Βελτιστοποίηση με την χρήση του λογισμικού Optimus

Ιωάννης Γ. Τσούλος

Έκδοση 1.00 - Οκτώβριος 2025

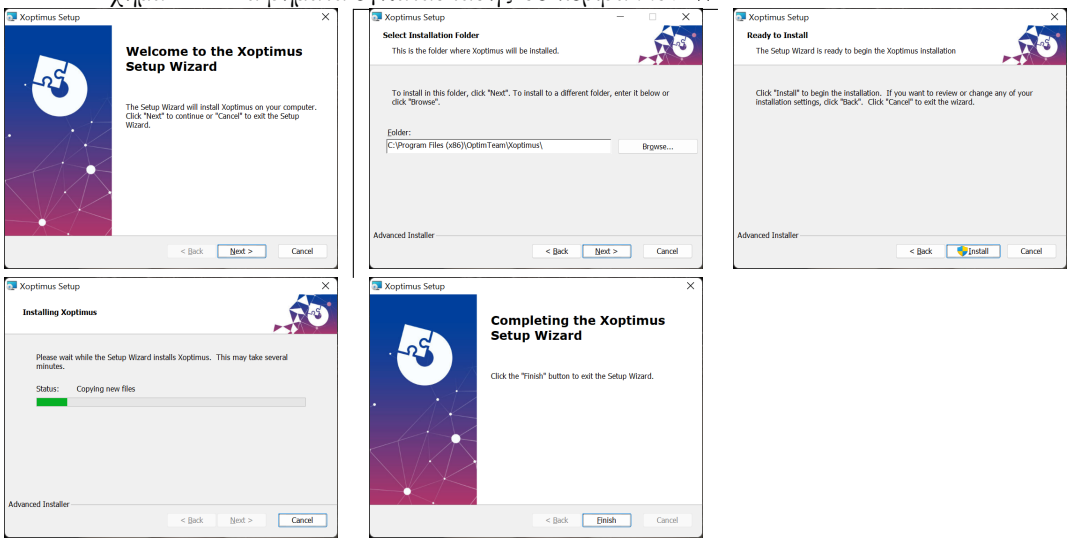
Κεφάλαιο 1

Το γραφικό περιβάλλον Xoptimus

1.1 Εγκατάσταση

Το λογισμικό Xoptimus μπορεί να εγκατασταθεί είτε αυτόνομα σε υπολογιστές Windows και χωρίς την ανάγκη εγκατάστασης κάποιας εξωτερικής βιβλιοθήκης με την χρήση του προγράμματος εγκατάστασης που διατίθεται στον επόμενο δικτυακό τόπο `https://www.dit.uoi.gr/files/GlobalOptimus.msi`. Η εγκατάσταση με την χρήση του συγκεκριμένου εκτελέσιμου αποτελείται από μερικά βήματα όπως παρουσιάζεται και στην εικόνα 1.1.

Σχήμα 1.1: Τα βήματα εγκατάστασης σε περιβάλλον Windows.

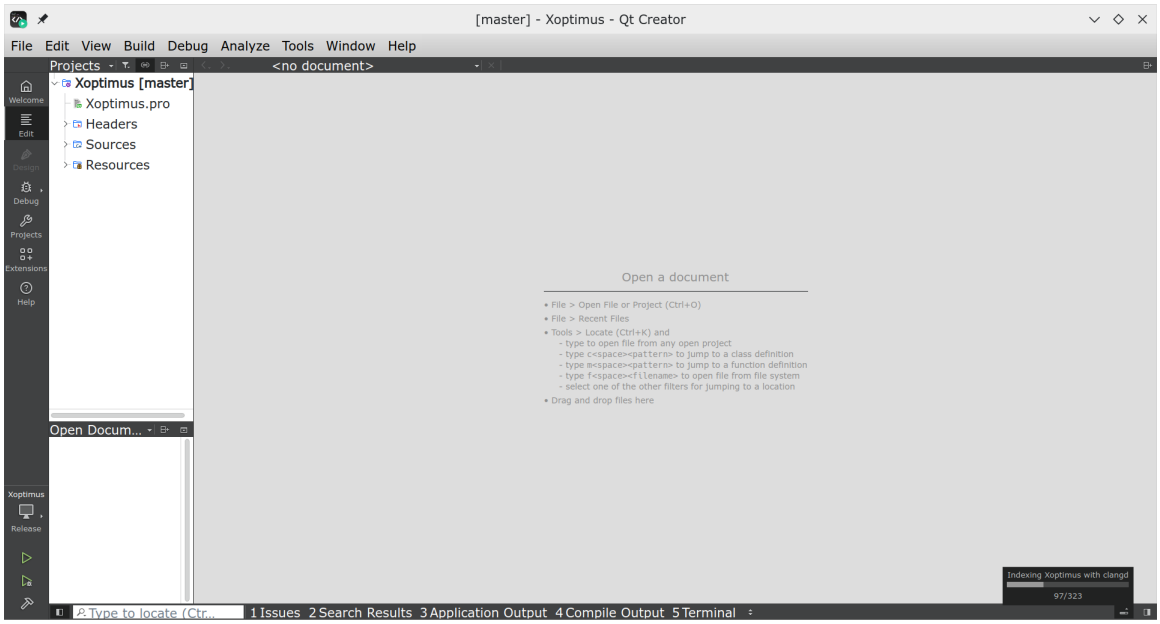


Εναλλακτικά μπορεί να γίνει εγκατάσταση μέσω της εφαρμογής choco των windows δίνοντας την ακόλουθη εντολή στο Power Shell:

choco install -y make qt5-default cmake qtcreator

Σε περιβάλλον Linux αρκεί ο χρήστης να εγκαταστήσει τις βιβλιοθήκες προγραμματισμού της QT Library καθώς και το περιβάλλον εργασίας Qt Creator. Αφού γίνει η εγκατάσταση του Qt Creator ο χρήστης πρέπει να κατεβάσει την τελευταία έκδοση του GlobalOptimus περιβάλλοντος από τον ακόλουθο δικτυακό τόπο `https://github.com/itsoulos/GlobalOptimus.git`. Στην συνέχεια ο χρήστης αφού κατεβάσει το πακέτο θα πρέπει μέσα από την εφαρμογή Χτ῎ρεατορ να ανοίξει μέσω της επιλογής File -> Open file or Project το project file με το όνομα Xoptimus.pro Η οθόνη που θα εμφανιστεί θα πρέπει να είναι ίδια με την εικόνα 1.2.

Σχήμα 1.2: Η εφαρμογή Xoptimus μέσα από το περιβάλλον Qt Creator.



Για την μεταγλώττιση και εκτέλεση της εφαρμογής αρκεί ο χρήστης να πατήσει το πράσινο κουμπί αριστερά (RUN).

1.2 Λειτουργία

Το γραφικό περιβάλλον Xoptimus μπορεί να χρησιμοποιηθεί για την εφαρμογή τεχνικών τοπικής ελαχιστοποίησης για την εύρεση του τοπικού ή του καθολικού

ελαχίστου συναρτήσεων:

$$f : S \to \mathbb{R},\ S \subset R^n \tag{1.1}$$

Γενικά θεωρούμε πως οι συναρτήσεις έχουν διάσταση n και το πεδίο ορισμό τους είναι υποσύνολο του συνολικού χώρου. Η αρχική οθόνη του περιβάλλοντος Xoptimus παρουσιάζεται στην εικόνα 1.3.

Σχήμα 1.3: Η αρχική οθόνη της εφαρμογής Xoptimus.

Η σημασία των εμφανιζόμενων μενού είναι η ακόλουθη:

1. Μενού **LOAD**. Με την επιλογή αυτή ο χρήστης μπορεί να επιλέξει το επιθυμητό πρόβλημα βελτιστοποίησης ή την επιθυμητή μέθοδο τοπικής ή καθολικής βελτιστοποίησης.

2. Μενού **SETTINGS**. Με την επιλογή αυτή ο χρήστης μπορεί να διαμορφώσει τις παραμέτρους είτε του προβλήματος ελαχιστοποίησης είτε της ακολουθούμενης τεχνικής τοπικής ή καθολικής βελτιστοποίησης. Επιπλέον μπορεί να αλλάξει την τιμή αρχικοποίησης για την γεννήτρια τυχαίων αριθμών. Το λογισμικό χρησιμοποιεί την συνάρτηση rand() της γλώσσας προγραμματισμού C++ για την παραγωγή τυχαίων αριθμών.

3. Μενού **EXECUTE**. Η επιλογή EXECUTE έχεις τρεις υπο - επιλογές

   (a) RUN. Με την επιλογή αυτή ξεκινάει η εκτέλεση της μεθόδου βελτιστοποίησης για το πρόβλημα που έχει επιλέξει ο χρήστης. Ο χρήστης θα πρέπει να εισάγει πλήθος εκτελέσεων με προκαθορισμένη τιμή τις 30 εκτελέσεις. Σε κάθε εκτέλεση διαφορετικοί τυχαίοι αριθμοί χρησιμοποιούνται.

(b) STATISTICS. Με την επιλογή αυτή εμφανίζονται στατιστικά για την τελευταία εκτέλεση της μεθόδου βελτιστοποίησης.

(c) TEST. Με αυτήν την επιλογή εμφανίζεται στην καρτέλα main η τιμή της συνάρτησης στο τελευταίο ελάχιστο που βρήκε η μέθοδος βελτιστοποίησης αλλά και η τιμή της παραγώγου σε εκείνο το σημείο.

(d) DARK THEME. Με αυτήν την επιλογή ο χρήστης μπορεί να αλλάξει την εμφάνιση της εφαρμογής σε σκοτεινό θέμα.

(e) LIGHT THEME. Με αυτήν την επιλογή ο χρήστης επιλέγει την εφαρμογή φωτεινού θέματος για την εφαρμογή.

4. Μενού **HELP**. Με αυτήν την επιλογή εμφανίζονται στοιχεία σχετικά με την εφαρμογή, όπως για παράδειγμα η ερευνητική ομάδα της εφαρμογής καθώς και η αντίστοιχη δημοσίευση στο περιοδικό Journal of Open Source Software.

Κεφάλαιο 2

Προβλήματα βελτιστοποίησης

Τα προβλήματα βελτιστοποίησης είναι οι αντικειμενικές συναρτήσεις που πρέπει να ελαχιστοποιηθούν από μια επιλεγμένη μέθοδο τοπικής βελτιστοποίησης. Κάθε τέτοιο πρόβλημα είναι μια νέα κατηγορία σε c++ που κληρονομεί την βασική κατηγορία Problem. Τα προβλήματα που έχουν υλοποιηθεί βρίσκονται στον φάκελο PROBLEMS.

2.1 Παράμετροι

Οι παράμετροι χρησιμοποιούνται για να αλλάξουν την συμπεριφορά των αντικειμενικών προβλημάτων όταν εκτελείται η μέθοδος βελτιστοποίησης αλλά και για να περάσουν πληροφορία από τον έξω κόσμο στην αντικειμενική συνάρτηση. Για παράδειγμα στην συνάρτηση test2n με τύπο:

