεταξύ απόδοσης του αλγορίθμου και εγγύησης εύρεσης του καθολικού ελαχίστου/μεγίστου. Έχοντας λοιπόν τα παραπάνω σαν κεντρική ιδέα, τα γενικότερα κριτήρια τερματισμού του αλγορίθμου είναι τα εξής:
   I. Ένα καθορισμένο πλήθος αποτιμήσεων της συνάρτησης.
   II. Εισαγωγή προκαθορισμένου αριθμού βημάτων του αλγορίθμου. [5]

1.4.2 Κατηγορίες μεθόδων καθολικής βελτιστοποίησης

Βασική διάκριση των μεθόδων βελτιστοποίησης είναι σε αιτιοκρατικές και στοχαστικές μεθόδους.

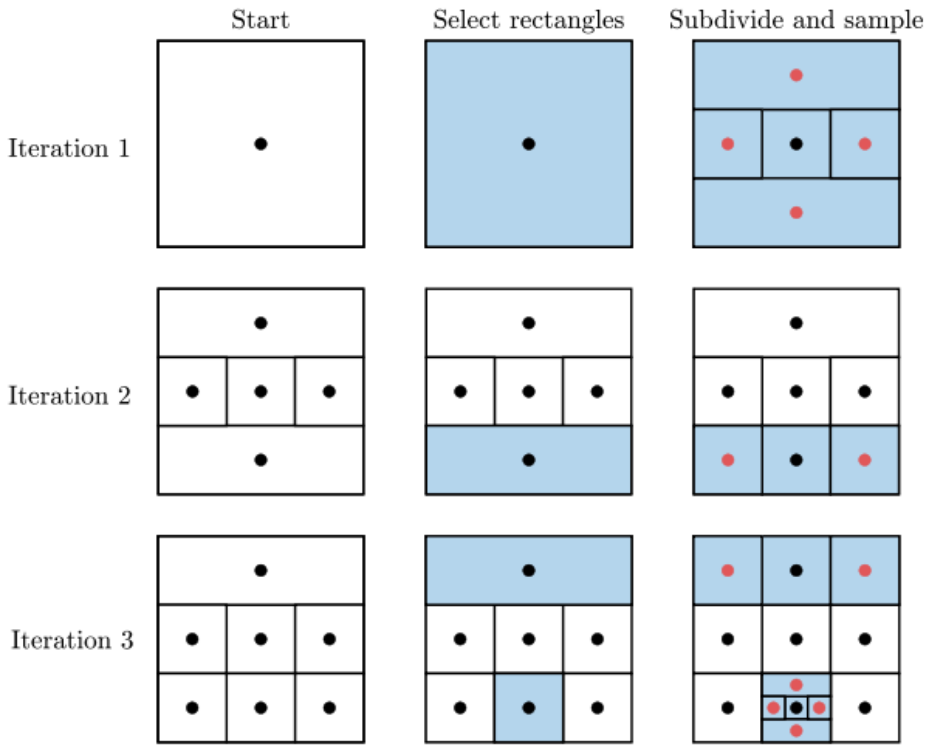
Αιτιοκρατικές μέθοδοι:

Οι αιτιοκρατικές ή αλλιώς ντετερμινιστικές μέθοδοι (deterministic methods), επιχειρούν να βρουν το καθολικό ελάχιστο εφαρμόζοντας μία εξαντλητική αναζήτηση σε όλο το πεδίο ορισμούτων μεταβλητών της συνάρτησης. Τα αποτελέσματα είναι εγγυημένα όταν βέβαια η συνάρτηση πληροί κάποιες προϋποθέσεις. Οι μέθοδοι αυτές καταναλώνουν μεγάλο υπολογιστικό χρόνο, αυξανόμενος με εκθετικό ρυθμό με την αύξηση του πλήθους των μεταβλητών του προβλήματος, μέχρι την ικανοποίηση κάποιου κριτηρίου τερματισμού, όπου, η εύρεση του κατάλληλου, αποτελεί μεγάλο πρόβλημα τόσο για τις αιτιοκρατικές όσο και για τις στοχαστικές μεθόδους. Οι αιτιοκρατικές μέθοδοι βασίζονται στις υποθέσεις πως ο χώρος τιμών των μεταβλητών καθώς και τα όρια των περιορισμών είναι γνωστά, σταθερά και πως ο υπολογισμός της αντικειμενικής συνάρτησης και των συναρτήσεων περιορισμών είναι ντετερμινιστικές διαδικασίες, δηλαδή για τα ίδια επιχειρήματα θα επιστρέφονται κάθε φορά τα ίδια αποτέλεσμα. Υποθέσεις που ισχύουν βέβαια μόνο σε ιδανικές συνθήκες. Αν και συγκλίνουν γρήγορα στη βέλτιστη λύση, υπάρχει πάντα ο κίνδυνος να βρεθεί τοπικό και όχι ολικό ακρότατο, κάτι που εξαρτάται κυρίως από το σημείο εκκίνησης.Η ιδέα των αιτιοκρατικών μεθόδων είναι η εύρεση μίας αρχικής λύσης, όπου αργότερα, σε κάθε επανάληψη ο αλγόριθμος παρακολουθεί και βελτιώνει τη λύση αυτή. Σε έναν μεγάλο βαθμό, από την συγκεκριμένη αρχική λύση που θα βρεθεί, εξαρτάται το ποσοστό κινδύνου για εγκλωβισμό του αλγορίθμου σε τοπικό ακρότατο. Παρακάτω αναλύονται ορισμένες αιτιοκρατικές μέθοδοι. [5][18][19]

- *Μέθοδος Direct*:

Η μέθοδος DIRECT δημιουργήθηκε από τους Jones, Perttunen και Stuckman το 1993, οι οποίοι εμπνεύστηκαν από την Lipschitzian καθολική βελτιστοποίηση. Η DIRECT είναι μία αυστηρή και ντετερμινιστική μέθοδος βασισμένη στη μέθοδο διακλάδωσης και φράγματος (branch and bound), όπου τα φράγματα υπολογίζονται βασισμένα στη γνώση μίας Lipschitz σταθεράζγια την αντικειμενική συνάρτηση. Η DIRECT προσέφερε κάποιες παραλλαγές στην Lipschitzian προσέγγιση που την έκανε πιο ανιχνεύσιμη σε προβλήματα πολλών διαστάσεων, καθώς και εξάλειψε την ανάγκη να γνωρίζει κανείς την Lipschitz σταθερά. Το αποτέλεσμα ήταν ένας αλγόριθμος απλός, ντετερμινιστικός, ο οποίος είχε μία μόνο υπερπαράμετρο, την επιθυμητή ακρίβεια. Η DIRECT, αν και πλήρως αποδοτική σε προβλήματα λίγων διαστάσεων (2-6 μεταβλητές), αρκετά χρήσιμη δηλαδή για προβλήματα σε αρκετά μεγάλο εύρος επιστημονικών πεδίων, είναι πολύ λιγότερο ανταγωνιστική όταν έχει να κάνει με προβλήματα πολλών διαστάσεων. Πιο συγκεκριμένα, αρκετοί γνώστες έχουν παρατηρήσει την τάση της μεθόδου να φτάνει γρήγορα στο καθολικό ακρότατο, αλλά χρειάζεται αρκετή ώρα ώστε να βελτιστοποιήσει τη λύση σε υψηλή ποιότητα. Αν και, ανά τα χρόνια, έχουν δημοσιευθεί αρκετές παραλλαγές της, όπως είναι η DIRECT (glcCluster) η οποία σε μία σχετικά πρόσφατη σύγκριση με άλλους αλγορίθμους για τον χειρισμό προβλημάτων μέχρι και 300 μεταβλητών, κατέληξε να είναι από τους κορυφαίους αλγορίθμους βάσει απόδοσης, ή η eDIRECT-C η οποία πάλι σχετικά πρόσφατα, επιτυχώς κατάφερε να βελτιστοποιήσει έναν κβαντικό προσομοιωτή 15 μεταβλητών, παρακάτω θα αναπτυχθεί η λογική του πρώτου αλγορίθμου DIRECT, που αποτέλεσε βάση για αρκετές παραλλαγές ή παρόμοιους αλγορίθμους σε βάθος χρόνου.

Η λογική λοιπόν πίσω από την πρώτη έκδοση της DIRECT μεθόδου, έχοντας ως χώρο σχεδιασμού έναν υπερκύβο και υπακούοντας σε περιορισμούς που ορίζουν τα κάτω και άνω όρια του πεδίου ορισμού, είναι να διαιρεί τον υπερκύβο σε μικρότερα ορθογώνια τμήματα, με την ιδιότητα ότι η αντικειμενική συνάρτηση αξιολογείται στο κάθε κεντρικό σημείο του κάθε νέου ορθογωνίου. Πιο συγκεκριμένα, σε κάθε επανάληψη, ορισμένα "πιθανώς βέλτιστα" ορθογώνια επιλέγονται για περαιτέρω αναζήτηση. Αυτά τα ορθογώνια τότε υποδιαιρούνται και η αντικειμενική συνάρτηση αξιολογείται στο κεντρικό σημείο των νέων ορθογωνίων που προέκυψαν. Η παραπάνω λογική φαίνεται καλύτερα και στηνεικόνα 1.4, όπου φαίνεται οπτικά η έκβαση μέσω τριών επαναλήψεων.



Εικόνα1.6 Οπτική απεικόνιση της DIRECT μεθόδου για 3 επαναλήψεις ([32])

Στην παραπάνω εικόνα φαίνεται η διαδικασία των τριών πρώτων επαναλήψεων του αλγορίθμου για μία αντικειμενική συνάρτηση με 2 μεταβλητές. Αρχικά, υπάρχει ένα ορθογώνιο (ο λεγόμενος υπερκύβος), όπου και επιλέγεται. Ο υπερκύβος χωρίζεται σε μικρότερα ορθογώνια τμήματα (η διάσταση πάνω στην κατεύθυνση της οποίας θα γίνει ο χωρισμός είναι τυχαία), και το κεντρικό σημείο του κάθε ορθογωνίου που προκύπτει (όχι το κεντρικό όλων, καθώς αυτό ανήκει στον υπερκύβο) αξιολογείται με βάση την αντικειμενική συνάρτηση (κόκκινες κουκίδες). Στη δεύτερη επανάληψη, επιλέγεται ένα ορθογώνιο, χωρίζεται σε ακόμη μικρότερα ορθογώνια τμήματα, και ξανά τα κεντρικά τους σημεία αξιολογούνται. Στην τρίτη επανάληψη, επιλέγονται 2 ορθογώνια και ακολουθείται η ίδια φιλοσοφία. Η διαδικασία αυτή επαναλαμβάνεται μέχρι ο αλγόριθμος να φτάσει είτε σε όριο επαναλήψεων, είτε σε όριο αξιολογήσεων την συνάρτησης. [32][33]

- *Μέθοδοι ανίχνευσης κατά γραμμή*:
  Η ιδέα των επαναληπτικών αυτών μεθόδων ανίχνευσης κατά γραμμή, είναι ο υπολογισμός της επόμενης προσέγγισης $\vec{x}^{n+1}$ της λύσης, χρησιμοποιώντας την τρέχουσα προσέγγιση $\vec{x}^{n}$και μία μεταβλητή, την κατεύθυνση ανίχνευσης$\vec{p}^{n}$ πολλαπλασιαζόμενη με μία βαθμωτή ποσότητα η που ρυθμίζει το μέγεθος του

βήματος (step size) στην κατεύθυνση$\vec{p}^n$. Παρακάτω φαίνεται και ο μαθηματικός τύπος:

$$\vec{x}^{n+1} = \vec{x}^n + \eta * \vec{p}^n,$$

με τον δείκτη n να συμβολίζει τον αριθμό της επανάληψης. Μεταξύ των διαφορετικών μεθόδων ανίχνευσης κατά γραμμή, η διαφορά βρίσκεται στον τρόπο που υπολογίζεται η τιμή της κατεύθυνσης ανίχνευσης και η τιμή του η. Άλλοτε η τιμή του η μπορεί να είναι σταθερή για όσο εκτελείται η αντίστοιχη μέθοδος και άλλοτε να υπολογίζεται σε κάθε βήμα εκ νέου, μετά τον υπολογισμό της κατεύθυνσης ανίχνευσης $\vec{p}^n$. Στην τελευταία περίπτωση, ο παραπάνω μαθηματικός τύπος μετατρέπεται στην παρακάτω μορφή:

$$\vec{x}^{n+1} = \vec{x}^n + \eta^n * \vec{p}^n$$

Παρακάτω εμφανίζονται ορισμένες μέθοδοι ανίχνευσης κατά γραμμή:

Η μέθοδος της απότομης καθόδου:

Η μέθοδος της απότομης καθόδου (steepest descent method), υπολογίζει την επόμενη προσέγγιση $\vec{x}^{n+1}$, επιλέγοντας ως κατεύθυνση ανίχνευσης $\vec{p}^n$ το αντίθετο του τοπικού διανύσματος κλίσης της συνάρτησης $F(\vec{x}^n)$, όπως φαίνεται και παρακάτω:

$$p^n = -\nabla F(\vec{x}^n)$$

Για λόγους πληρότητας, παρακάτω εμφανίζεται και το ανάπτυγμα Taylor:

$$F(\vec{x}^n + \eta^n * \vec{p}^n) = F(\vec{x}^n) + \eta^n * \vec{p}^{nT} * \nabla F(\vec{x}^n) + O(n^2)$$

όπου, το πρόσημο του δεύτερου όρου στο δεξί μέρος (αρνητική ποσότητα για $\eta^n > 0$) δικαιολογεί και την επιλογή της προηγούμενης σχέσης. Χαρακτηριστικό της μεθόδου απότομης καθόδου είναι πως απαιτεί τον υπολογισμό της κλίσης της αντικειμενικής συνάρτησης, και ότι δεν απαιτεί πληροφορία για το μητρώο Hess (δεύτερες παράγωγοι). Παρόλα αυτά, η μέθοδος είναι αποτελεσματική ΜΟΝΟ για απλά προβλήματα βελτιστοποίησης, ενώ σε πολύπλοκα προβλήματα γίνεται αργή και πλήρως αναποτελεσματική.

Η μέθοδος Newton:
Η μέθοδος Newton, ως μία από τις πιο κλασσικές μεθόδους ανίχνευσης κατά γραμμή, χρησιμοποιεί το ανάπτυγμα Taylor διατηρώντας τον όρο της δεύτερης τάξης:

$$F(\vec{x}^n + \vec{p}^n) \approx F(\vec{x}^n) + \vec{p}^{nT} * \nabla F(\vec{x}^n) + \frac{1}{2} * \vec{p}^{nT} * \nabla^2 F(\vec{x}^n) * \vec{p}^n$$

Η απαίτηση μηδενισμού της πρώτης παραγώγου της $F(\vec{x}^n + \vec{p}^n)$ ως προς $\vec{p}^n$ δίνει:

$$\nabla F(\vec{x}^n) + \nabla^2 F(\vec{x}^n) * \vec{p}^n = 0$$

που καθορίζει την κατεύθυνση ανίχνευσης για την τρέχουσα επανάληψη ως:

$$\vec{p}^n = - (\nabla^2 F(\vec{x}^n))^{-1} * \nabla F(\vec{x}^n)$$

Εδώ πρέπει να σημειωθεί πως η παραπάνω σχέση προϋποθέτει ότι το μητρώο Hess $\nabla^2 F(\vec{x}^n)$ είναι θετικά ορισμένο, αλλιώς η κατεύθυνση $\vec{p}^n$ δεν μπορεί να οριστεί αφού δεν μπορεί να υπολογιστεί η αντίστροφος του μητρώου Hess. Αξίζει να σημειωθεί επίσης ότι στις περισσότερες εφαρμογές της μεθόδου Newton, η βαθμωτή ποσότητα η^nθεωρείται πως είναι ίση με 1. Η μέθοδος αυτή σε αντίθεση με την μέθοδο της απότομης καθόδου απαιτεί τον υπολογισμό του μητρώου Hess της αντικειμενικής συνάρτησης, γεγονός που προκαλεί αυξημένες υπολογιστικές δυσκολίες και κόστος. Η περίπτωση που η μέθοδος Newton αξίζει να χρησιμοποιηθεί, είναι όταν μπορεί να προκαλέσει αισθητά ταχύτερη σύγκλιση.

- <u>*Μέθοδος ανίχνευσης κατά περιοχές εμπιστοσύνης*</u>:
  Η μέθοδος ανίχνευσης κατά περιοχές εμπιστοσύνης είναι μία εναλλακτική μέθοδος της ανίχνευσης της βέλτιστης λύσης κατά γραμμή. Η βασική αρχή της μεθόδου αυτής στηρίζεται στην επαναληπτική δημιουργία ενός τοπικού μοντέλου της αντικειμενικής συνάρτησης $F(\vec{x})$, έστω $\Phi^n(\vec{x})$, όπου n, ο αριθμός των επαναλήψεων στη γειτονιά της βέλτιστης λύσης και στη χρήση του, ώστε σταδιακά, να προσεγγίζεται όλο και περισσότερο η βέλτιστη λύση. Συνηθέστερη τεχνική για τη δημιουργία του μοντέλου είναι μία συνάρτηση τετραγωνικής μορφής. Αν λοιπόν,$\vec{x}^n$ είναι η τρέχουσα λύση, η δημιουργία του νέου τοπικού μοντέλου $\Phi(\vec{x}^n + \vec{p})$ υπολογίζεται ξανά από το ανάπτυγμα Taylor ως εξής:

$$\Phi^n(\vec{x}^n + \vec{p}) = F(\vec{x}^n) + \vec{p}^T * \nabla F(\vec{x}^n) + \frac{1}{2} * \vec{p}^T * \nabla^2 F(\vec{x}^n) * \vec{p}$$

Μαθηματικά το παραπάνω ανάπτυγμα διατυπώνεται ως:

$$\min\Phi(\vec{x}^n+\vec{p}),\ (\vec{x}^n+\vec{p})\in T(\vec{x}^n),$$

με $T(\vec{x}^n)$, να είναι η περιοχή εμπιστοσύνης γύρω από το $\vec{x}^n$. Η περιοχή εμπιστοσύνης ωστόσο, ισοδυναμεί με το χώρο γύρω από το $\vec{x}^n$ ο οποίος καθορίζεται από την παρακάτω σχέση:

$$\|\vec{p}\| \leq \Delta,$$

όπου Δ, καλείται η ακτίνα περιοχής εμπιστοσύνης (trust region radius).

Βασική προσεγγιστική διαφορά ανάμεσα στις μεθόδους ανίχνευσης κατά γραμμή και στη μέθοδο ανίχνευσης κατά περιοχές εμπιστοσύνης, είναι πως, ενώ στις μεθόδους ανίχνευσης κατά γραμμή, υπολογίζεται αρχικά η τρέχουσα λύση $\vec{x}^n$, η κατεύθυνση ανίχνευσης $\vec{p}^n$, και βάσει αυτής, το μέγεθος του βήματος (step size), στη μέθοδο ανίχνευσης κατά περιοχές εμπιστοσύνης, πρώτα υπολογίζεται η μέγιστη ακτίνα Δ στο σημείο $\vec{x}^n$, άρα συνεπώς πρώτα καθορίζεται η διαφορά απόστασης από το $\vec{x}^n$ όπου αναζητείται η λύση, και αργότερα υπολογίζεται η κατεύθυνση αναζήτησης. [19]

Στοχαστικές μέθοδοι:

Στοχαστικές μέθοδοι ονομάζονται οι μέθοδοι βελτιστοποίησης που παράγουν και χρησιμοποιούν τυχαίες μεταβλητές[23]. Χρησιμοποιούν δηλαδή, στοιχεία τυχαίας ή οργανωμένα τυχαίας αναζήτησης για την εύρεση της βέλτιστης λύσης. Αποτελούν μία γενίκευση των αιτιοκρατικών μεθόδων, και χρησιμεύουν στην επίλυση στοχαστικών προβλημάτων (προβλημάτων που εμφανίζουν τυχαίες μεταβλητές) ή ακόμη και μη-στοχαστικών προβλημάτων χρησιμοποιώντας τυχαίες επαναλήψεις. Μπορούν να χαρακτηριστούν ως περισσότερο γενικές μέθοδοι, με την έννοια ότι εύκολα προγραμματίζονται σε διαφορετικούς τύπους προβλημάτων. Πρέπει να σημειωθεί βέβαια ότι καταλήγουν να είναι περισσότερο αργές, με κέρδος όμως, το να μπορούν να εντοπίσουν το ολικό ακρότατο ανεξάρτητα από τη λύση ή τις λύσεις με τις οποίες εκκινούν. Μεγάλο μειονέκτημα των στοχαστικών μεθόδων είναι ότι ανακαλύπτουν το καθολικό ελάχιστο με κάποια πιθανότητα, μέσω ενός πεπερασμένου αριθμού επαναλήψεων. Αυτό σημαίνει ότι εφόσον το ασυμπτωτικό όριο (άπειρες επαναλήψεις) σε πρακτικές εφαρμογές δεν είναι εφικτό, θα πρέπει να τερματίσει η διαδικασία πρώιμα διακινδυνεύοντας την πλήρη επιτυχία της. Παρόλα αυτά, υπό κάποιες προϋποθέσεις που αφορούν την κατανομή των σημείων του αρχικού δείγματος και της συνάρτησης όσο αυξάνεται το πλήθος των σημείων των δειγμάτων αυτών, η πιθανότητα εγγυημένων

λύσεων πλησιάζει τη μονάδα. Στη διαδικασία εύρεσης της βέλτιστης λύσης, οι στοχαστικές μέθοδοι περιλαμβάνουν συνήθως δύο μεγάλες φάσεις:

1. Την *ολική φάση*, που η συνάρτηση αποτιμάται σε έναν τυχαίο αριθμό σημείων.
2. Την*τοπική φάση*, που κάποια από τα σημεία του αρχικού δείγματος αποτελούν αρχικά σημεία για μεθόδους τοπικής βελτιστοποίησης.

Το μεγαλύτερο πλεονέκτημα που έχουν οι στοχαστικές μέθοδοι έναντι των ατιοκρατικών μεθόδων, είναι πως μπορούν να αντιμετωπίσουν το πρόβλημα του εγκλωβισμού σε τοπικό ακρότατο.
Παρακάτω εμφανίζονται δύο βασικές δράσεις για το πρόβλημα αυτό:

1. Μη χρήση μόνο αιτιοκρατικών κριτηρίων βελτίωσης της τρέχουσας λύσης. Ταυτόχρονα μπορεί να χρησιμοποιηθούν στοχαστικά κριτήρια που θα επιφέρουν μία τυχαιότητα στη χρησιμοποιούμενη μέθοδο. Η στοχαστικότατα δίνει ευκαιρίες ανίχνευσης και άλλων υποπεριοχών του πεδίου ορισμού. Αυτό όμως επιφέρει βέβαια και υψηλότερο υπολογιστικό κόστος.
2. Εγκαταλείπεται η ιδέα για παρακολούθηση και βελτίωση μίας μόνο λύσης ανά επανάληψη. Αντ'αυτού, πολλές υποψήφιες λύσεις ανά επανάληψη, παρέχει μεγαλύτερη σιγουριά πως η λύση που θα βρεθεί θα είναι το ολικό ακρότατο. Τέτοιες μέθοδοι ονομάζονται μέθοδοι βελτιστοποίησης σε πληθυσμούς λύσεων (population-based optimization methods).

Οι δύο παραπάνω τρόποι αν συνδυαστούν, δημιουργούν μία κατηγορία στοχαστικών μεθόδων που βασίζεται σε πληθυσμούς λύσεων· εκεί ανήκουν και οι εξελικτικοί αλγόριθμοι (οι οποίοι θα αναλυθούν παρακάτω). Όσον αφορά τον τερματισμό των στοχαστικών μεθόδων, κύριο κριτήριο τερματισμού, αποτελεί ένα κριτήριο που πιστοποιεί ότι η πιθανότητα για την ορθότητα των αποτελεσμάτων πλησιάζει τη μονάδα.[5][19]Παρακάτω αναλύονται ορισμένες στοχαστικές μέθοδοι.

1. <u>*Μέθοδος αναρρίχησης λόφου και παραλλαγές αυτής*</u>:
   Η μέθοδος αναρρίχησης λόφου λαμβάνει το όνομά της λόγω του ότι απευθύνεται σε προβλήματα μεγιστοποίησης. Εύκολα μπορεί όμως να παραμετροποιηθεί σε σχήμα κατάβασης λόφου για την επίλυση προβλημάτων ελαχιστοποίησης. Δεδομένου λοιπόν, ότι η μέθοδος αποσκοπεί στη μεγιστοποίηση ενός προβλήματος, αρχική προσέγγιση της ανάλυσής της, θα είναι η επεξήγηση μίας τοπικής μεθόδου ανίχνευσης πάνω στην οποία στηρίζεται ο αλγόριθμος, και σταδιακά, θα αναπτυχθεί στην τελική μορφή της μεθόδου.

   Ακολουθεί η μορφή του απλούστερου αλγορίθμου αναρρίχησης λόφου με τοπική ανίχνευση:

- Βήμα 1: Αρχικοποίηση του μετρητή επαναλήψεων n = 0, τυχαία επιλογή της αρχικής λύσης $\vec{x}^n$ όπου $\vec{x}^n \in \Omega$. Υπολογισμός τιμής της αντικειμενικής συνάρτησης στο $\vec{x}^n$.
- Βήμα 2: Αφού εντοπιστούν όλες οι λύσεις στη γειτονιά του $\vec{x}^n$, για καθεμία από τις οποίες υπολογίζεται η τιμή της αντικειμενικής συνάρτησης, επιλέγεται η καλύτερη (για πρόβλημα μεγιστοποίησης αυτή η λύση, δηλαδή, με τη μεγαλύτερη αντικειμενική συνάρτηση F), έστω $\vec{x}^{\#}$.
- Βήμα 3: Αν $\vec{x}^{\#} > \vec{x}^n$ ($\vec{x}^{\#} < \vec{x}^n$ για πρόβλημα ελαχιστοποίησης), τότε $\vec{x}^n = \vec{x}^{\#}$, n = n+1, επιστροφή του αλγορίθμου στο βήμα 2.
- Βήμα 4: Ο αλγόριθμος τερματίζει με τη $\vec{x}^n$ ως βέλτιστη λύση.

Με την εισαγωγή ενός εξωτερικού βρόχου στον παραπάνω αλγόριθμο, με σκοπό σε κάθε επανάληψη να ξεκινά με διαφορετική τυχαία λύση, και καταλήγοντας, να καταγράφει ως βέλτιστη την καλύτερη λύση μεταξύ του πλήθους των βέλτιστων λύσεων κάθε επανάληψης μέχρι στιγμής, δημιουργείται ο αλγόριθμος αναρρίχησης λόφου με επανεκκίνηση (iterated hill climber). Ακολουθεί η μορφή του αλγορίθμου:

- Βήμα 1: Αρχικοποίηση του μετρητή επαναλήψεων στον εξωτερικό βρόχο, k = 0, ανάθεση πολύ μικρής τιμής στο τελικό διάνυσμα $\vec{x}^*$, στην αρχική φάση του αλγορίθμου (για πρόβλημα ελαχιστοποίησης ανατίθεται μία πολύ μεγάλη τιμή).
- Βήμα 2: Αρχικοποίηση του μετρητή επαναλήψεων στον εσωτερικό βρόχο, n = 0, τυχαία επιλογή της αρχικής λύσης $\vec{x}^n$. Υπολογισμός τιμής της αντικειμενικής συνάρτησης στο $\vec{x}^n$.
- Βήμα 3: Αφού εντοπιστούν όλες οι λύσεις στη γειτονιά του $\vec{x}^n$, για καθεμία από τις οποίες υπολογίζεται η τιμή της αντικειμενικής συνάρτησης, επιλέγεται η καλύτερη (για πρόβλημα μεγιστοποίησης αυτή η λύση, δηλαδή, με τη μεγαλύτερη αντικειμενική συνάρτηση F), έστω $\vec{x}^{\#}$.
- Βήμα 4: Αν $\vec{x}^{\#} > \vec{x}^n$ ($\vec{x}^{\#} < \vec{x}^n$ για πρόβλημα ελαχιστοποίησης), τότε $\vec{x}^n = \vec{x}^{\#}$, n = n+1, και ο αλγόριθμος επιστρέφει στο βήμα 3.
- Βήμα 5: Ορισμός του $\vec{x}^n$ ως καλύτερη λύση για την επανάληψη αυτή ($\vec{x}^{*,k} = \vec{x}^n$).
- Βήμα 6: Αν $\vec{x}^{*,k} > \vec{x}^*$ ($\vec{x}^{*,k} < \vec{x}^*$ για πρόβλημα ελαχιστοποίησης), τότε $\vec{x}^* = \vec{x}^{*,k}$, k = k+1, επιστροφή του αλγορίθμου στο βήμα 2.

Στη συνέχεια τροποποιείται ο αλγόριθμος αναρρίχησης λόφου με επανεκκίνηση με την εισαγωγή ορισμένων στοιχείων στοχαστικότατας (αν και ήδη περιέχει κάποια στοιχεία στοχαστικότατας, εφόσον σε κάθε επανάληψη ξεκινά με μία νέα,

τυχαία επιλεγμένη, αρχική λύση), ώστε να προκύψει ο στοχαστικός αλγόριθμος αναρρίχησης λόφου ο οποίος εμφανίζεται παρακάτω:

- Βήμα 1: Αρχικοποίηση του μετρητή επαναλήψεων στον εσωτερικό βρόχο, n = 0, τυχαία επιλογή της αρχικής λύσης $\vec{x}^n$. Υπολογισμός τιμής της αντικειμενικής συνάρτησης στο$\vec{x}^n$.
- Βήμα 2: Επιλογή ενός τυχαίου σημείου στη γειτονιά του $\vec{x}^n$, έστω $\vec{x}$, και ακολουθεί ο υπολογισμός της αντικειμενικής συνάρτησης στο $\vec{x}$. Έπειτα υπολογίζεται η πιθανότητα P, βάσει του παρακάτω τύπου:

$$P = \left(\frac{1}{1+e^{\frac{F(\vec{x}^n)-F(\vec{x})}{T}}} \right), T \in [0,1]$$

- Βήμα 3: Αν T < P, τότε $\vec{x}^n = \vec{x}$.
- Βήμα 4: (αλλιώς) $\vec{x}^n = \max(\vec{x}^n, \vec{x})$, n = n + 1, επιστροφή του αλγορίθμου στο βήμα 2.

Ο παραπάνω αλγόριθμος, όπως φαίνεται, δε χρειάζεται εξωτερικό βρόχο· και αυτό, γιατί το ρόλο της επανεκκίνησης έχει η δυνατότητα αλλαγής επίκεντρου ανίχνευσης, ακόμη και αν το νέο επίκεντρο είναι χειρότερο του προηγούμενου. Αυτό φαίνεται και στο βήμα 2, που επιλέγεται ένα μόνο τυχαίο στοιχείο στη γειτονιά του $\vec{x}^n$, το οποίο ακόμη και αν δεν υπερτερεί του $\vec{x}^n$, βάσει πιθανότητας P, μπορεί και να το αντικαταστήσει. Η μεταβλητή T, είναι ένας πραγματικός αριθμός, που κυμαίνεται από το 0 έως το 1 και στόχος της είναι η ρύθμιση των επιπέδων τιμής για την πιθανότητα P. Το $\vec{x}$, μπορεί να γίνει το νέο επίκεντρο ανίχνευσης, είτε με το να υπερτερεί του $\vec{x}^n$, είτε βάσει της πιθανότητας P, η οποία εξαρτάται από την απόκλιση της καταλληλότητας μεταξύ των $\vec{x}^n$ και $\vec{x}$, με τέτοιο τρόπο, ώστε να μειώνεται όσο χειρότερη είναι η τιμή καταλληλότητας του $\vec{x}$ συγκριτικά με του $\vec{x}^n$. Το δεύτερο σενάριο, που καθιστά το επίκεντρο ανίχνευσης να μπορεί να ανήκει σε σημείο που μέχρι τώρα δεν είναι η βέλτιστη λύση, είναι μία έξυπνη τεχνική για να μπορεί ο αλγόριθμος να ξεφεύγει από τα τοπικά ακρότατα. [19]

2. *Εξελικτικοί αλγόριθμοι*:
   Οι εξελικτικοί αλγόριθμοι (evolutionary algorithms) σε αντίθεση με τις περισσότερες τεχνικές βελτιστοποίησης, χρησιμοποιούν έναν πιθανό πληθυσμό λύσεων, στον οποίο εφαρμόζοντας ανταγωνιστικά κάποιους τελεστές καταλληλότητας, τείνουν να βρίσκουν μία αρκετά ικανοποιητική αν όχι ολικά βέλτιστη, λύση. Έχουν εφαρμοστεί σε πολλά προβλήματα αναζήτησης

βελτιστοποίησης καθώς και μηχανικής μάθησης (Holland 1975, Goldberg 1989), και σε αντίθεση με τις συμβατικές μεθόδους που εφαρμόζονται σε ορισμένη κατηγορία προβλημάτων για ταχύτερη αναζήτηση (π.χ. οι μέθοδοι επίλυσης γραμμικών προβλημάτων εφαρμόζονται σε γραμμικά προβλήματα), οι εξελικτικοί αλγόριθμοι χρησιμοποιούνται αποδοτικά μόνο σε προβλήματα υψηλής πολυπλοκότητας, μεταξύ της διαφορετικής φύσης των οποίων, δεν επηρεάζεται σημαντικά η ταχύτητά τους. Η ιδέα πίσω από τους εξελικτικούς αλγορίθμους βρίσκεται στο χειρισμό ενός πληθυσμού από υποψήφιες λύσεις, για την αξιολόγηση των οποίων χρησιμοποιούνται ορισμένοι τελεστές εμπνευσμένοι από τη φυσική γενετική μεταβολή και τη φυσική επιλογή. Τα χαρακτηριστικά που θα πρέπει να έχει ένας αλγόριθμος για να θεωρηθεί εξελικτικός είναι τα εξής:

- Να χρησιμοποιεί πληθυσμούς ατόμων και όχι μεμονωμένα άτομα.
- Η επιλογή του κάθε ατόμου για να επιβιώσει και να περάσει στην επόμενη γενιά, να καθορίζεται από την τιμή καταλληλότητας (πρόβλημα μεγιστοποίησης), ή την τιμή κόστους (πρόβλημα ελαχιστοποίησης), με βάση την αντικειμενική συνάρτηση.
- Το μέγεθος του πληθυσμού να μπορεί να μεταβάλλεται δυναμικά, με τη δημιουργία νέων ή την εξαφάνιση ήδη υπαρχόντων ατόμων βάσει της τιμής καταλληλότητας τους.
- Κατά τη μετάβαση από γενιά σε γενιά να εφαρμόζεται η κληρονομικότητα χαρακτηριστικών από τους γονείς προς τους απογόνους, καθώς και στοχαστική εμφάνιση νέων χαρακτηριστικών. [19][24][25]

Χρησιμοποιούμενη ορολογία:

Ο *πληθυσμός ατόμων* ή *πληθυσμός λύσεων* οδηγείται (εξελίσσεται) προς τη λύση με χρήση τελεστών στα άτομα του πληθυσμού αυτού. Ο σκοπός των τελεστών είναι να τροποποιήσουν, να εισάγουν ή να αφαιρέσουν άτομα από τον πληθυσμό του προβλήματος.

Ένα *άτομο*, είναι μία πιθανή λύση ανάμεσα στην πληθώρα λύσεων που περιέχει το πρόβλημα βελτιστοποίησης και αναπαριστά τις μεταβλητές της λύσης αυτής. Οι μεταβλητές που περιέχει, μπορεί να είναι πληροφορίες για τη θέση της λύσης στο χώρο των λύσεων, ή πληροφορίες για την καταλληλότητα της συνάρτησης, για το πόσο δηλαδή πλησιάζει τη βέλτιστη λύση.

Οι *τελεστές* χωρίζονται σε δύο κατηγορίες. Στην πρώτη ανήκουν οι τελεστές που τροποποιούν τα άτομα του πληθυσμού, και στη δεύτερη οι τελεστές που προσθέτουν ή αφαιρούν άτομα από τον πληθυσμό.

Στην πρώτη κατηγορία των τελεστών ανήκουν οι τελεστές μετάλλαξης (mutation) και διασταύρωσης (crossover). Ο τελεστής μετάλλαξης προσομοιώνει

τη διαδικασία της φυσικής μετάλλαξης αλλάζοντας κάποια χαρακτηριστικά στο άτομο. Ο τελεστής διασταύρωσης χρησιμοποιώντας τουλάχιστον δύο άτομα, που στην προκειμένη χρονική στιγμή αποκαλούνται γονείς, και με βάση αυτών δημιουργεί έναν ή περισσότερους απογόνους (καινούρια άτομα), οι οποίοι κληρονομούν κάποια από τα χαρακτηριστικά των γονέων. Χάρις των δύο αυτών τελεστών κατά την έκβαση του αλγορίθμου, δημιουργούνται διαφοροποιημένα άτομα, και ο πληθυσμός των ατόμων αποκτά μεγαλύτερη ποικιλία, που κατά συνέπεια οδηγεί στη δημιουργία νέων πιθανών λύσεων.

Στη δεύτερη κατηγορία των τελεστών ανήκει ο τελεστής επιλογής (selection), ή αλλιώς, αναπαραγωγής (reproduction). Σκοπός του είναι να επιλέξει τα καλύτερα άτομα του πληθυσμού που αργότερα θα αναλάβει ο τελεστής διασταύρωσης.

Η *συνάρτηση κόστους* (pay-off function) ή *συνάρτηση καταλληλότητας* (fitness function), αποτελεί το κριτήριο επιλογής ενός ατόμου από τον τελεστή επιλογής. Σκοπός της είναι να εκτιμήσει πόσο κοντά βρίσκεται ένα άτομο στη βέλτιστη λύση.

Οι εξελικτικοί αλγόριθμοι διαχωρίζονται στις επόμενες κατηγορίες βάσει του τρόπου που αντιμετωπίζεται το πλήθως των ατόμων του προβλήματος:

- Γενεαλογικοί ΕΑ (generational EA): Ο αριθμός των απογόνων ισούται με τον αριθμό των γονέων και οι απόγονοι αντικαθιστούν τους γονείς. Έτσι, ο αριθμός των ατόμων στον πληθυσμό παραμένει σταθερός.

- Επιλεκτικοί ή ελιστικοί ΕΑ (elitist EA): Ένας αριθμός βέλτιστων ατόμων διατηρείται από γενιά σε γενιά. Σε αντίθεση με τους γενεαλογικούς εξελικτικούς αλγορίθμους, ο αριθμός των ατόμων στον πληθυσμό, μπορεί είτε να παραμείνει σταθερός, είτε να μεταβληθεί.

- Μόνιμης κατάστασης ΕΑ (steady-state EA): Είναι συνδυασμός των παραπάνω κατηγοριών. Τα άτομα κάθε γενιάς αποτελούνται από απογόνους αλλά και γονείς της προηγούμενης γενιάς, που επιλέγονται βάσει της καταλληλότητάς τους. [24]

Κύριες μεθοδολογίες εξελικτικών αλγορίθμων:

Οι εξελικτικοί αλγόριθμοι παρουσιάζονται σε τέσσερις διαφορετικές μορφές, οι οποίες είναι: οι **Γενετικοί Αλγόριθμοι** (Genetic Algorithms), ο **Εξελικτικός Προγραμματισμός** (Evolutionary Programming), οι **Εξελικτικές Στρατηγικές**

(Evolution Strategies) και ένα νέο παρακλάδι των Γενετικών Αλγορίθμων που εξελίχθηκε πρόσφατα, ο **Γενετικός Προγραμματισμός** (Genetic Programming).

- Γενετικοί Αλγόριθμοι: Προτάθηκαν από τον John Holland το 1975, με στόχο να εξετάσει το φαινόμενο της προσαρμογής, όπως παρουσιάζεται στη φύση, και να αναπτύξει τρόπους όπου οι τεχνικές της προσαρμογής αυτής να μπορούν να εισαχθούν σε υπολογιστικά συστήματα. Οι γενετικοί αλγόριθμοι αναλύονται εκτενέστερα σε επόμενο κεφάλαιο. [24]

- Εξελικτικές στρατηγικές: Οι εξελικτικές στρατηγικές αναπτύχθηκαν το 1964 στο Technical University of Berlin από τους Rechenberg και Schwefel. Η πρώτη έκδοση των εξελικτικών στρατηγικών, αφορούσε δύο άτομα (διμελής ΕΣ / (1+1)-ES), έναν γονέα και έναν απόγονο, όπου για κάθε επανάληψη θα δρούσε ως γονέας για την επόμενη γενιά το καλύτερο, εκ των δύο, άτομο. Αργότερα η έκδοση αυτή αντικαταστάθηκε, κυρίως σε υπολογιστικές εφαρμογές, από μία έκδοση που χειρίζεται πλήθος γονέων και απογόνων (πολυμελής ΕΣ / (μ+1)-ES). Εκεί εισάγεται η έννοια του ανασυνδυασμού, όπου, κατά τον σχηματισμό του νέου απογόνου και μετά τη μετάλλαξη και αξιολόγηση ποιότητας, ο απόγονος αυτός θα αντικαταστήσει τον χειρότερο από τους γονείς, δεδομένου ότι η ποιότητα του απογόνου είναι τουλάχιστον ίση με την ποιότητα του εκάστοτε χειρότερου γονέα. [24]

- Εξελικτικός προγραμματισμός: Ο εξελικτικός προγραμματισμός αναπτύχθηκε το 1966 από τους Fogel, Owens και Walsh. Η αρχική ιδέα πίσω από τον εξελικτικό προγραμματισμό ήταν η επιτυχημένη εφυής συμπεριφορά μέσω της προσομοίωσης της φυσικής εξέλιξης. Συγκεκριμένα ο D. Fogel όρισε τη νοημοσύνη ως την *ικανότητα ενός συστήματος να προσαρμόσει την συμπεριφορά του για να πετύχει τους στόχους του σε διάφορα περιβάλλοντα,* διευκρινίζοντας πως αυτό μπορεί να επιτευχθεί έχοντας σαν βάση την προσομοίωση της φυσικής εξέλιξης. Υλοποιώντας λοιπόν την παραπάνω φιλοσοφία, πράγματι, ο εξελικτικός αλγόριθμος χρησιμοποιεί αναπαραστάσεις διαφορετικής μορφής ανάλογα με τη φύση του προβλήματος. Για παράδειγμα, σε προβλήματα βελτιστοποίησης πραγματικών αριθμών, τα άτομα είναι διανύσματα πραγματικών αριθμών, ενώ για εφαρμογές μηχανών πεπερασμένων καταστάσεων χρησιμοποιούνται γράφοι κτλ. Το μοντέλο του Fogel, χρησιμοποιούσε μ>1 πληθυσμό ατόμων, από τα οποία δημιουργείται μ πλήθος απογόνων, μέσω μετάλλαξης του κάθε γονέα. Η κάθε μετάλλαξη μπορεί, τυχαία, να είναι αλλαγή ενός συμβόλου εξόδου, αλλαγή μίας κατάστασης μετάβασης, πρόσθεση μίας κατάστασης, διαγραφή μίας κατάστασης ή αλλαγή της αρχικής κατάστασης. Σε αντίθεση με τις εξελικτικές στρατηγικές, ο εξελικτικός προγραμματισμός δεν

χρησιμοποιεί ανασυνδυασμό και οι μεταλλάξεις γίνονται με ομοιόμορφη κατανομή, και ο αριθμός των μεταλλάξεων για ένα άτομο μπορεί είτε να είναι σταθερός, είτε να προκύπτει μέσω μίας πιθανότητας. Μετά την εφαρμογή των μεταλλάξεων και τη δημιουργία των απογόνων, επιλέγονται τα μ καλύτερα άτομα από το σύνολο των γονέων και των απογόνων. [24]

- Γενετικός προγραμματισμός: Ο γενετικός προγραμματισμός δημιουργήθηκε από τον J. Koza το 1992. Είναι μία "domain-independent, problem-solving" μέθοδος, σκοπός της οποίας είναι η αυτοματοποιημένη παραγωγή υπολογιστικών προγραμμάτων. Η φιλοσοφία του γενετικού προγραμματισμού απαντά στο ερώτημα που είχε θέσει ο Arthur Samuel το 1959 :

  > *Πως θα μπορούσαν να φτιαχτούν οι υπολογιστές, ώστε να κάνουν αυτό που πρέπει, χωρίς να τους πει κάποιος ακριβώς πως να το κάνουν;*

  Απορρέοντας από τους γενετικούς αλγορίθμους, παρόμοιας τεχνικής είναι και ο γενετικός προγραμματισμός όπου, ξεκινώντας από χιλιάδες ή εκατομμύρια, τυχαίως παραγόμενα, κατάλληλα υπολογιστικά προγράμματα, χρησιμοποιώντας τη λογική της "φυσικής επιλογής" του Δαρβίνου, καθώς και τους κατάλληλους τελεστές διασταύρωσης και κάποιες φορές μετάλλαξης, δημιουργεί την επόμενη γενιά πληθυσμού προγραμμάτων, φροντίζοντας πάντα να επιβιώσουν τα καταλληλότερα προγράμματα. Ο γενετικός προγραμματισμός δημιουργεί υπολογιστικά προγράμματα, ικανά για επίλυση προβλημάτων, ακολουθώντας τα παρακάτω τρία βήματα:

  1. Παραγωγή αρχικού πληθυσμού από τυχαίες συνθέσεις των συναρτήσεων και τερματικών συνθηκών του προβλήματος.

  2. Επαναλαβμανόμενα εκτελούνται τα παρακάτω βήματα μέχρι να ικανοποιηθεί το ορισμένο κριτήριο τερματισμού:
     a. Εκτέλεση κάθε προγράμματος και ανάθεση αντίστοιχης τιμής καταλληλότητας.
     b. Δημιουργία νέου πληθυσμού υπολογιστικών προγραμμάτων, εφαρμόζοντας στα προγράμματα που πιθανοτικά επιλέχθηκαν βάσει τιμής καταλληλότητας, οι τελεστές αναπαραγωγής (με αντιγραφή του προγράμματος στη νέα γενιά), διασταύρωσης (με δημιουργία δύο νέων προγραμμάτων από δύο γονείς του πληθυσμού) και μετάλλαξης (με δημιουργία ενός προγράμματος, αλλάζοντας ένα τυχαίο μέρος του γονέα).

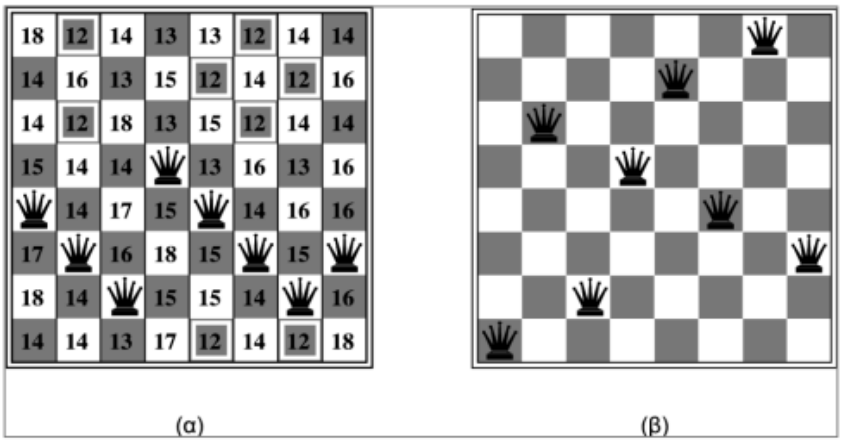
3. Το πρόγραμμα που θα οριστεί ως καλύτερο βάσει της παραπάνω μεθόδου, ορίζεται και ως η καλύτερη λύση μέχρι στιγμής, υποψήφιο για να είναι η λύση ή τουλάχιστον η προσεγγιστικά καλύτερη λύση για το εκάστοτε πρόβλημα.[24][26][27]

1.5 Παραδείγματα προβλημάτων βελτιστοποίησης

Πάρα πολλά προβλήματα βελτιστοποίησης έχουν προκύψει, καταγραφεί και συνεχίζουν να προκύπτουν καθημερινώς, άλλοτε απλά και άλλοτε περίπλοκα αρκετά, ώστε ελάχιστες μέθοδοι βελτιστοποίησης να είναι ικανές να τα λύσουν, έστω και προσεγγιστικά. Παρακάτω εμφανίζονται ορισμένα γνωστά προβλήματα βελτιστοποίησης που έχουν καταγραφεί μέσα στα χρόνια.

1.5.1 Το πρόβλημα των 8 βασιλισσών

Ένα κλασσικό παράδειγμα προβλήματος περιορισμών, καθώς η απλή του διαδεδομένη μορφή απαιτεί να τοποθετηθούν 8 βασίλισσες σε μία σκακιέρα, διαστάσεων 8 x 8, με την κάθε μία σε ξεχωριστή γραμμή και ξεχωριστή στήλη έτσι ώστε να μην απειλούν η μία την άλλη, είναι το πρόβλημα των 8 βασιλισσών. Το πρόβλημα αυτό σαφώς και ορίζεται για περισσότερες από 8 βασίλισσες, με τη δυσκολία του προβλήματος να αυξάνεται εκθετικά, αλλά για λόγους απλότητας, εδώ θα αναφερθεί το πρόβλημα των 8 βασιλισσών. Η πρώτη έκδοση αυτού του προβλήματος εμφανίστηκε το 1848 και μετά από αρκετά χρόνια αποδείχθηκε ότι το πρόβλημα αυτό έχει 92 διαφορετικές λύσεις. Μπορεί να επιλυθεί εύκολα με τη χρήση αλγορίθμων τοπικής βελτιστοποίησης (π.χ. με χρήση του αλγορίθμου “Αναρρίχησης λόφου”, που έχει αναλυθεί σε προηγούμενη ενότητα), αλλά είναι και αρκετά χρήσιμο για την αξιολόγηση της απόδοσης διαφόρων αλγορίθμων βελτιστοποίησης, όσον αφορά την ικανοποίηση περιορισμών. Η επίλυση του προβλήματος ξεκινά με τις βασίλισσες να βρίσκονται σε τυχαία αρχική θέση, καθεμία στη στήλη της, και τη συνάρτηση διαδοχών να επιστρέφει όλες τις δυνατές καταστάσεις που παράγονται από τη μετακίνηση μίας μόνο βασίλισσας σε άλλο τετράγωνο της ίδιας στήλης (βλέπε εικόνα 1.5α), έστω h, που είναι ο αριθμός των ζευγών που αλληλεπιδρούν είτε άμεσα είτε έμμεσα μεταξύ τους. Στη συγκεκριμένη περίπτωση η καλύτερη τιμή είναι η h = 12, οπότε ο αλγόριθμος, αν υπάρχουν περισσότερες από μία καλύτερες λύσεις, επιλέγει μία τυχαία και τοποθετεί εκεί τη μία βασίλισσα. Αυτό επαναλαμβάνεται μέχρι καμία βασίλισσα να απειλεί άλλη (βλέπε εικόνα 1.5β για μία από τις πιθανές λύσεις). [38][39][40]



Εικόνα 1.7 α) Τυχαία αρχική κατάσταση, Εικόνα 1.7 β) Μία από τις πιθανές λύσεις

1.5.2 Το πρόβλημα του πλανόδιου πωλητή

Ένα ακόμη κλασσικό πρόβλημα περιορισμών, είναι το πρόβλημα του πλανόδιου πωλητή. Στο πρόβλημα αυτό, αναπαρίσταται ένας πωλητής ο οποίος ξεκινά από την πόλη του και προσπαθεί να περάσει από όλες τις άλλες πόλεις μία μόνο φορά καταλήγοντας πάλι στη δική του. Η μορφή του προβλήματος αποτελείται από ένα τυχαίο μη κατευθυνόμενο γράφημα, έστω $G_n = (V_n, E_n)$, όπου V το πλήθοςτων κόμβων, όπου ένας κόμβος αναπαριστά μία πόλη, και Ε το πλήθος των ακμών, με κάθε ακμή να ενώνει δύο πόλεις και να φέρει το κόστος που απαιτεί η μεταφορά από τη μία πόλη στην άλλη. Το μέγεθος του γραφήματος δεν έχει περιορισμό, με προφανέστατη αυξανόμενη δυσκολία ανάλογη του μεγέθους. Σκοπός του αλγορίθμου είναι να βρει τη διαδρομή με το μικρότερο συνολικό κόστος. Ορισμένοι αλγόριθμοι κατάλληλοι για την επίλυση του προβλήματος αυτού, οι οποίοι έχουν αναλυθεί σε προηγούμενη ενότητα, είναι η μέθοδος διακλάδωσης και φράγματος (branch and bound) και η μέθοδος προσομοιωμένης ανόπτησης (simulated annealing). [41]

2 Μέθοδοι Βελτιστοποίησης

2.1 Διαχωρισμός μεθόδων βελτιστοποίησης

Όπως αναφέρθηκε και στην ενότητα της βελτιστοποίησης, δεν υπάρχει αλγόριθμος που να μπορεί να επιλύσει κάθε πρόβλημα βελτιστοποίησης πλήρως αποδοτικά. Συνήθως, η κάθε μέθοδος βελτιστοποίησης δημιουργείται και εφαρμόζεται σε μία συγκεκριμένη κατηγορία προβλημάτων. Φυσικά, μία μέθοδος μπορεί να εφαρμοστεί και σε άλλες κατηγορίες προβλημάτων, αλλά για να είναι πλήρως αποδοτική θα πρέπει να μετατραπεί, και συνήθως σε βαθμό που καταλήγει να αλλάζει η λογική του αλγορίθμου, δημιουργώντας έτσι έναν άλλον αλγόριθμο. Ο γενικός διαχωρισμός ανάμεσα στις μεθόδους αυτές, αναφέρεται κυρίως σε μαθηματικές και μεταυριστικές μεθόδους βελτιστοποίησης.

2.1.1 Μαθηματικές μέθοδοι

Ο κλάδος που χρησιμοποιεί εφαρμοσμένα μαθηματικά για την επίλυση προβλημάτων βελτιστοποίησης ονομάζεται μαθηματικός προγραμματισμός (Mathematical Programming). Ο μαθηματικός προγραμματισμός αποσκοπεί στην επιλογή του καλύτερου στοιχείου, σε σχέση με κάποιο κριτήριο, από ένα σύνολο διαθέσιμων λύσεων. Αυτό, στις περισσότερες περιπτώσεις επιτυγχάνεται μέσω μίας επαναληπτικής αναζήτησης, όπου σε κάθε βήμα της πραγματοποιείται ανάλυση ευαισθησίας για τον υπολογισμό των απαραίτητων παραγώγων των αντικειμενικών συναρτήσεων και συναρτήσεων περιορισμών. Κάποιες υποκατηγορίες του μαθηματικού προγραμματισμού εμφανίζονται παρακάτω:

- *Κυρτός προγραμματισμός* (*Convex Programming*). Επιλύει προβλήματα στα οποία οι αντικειμενικές συναρτήσεις και οι συναρτήσεις περιορισμών είναι κυρτές, κάτι που καθιστά εύκολη την επίλυσή τους αφού το τοπικό ελάχιστο είναι και το ολικό.

- *Γραμμικός προγραμματισμός* (*Linear Programming*). Αποτελεί μία ειδική περίπτωση του κυρτού προγραμματισμού, όπου οι αντικειμενικές συναρτήσεις και οι συναρτήσεις περιορισμών είναι γραμμικές, οπότε η περιοχή του συνόλου των εφικτών σχεδιασμών είναι ένα κυρτό πολύεδρο. Μία μέθοδος γραμμικού προγραμματισμού τείνει να αναζητά τη βέλτιστη λύση στα όρια του πολυέδρου και κυρίως στις γωνίες του.

- *Μη γραμμικός προγραμματισμός* (*Non Linear Programming*). Αντιμετωπίζει προβλήματα που κάποιες αντικειμενικές συναρτήσεις ή συναρτήσεις περιορισμών τους είναι μη γραμμικές. Η φιλοσοφία του μη γραμμικού προγραμματισμού προσεγγίζει περισσότερο τη ρεαλιστικότητα στα πραγματικά προβλήματα βελτιστοποίησης, καθώς οι τάσεις και οι παραμορφώσεις, που συνήθως ορίζονται ως περιορισμοί, παρουσιάζουν έντονη μη γραμμικότητα.

- *Ακέραιος προγραμματισμός* (*Integer Programming*). Στην κατηγορία αυτή, ανήκουν οι μέθοδοι βελτιστοποίησης που επιλύουν προβλήματα στα οποία κάποιες (Mixed Integer Programming), ή όλες (Pure Integer Programming) οι μεταβλητές σχεδιασμού παίρνουν μόνο διακριτές ακέραιες τιμές. Οι αντικειμενικές συναρτήσεις και οι συναρτήσεις περιορισμών μπορούν να είναι είτε γραμμικές (Linear Integer Programming) είτε μη γραμμικές (Non Linear Integer Programming).

- *Τετραγωνικός προγραμματισμός* (*Quadratic Programming*). Τα προβλήματα που επιλύει η κατηγορία αυτή περιέχουν αντικειμενικές συναρτήσεις δευτέρας τάξης, και συναρτήσεις περιορισμών που είναι γραμμικές συναρτήσεις των μεταβλητών σχεδιασμού. Σε κάποιες περιπτώσεις ειδικής μορφής των αντικειμενικών συναρτήσεων, το πρόβλημα βελτιστοποίησης μπορεί να είναι κυρτό.

Οι επαναληπτικοί αυτοί αλγόριθμοι συνήθως εγγυώνται την εύρεση της βέλτιστης λύσης μετά από έναν πεπερασμένο αριθμό επαναλήψεων, και χρησιμοποιούν τις

πρώτες (gradient) ή τις δεύτερες (hessian) παραγώγους των αντικειμενικών συναρτήσεων και συναρτήσεων περιορισμών, σε κάθε τους βήμα, ώστε να οδηγηθούν σταδιακά αλλά γρήγορα στη βέλτιστη λύση. Το σημαντικό τους όμως ελάττωμα είναι πως, αν και οι επαναλήψεις είναι λίγες, το υπολογιστικό κόστος για την ανάλυση ευαισθησίας σε κάθε επανάληψη είναι μεγάλο. Επιπλέον, οι μέθοδοι αυτές τείνουν να παγιδεύονται εύκολα σε τοπικά ακρότατα, και δεν είναι ικανοποιητικά αποδοτικές σε προβλήματα με έντονη μη γραμμικότητα ή με πολλούς περιορισμούς. [18]

2.1.2 Μεταευτερικές μέθοδοι

Οι μεταευριστικές μέθοδοι παίζουν έναν πολύ σημαντικό ρόλο στην τεχνολογία των καθολικών μεθόδων βελτιστοποίησης. Ως επί το πλείστων, αυτοί οι αλγόριθμοι υιοθετούν τη λογική της φύσης, με το μεγαλύτερο υποσύνολο των αλγορίθμων αυτών να εφαρμόζει τη λογική των αλγορίθμων σμήνους, αφού μιμούνται τη συμπεριφορά ζώων όπως ψαριών, πτηνών, μελισσών κτλ. Μέθοδοι όπως η "Βελτιστοποίηση σμήνους σωματιδίων (PSO)" και οι "Γενετικοί αλγόριθμοι (Genetic algorithms)" (οι οποίες θα αναλυθούν παρακάτω) καθώς και η "Προσομοιωμένη ανόποτηση (Simulated annealing)" (η οποία έχει ήδη αναλυθεί σε προηγούμενη ενότητα), έχουν αποδειχθεί χρήσιμες στην επίλυση σημαντικών προβλημάτων μηχανικής λήψης αποφάσεων. [34] Αν και ο σκοπός τους είναι παρόμοιος με τις μαθηματικές μεθόδους, οι μεταευριστικές μέθοδοι ωστόσο, δεν απαιτούν παραπάνω πληροφορίες σχετικές με το πρόβλημα, πέρα από τον υπολογισμό των συναρτήσεων περιορισμών και της αντικειμενικής συνάρτησης. Έτσι, μπορούν να εφαρμοστούν σε μεγαλύτερο εύρος προβλημάτων βελτιστοποίησης, ακόμη και αν οι υπολογισμοί των συναρτήσεων δεν είναι γνωστοί, οι αντικειμενικές συναρτήσεις του προβλήματος δεν είναι διαφορίσιμες, ή οι ακριβείς τιμές των παραμέτρων παρουσιάζουν αβεβαιότητες ή διακυμάνσεις.

Στους μεταευριστικούς αλγορίθμους χρησιμοποιείται ένα σύνολο πιθανών λύσεων, καθεμία από τις οποίες αντιπροσωπεύει έναν σχεδιασμό. Παρά το γεγονός ότι οι πιθανές αυτές λύσεις βρίσκονται στον χώρο ανεξάρτητα η μία με την άλλη, σε κάθε επανάληψη επικοινωνούν μεταξύ τους για να οριστεί η μέχρι τότε καλύτερη λύση. Έτσι εξερευνάται ο χώρος σχεδιασμού, και ο αλγόριθμος καθοδηγείται προς τη βέλτιστη λύση βασιζόμενος στην εμπειρία που έχει αποκτήσει από τις παραμέτρους σχεδιασμού που έχουν ήδη δοκιμαστεί, αντί της παραγώγου. Σε κάθε επανάληψη, για τον υπολογισμό των νέων διανυσμάτων μεταβλητών σχεδιασμού χρησιμοποιούνται συνήθως στοχαστικοί τελεστές και τυχαίοι αριθμοί σε αντίθεση με τους αιτιοκρατικούς μαθηματικούς υπολογισμούς. Τυχαίους αριθμούς μπορεί επίσης να περιέχουν οι αρχικές πιθανές λύσεις του αλγορίθμου.

Η τελική λύση που φτάνει ένας μεταυεριστικός αλγόριθμος, δεν είναι σίγουρο ότι θα είναι και η ολική βέλτιστη, ακόμη και αν αυτή υπάρχει. Αυτό συμβαίνει γιατί η μέθοδος κινείται πάντα προσεγγιστικά στη λύση, η ακρίβεια της οποίας εξαρτάται από τις συνθήκες του προβλήματος και τις παραμέτρους της μεθόδου. Το θετικό με την παραπάνω προσέγγιση, είναι πως η τυχαιότητα που διέπει τον αλγόριθμο στην

αναζήτηση της λύσης, προκαλεί συνήθως να μην εγκλωβίζεται σε τοπικό ακρότατο, κατάλληλη στρατηγική για αποδοτική επίλυση προβλημάτων καθολικής βελτιστοποίησης.

Βασικό μειονέκτημα των μεταυριστικών αλγορίθμων είναι πως αργούν περισσότερο στη σύγκλιση προς τη βέλτιστη λύση σε αντίθεση με τους μαθηματικούς αλγορίθμους, εξαιτίας της απαίτησης πολλών επαναλήψεων και δοκιμών. Παρόλα αυτά, ο υπολογισμός παραγώγων κατά την ανάλυση ευαισθησίας των μαθηματικών μεθόδων καταλήγει να είναι πιο χρονοβόρος από τις περισσότερες επαναλήψεις των μεταευριστικών μεθόδων, μιας και στις μεθόδους αυτές αποφεύγεται η ανάλυση ευαισθησίας. [18]

2.2 Η μέθοδος Gradient Descent

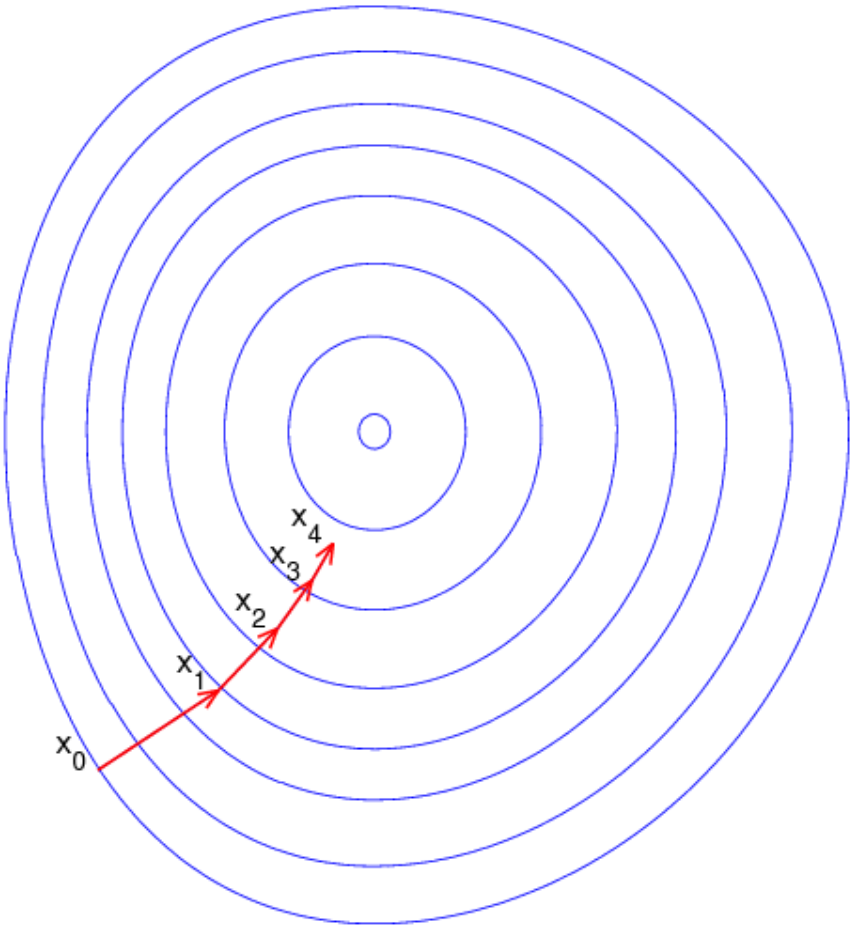
Η μέθοδος κλίσης κατάβασης (Gradient Descent) είναι ένας αλγόριθμος τοπικής επαναληπτικής αναζήτησης με διορθώσεις πρώτης τάξης, για τη βελτιστοποίηση κυρτών διαφορίσιμων αντικειμενικών συναρτήσεων. Είναι μία από τις πιο γνωστές μεθόδους βελτιστοποίησης και χρησιμοποιείται στους κλάδους των machine learning και deep learning. Η ιδέα πίσω από τη λογική του αλγορίθμου, είναι να κάνει επαναλαμβανόμενα βήματα προς την αντίθετη κατεύθυνση της κλίσης της συνάρτησης στο τρέχον σημείο, καθώς αυτή είναι και η κατεύθυνση της πιο απότομης καθόδου (Αντίστοιχα για τη μέθοδο κλίσης ανάβασης, η μέθοδος εκτελεί τα βήματα προς την κατεύθυνση της κλίσης της συνάρτησης). Για την καλύτερη κατανόηση της μεθόδου, χρήσιμο είναι να κατανοήσει κανείς την έννοια της κλίσης γενικότερα. Η κλίση (Gradient), είναι μία "Hill-Climbing" τεχνική και μπορεί κανείς να φανταστεί την κλίση ως την πλαγιά της συνάρτησης, υψηλότερη κλίση ισοδυναμεί με πιο απότομη πλαγιά. Αν υποθέσουμε ότι ένας άνθρωπος έχει σκοπό να ανέβει ένα λόφο με τα λιγότερα δυνατά βήματα, μπορεί να το επιτύχει με το να κάνει μεγαλύτερα βήματα όσο είναι στην πιο απότομη περιοχή του λόφου, και μικρότερα βήματα όσο πλησιάζει σε μικρότερη κλίση, όσο δηλαδή πλησιάζει στην κορυφή, ώστε να μην ξεπεράσει το επιθυμητό σημείο. Μαθηματικά η κλίση μπορεί να αναπαρασταθεί ως ένα διάνυσμα που περιέχει το σύνολο των παραγώγων ενός σημείου σε κάθε άξονά της:

$$\nabla F(p) = \begin{bmatrix} \frac{\partial F}{\partial x_1}(p) \\ . \\ . \\ . \\ \frac{\partial F}{\partial x_n}(p) \end{bmatrix}$$

Αν για παράδειγμα θέλουμε την κλίση στο σημείο (10, 10) για τη συνάρτηση $F(x) = 0.5x^2 + y^2$ τότε η παράγωγος ισούται με $x + 2y$ και έχοντας το διάνυσμα,

$$\nabla F(x,y) = \begin{bmatrix} x \\ 2y \end{bmatrix}, \text{ αυτό αντικαθίσταται με } \nabla F(10, 10) = \begin{bmatrix} 10 \\ 20 \end{bmatrix}$$

Παρατηρώντας και την εικόνα 2.1 παρακάτω που απεικονίζει ένα λόφο από ψηλά, φαίνεται πως το κόκκινο βέλος ανάμεσα στο X_0 και το X_1 είναι αρκετά μακρύτερο από το βέλος ανάμεσα στο X_3 και το X_4. Αν σκεφτεί κανείς την κλίση ως ένα διάνυσμα που περιέχει την κατεύθυνση και το μέγεθος κάθε βέλους (βήματος), τότε το συμπέρασμα είναι πως μικρότερη κλίση σημαίνει λιγότερο απότομη πλαγιά και μικρότερο μέγεθος βήματος(stepsize).



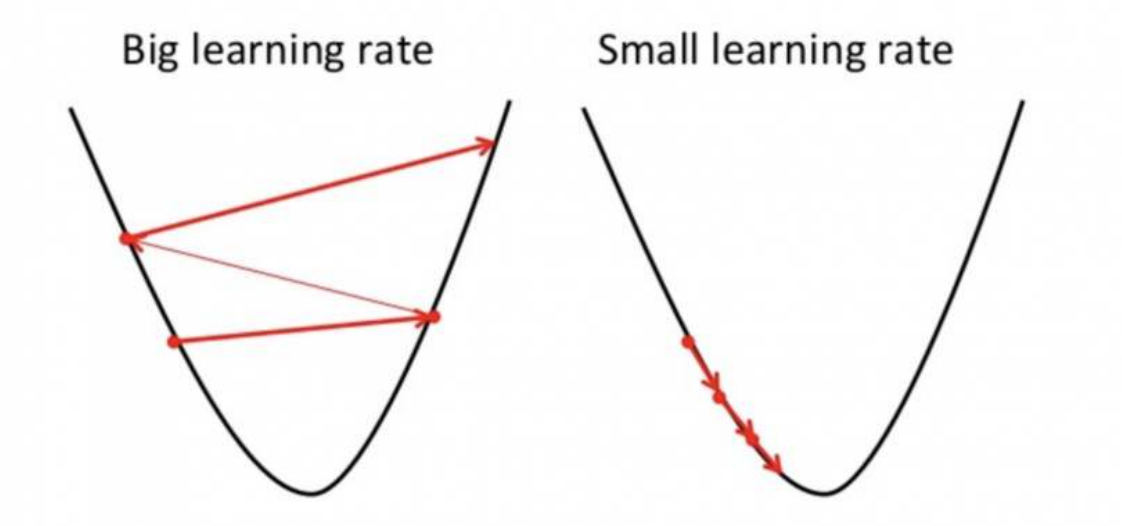
Εικόνα1.1 Επαναλαμβανόμενα βήματα προς λόφο με οπτική από πάνω προς τα κάτω

Αν φανταστεί κανείς τώρα ότι αντί να ανεβαίνει το λόφο κινείται προς τα κάτω, αναφέρεται στην κατάβαση, δηλαδή στη μέθοδο Gradient Descent. Η μέθοδος αυτή λοιπόν, επαναλαμβανόμενα υπολογίζει το επόμενο σημείο χρησιμοποιώντας την κλίση της συνάρτησης $\nabla F(p_n)$ στο τωρινό σημείο p_n, την κλιμακώνει (scale) με το learning rate (n) και την αφαιρεί από το τωρινό σημείο, δημιουργώντας την παρακάτω ισότητα:

$$p_{n+1} = p_n - n * \nabla F(p_n)$$

Από τα παραπάνω παρατηρείται ότι όσο πιο μικρό είναι το learning rate, τόσο μικρότερο το βήμα του αλγορίθμου, άρα τόσο μεγαλύτερη ακρίβεια έχει η μέθοδος στο να βρει το ακρότατο (μπορεί και να αργήσει αρκετά λόγω μικρών βημάτων), αλλά και τόσο πιο επικίνδυνο είναι το να μην προλάβει η μέθοδος να φτάσει στο επιθυμητό σημείο λόγω περιορισμένου προκαθορισμένου αριθμού βημάτων. Από την άλλη, όσο πιο μεγάλο είναι το learning rate, τόσο πιο πολύ θα αποκλίνει ο αλγόριθμος (πηδώντας μπρος πίσω με μεγάλα άλματα), με αποτέλεσμα να υπάρχει η πιθανότητα να μην μπορέσει να βρει ποτέ το επιθυμητό σημείο λόγω μεγάλων αλμάτων. Παρακάτω στην εικόνα 2.2, στα αριστερά

φαίνεται μία περίπτωση αναζήτησης με μεγάλο learning rate ενώ στα δεξιά φαίνεται μία περίπτωση αναζήτησης με μικρό learning rate. [3][29][30][31]



Εικόνα 2.2 α) Αναζήτηση με μεγάλο learningrate, b) Αναζήτηση με μικρό learningrate

2.3 Η μέθοδος BFGS

Η μέθοδος BFGS δημιουργήθηκε το 1970 και πήρε το όνομά της από τα αρχικά των ονομάτων των δημιουργών της, Broyden, Fletcher, Goldfarb και Shanno. Η απλή της έκδοση, είναι μία τοπική επαναληπτική μέθοδος που χρησιμεύει στην επίλυση μη γραμμικών, χωρίς περιορισμούς, κυρτών προβλημάτων βελτιστοποίησης. Ανήκει στην ομάδα των Quasi-Newton μεθόδων, επεκτάσεις της μεθόδου Newton (η οποία έχει αναλυθεί σε προηγούμενη ενότητα), με διορθώσεις δεύτερης τάξης, και συγκεκριμένα θεωρείται η πιο αποδοτική Quasi-Newton μέθοδος. Η λογική που χρησιμοποιεί για την προσέγγιση της λύσης σε κάθε βήμα είναι ο καθορισμός της κατεύθυνσης καθόδου με το να προβλέπει την κλίση της συνάρτησης χρησιμοποιώντας πληροφορίες καμπύλης. Χειρίζεται διπλά διαφοροποιήσιμες κυρτές αντικειμενικές συναρτήσεις, ξεκινώντας από ένα τυχαίο σημείο στο χώρο σχεδιασμού και αρχίζοντας να κινείται κατηφορικά με επαναληπτικά βήματα, ανάλογα με την αρνητική τιμή κλίσης της συνάρτησης σε κάθε σημείο. Αν και η μέθοδος Newton έχει την ικανότητα να χρησιμοποιεί τον όρο της δεύτερης τάξης, υπολογίζοντας την ανάστροφο του Εσσιανού πίνακα (hessian matrix), ώστε να μειώνει αρκετά τα βήματα του αλγορίθμου και το χρόνο σύγκλισης προς το ελάχιστο, αυτή είναι μία πράξη αρκετά χρονοβόρα καθώς ο υπολογισμός του Εσσιανού πίνακα και η αναστροφή του σε κάθε βήμα, χρειάζεται αρκετούς υπολογιστικούς πόρους. Αντ' αυτού, η μέθοδος BFGS υπολογίζει μία προσέγγιση του Εσσιανού πίνακα σε κάθε βήμα:

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{s_k^T y_k},$$

όπου k είναι ο αριθμός επανάληψης μέχρι τώρα, S είναι η διαφορά μεταξύ των σημείων x_{k+1} και $x_κ$, y είναι η διαφορά μεταξύ των τιμών κλήσης της συνάρτησης των παραπάνω σημείων $(\nabla F(x_{k+1}) - \nabla F(x_k))$ και B είναι η προσέγγιση του Εσσιανού πίνακα. Έχοντας υπολογιστεί η νέα προσέγγιση B_{k+1},κατ' επέκταση μπορεί να ικανοποιηθεί και η μέθοδος τέμνουσας (secant method), η οποία μπορεί να θεωρηθεί και ως προσέγγιση της πεπερασμένης διαφοράς της μεθόδου Newton. Παρακάτω φαίνεται η μέθοδος τέμνουσας

$$B_{k+1} * S_k = y_k$$

Σημαντικό είναι να σημειωθεί πως η μέθοδος τέμνουσας είναι δυνατόν να ικανοποιηθεί μόνο αν ισχύει η παρακάτω ανίσωση:

$$S_k^T * y_k > 0,$$

η οποία είναι γνωστή και ως συνθήκη κυρτότητας. Για προβλήματα μίας διαστάσεως, η BFGS πετυχαίνει την προσέγγιση αυτή με το να χρησιμοποιήσει τη μέθοδο της τέμνουσας (secant method), όπως φαίνεται και παραπάνω. Για προβλήματα περισσότερων της μίας διαστάσεως όμως, το αποτέλεσμα της παραπάνω διαδικασίας καταντά απροσδιόριστο. Για να μπορέσει η μέθοδος λοιπόν να προσδιορίσει την επόμενη προσέγγιση B, θα χρειαστεί να εφαρμοστούν περισσότεροι περιορισμοί. Τέτοιοι περιορισμοί είναι ο πίνακας B να είναι συμμετρικός και θετικά ορισμένος, κάτι που θα πρέπει η μέθοδος να διατηρεί σε κάθε επανάληψη, καθώς και αρκετά επιθυμητό, είναι η απόσταση του B_{k+1} από τον B_k να είναι αρκετά μικρή σε κάθε επανάληψη k+1, γι' αυτό σκοπός είναι να ελαχιστοποιηθεί η τιμή της πράξης $||B_{k+1} - B_k||$. [2]42][43]

2.4 Γενετικοί αλγόριθμοι

Οι γενετικοί αλγόριθμοι δημιουργήθηκαν από τον John Holland και τους συνεργάτες τουτις δεκαετίες του 60' και 70' στο πανεπιστήμιο του Μίσιγκαν, βασιζόμενοι στη θεωρία της "φυσικής επιλογής" που ανέπτυξε ο θεμελιωτής της ιστορίας της εξέλιξης, Κάρολος Δαρβίνος, το 1859 στο βιβλίο του με τίτλο "Η καταγωγή των ειδών". Ο Holland εισήγαγε την έννοια των προτύπων ομοιότητας ή αλλιώς σχημάτων, στην προσπάθεια να ποσοτικοποιηθεί η λειτουργία των γενετικών αλγορίθμων. [6][7]

Οι γενετικοί αλγόριθμοι χρησιμοποιούνται σε ευρύ πεδίο εφαρμογών και χρησιμεύουν πολύ σε προβλήματα βελτιστοποίησης. Ιδίως ωφέλιμοι σε περιπτώσεις που η αντικειμενική συνάρτηση (Objective function) έχει πολλά ακρότατα ή ασυνεχείςπαραγώγους. Πιθανή λύση αποτελεί ένα άτομο, το οποίο αναπαρίσταται με μία αλυσίδα γονιδίων (χρωμόσωμα) (π.χ. [11010001]) όπου γονίδιο είναι το κάθε ψηφίο της

αλυσίδας και αλληλόμορφο ονομάζεται η κάθε διαφορετική τιμή που μπορεί ναπάρει το γονίδο αυτό. Το πλήθος των γονιδίων εξαρτάται από τις παραμέτρους του προβλήματος καθώς και την επιθυμητή ακρίβεια. Από όσα πιο πολλά γονίδια αποτελείται η κάθε αλυσίδα γονιδίων τόσο μεγαλύτερο είναι το μήκος του χρωμοσώματος και τόσο πιο χρονοβόρες και δύσκολες είναι οι προσομοιώσεις του αλγορίθμου. Αν και παρακάτω που αναλύονται οι λειτουργίες της διασταύρωσης και της μετάλλαξηςπαρουσιάζονται με δυαδική και δεκαδική κωδικοποίηση, σημαντικό είναι να σημειωθεί πως στην εργασία αυτή χρησιμοποιούνται προβλήματα βελτιστοποίησης σε δεκαδική κωδικοποίηση. [6]

Πρώτη βασική παράμετρος ενός γενετικού αλγορίθμου είναι ο ορισμός του μεγέθους του πληθυσμού (έστω PS) από δυνατές τυχαίες λύσεις (χρωμοσώματα) του προβλήματος που μελετάται (συνήθως η PS καθορίζεται από τον χρήστη και μένει σταθερή κατά την διαδικασία της βελτιστοποίησης). Τα χρωμοσώματα ανήκουν σε γενιές και κάθε χρωμόσωμα της πρώτης γενιάς αποτιμάται μέσω μίας σχετικής συνάρτησης (Fitness function) ώστε να προκύψει μία τιμή καταλληλότητας (Fitness Value) (μεγαλύτερη τιμή για πρόβλημα μεγιστοποίησης, μικρότερη για πρόβλημα ελαχιστοποίησης) για κάθε χρωμόσωμα. Σκοπός του αλγορίθμου είναι η παραγωγή της επόμενης γενιάς χρωμοσωμάτων μέσω συγκεκριμένων βιολογικών τελεστών (επιλογή, διασταύρωση, μετάλλαξη).

2.4.1 Επιλογή

Η μέθοδος της επιλογής (selection) βασίζεται στην θεωρία της φυσικής επιλογής και της επιβίωσης του ικανότερου – καταλληλότερου χρωμοσώματος. Έτσι, ανάμεσα σε δύο γενεές δημιουργείται ένας ενδιάμεσος πληθυσμός με έναν προκαθορισμένα τυχαίο τρόπο καθώς σε κάθε χρωμόσωμα αντιστοιχεί μία πιθανότητα επιβίωσης ανάλογα με την τιμή καταλληλότητάς του. Τα καλύτερα χρωμοσώματα έχουν περισσότερες πιθανότητες να επιλεχθούν για τον ενδιάμεσο πληθυσμό. Στην εργασία αυτή θα αναφερθούν η μέθοδος επιλογής της ρουλέτας (roulette wheel) και η μέθοδος επιλογής διαγωνισμού (tournament). [6]

2.4.1.1 Μέθοδος επιλογής διαγωνισμού (tournament)

Στην περίπτωση της μεθόδου του διαγωνισμού (tournament) η πιο κοινή έκδοση είναι η "n-τυχαία επιλογή" που βασική παράμετρος της μεθόδου είναι το μέγεθός της, η πίεση επιλογής (selection pressure) n, η οποία παίρνει ακέραιες τιμές και ορίζει το πλήθος των χρωμοσωμάτων που θα συμμετέχουν κάθε φορά σε έναν διαγωνισμό. Το χρωμόσωμα με την καταλληλότερη Fitness value νικάει και περνάει ως γονέας στον ενδιάμεσο πληθυσμό. Η διαδικασία αυτή συνεχίζεται Ps επαναλήψεις, μέχρι δηλαδή το πλήθος των υποψήφιων γονέων να είναι ίδιο με τον πλήθος του αρχικού πληθυσμού [15]. Να σημειωθεί ότι όσο μεγαλύτερη είναι η τιμή της n, τόσο μικρότερη πιθανότητα έχει να περάσει ως γονέας ένα χρωμόσωμα με μικρή Fitness value, γιατί τόσο μεγαλύτερη πιθανότητα έχει στο ίδιο τουρνουά να βρίσκεται ένα χρωμόσωμα με καλύτερη Fitness value. Τέλος, να σημειωθεί ότι στην έκδοση αυτή υπάρχει η δυνατότητα παραλλαγής, κατά την οποία μπορεί να διαφοροποιηθεί το μέγεθος n για κάθε βήμα και έτσι να κλιμακωθεί η επιλογή των γονέων με κατάλληλες τιμές της παραμέτρου n. [6][7][8][9]

2.4.1.2 Μέθοδος επιλογής της ρουλέτας (roulette wheel)

Στην περίπτωση της μεθόδου της ρουλέτας (roulette wheel), αρχικά καθορίζεται ο αριθμός n ο οποίος αντιπροσωπεύει το πλήθος των χρωμοσωμάτων που θα επιλεχθούν για διασταύρωση. Έπειτα χρησιμοποιείται η Fitness value του κάθε χρωμοσώματος για να υπολογιστεί η πιθανότητα p για το κάθε χρωμόσωμα αντίστοιχα, η οποία ισούται με το πηλίκο της Fitness value του προς το άθροισμα όλων των Fitness values. Παρακάτω φαίνεται και ο αντίστοιχος τύπος:

$$p_i = \frac{f_i}{\sum_{j=1}^{N} f_i},$$

όπου N είναι ο συνολικός πληθυσμός των χρωμοσωμάτων. Στην συνέχεια υπολογίζεται η αθροιστική πιθανότητα q για κάθε χρωμόσωμα, η οποία ισούται με το άθροισμα της p_i με το συνολικό άθροισμα όλων των p των προηγούμενων χρωμοσωμάτων. Αργότερα υπολογίζεται ένας τυχαίος αριθμός r, όπου $r \in (0, 1]$. Έχοντας υπολογίσει την q για κάθε χρωμόσωμα και τον αριθμό r, ο αλγόριθμος ξεκινά σειριακά και συγκρίνει την κάθε q_i με τον αριθμό r. Το πρώτο χρωμόσωμα που θα βρεθεί να έχει q_i μεγαλύτερη ή ίση του r επιλέγεται ως υποψήφιο προς διασταύρωση χρωμόσωμα και περνά στον ενδιάμεσο πληθυσμό. Για n-1 υπολειπόμενες φορές υπολογίζεται καινούριος τυχαίος αριθμός r και γίνονται συγκρίσεις της κάθε q_i με τον r από την αρχή. [7][8][9][10]

2.4.2 Διασταύρωση

Όταν ολοκληρωθεί η διαδικασία του τελεστή επιλογής και έχει συμπληρωθεί ο ενδιάμεσος πληθυσμός, ο επόμενος τελεστής που παίρνει μέρος είναι ο τελεστής της διασταύρωσης. Κατά τη διαδικασία αυτή τα νέα χρωμοσώματα ανασυνδιάζονται είτε από 2 διαδοχικούς είτε από 2 τυχαίους γονείς κάτι που εξαρτάται από τον αλγόριθμο. Από τον αλγόριθμο εξαρτάται επίσης και η πιθανότητα διασταύρωσης έστω p_c, η οποία σε μεγάλες τιμές (κοντά στο 1) προκαλεί η αναζήτηση του αλγορίθμου να γίνει με μικρό βήμα, που συνεπάγεται αναζήτηση σε όλο τον χώρο του προβλήματος και κατ' επέκταση τον αλγόριθμο να συγκλίνει προς την βέλτιστη λύση, πράξη πλήρως αποδοτική αλλά χρονοβόρα, ενώ σε μικρές τιμές προκαλεί ο αλγόριθμος να αρχίζει να αναζητά με άλματα, πιθανότατα να συγκλίνει γρηγορότερα στην βέλτιστη λύση, με τον κίνδυνο όμως να την ξεπεράσει ("αναπηδήσει") και να αρχίσει να αποκλίνει από αυτή.

2.4.2.1 Διασταύρωση με δυαδική κωδικοποίηση

Η πιο κλασσική και ευρέως αποδεκτή διαδικασία διασταύρωσης δυαδικής κωδικοποίησης είναι η "διασταύρωση ενός σημείου (single point crossover)".Η τεχνική ξεκινά με τα ζεύγη των γονέων να χωρίζονται σε ένα τυχαίο σημείο (γονίδιο) έστω KK, το οποίο παράγεται τυχαία και ανήκει στο διάστημα [0, L-1], όπου L το μήκος των γονέων. Κατά τον διαχωρισμό αυτόν τα δύο αποκομμένα σημεία από κάθε γονέα εναλλάσσονται και έτσι δημιουργούνται δύο νέα χρωμοσώματα τα οποία θα αντικαταστήσουν τους γονείς τους στη νέα γενιά. Σκοπός της παραπάνω διαδικασίας είναι να προκύψει έστω και ένα χρωμόσωμα φέροντας τα επιθυμητά χαρακτηριστικά και από τους δύο γονείς. [6][7][8][9][11]

Έστω πχ. δύο γονείς Α και Β όπου:

Γονιός Α: 0010011101

Γονιός Β: 1110010101

Και έστω ότι το ΚΚ ορίζεται τυχαία ως 4, τότε η διασταύρωση θα συμβεί στο $4^{ο}$ γονίδιο, άρα ο διαχωρισμός θα μοιάζει ως εξής: Α: 0010/011101 και Β: 1110/010101. Ύστερα από την εναλλαγή θα έχουμε τα δύο παιδιά. [6]

Απόγονος Α: 0010010101

Απόγονος Β: 0111011110

2.4.2.2 Διασταύρωση με δεκαδική κωδικοποίηση

Στην περίπτωση της διασταύρωσης με δεκαδική κωδικοποίηση, ο τρόπος εκτέλεσης του τελεστή είναι διαφορετικός. Πιο συγκεκριμένα ο πιο συνήθης τρόπος διασταύρωσης ονομάζεται "αριθμητική διασταύρωση (arithmetic)" και είναι ο εξής:

- Απόγονος1 = α * Γονέας1 + (1 - α) * Γονέας2
- Απόγονος2 = (1-α) * Γονέας1 + α * Γονεάς2

Όπου:

- Απόγονος1 και Απόγονος2 τα χρωμοσώματα που προκύπτουν μετά την διασταύρωση.
- Γονέας1 και Γονέας2 τα χρωμοσώματα του ενδιάμεσου πληθυσμού που υπέστησαν διασταύρωση.
- α, ένας τυχαίος αριθμός που ανήκει στο διάστημα [0, 1]. [14][15]

2.4.3 Μετάλλαξη

Η τελευταία διαδικασία είναι εκείνη της μετάλλαξης. Η μετάλλαξη όπως και στην φύση δρα σπάνια με βελτιωτική συμπεριφορά για τους οργανισμούς. Καθώς περνούν τα γονίδια στον απόγονο, με τη διαδικασία της αντιγραφής, υπάρχει μία μικρή πιθανότητα, η πιθανότητα μετάλλαξης, έστω p_m, να μεταβληθεί κάποιο γονίδιο. Είναι σημαντικό η πιθανότητα p_m να είναι πολύ μικρή γιατί διαφορετικά ο γενετικός αλγόριθμος καταλήγει να αναζητά την βέλτιστη λύση εντελώς τυχαία. Σημαντικό επίσης είναι η ανάγκη του να μικραίνει γραμμικά ανάλογα με το πλήθος των γενεών σύμφωνα με τον Hibbert [13], καθώς όσο περνούν οι γενιές και καλυτερεύουν τα χρωμοσώματα τόσο λιγότερη ανάγκη υπάρχει να αλλάξουν. Χάρις τη διαδικασία αυτή υπάρχει η πιθανότητα εισαγωγής καινούριας πληροφορίας και καλύτερων χαρακτηριστικών στον πληθυσμό, με αποτέλεσμα την αποφυγή εγκλωβισμού του αλγορίθμου σε τοπικό ακρότατο της αντικειμενικής συνάρτησης (objective function).Πιο συγκεκριμένα όσον αφορά τον παράγοντα για το αν θα συμβεί ή όχι η μετάλλαξη, δημιουργείται ξανά ένας τυχαίος αριθμός, έστω r_c, στο διάστημα [0,1]. Αν ο αριθμός r_c είναι μικρότερος ή ίσος της πιθανότητας p_m, τότε πραγματοποιείται μετάλλαξη στον απόγονο [6][8][12][13].

2.4.3.1 Μετάλλαξη με δυαδική κωδικοποίηση

Ο πιο συνήθης τελεστής μετάλλαξης στο τεχνητό περιβάλλον για προβλήματα δυαδικής κωδικοποίησης είναι η μετάλλαξη σε επίπεδο "μπιτ". Ο τελεστής αυτός επιλέγει κάποιο γονίδιο ενός συγκεκριμένου χρωμοσώματος βάσει της πιθανότητας p_m και το αλλάζει από 0 σε 1 και αντίστροφα. Ανάλογα την φύση προβλήματος, την κωδικοποίηση του πληθυσμού και την τεχνική της μετάλλαξης που απαιτείται, στη μετάλλαξη εφαρμόζονται διαδοχικά τέσσερεις υποτελεστές: η πρόσθεση (add), ηδιαγραφή (omit), η αντικατάσταση (replace) και η ανταλλαγή (swap).[8]

- Στην πρόσθεση επιλέγεται ένα γονίδιο που δεν βρίσκεται στον απόγονο και προστίθεται σε αυτό σε μία τυχαία θέση της αλυσίδας.
- Στη διαγραφή επιλέγεται ένα γονίδιο από τον απόγονο και διαγράφεται.
- Στην αντικατάσταση επιλέγεται ένα γονίδιο που δεν βρίσκεται στον απόγονο και αντικαθιστά ένα τυχαίο γονίδιο του απογόνου.
- Στην ανταλλαγή επιλέγονται δύο τυχαία γονίδια μέσα στον απόγονο και αλλάζουν θέση μεταξύ τους.

2.4.3.2 Μετάλλαξη με δεκαδική κωδικοποίηση

Αν όμως χρησιμοποιηθεί δεκαδική κωδικοποίηση, η γενική διαφορά είναι ότι η διαδικασία μετάλλαξης θα αλλάξει την τιμή του χρωμοσώματος με μία άλλη τυχαία η οποία θα βρίσκεται μέσα σε ένα προκαθορισμένο διάστημα ορισμού τιμών. Πέραν αυτού, υπάρχουν όμως και κάποιοι τελεστές μετάλλαξης φτιαγμένοι να χειρίζονται αποκλειστικά προβλήματα δεκαδικής κωδικοποίησης. Τρείς από αυτούς είναι οι τελεστές ομοιόμορφης μετάλλαξης (uniform mutation), μετάλλαξης ορίων (boundary mutation) και ανομοιόμορφης μετάλλαξης (non-uniform mutation). [15]

2.4.3.2.1 Ομοιόμορφη μετάλλαξη (Uniform mutation)

Στην ομοιόμορφη μετάλλαξη επιλέγεται ένας γονέας X για να παραχθεί ο μεταλλαγμένος απόγονός του X'. Αρχικά επιλέγεται ένα τυχαίο γονίδιο έστω k από τον γονέα $X = (x_1,\ldots,x_k,\ldots,x_m)$ και παράγεται ο απόγονός του $X' = (x_1,\ldots,x_{k'},\ldots,x_m)$. Το γονίδιο που μεταλλάσσεται παίρνει την νέα τυχαία του τιμή μέσω ενός εύρους τιμών που έχει οριστεί προηγουμένως. [16]

2.4.3.2.2 Μετάλλαξη ορίων (Boundary mutation)

Όπως και η ομοιόμορφη μετάλλαξη έτσι και η μετάλλαξη ορίων χρησιμοποιεί έναν γονέα X για να πράξει έναν απόγονο X', με τη μόνη διαφορά ότι το νέο γονίδιο θα πάρει μία από τις ακραίες τιμές του προκαθορισμένου εύρους τιμών (κάθε μία εκ των δύο με ίση πιθανότητα). Ο τελεστής αυτός χρησιμεύει για προβλήματα βελτιστοποίησης που η λύση βρίσκεται ακριβώς ή κοντά στα όρια του διαστήματος των εφικτών λύσεων. [16]

2.4.3.2.3 Ανομοιόμορφη μετάλλαξη (Non-uniform mutation)

Για τη μείωση όμως τυχαίων μεταλλάξεων ο Michalewicz πρότεινε έναν δυναμικό ανομοιόμορφο τελεστή μετάλλαξης, ο οποίος έχει ως εξής:

Για κάθε χρωμόσωμα $X_{t,i} = \{X_1,\ldots,X_k,\ldots,X_m\}$ (όπου t ο αριθμός την τωρινής γενιάς) που το γονίδιο X_k επιλεχθεί για μετάλλαξη, το αποτέλεσμα θα είναι $X_{t,i} = \{X_1,\ldots,X_{k'},\ldots,X_m\}$ όπου,

$$X_{k'} = \begin{cases} x_k + \Delta(t, UB - x_k), \alpha\nu q = 0 \\ x_k - \Delta(t, x_k - LB), \alpha\nu q = 1 \end{cases}$$

με UBκαιLB να είναι τα πάνω και κάτω όρια αντίστοιχα για το εύρος τιμών που έχουμε θέσει για το $X_{k'}$, το q να είναι ένας τυχαίος δυαδικός αριθμός που ανήκει στο [0,1] και η συνάρτηση Δ(t,y), η οποία επιστρέφει έναν αριθμό που ανήκει στο εύρος [0,y] και ορίζεται από τον παρακάτω τύπο:

$$\Delta(t,y) = y * (1 - r^{(1 - \frac{t}{T})^b}),$$

όπου r είναι ένας τυχαίος αριθμός που ανήκει στο [0,1], T ο μέγιστος αριθμός γενεών που έχει τεθεί εξαρχής και b για λόγους απλότητας θα θεωρήσουμε ότι είναι μία σταθερά η οποία όσο αυξάνεται τόσο μειώνεται ο μέσος όρος ταχύτητας μείωσης του μεγέθους βήματος (step size). Τέλος, μπορεί να σημειωθεί ότι η τιμή που επιστρέφει η συνάρτηση Δ(t,y) πλησιάζει το 0 όσο παράγονται νέες γενεές (αύξηση του t). Σαν αποτέλεσμα επιτυγχάνεται ο αλγόριθμος να μπορεί να πραγματοποιήσει μία ομοιόμορφη αναζήτηση (όταν το t είναι μικρό) και αρκετά τοπικά σε πιο τελικά στάδια (όταν το t είναι μεγάλο). [16][17]

2.5 Βελτιστοποίηση σμήνους σωματιδίων (PSO)

Είναι μία από τις πιο διάσημες μεταευριστικές μεθόδους βελτιστοποίησης εμπνευσμένες από τη φύση. Δημιουργήθηκε από τους James Kennedy και Russell Eberhart το 1995. Η φιλοσοφία της μεθόδου ακολουθεί την κοινωνική συμπεριφορά ορισμένων πληθυσμών, όπως πτηνά, ψάρια, μέλισσες κτλ., όπου προς την αναζήτηση κάποιου συγκεκριμένου πόρου, επικοινωνούν μεταξύ τους, μοιράζονται την εμπειρία τους που έχουν αποκτήσει τα ίδια, αλλά και μέρος του πληθυσμού, σχετικά με την καλύτερη περιοχή αναζήτησης. Ο αλγόριθμος, υιοθετώντας τη φυσική αυτή συμπεριφορά, ακολουθεί τις εξής μετατροπές ώστε να μπορεί να εφαρμοστεί σε προβλήματα βελτιστοποίησης:

- Ο αλγόριθμος χρησιμοποιεί ένα σμήνος σωματιδίων (πληθυσμός).
- Κάθε σωματίδιο καταλαμβάνει μία θέση στο χώρο σχεδιασμού, η οποία θέση αποτελεί μία πιθανή λύση.

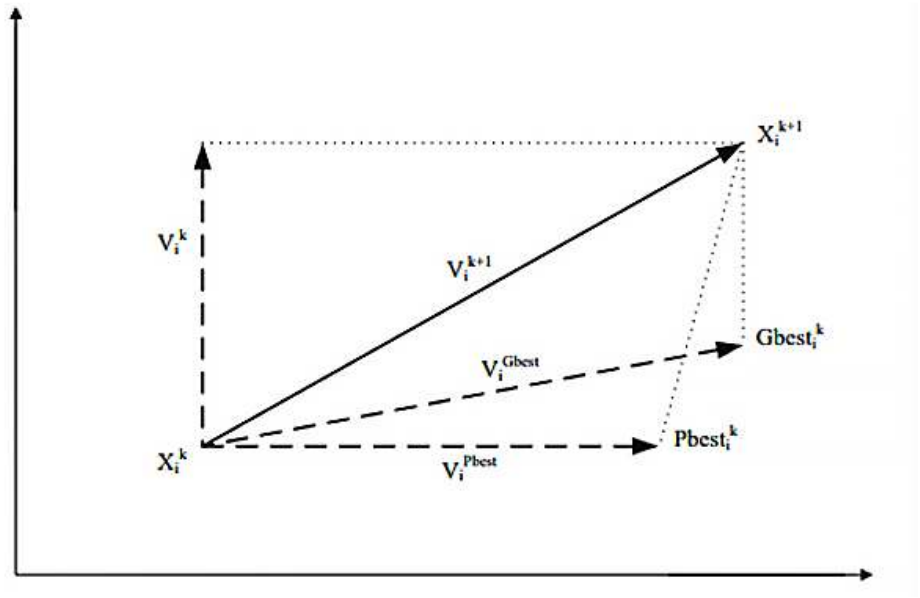
- Η καταλληλότητα κάθε σωματιδίου αντιστοιχεί στην τιμή της αντικειμενικής συνάρτησης ή της συνάρτησης ποιότητας σε περίπτωση ποινής λόγω περιορισμών.
- Τα σωματίδια κινούνται στο χώρο σχεδιασμού με κάποια ταχύτητα.
- Η κατεύθυνση και το μέτρο της ταχύτητας επηρεάζονται από την καλύτερη λύση που έχει βρει μέχρι στιγμής το σωματίδιο. Επομένως πραγματοποιείται τοπική αναζήτηση όπου χρησιμοποιείται η προσωπική μνήμη και εμπειρία του σωματιδίου αυτού.
- Η κατεύθυνση και το μέτρο της ταχύτητας επηρεάζονται από την καλύτερη λύση που έχουν βρει μέχρι τώρα οι γείτονες του σωματιδίου. Επομένως πραγματοποιείται καθολική αναζήτηση όπου χρησιμοποιείται η κοινωνική μνήμη και εμπειρία ολόκληρου του σμήνους.
- Εν τέλει, όλα τα σωματίδια συγκλίνουν σε τοπικά και ολικά βέλτιστες λύσεις.

Ο αρχικός πληθυσμός του αλγορίθμου, μεγέθους *N* και διαστάσεων *D* εκφράζεται ως $X = [X_1, X_2, \ldots, X_N]^T$, όπου *T* είναι ο τελεστής μετακίνησης. Το κάθε σωματίδιο X_iμε i = (1, 2, ..., N), εκφράζεται ως $X = [X_{i,1}, X_{i,2}, \ldots, X_{i,D}]$.Αντίστοιχα, η αρχική ταχύτητα του πληθυσμού εκφράζεται ως $V = [V_1, V_2, \ldots, V_N]^T$, με την ταχύτητα του κάθε σωματιδίου X_i να είναι $V = [V_{i,1}, V_{i,2}, \ldots, V_{i,D}]$. Παρακάτω εμφανίζονται οι βασικές εξισώσεις νέας θέσης και ταχύτητας με τις μεταβλητές i να κυμαίνεται από το 1 έως το *N* και j από το 1 έως το *D*.

$$X_{i,j}^{k+1} = X_{i,j}^{k} + V_{i,j}^{k+1},$$

$$V_{i,j}^{k+1} = w * V_{i,j}^{k} + c_1 * r_1 * (P\text{best}_{i,j}^{k} - X_{i,j}^{k}) + c_2 * r_2 * (G\text{best}_{i,j}^{k} - X_{i,j}^{k}),$$

όπου k είναι η τρέχουσα επανάληψη του αλγορίθμου, $X_{i,j}^{k}$και $V_{i,j}^{k}$είναι η τρέχουσα θέση και ταχύτητα του σωματιδίου αντίστοιχα, $P\text{best}_{i,j}^{k}$και $G\text{best}_{i,j}^{k}$είναι η τρέχουσα καλύτερη λύση που έχει βρει το σωματίδιο και το σμήνος αντίστοιχα μέχρι στιγμής, c_1και c_2είναι δύο συντελεστές επιτάχυνσης, που επηρεάζουν την ταχύτητα και εκφράζουν την εμπιστοσύνη στην καλύτερη λύση που έχει βρει το σωματίδιο και το σμήνος αντίστοιχα,r_1 καιr_2είναι δύο τυχαίοι αριθμοί που ανήκουν στο διάστημα [0,1] και τέλος, η μεταβλητή w που είναι ο τελεστής αδράνειας, ανήκει στο διάστημα [w_{min}, w_{max}] και εκφράζει την τάση του σωματιδίου να διατηρεί την ταχύτητα που είχε στην προηγούμενη επανάληψη. Παρακάτω στην εικόνα 2.3 εμφανίζεται οπτικά η τεχνική αναζήτησης της μεθόδου ΒΣΣ σε έναν πολυδιάστατο χώρο αναζήτησης.



Εικόνα 2.3 Τεχνική αναζήτησης της μεθόδου ΒΣΣ σε πολυδιάστατο χώρο αναζήτησης ([36])

Παρακάτω εμφανίζονται τα βήματα του αλγορίθμου της μεθόδου ΒΣΣ:

1. Ορισμός παραμέτρων w_{min}, w_{max}, c_1, και c_2
2. Αρχικοποίηση πληθυσμού σωματιδίων με αρχικές θέσεις X και ταχύτητες V
3. k = 1
4. Υπολογισμός συνάρτησης ποιότητας F για να οριστεί το καλύτερο σωματίδιο στην τρέχουσα θέση
5. Αν στην τρέχουσα θέση η τιμή ποιότητας είναι καλύτερη από την προηγούμενη *P*best, τότε *P*best = με την τιμή ποιότητας της τρέχουσας τιμής και την τρέχουσα θέση ως τη θέση με την καλύτερη τιμή ποιότητας μέχρι στιγμής. *G*best = με την τιμή ποιότητας του τρέχοντος καλύτερου σωματιδίου
6. $w = w_{max} - k * (w_{max} - w_{min})$ / Maxite (όπου Maxite η τιμή των συνολικών επαναλήψεων που έχουν οριστεί εξαρχής)
7. Εκτέλεση των δύο παραπάνω εξισώσεων για την ανανέωση θέσης και ταχύτητας για κάθε σωματίδιο για k = k+1
8. Εκτίμηση της νέας τιμής της συνάρτησης ποιότητας F για να οριστεί το καλύτερο σωματίδιο στη νέα τρέχουσα θέση
9. Ανανέωση της *P*best. Αν η νέα τιμή της συνάρτησης ποιότητας είναι μικρότερη της προηγούμενης, τότε η νέα *P*best ισούται με την τιμή της νέας θέσης του σωματιδίου, αλλιώς διατηρεί την προηγούμενη τιμή της
10. Ανανέωση της *G*best. Αν η νέα τιμή της συνάρτησης ποιότητας του τρέχοντος καλύτερου σωματιδίου σε όλο το σμήνος είναι μικρότερη της τιμής του προηγούμενα καλύτερου, τότε η νέα *G*best ισούται με την τιμή της *Pbest* του τρέχοντος καλύτερου σωματιδίου, αλλιώς διατηρεί την προηγούμενη τιμή της

11. Αν k<Maxite τότε k=k+1 και επιστροφή στο βήμα 6, αλλιώς συνέχεια στο βήμα 12
12. Εμφάνιση της τιμής της *G*best ως τη βέλτιστη λύση.

Η μέθοδος ΒΣΣ παρουσιάζει γρήγορη σύγκλιση στη βέλτιστη λύση, είναι εύκολη στον προγραμματισμό της εφόσον χρησιμοποιεί μόνο πρόσθεση και πολλαπλασιασμό διανυσμάτων και απαιτεί τη ρύθμιση λίγων παραμέτρων. Είναι αποδοτική για μη γραμμικούς, μη κυρτούς, και, ή συνεχείς χώρους σχεδιασμού. [18][36]

3. Το λογισμικό OPTIMUS

3.1 Μεταγλώττιση

Το λογισμικό OPTIMUS έχει αναπτυχθεί στο περιβάλλον προγραμματισμού Qt και σε γλώσσα C++. Είναι ένα αρχείο που περιέχει αρκετές μεθόδους βελτιστοποίησης καθώς και αρκετά προβλήματα προγραμματισμού. Για να χρησιμοποιήσει κάποιος το λογισμικό OPTIMUS, αρχικά θα πρέπει να έχει εγκατεστημένο το περιβάλλον Qt στον υπολογιστή του με τη γλώσσα C++, έπειτα θα πρέπει να ακολουθήσει τα εξής βήματα:

- Η πηγή του κώδικα βρίσκεται σε αυτό τον σύνδεσμο https://github.com/itsoulos/OPTIMUS/. Ο χρήστης θα πρέπει να μπει στο σύνδεσμο αυτό ώστε να κατεβάσει τον πηγαίο κώδικα.
- Στη συνέχεια θα χρειαστεί να γίνει αποσυμπίεση (unzip) του αρχείου.
- Μετά την αποσυμπίεση, μέσω του τερματικού (terminal/cmd), ο χρήστης θα πρέπει να μπει στον φάκελο του αρχείο με την εντολή cd ακολουθώντας με το μονοπάτι του αρχείου στη μνήμη του υπολογιστή. Σε περίπτωση που το αρχείο έχει κατεβεί στον φάκελο Home, στον οποίο ήδη κοιτάζει το τερματικό, η απλή εντολή θα είναι:
  cd OPTIMUS. (Για σιγουριά βέβαια, θα πρέπει πριν από την εντολή cd ο χρήστης να τρέξει την εντολή ls που εμφανίζει όλα τα αρχεία μέσα στον φάκελο που βρίσκεται το terminal εκείνη τη στιγμή).
- Απαραίτητη προϋπόθεση για να μπορεί να τρέξει το OPTIMUS, θα πρέπει να οριστούν δύο μονοπάτια ώστε να μπορεί το λογισμικό να ορίσει που είναι φάκελος που περιέχει τα προβλήματα, καθώς και για να μπορεί να βρει τις απαραίτητες βιβλιοθήκες για να τις χρησιμοποιήσει. Θα πρέπει λοιπόν ο χρήστης να εισάγει στα αρχεία /.bashrc και /etc/environment, τις εξής δύο γραμμές:
  1. export OPTIMUSPATH=~/OPTIMUS/
  2. export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib/:$OPTIMUSPATH/lib/:

Το παραπάνω βήμα θέλει αρκετή προσοχή καθώς οι δύο αυτές γραμμές θα πρέπει να είναι ακριβώς ίδιες και στα δύο αρχεία. Για να γίνει αυτό, θα πρέπει ο χρήστης να ακολουθήσει τα παρακάτω βήματα:

1. /.bashrc: Για να εισάγει τις δύο γραμμές στο αρχείο /.bashrc, το οποίο είναι κρυφό αρχείο, θα πρέπει να βρίσκεται στον φάκελο Home, και πατώνας "ctrl + H" εμφανίζονται τα κρυφά αρχεία. Μπαίνει στο αρχείο /.bashrc και εισάγει αυτές τις δύο γραμμές.
2. /etc/environment: Για να εισάγειτις δύο γραμμές στο αρχείο environment θα πρέπει να βρίσκεται στο τερματικό και ενώ βρίσκεται στον φάκελο Home, θα πρέπει να εκτελέσει την εντολή "sudo nano –w /etc/environment". Τότε χρησιμοποιώντας κειμενογράφο nano, ανοίγει το εσωτερικό του αρχείου και μπορεί να γράψει αυτές τις δύο γραμμές. Πατώντας "ctrl + X", αποθηκεύει τις αλλαγές που πραγματοποίησε στο αρχείο.

- Τέλος για να ολοκληρωθεί η μεταγλώττιση του λογισμικού OPTIMUS, θα πρέπει ο χρήστης να μπει μέσω τερματικού στον φάκελο του λογισμικού εκτελώντας την εντολή "cd OPTIMUS-master/", τότε θα εκτελέσει την εντολή "./compile.sh". Όταν ολοκληρωθεί η μεταγλώττιση το λογισμικό θα είναι έτοιμο για χρήση.

3.2 Κωδικοποίηση προβλημάτων

Τα προβλήματα βελτιστοποίησης αποτελούνται από παρόμοιες συναρτήσεις, οι οποίες ευθύνονται για την αρχικοποίηση του κάθε προβλήματος και την λογική πίσω από τον χειρισμό και αναζήτηση νέων καλύτερων λύσεων. Αν παραδείγματος χάρην αναλυθεί το πρόβλημα "rastrigin", παρακάτω εξηγούνται περιληπτικά οι συναρτήσεις που περιέχει:

- void init(QJsonObject): Η συνάρτηση αυτή ευθύνεται για την αρχικοποίηση του προβλήματος με το να περνά το αρχικοποιημένο σύνολο πιθανών λύσεων.
- int getdimension(): Επιστρέφει πάντα 2 καθώς τα προβλήματα είναι δύο διαστάσεων
- void getmargins(vector<Interval>): Σκοπός της είναι ο χειρισμός των περιθωρίων μεταξύ των πιθανών λύσεων στο χώρο σχεδιασμού.
- double funmin(vector<double>): Η συνάρτηση αυτή είναι υπεύθυνη για τον υπολογισμό της τιμής της αντικειμενικής συνάρτησης στο σημείο αυτό.
- double granal(vector<double>): Αντίστοιχα, η συνάρτηση αυτή είναι υπεύθυνη για τον υπολογισμό της τιμής της αντικειμενικής συνάρτησης στο επόμενο σημείο του αργορίθμου.
- QJsonObject done(vector<double>): Καλώντας τη συνάρτηση αυτή σημαίνει ότι η αναζήτηση έχει τελειώσει, καθώς η συνάρτηση αυτή καλείται στο τέλος μιας

και είναι ο destructor (συνάρτηση καταστροφής) της κλάσης Problem, όπου Problem είναι η κλάση από την οποία προέρχονται όλα τα προβλήματα.

3.3 Παράμετροι προβλημάτων

Κάθε είδος προβλήματος μπορεί να απαιτεί διαφορετική προσέγγιση της καλύτερη λύσης, κατ' επέκταση διαφορετική μέθοδο βελτιστοποίησης. Διαφορετικές μέθοδοι βελτιστοποίησης απαιτούν και την εκτίμηση διαφορετικών παραμέτρων για την ομαλή λειτουργία τους και την εύρεση του καλύτερου δυνατού αποτελέσματος. Υπάρχουν παράμετροι κοινές για όλες τις μεθόδους βελτιστοποίησης και παράμετροι για κάθε μέθοδο ξεχωριστά.

Κοινές παράμετροι:

- Adam_b1: Η τιμή της παραμέτρου b1 του αλγορίθμου adam.
- Adam_b2: Η τιμή της παραμέτρου b2 του αλγορίθμου adam.
- Adam_iterations: Ο μέγιστος αριθμός επαναλήψεων για τον αλγόριθμο adam.
- Adam_rate: Το μέγεθος βήματος (step size) του αλγορίθμου adam.
- Bfgs_iterations: Ο μέγιστος αριθμός επαναλήψεων για τον αλγόριθμο bfgs.
- Gradient_iterations: Ο μέγιστος αριθμός επαναλήψεων για τον αλγόριθμο gradient.
- Gradient_rate: Το μέγεθοςβήματος (step size) τουαλγορίθμου gradient.

Παρακάτω εμφανίζονται παραδειγματικά οι παράμετροι που χρειάζονται οι μέθοδοι βελτιστοποίησης Genetic και Pso.

Παράμετροι της μεθόδου Genetic:

- Chromosome_count: Ο αριθμός χρωμοσωμάτων.
- Genetic_crossover_type: Ο τύπος διασταύρωσης της μεθόδου (τιμές μεταξύ double, uniform, laplace και onepoint).
- Genetic_mutation_type: Ο τύπος μετάλλαξης της μεθόδου (τιμές μεταξύ double, random, pso).
- Localsearch_method: Η τοπική μέθοδος βελτιστοποίησης που θα χρησιμοποιηθεί (τιμές μεταξύ adam, bfgs, gradient, lbfgs).
- Localsearch_rate: Το μέγεθος βήματος (step size) της τοπικής μεθόδου που θα επιλεχθεί
- Max_generations: Ο μέγιστος αριθμός παραγωγής νέων γενεών.
- Max_iterations_of_algorithm: Μέγιστος αριθμός επαναλήψεων της μεθόδου Genetic
- Mutation_rate: Πιθανότητα να συμβεί μετάλλαξη.
- Selections_rate: Πιθανότητα να συμβεί επιλογή.

Παράμετροι της μεθόδου Pso:

- Localsearch_method: Η τοπική μέθοδος βελτιστοποίησης που θα χρησιμοποιηθεί (τιμές μεταξύ adam, bfgs, gradient, lbfgs).
- Pso_particles: Ο αριθμός του αρχικού πληθυσμού που θα χρησιμοποιήσει η μέθοδος Pso.
- Pso_max_generations: Μέγιστος αριθμός επαναλήψεων της μεθόδου Pso.
- Pso_c1: Η τιμή του συντελεστή επιτάχυνσης c1 της μεθόδου Pso.
- Pso_c2: Η τιμή του συντελεστή επιτάχυνσης c2 της μεθόδου Pso.
- Pso_inertia_starting_value: Αρχική τιμή του συντελεστή αδράνειας.
- Pso_inertia_ending_value: Τελική τιμή του συντελεστή αδράνειας.
- Pso_localsearch_rate: Τιμή ρυθμού εκκινήσεων μεθόδου τοπκής βελτιστοποίησης από τη μέθοδο Pso.

3.4 Κωδικοποίηση μεθόδου βελτιστοποίησης

Παρόμοιας λογικής με την ενότητα 2 είναι και αυτή η ενότητα, αλλά για την ανάλυση μεθόδων βελτιστοποίησης και όχι προβλημάτων. Παίρνοντας λοιπόν ως παράδειγμα τη μέθοδο βελτιστοποίησης “Genetic”, παρακάτω εξηγούνται περιληπτικά οι συναρτήσεις που περιέχει:

- Genetic(Problem): Είναι ο constructor (συνάρτηση κατασκευής) της μεθόδου, όταν καλείται αρχικοποιεί τις πρώτες απαραίτητες παραμέτρους της μεθόδου.
- void init(): Συνάρτηση υπεύθυνη για την αρχικοποίηση όλων των πεδίων της μεθόδου, των τύπων της διασταύρωσης και την μετάλλαξης καθώς και τη δημιουργία του αρχικού πληθυσμού των χρωμοσωμάτων.
- double fitness(Data): Συνάρτηση υπολογισμού της τιμής καταλληλότητας της αντικειμενικής συνάρτησης σε ένα σημείο.
- bool terminated(): Είναι η συνάρτηση τερματισμού της μεθόδου ανάλογα με τα κριτήρια τερματισμού που έχουν τεθεί.
- bool checkForGradientCriterion(Data): Ελέγχει για το αν χρειάζεται να εφαρμόσει τοπική βελτιστοποίηση ή όχι.
- void calcFitnessArray(): Υπολογίζει τις τιμές καταλληλότητας όλων των χρωμοσωμάτων.
- void select(): Επιλέγει το χρωμόσωμα με την καλύτερη τιμή καταλληλότητας ανάμεσα στο τρέχον και το επόμενο.
- void getTournamentElement(Data): Συνάρτηση δημιουργίας της διαδικασίας επιλογής μέσω διαγωνισμού.
- void tournament(Data, Data): Συνάρτηση εισαγωγής δύο χρωμοσωμάτων, για να πάρουν μέρος στη μέθοδο επιλογής μέσω διαγωνισμού.

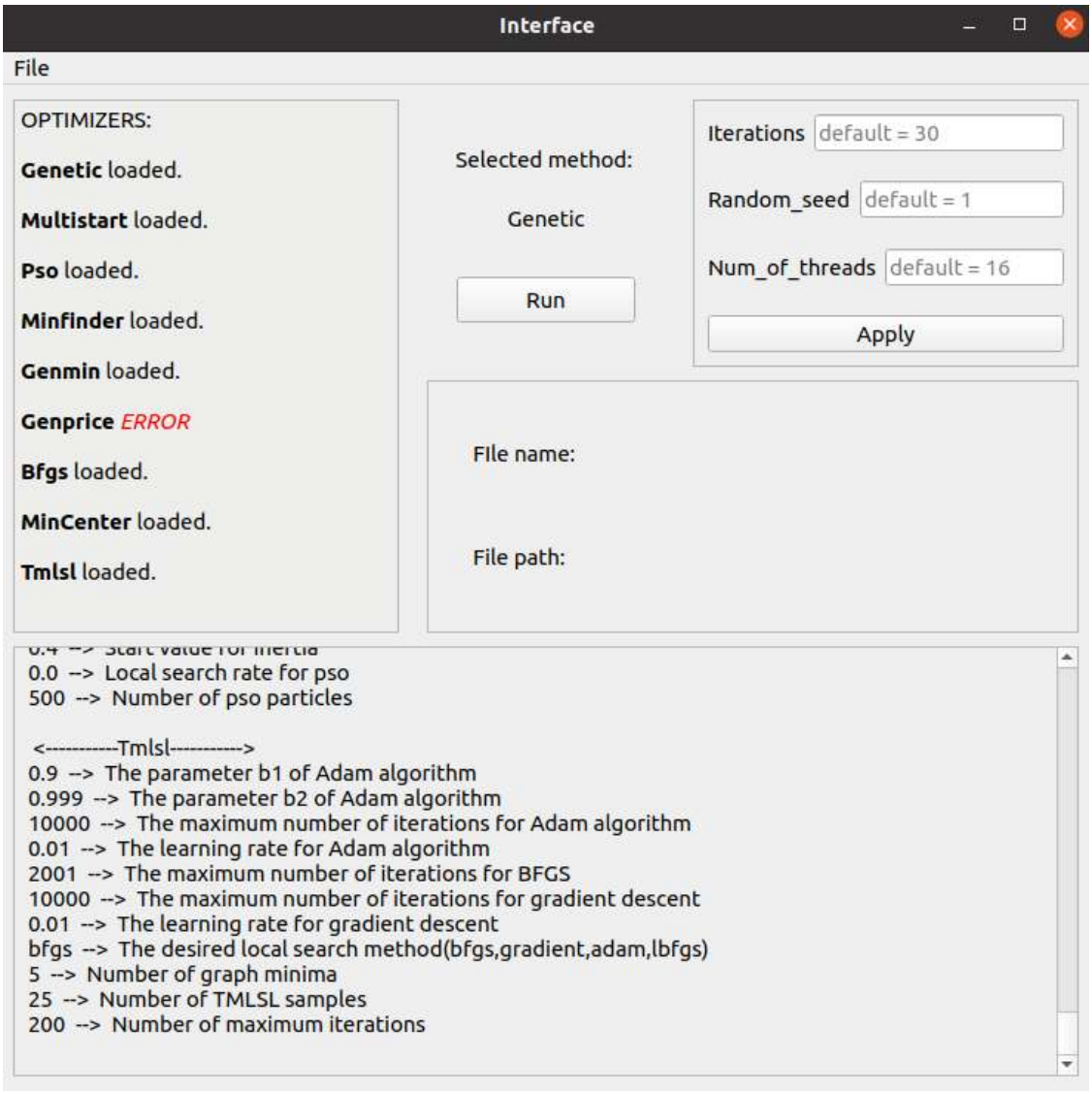
- void doubleCrossover(), void laplaceCrossover(), void uniformCrossover(), void onePointCrossover(): Οι 4 αυτές συναρτήσεις είναι υπεύθυνες για τον υπολογισμό τηςδιασταύρωσης δύο σημείων, διασταύρωσηςlaplace, ομοιόμορφης διασταύρωσης και διασταύρωσης ενός σημείου αντίστοιχα.
- void randomSearch(int): Υπεύθυνη για την τυχαία αναζήτηση μεταξύ χρωμοσωμάτων.
- void crossover(): Χειρίζεται τη διαδικασία της διασταύρωσης καθώς και ποια από τις παραπάνω 4 θα εφαρμοστεί κάθε φορά.
- void mutate(): Χειρίζεται την εφαρμογή και διαδικασία της μετάλλαξης.
- void step(): Συνάρτηση που υπολογίζει κάθε φορά το μέγεθος βήματος του αλγορίθμου στην αναζήτηση του επόμενου σημείου στο χώρο σχεδιασμού.
- void Solve():Η συνάρτηση αυτή καλείται για να ξεκινήσει τη διαδικασία λύσης του προβλήματος.
- void done(): Συνάρτηση που καλείται αφού ολοκληρωθεί η βελτιστοποίηση του προβλήματος και αρχικοποιεί τους δυναμικούς πίνακες της κλάσης.
- extern "C" GENETIC_EXPORT Optimizer *createOptimizer(Problem): Συνάρτησηδημιουργίαςνέουβελτιστοποιητή.

4. Η εφαρμογή

4.1 Εισαγωγή

Η εφαρμογή που έχει αναπτυχθεί σε αυτή την εργασία, αποτελεί μία διαπροσωπεία χρήστη (User Interface / UI) για το λογισμικό OPTIMUS, μέσω της οποίας ο χρήστης μπορεί να βρει τη βέλτιστη λύση σε προβλήματα βελτιστοποίησης επιλέγοντας κάθε φορά το επιθυμητό πρόβλημα και την κατάλληλη μέθοδο βελτιστοποίησης από τη λίστα των διαθέσιμων μεθόδων. Η εφαρμογή αναπτύχθηκε σε γλώσσα προγραμματισμού C++ και σε λειτουργικό σύστημα Linux (εξού και τα προβλήματα βελτιστοποίησης που χρησιμοποιούνται, τα οποία έχουν επέκταση ".so"). Η εφαρμογή δεν απαιτεί ταυτοποίηση του χρήστη ή οποιαδήποτε μορφή δημιουργίας λογαριασμού. Η εφαρμογή ξεκινά να τρέχει εμφανίζοντας το αρχικό της παράθυρο (βλέπε εικόνα 4.1). Στο πάνω μέρος βρίσκεται η μπάρα μενού με την επιλογή File, στα αριστερά εμφανίζεται σε μορφή λίστας η κατάσταση φόρτωσης των μεθόδων βελτιστοποίησης (παραδείγματος χάριν η Genprice εμφανίζει σφάλμα φόρτωσης που την καθιστά αδύνατη για χρήση), στη μέση βρίσκονται η μέθοδος που έχει επιλεχθεί από το χρήστη για τη λύση του προβλήματος (στην αρχή είναι προεπιλεγμένη η μέθοδος Genetic) και το κουμπί εκτέλεσης της διαδικασίας λύσης του προβλήματος, στα δεξιά 3 παράμετροι, κοινές για όλες τις μεθόδους βελτιστοποίησης, και τέλος, στο κάτω μέρος, ένα τερματικό για την εμφάνιση των αποτελεσμάτων των διαδικασιών βελτιστοποίησης καθώς και χρήσιμων πληροφοριών γιατο χρήστη. Όλες οι παράμετροι έχουν αρχικές προκαθορισμένες τιμές,

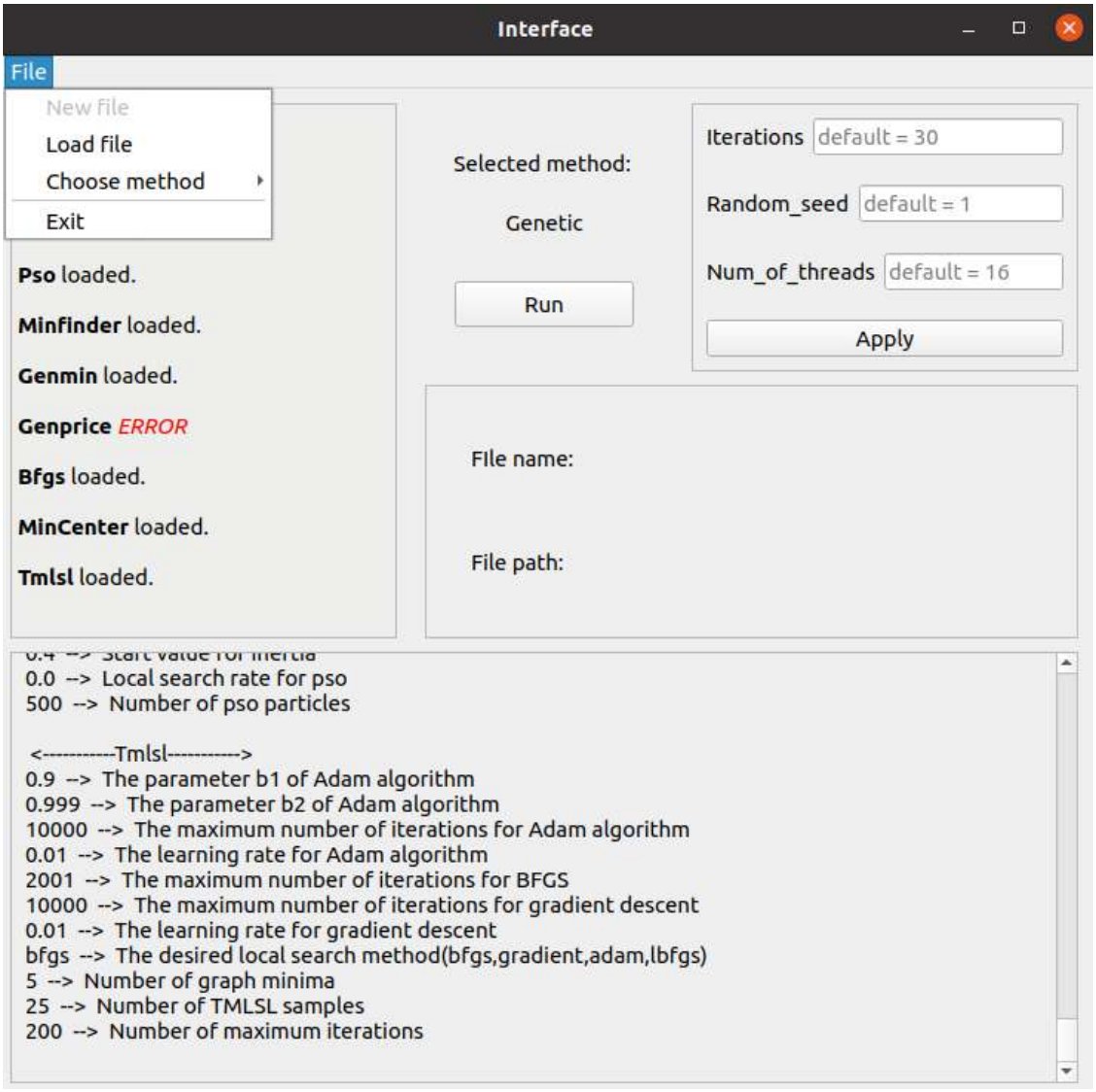
τις οποίες εννοείται πως ο χρήστης μπορεί να αλλάξει, όπως επίσης και όλα τα πεδία εμφανίζουν αρχικά την κάθε προκαθορισμένη τιμή της αντίστοιχης παραμέτρου.



Εικόνα 4.1 - Αρχικό παράθυρο της εφαρμογής κατά την εκτέλεσή της

4.2 Το μενού File

Μέσω του μενού File ο χρήστης μπορεί να αλληλεπιδράσει με τις περισσότερες επιλογές που παρέχει η εφαρμογή. Πιο συγκεκριμένα, η επιλογή File εμφανίζει ένα μενού με τις επιλογές New file / Νέο αρχείο, Load file / Φόρτωση αρχείου, Choose method / Επιλογή μεθόδου, Exit / Έξοδος (βλέπε εικόνα 4.2).

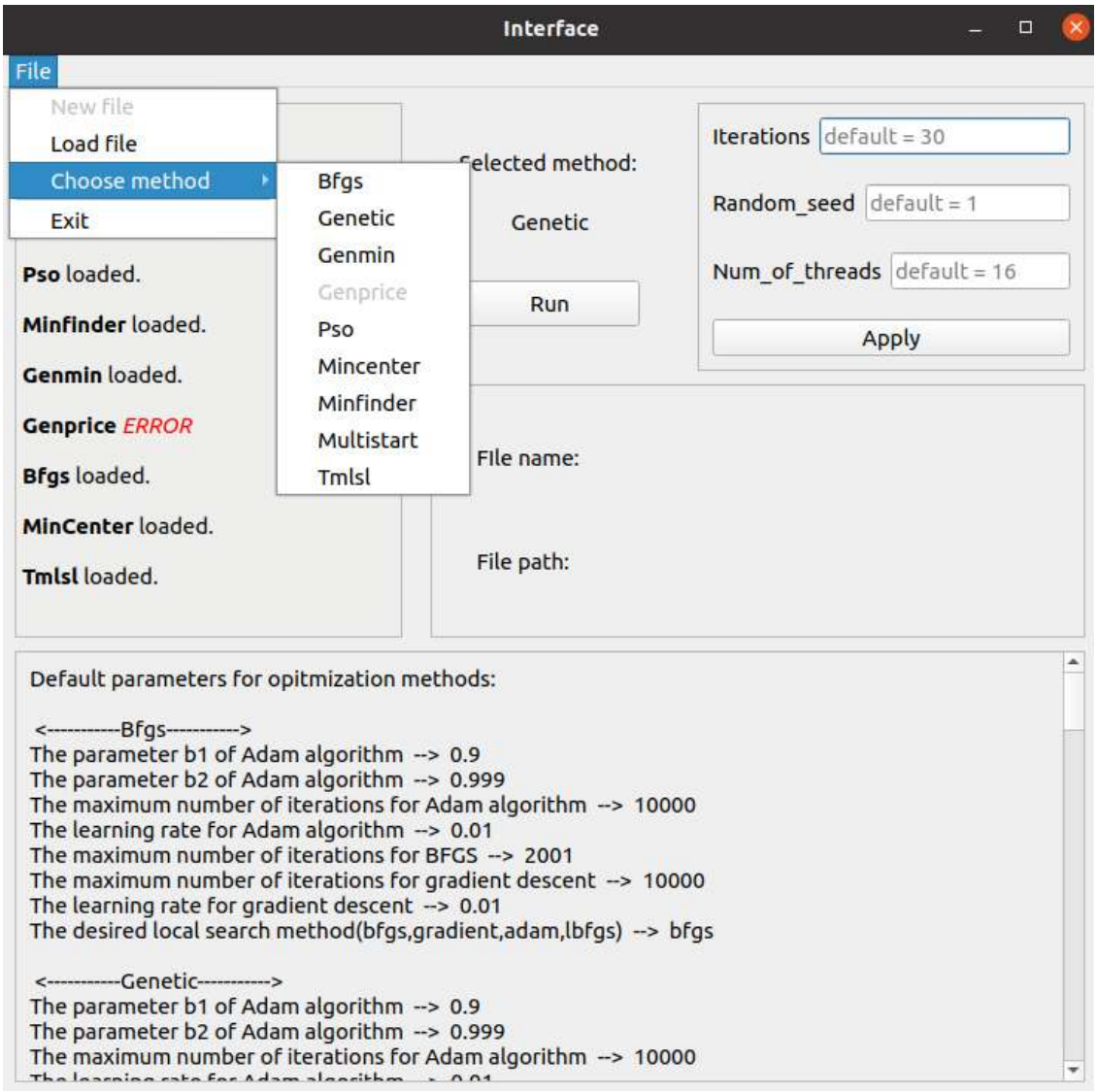


Εικόνα 4.2 Εμφάνιση επιλογών του μενού File

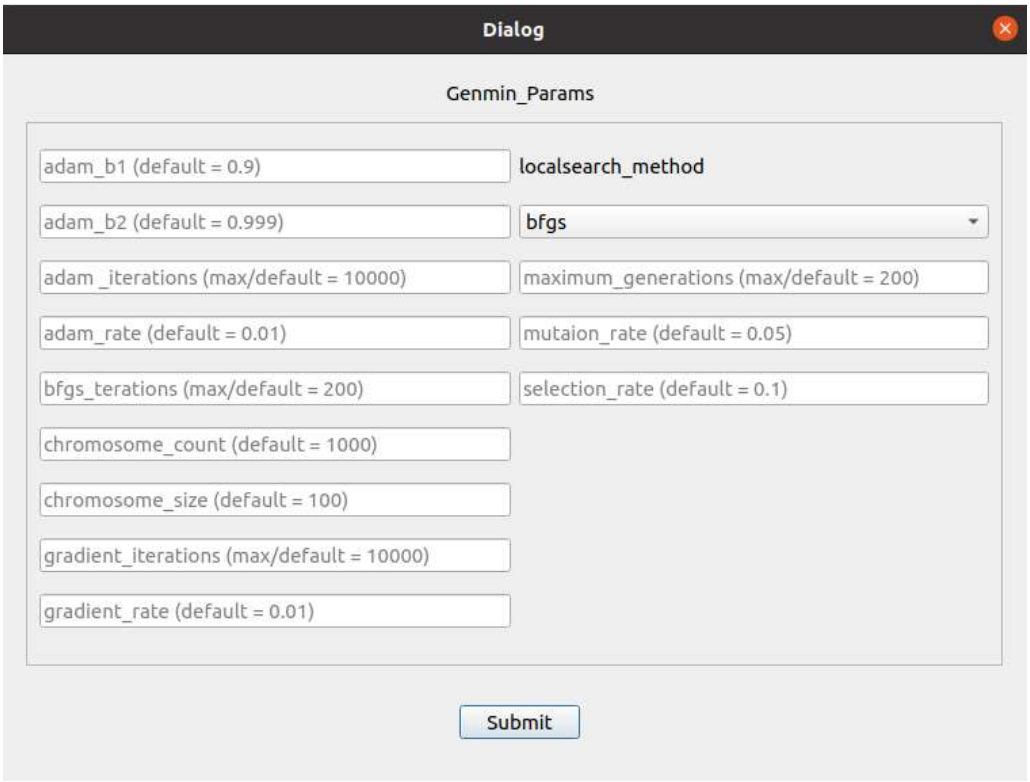
- Η επιλογή του Νέου αρχείου επανεκκινεί την εφαρμογή με το να αρχικοποιήσει ξανά όλες τις παραμέτρους στις αρχικές τους τιμές, που έχουν οριστεί από την εφαρμογή και να αποεπιλέξει το προηγουμένως επιλεγμένο αρχείο προβλήματος προς βελτιστοποίηση. Οι παραπάνω ενέργειες γίνονται αυτόματα όταν ανοίγει οι εφαρμογή, γι' αυτό και η επιλογή του νέου αρχείου δεν είναι διαθέσιμη στην αρχή.
- Η επιλογή της φόρτωσης αρχείου ανοίγει τον φάκελο στο μονοπάτι που έχει οριστεί στην εφαρμογή ώστε να επιλέξει ο χρήστης το επιθυμητό πρόβλημα προς βελτιστοποίηση.
- Η επιλογή της επιλογής μεθόδου, επεκτείνει το μενού με μία λίστα για την επιλογή της επιθυμητής μεθόδου βελτιστοποίησης (βλέπε εικόνα 4.3). Στην

εικόνα φαίνεται επίσης ότι η επιλογή της μεθόδου Genpriceείναι μη διαθέσιμη καθώς έχει υπάρξει πρόβλημα φόρτωσης. Η επιλογή κάποιας μεθόδου βελτιστοποίησης θα εμφανίσει το κατάλληλο παράθυρο με τις απαραίτητες παραμέτρους για τη μέθοδο αυτή (βλέπε εικόνα 4.4 στην οποία έχει επιλεχθεί η μέθοδος Genmin).

- Η επιλογή Exit, κάνει το προφανές με το να κλείσει το παράθυρο της εφαρμογής και να τερματίσει το πρόγραμμα.



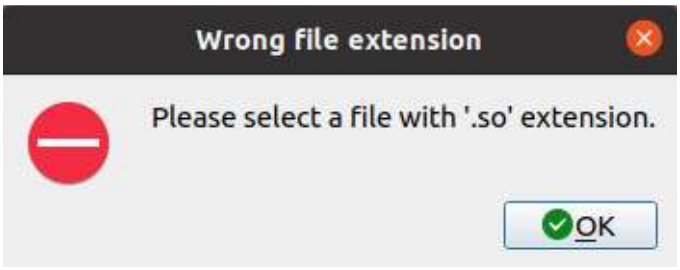
Εικόνα 4.3 Εμφάνιση λίστας μεθόδων βελτιστοποίησης



Εικόνα 4.4 Εμφάνιση αντίστοιχου παραθύρου κατά την επιλογή της μεθόδου Genmin από τη λίστα μεθόδων

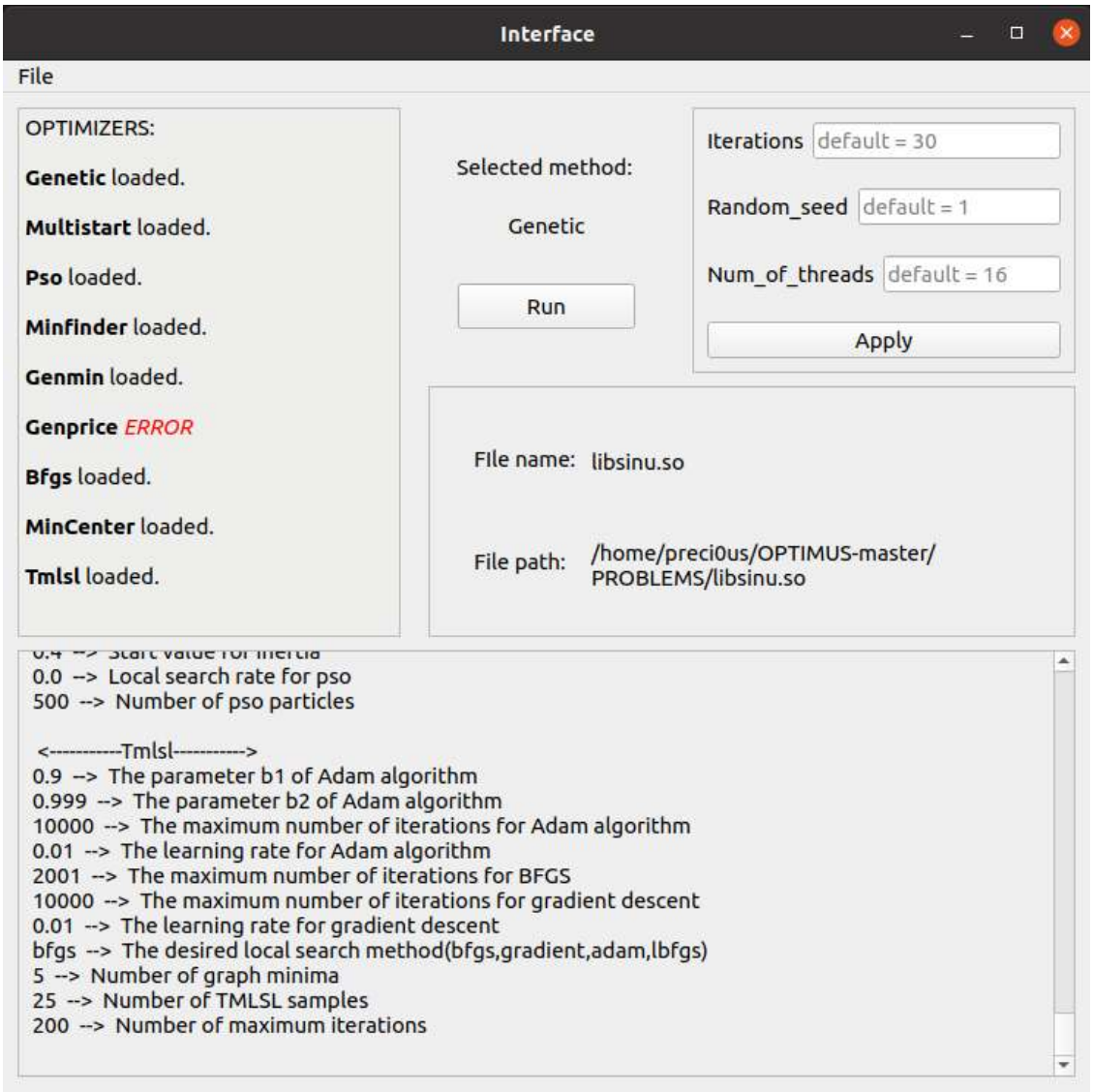
4.3 Παράδειγμα χρήσης της εφαρμογής

Ενώ βρίσκεται στο αρχικό παράθυρο ο χρήστης (βλέπε εικόνα 4.1), αυτό που σίγουρα πρέπει να κάνει ώστε να μπορέσει να τρέξει η εφαρμογή είναι να φορτώσει το επιθυμητό πρόβλημα προς βελτιστοποίηση. Σε περίπτωση επιλογής αρχείου με λανθασμένη επέκταση η εφαρμογή επιστρέφει τον χρήστη στο αρχική παράθυρο και παρέχει αντίστοιχο μήνυμα ειδοποίησης του χρήστη (βλέπε εικόνα 4.5 που αποτρέπει το χρήστη από την επιλογή αρχείου με επέκταση διαφορετική της “.so”).



Εικόνα 4.5 Ειδοποίηση του χρήστη για την επιτρεπτή επιλογή αρχείου προβλήματος

Όταν ο χρήστης επιλέξει επιτυχώς το πρόβλημα βελτιστοποίησης, ο χρήστης επιστρέφει στο αρχικό παράθυρο της εφαρμογής και στη μέση του παραθύρου (βλέπε εικόνα 4.6), αναγράφονται το όνομα του αρχείο και το μονοπάτι του στη μνήμη του υπολογιστή.



Εικόνα 4.6 Εμφάνιση πληροφοριών για το επιλεγμένο πρόβλημα βελτιστοποίησης

Όλα τα άλλα στοιχεία της εφαρμογής είναι ήδη αρχικοποιημένα. Έτσι λοιπόν, αν ο χρήστης πατήσει το κουμπί "Run" πριν επιλέξει κάποιο πρόβλημα θα εμφανιστεί το ανάλογο μήνυμα αναφέροντας την έλειψη επιλογής προβλήματος βελτιστοποίησης (βλέπε εικόνα 4.7).

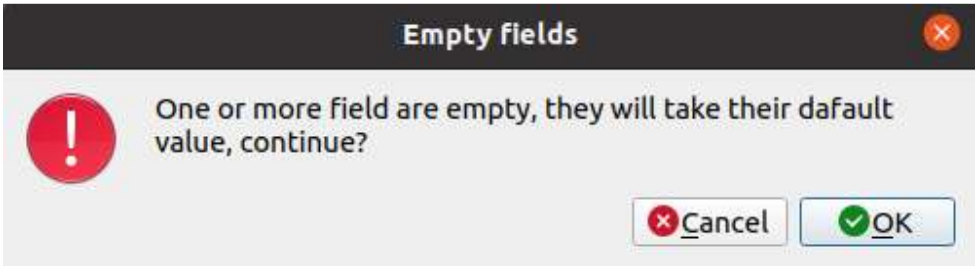
Εικόνα 4.7 Διάλογος ενημέρωσης για επιλογή προβλήματος βελτιστοποίησης

Εφόσον λοιπόν ο χρήστης επιλέξει το επιθυμητό πρόβλημα βελτιστοποίησης, το πρόγραμμα είναι ικανό να τρέξει και να παράξει ένα αποτέλεσμα. Αν ο χρήστης επιθυμεί να επιλέξει άλλη μέθοδο βελτιστοποίησης, επιλέγει μία από τη λίστα και εν συνεχεία εμφανίζεται το αντίστοιχο παράθυρο για να εισάγει τις παραμέτρους που επιθυμεί (βλέπε εικόνα 4.4 για παράδειγμα που ο χρήστης επιλέξει τη μέθοδο Genmin). Αν ο χρήστης θελήσει να ακυρώσει την επιλογή του, μπορεί να επιλέξει να κλείσει το παράθυρο παραμέτρων και μετά από αντίστοιχο διάλογο που εμφανίζεται στην εικόνα 4.8, οποιαδήποτε αλλαγή έκανε στο παράθυρο αυτό ακυρώνεται. Φυσικά, ο χρήστης μπορεί να κλείσει τον διάλογο και να συνεχίσει κανονικά την παραμετροποίηση της επιλεγμένης μεθόδου.

Εικόνα 4.8 Διάλογος προειδοποίησης κατά το κλείσιμο παραθύρου παραμέτρων

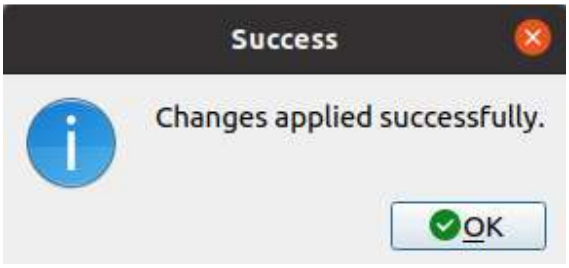
Αν εν τέλει ο χρήστης συνεχίσει με την παραμετροποίηση, και αφού επιλέξει το κουμπί “submit”, ακολουθούν δύο περιπτώσεις. Αν δεν έχει συμπληρώσει όλα τα πεδία παραμέτρων, εμφανίζεται διάλογος αναφέροντας πως οποιοδήποτε πεδίο που δεν έχει συμπληρωθεί θα του ανατεθεί η προκαθορισμένη τιμή από την εφαρμογή (βλέπε εικόνα 4.9). Μπορεί είτε να τον ακυρώσει και να επιστρέψει στο παράθυρο παραμετροποίησης, είτε να δεχθεί την ανάθεση των προκαθορισμένων τιμών για τα μη συμπληρωμένα πεδίακαι να επιστρέψει στο αρχικό παράθυρο. Αν από την άλλη, έχει συμπληρώσει όλα τα πεδία, οι νέες τιμές ενημερώνουν το πρόγραμμα, ο χρήστης επιστρέφει στο αρχικό παράθυρο και στη μέση αναγράφεται η μέθοδος που επέλεξε ο χρήστης. Η εφαρμογή έχει επίσης την ικανότητα να αποθηκεύει κάθε αλλαγή του χρήστη στις τιμές των

παραμέτρων, ώστε να τον διευκολύνει σε περίπτωση που επιστρέψει πάλι για περαιτέρω αλλαγές.



Εικόνα 4.9 Διάλογος ενημέρωσης του χρήστη κατά την έλειψη συμπλήρωσης όλων των πεδίων της μεθόδου

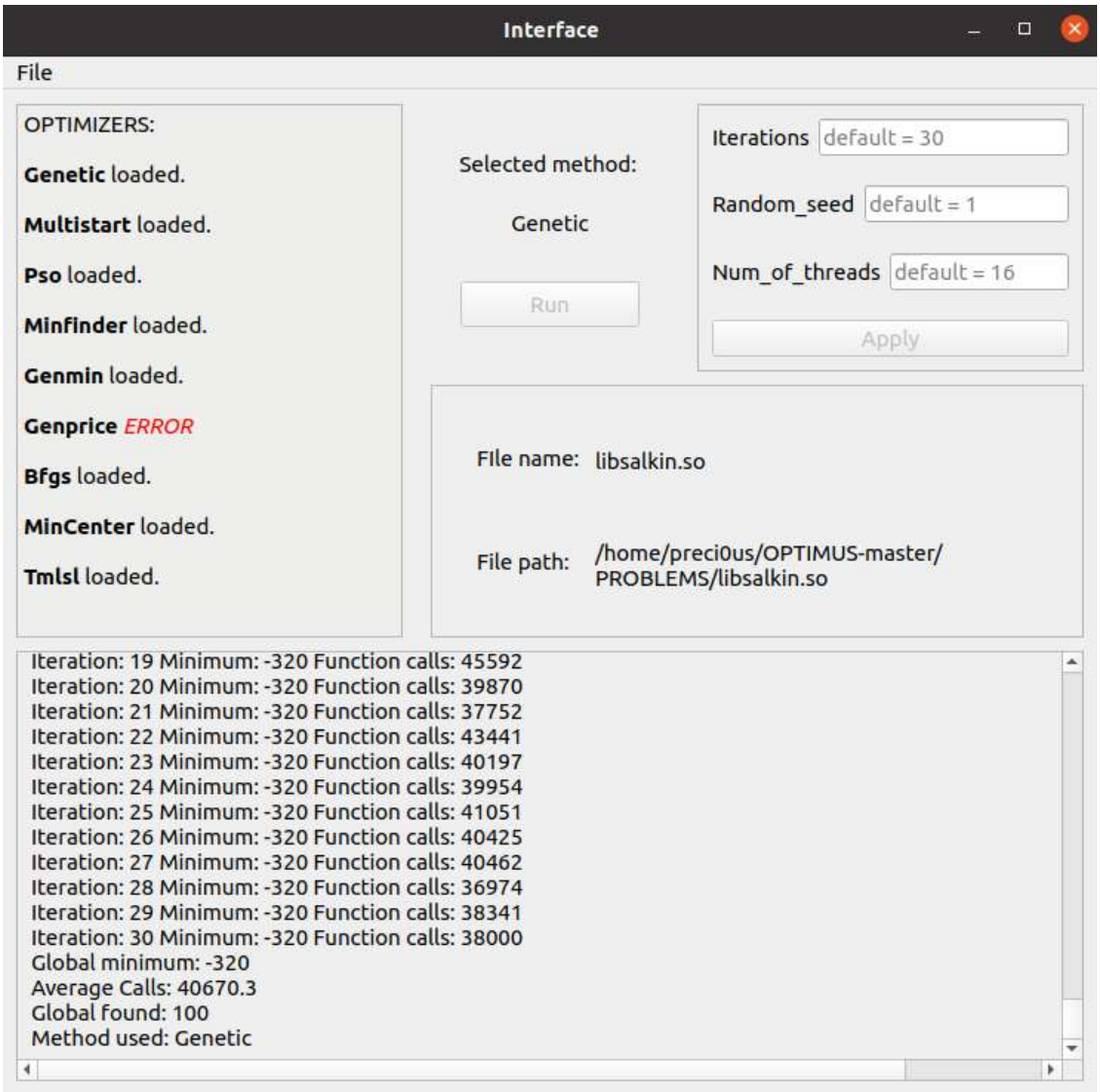
Τελευταία επιρροή που μπορεί να έχει ο χρήστης στη διαδικασία βελτιστοποίησης του προβλήματος, είναι με την αλλαγή των τιμών των τριών κοινών παραμέτρων, “Iterations”, “Random_seed”, “Num_of_threads”, που βρίσκονται στα δεξιά του αρχικού παραθύρου. Ο χρήστης έχει τη δυνατότητα να ενημερώσει οποιαδήποτε παράμετρο ξεχωριστά και πατώντας το κουμπί “submit”, εμφανίζεται ένα αντίστοιχο μήνυμα, ενημερώνοντας το χρήστη ότι οι αλλαγές που έκανε έχουν καταχωρηθεί (βλέπε εικόνα 4.10), αν όμως επιλέξει το κουμπί “submit” χωρίς να έχει συμπληρώσει κανένα από τα τρία πεδία, τότε εμφανίζεται μήνυμα αναφέροντας πως δεν είναι εφικτή κάποια αλλαγή καθώς δεν έχει συμπληρωθεί κανένα πεδίο (βλέπε εικόνα 4.11).



Εικόνα 4.10 Μήνυμα ενημέρωσης για την επιτυχημένη ενημέρωση των παραμέτρων

Εικόνα 4.11 Μήνυμα ενημέρωσης για την έλειψη συμπλήρωσης πεδίων

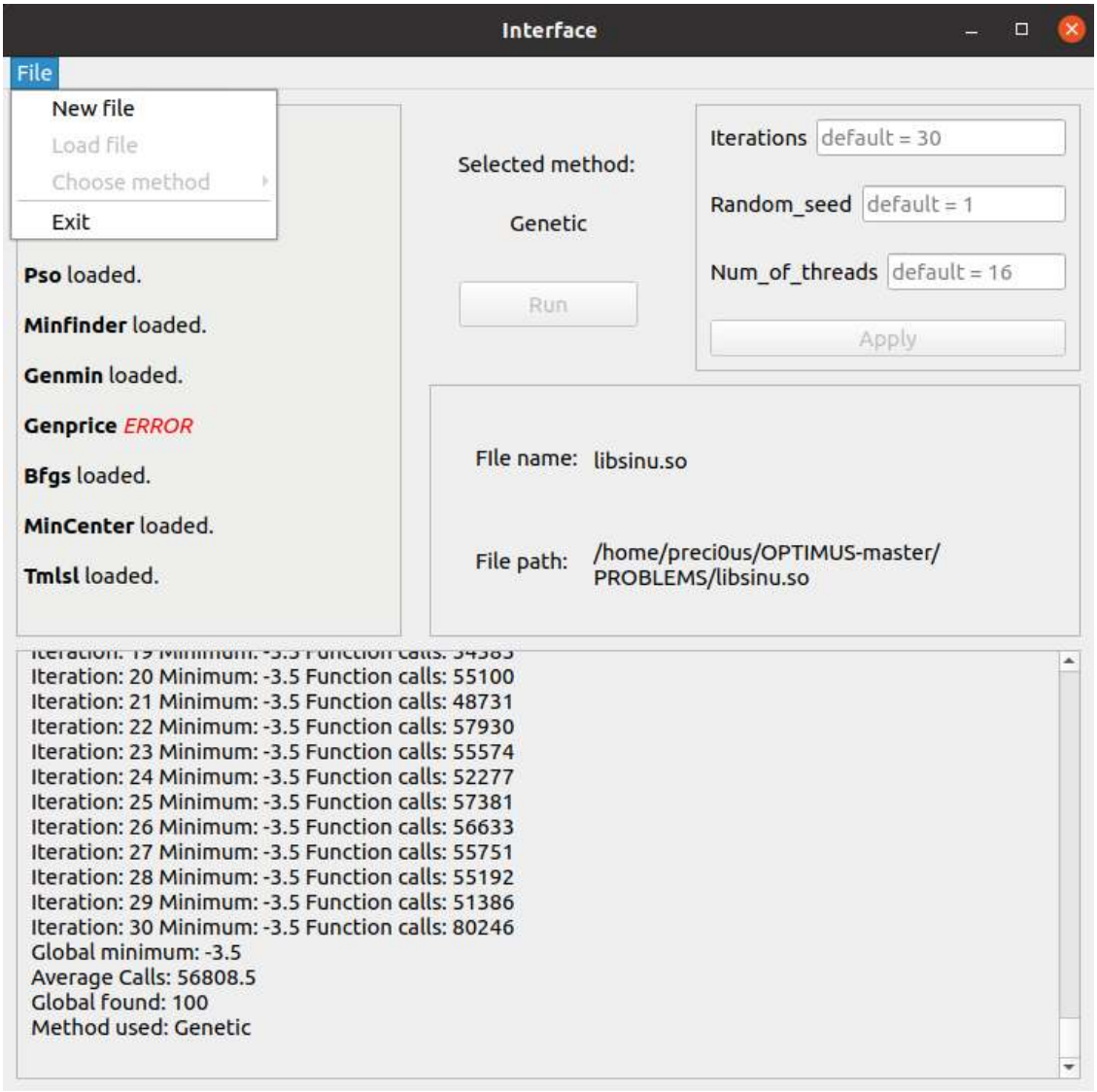
Έχοντας ακολουθήσει την απαραίτητη διαδικασία, η εφαρμογή είναι ικανή να τρέξει ομαλά και να εμφανίσει στο τερματικό (terminal) το αποτέλεσμα της αντίστοιχης μεθόδου βελτιστοποίησης. Παρακάτω, στην εικόνα 4.12, εμφανίζεται το αποτέλεσμα της μεθόδου Genetic.



Εικόνα 4.12 Αποτέλεσμα εκτέλεσης προβλήματος με τη μέθοδο Bfgs

Μετά την εκτέλεση του προγράμματος και εμφάνιση των αποτελεσμάτων, η εφαρμογή παρέχει ως μόνη επιλογή στο χρήστη, την επιλογή “New file” και “Exit” (βλέπε εικόνα

4.13). Έτσι, ο χρήστης μπορεί είτε να αρχικοποιήσει πάλι όλες τις παραμέτρους και να ξεκινήσει καινούρια συνεδρία με στην εφαρμογή, είτε να κλείσει την εφαρμογή.



Εικόνα 4.13 Εμφάνιση περιορισμένων επιλογών μετά την εκτέλεση του προγράμματος

5.Συμπεράσματα

Η εργασία αυτή εξήγησε τις βασικές έννοιες του κλάδου της βελτιστοποίησης, καθώς και αρκετές υποκατηγορίες αυτής. Ταυτόχρονα παρείχε και την αντίστοιχη εφαρμογή στο χρήστη μέσω της οποίας έχει την ικανότητα να παράξει τα δικά του αποτελέσματα βελτιστοποίησης. Για την καλύτερη και ομαλότερη χρήση από την πλευρά του χρήστη θα μπορούσε η εφαρμογή μελλοντικά να αποκτήσει από απλές επεκτάσεις, όπως είναι τα όρια τιμών για όλες τις παραμέτρους μεθόδων, χρήση νημάτων (threads)για ομαλότερη λειτουργία του τερματικού, μέχρι και τις πιο συμαντικές επεκτάσεις, όπως να παρέχονται περισσότερες μέθοδοι βελτιστοποίησης, τα αρχεία προβλημάτων να μπορούν να επιστρέφουν μία λίστα με τις δικές τους παραμέτρους ώστε να μπορεί ο χρήστης να θέσει τις δικές του τιμές για τις παραμέτρους των προβλημάτων, ή ακόμη, και την υποστήριξη προβλημάτων με διαφορετικό τρόπο δημιουργίας τυχαίων σημείων, εφόσον μέχρι στιγμής υποστηρίζονται μόνο προβλήματα με ομοιόμορφη κατανομή. Επεκτάσεις σαν κι αυτές που αναφέρθηκαν παραπάνω, θα αποτελέσουν μία αρκετά χρήσιμη εφαρμογή από τη στιγμή που σήμερα η βελτιστοποίηση είναι αναγκαία στο μεγαλύτερο ποσοστό των επιστημονικών κλάδων ανά τον κόσμο.

Παράρτημα: Κώδικας εφαρμογής

Ο κώδικας της εφαρμογής αποτελείται από τρία βασικά τμήματα. Την κλάση OpBackEnd, η οποία περιέχει τις βασικές συναρτήσεις που ευθύνονται για την ομαλή λειτουργία του λογισμικού OPTIMUS, όπως την ανάγνωση και ανάλυση των προβλημάτων βελτιστοποίησης, τη δημιουργία, αρχικοποίηση και χρήση της κάθε μεθόδου βελτιστοποίησης, τον χειρισμό των παραμέτρων και την εμφάνιση των τελικών αποτελεσμάτων, την κλάσηInterface, η οποία ευθύνεται για τη σωστή σειρά εκτέλεσης των συναρτήσεων της κλάσης OpBackEnd, την επικοινωνία μεταξύ των διαλόγων παραμέτρων με κάποιες από τις συναρτήσεις της κλάσης OpBackEnd,καθώς και κάποιων απαραίτητων διεργασιών για την ομαλή συνεδρία της διαδικασίας βελτιστοποίησης,και τέλος, οι διάλογοι, που αποτελούν ξεχωριστά ένα παράθυρο για την εισαγωγή των απαραίτητων παραμέτρων για την κάθε μέθοδο βελτιστοποίησης αντίστοιχα.

Συναρτήσεις της κλάσης OpBackEnd:

- OpBackEnd(): Ησυνάρτησηδημιουργίας(constructor) τηςκλάσηςOpBackEnd. Αρχικοποιεί κάποιες απαραίτητες παραμέτρους.
- QStringListloadOptimizers(): Συνάρτηση δημιουργίας των μεθόδων βελτιστοποίησης, μαζί με κάποιες αρχικές παραμέτρους τους.
- voidloadProblem(): Ευθύνεται για τη φόρτωση του επιλεγμένου προβλήματος βελτιστοποίησης από το χρήστη.
- voidunloadOptimizers(): Συνάρτηση διαγραφής των μεθόδων βελτιστοποίησης κατά το τέλος του προγράμματος.
- voidprintStatistics(): Συνάρτηση εκτύπωσης κάποιων απαραίτητων αποτελεσμάτων για κάποιες μεθόδους βελτιστοποίησης.
- voidrunSelectedMethod(): Επιλογή της μεθόδου βελτιστοποίησης από τον χρήστη και επίλυση του προβλήματος μέσω αυτής.
- QJsonObjectgetAvailableParams(): Συνάρτηση επιστροφής όλων των παραμέτρων των μεθόδων βελτιστοποίησης.
- voidsetOptimusPath(QString): Συνάρτηση ενημέρωσης του μονοπατιού για τον φάκελο των προβλημάτων βελτιστοποίησης.
- voidaddProblemParam(QString, QString):Συνάρτησηεισαγωγής παραμέτρων προβλημάτων.
- QJsonObjectgetProblemParams(): Συνάρτηση επιστροφής παραμέτρων προβλημάτων.
- QStringListgetOptimizersNames(): Συνάρτηση επιστροφής λίστας με τα ονόματα των μεθόδων βελτιστοποίησης.
- voidsetOptMethod(): Συνάρτηση ενημέρωσης επιλεγμένης μεθόδου βελτιστοποίησης από τον χρήστη.
- QStringgetOptMethod(): Συνάρτηση επιστροφής επιλεγμένης μεθόδου βελτιστοποίησης.
- voidsetInterfaceParams(constQstring, constQString, constQString):Συνάρτησηενημέρωσηςτωντριώνπαραμέτρων στην αρχική σελίδα της εφαρμογής (iterations, randomseed, threads).
- int&getIterations():Συνάρτηση επιστροφής τιμής της παραμέτρου iterations.
- int&getRandomSeed():Συνάρτηση επιστροφής τιμής της παραμέτρου randomseed.
- int&getThreads():Συνάρτηση επιστροφής τιμής της παραμέτρου threads.
- QMap<QString, Optimizer*>&getOptimizerList():Συνάρτησηεπιστροφής των μεθόδων βελτιστοποίησης.
- voidsetFileName(QString): Συνάρτηση ενημέρωσης του ονόματος του προβλήματος βνελτιστοποίησης.
- QStringgetFileName(): Συνάρτηση επιστροφής του ονόματος του προβλήματος βελτιστοποίησης.
- voideditParams(constQJsonObject):Συνάρτησηενημέρωσης παραμέτρων προβλημάτων.
- QStringListgetCorruptedMethods() : Συνάρτηση επιστροφής των μεθόδων με πρόβλημα φόρτωσης στην εφαρμογή.

Συναρτήσεις της κλάσης Interface:

- Interface():Ησυνάρτησηδημιουργίας (constructor) τηςκλάσηςInterface. Κυρίως, δημιουργείκαι εμφανίζει το κύριο παράθυρο της εφαρμογής.
- voidsetOpbObject():Συνάρτηση που αρχικοποιεί τα απαραίτητα στοιχεία του κύριου παραθύρου ώστε να είναι έτοιμο για χρήση. Εμφανίζει και κάποια αντίστοιχα μηνύματα ενημέρωσης για τον χρήστη, αρχικοποιεί το τερματικό και απενεργοποιεί προβληματικές μεθόδους βελτιστοποίησης.
- voidinterfaceSlot(constQJsonObject&,constQString&):Συνάρτησηενημέρωσης και αποθήκευσηςπαραμέτρων που άλλαξε ο χρήστης μέσω διαλόγου μεθόδου βελτιστοποίησης. Εμφανίζει και κάποια αντίστοιχα μηνύματα ενημέρωσης για τον χρήστη.
- Void printResults(): Συνάρτηση εμφάνισης μηνυμάτων στο τερματικό.
- voidon_apply_btn_clicked():ΣυναρτήσηενημέρωσηςτωνπαραμέτρωνIterations, Random_seedκαιNum_of_threads.
- voidon_run_whole_thing_btn_clicked():Συνάρτησηεκίννησης της διαδικασίας βελτιστοποίησης.
- voidon_actionExit_triggered():Συνάρτησηεξόδουαπό την εφαρμογή.
- voidon_actionLoad_file_triggered():Συνάρτησηεπιλογήςπροβλήματος βελτιστοποίησης.
- voidon_actionBfgs_triggered(): Συνάρτησηεμφάνισηςδιαλόγουγια τη μέθοδο BFGS.
- voidon_actionGenetic_triggered():Συνάρτησηεμφάνισηςδιαλόγουγια τη μέθοδο Genetic.
- voidon_actionGenmin_triggered(): Συνάρτησηεμφάνισηςδιαλόγουγια τη μέθοδο Genmin.
- voidon_actionGenprice_triggered(): Συνάρτησηεμφάνισηςδιαλόγουγια τη μέθοδο Gneprice.
- voidon_actionPso_triggered(): Συνάρτησηεμφάνισηςδιαλόγουγια τη μέθοδο Pso
- voidon_actionMincenter_triggered(): Συνάρτησηεμφάνισηςδιαλόγουγια τη μέθοδο Mincenter.
- voidon_actionMinfinder_triggered(): Συνάρτησηεμφάνισηςδιαλόγουγια τη μέθοδο Minfinder.
- voidon_actionMultistart_triggered(): Συνάρτησηεμφάνισηςδιαλόγουγια τη μέθοδο Multistart.
- voidon_actionTmlsl_triggered(): Συνάρτησηεμφάνισηςδιαλόγουγια τη μέθοδο Tmlsl.
- voidon_actionNew_file_triggered():Συνάρτησηπροετοιμασίας του κύριου παραθύρου της εφαρμογής για νέα συνεδρία βελτιστοποίησης.
- voidcloseEvent(QCloseEvent*): Συνάρτηση εξόδου από τον διάλογο κάποιας μεθόδου βελτιστοποίησης.

Συναρτήσεις της κλάσης Genmin_D:

- Genmin_D(QWidget*, constQJsonObject&, OpBackEnd*):Η συνάρτηση δημιουργίας (constructor)της κλάσης Genmin_D. Είναι υπεύθυνη για τη δημιουργία και εμφάνιση του διαλόγου παραμέτρων της συγκεκριμένης μεθόδου βελτιστοποίησης, περιέχοντας και τις αποθηκευμένες αλλαγές που έχει τυχόν κάνει ο χρήστης προηγουμένως. Κατά

τον τερματισμό του, επιστρέφει τις παραμέτρους που ενημερώθηκαν, στην κλάση Interface, για να αντικαταστήσουν τις παλιές.

- voidgenminSignal(constQJsonObject&, constQString&): Συνάρτησημέσωτηςοποίας επιστρέφονται οι παράμετροι που έχει ενημερώσει ο χρήστης.
- voidon_push_button_clicked():Ησυνάρτησηαυτήείναι υπεύθυνη για την αποθήκευση των παραμέτρων που άλλαξε ο χρήστης. Μόλις ολοκληρωθεί η διαδικασία παραμετροποίησης της μεθόδου, η συνάρτηση αυτή αποθηκεύει τις παραμέτρους αυτές σε έναν πίνακα, τον οποίο επιστρέφει ο constructor μέσω της συνάρτησης genminSignal(constQJsonObject&, constQString&)κατά τον τερματισμό του διαλόγου.

Βιβλιογραφία

1. Βόγκλης, Κ. (2010). *Μέθοδοι Καθολικής Βελτιστοποίησης*. [Τμήμα Πληροφορικής, Πανεπιστήμιο Ιωαννίνων]. https://www.cs.uoi.gr/~voglis/present_gr.pdf

2. Ibrahim, M.A.H, Mamat, M. & Leong, W.J. (2014). *BFGS Method: A New Search Direction*. Sains Malaysiana 43 (10) (2014): 1593-1599. Online: https://www.researchgate.net/publication/287267638_BFGS_method_A_new_search_direction

3. Wiki (χ.χ.). Κλίση κατάβασης. Online: https://wblog.wiki/el/Steepest_descent

4. Τσιαφάκη, Σ. (2007*). Βελτιστοποίηση μικροκυματικών διατάξεων επιπέδων διηλεκτρικών υλικών με χρήση γενετικών αλγορίθμων.* [Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης, Τμήμα Φυσικής, Τομέας Εφαρμογών Φυσικής & Φυσικής Περιβάλλοντος.] Ανακτήθηκε από:http://ikee.lib.auth.gr/record/100475/files/gri-2007-933.pdf

5. Χατζή, Σ. (2008). *Δειγματοληψία με χρήση Κανονικών Κανομών και Εφαρμογή στην Καθολική Βελτιστοποίηση*. (Μεταπτυχιακή εργασία). Ανακτήθηκε από: https://www.cs.uoi.gr/wp-content/uploads/publications/master_2008_14.pdf

6. Νούση, Β. (2020-2021). *Γενετικοί αλγόριθμοι και mudflow για βελτιστοποίηση απορρύπανσης υπογείου υδροφορέα.* [Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης, Τμήμα Πολιτικών Μηχανικών]. Ανακτήθηκε από: http://ikee.lib.auth.gr/record/336737/files/Nousi.pdf

7. Λεριότης, Ι.Σ. (2021). *Global Optimization of Metamaterials using genetic algorithms.*[Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης]. Ανακτήθηκε από: http://ikee.lib.auth.gr/record/335747/files/SykasLeriotis.pdf

8. Μπαλτζόγλου, Δ.Ε. (2020). *Γενετικός Αλγόριθμος στο πρόβλημα επιλογής μεταβλητών*. [Εθνικό Μετσόβιο Πολυτεχνείο, Σχολή Εφαρμοσμένων Μαθηματικών & Φυσικών Επιστημών] . Ανακτήθηκε από: https://dspace.lib.ntua.gr/xmlui/bitstream/handle/123456789/42721/%CE%93%CE%95%CE%9D%CE%95%CE%A4%CE%99%CE%9A%CE%9F%CE%A3%20%CE%91%CE%9B%CE%93%CE%9F%CE%A1%CE%99%CE%98%CE%9C%CE%9F%CE%A3.pdf?sequence=1

9. Σάτσιου, Α. (2005). *Γενετικοί Αλγόριθμοι στην Ανάπτυξη Μηχανών Διανυσμάτων Υποστήριξης: Μεθοδολογία και Εφαρμογές στην Εκτίμηση Πιστωτικού Κινδύνου*. (Διπλωματική εργασία). [Πολυτεχνείο Κρήτης, Τμήμα Μηχανικών Παραγωγής και Διοίκησης]. Ανακτήθηκε από: https://dias.library.tuc.gr/view/manf/25093

10. AKP's CS Lessons. (χ.χ). *Roulette Wheel Selection Method| Genetic Algorithm | Roulette Wheel Selection for Genetic Algorithm*. Ανακτήθηκε από: https://www.youtube.com/watch?v=-B15r-8WX48

11. Geeksforgeeks (2019). *Crossover in Genetic Algorithm*. Ανακτήθηκε από: https://www.geeksforgeeks.org/crossover-in-genetic-algorithm/

12. Balland, L., Estel, L., Cosmao, J.M., Mouhab, N. (2000). *A genetic algorithm with decimal coding for estimation of kinetic and energetic parameters*. Chemometrics and Intelligent Laboratory Systems 50 (2000) 121-135. DOI : 10.1016/S0169-7439(99)00057-X**.**

13. Hibbert, D.B. (1993). *A hybrid genetic algorithm for the estimation of kinetic parameters*. Chemometrics and Intelligent Laboratory Systems 19 (1993) 319–329. Doi: https://doi.org/10.1016/0169-7439(93)80031-C

14. Boozarjomehry, R.B., Masoori, M. (2007). *Which method is better for the kinetic modeling: Decimal encoded or Binary Genetic Algorithm?*. Chemical Engineering Journal 130 (2007) 29 -37. Doi: https://doi.org/10.1016/j.cej.2006.11.017

15. Brunetti, A. (2000). *A fast and precise Genetic Algorithm for a non-linear fitting problem*. Computer Physics Communications 124 (2-3) 204–211. DOI:10.1016/S0010-4655(99)00454-3

16. Su, C.T., Tyen, W, T. (x.x). *A Genetic Algorithm Approach Employing Floating Point Representation for Economic Dispatch of Electric Power Systems*. Institute of Electric Engineering, National Chung Cheng University, Chiayi 621, TAIWAN 1444-1449. Ανακτήθηκε από: https://www.mssanz.org.au/MODSIM97/Vol%204/Su.pdf

17. Zhao, X., Gao, X.S., Hu, Z.C. (2007). *Evolutionary programming based on non-uniform mutation*. Applied Mathematics and Computation 192 (2007) 1-11. Doi: https://doi.org/10.1016/j.amc.2006.06.107

18. Μπακαλάκος, Σ. (2015). *Βέλτιστος σχεδιασμός κατασκευών σε ετερογενή υπολογιστικά συστήματα cpu&gpu.* (Διπλωματική εργασία). Εθνικό Μετσόβιο Πολυτεχνείο. Ανακτήθηκε από: https://dspace.lib.ntua.gr/xmlui/bitstream/handle/123456789/42140/Σεραφείμ%20Μπακαλάκος%20-%20Διπλωματική%20-%20Extra.pdf

19. Ανδρεάδης, Π.Ν. (2008). *Βελτιστοποίηση παραμέτρων έγχυσης σε δίχρονους ναυτικούς κινητήρες diesel*. (Διπλωματική εργασία). Εθνικό Μετσόβιο Πολυτεχνείο. Ανακτήθηκε από: https://dspace.lib.ntua.gr/xmlui/bitstream/handle/123456789/2931/andreadisp_diesel.pdf

20. Geeksforgeeks (2022). *Branch and Bound Algorithm*. Ανακτήθηκε από:

https://www.geeksforgeeks.org/branch-and-bound-algorithm/

21. Grötschel, M., Römisch, W. (2009). *Nonconvex Mixed-Integer Nonlinear Programming*. DFG Research Center Matheon. Ανακτήθηκεαπό: https://www.math.hu-berlin.de/~stefan/B19/

22. Τσίκας, Π. (χ.χ). *Μέθοδοι βελτιστοποίησης: Μέθοδος Βελτιστοποίησης Προσομοιωμένης Ανόπτησης.* Ανακτήθηκε από: https://eclass.upatras.gr/modules/document/file.php/CIV1756/5-Simulated%20Annealing%20%28SA%29%20%26%20Threshold%20Accepting%20Method.pdf

23. Στοχαστική βελτιστοποίηση (χ.χ). Ανακτήθηκε από: https://hellenicaworld.com/Science/Mathematics/gr/StochastikiVeltistopoiisi.html

24. Κονσολάκη, Ε. (2008). *Εξελικτικοί Αλγόριθμοι και Εφαρμογή στην πράξη*. (Διπλωματική εργασία). Πολυτεχνείο Κρήτης, Τμήμα: Μηχανικών Παραγωγής και Διοίκησης. Ανακτήθηκς από: http://artemis.library.tuc.gr/DT2009-0010/DT2009-0010.pdf

25. Αδαμίδης, Π. (1999). *Εξελικτικοί Αλγόριθμοι*. [ΤΕΙ Θεσσαλονίκης, Τμήμα: Πληροφορικής]. Ανακτήθηκε από: https://people.iee.ihu.gr/~adamidis/IntelSys/EA_notes.pdf

26. Λυκοθανάσης, Σ. (χ.χ). *Εισαγωγή στο Γενετικό Προγραμματισμό*. [Τμήμα Μηχανικών Η/Υ και Πληροφορικής.] Ανακτήθηκε από: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwiencfzgvb3AhWDuaQKHXqJBIAQFnoECBAQAQ&url=https%3A%2F%2Feclass.upatras.gr%2Fmodules%2Fdocument%2Findex.php%3Fcourse%3DCEID1042%26download%3D%2F4f197a70ly2q%2F4f197bdak4s9.ppt&usg=AOvVaw0nGizJQb8Jvr8ccOrX_gOF

27. Koza, J.R. (1997). *Genetic Programming*. Encyclopedia of Computer Science and Technology, Second Version. Online: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.44.5416&rep=rep1&type=pdf

28. Venter, G. (2010). *Review of Optimization Techniques.* Department of Mechanical and Mechatronic Engineering, Stellenbosch University. Online: http://scholar.sun.ac.za/handle/10019.1/14646

29. Donges, N. (2021). *Gradient Descent: An Introduction to 1 of Machine Learning's Most Popular Algorithms.* Online: https://builtin.com/data-science/gradient-descent

30.Kathuria, A. (2018). *Intro to optimization in deep learning: Gradient Descent*. Online: https://blog.paperspace.com/intro-to-optimization-in-deep-learning-gradient-descent/

31.Ruder, S. (2017). *An overview of gradient descent optimization algorithms*. Machine Learning (2017) 1-14. Doi: https://doi.org/10.48550/arXiv.1609.04747

32. Jones, D.R. & Martins, J. R. R. A. (2021). *The DIRECT algorithm: 25 years later.*Journal of Global Optimization (2021) 79:521–566. Online: https://link.springer.com/article/10.1007/s10898-020-00952-6#auth-Joaquim_R__R__A_-Martins

33. Finkel, D.E. & Kelley, C.T. (χ.χ). *Convergence Analysis Of The Direct Algorithm.* Online: https://repository.lib.ncsu.edu/bitstream/handle/1840.4/459/crsc-tr04-28.pdf

34. Kvasov, D.E. & Mukhametzhanov, M.S. (2018). *Metaheuristic vs. deterministic global optimization algorithms: The univariate case*. Applied Mathematics and Computation 318 (2018) 245-259. Doi: https://doi.org/10.1016/j.amc.2017.05.014

35. Wiki (χ.χ.). Μαθηματική Βελτιστοποίηση. Ανακτήθηκε από: https://wblog.wiki/el/Optimization_%28mathematics%29

36. Alam, M.N. (2016). *Particle Swarm Optimization: Algorithm and its Codes in MATLAB*. Department of Electrical Engineering, Indian Institute of Technology. Online: https://www.researchgate.net/profile/Mahamad-Alam/publication/297245624_Particle_Swarm_Optimization_Algorithm_and_its_Codes_in_MATLAB/links/56dddba408aedf2bf0c86c83/Particle-Swarm-Optimization-Algorithm-and-its-C/odes-in-MATLAB.pdf

37. Java Point (2021). *Hill Climbing Algorithm in Artificial Intelligence*. Ανακτήθηκε από: https://www.javatpoint.com/hill-climbing-algorithm-in-ai

38. Μπάτσου, Μ. (2016). *Υλοποίηση και Πειραματική Μελέτη του Αλγορίθμου Breakout*. (Διπλωματική εργασία). [Πανεπιστήμιο Δυτικής Μακεδονίας]. Ανακτήθηκε από: https://dspace.uowm.gr/xmluibitstream/handle/123456789/420/Διπλωματική%20-Μπάτσου%20Μαρία.pdf?sequence=1&isAllowed=y

39. Καλογιαννίδου, Ε. (χ.χ). Τοπική Ακτινωτή Αναζήτηση για Προβλήματα Ικανοποίησης Περιορισμών. (Διπλωματική εργασία). [Πανεπιστήμιο Δυτικής Μακεδονίας]. Ανακτήθηκε από: https://dspace.uowm.gr/xmlui/bitstream/handle/123456789/2575/ΚΑΛΟΓΙΑΝΝΙΔΟΥ%20ΕΥΜΟΡΦΙΑ-ΔΙΠΛΩΜΑΤΙΚΗ.pdf?sequence=1&isAllowed=y

40.ΜΗΧΑΝΗ ΤΟΥ ΧΡΟΝΟΥ.*"Οι 8 βασίλισσες". Το σκακιστικό πρόβλημα, για μαθηματικά μυαλά, που λύθηκε μετά από 150 χρόνια...* Ανακτήθηκε από: https://www.mixanitouxronou.gr/quot-oi-8-vasilisses-quot-to-skakistiko-provlima-gia-mathimatika-myala-poy-lythike-meta-apo-150-chronia/

41. Jünger, M., Reinelt, G. & RinaMi, G. (1995). *The Traveling Salesman Problem*. Elsevier Science B.V.: 225-330. Online: http://www.iasi.cnr.it/ResearchReports/R00375.pdf

42. Brownlee, J. (2021). *A Gentle Introduction to the BFGS Optimization Algorithm.* Machine Learning Mastery. Ανακτήθηκεαπό: https://machinelearningmastery.com/bfgs-optimization-in

43. Lam, A. (2020). *BFGS in a Nutshell: An Introduction to Quasi-Newton Methods*. Online: https://towardsdatascience.com/bfgs-in-a-nutshell-an-introduction-to-quasi-newton-methods-21b0e13ee504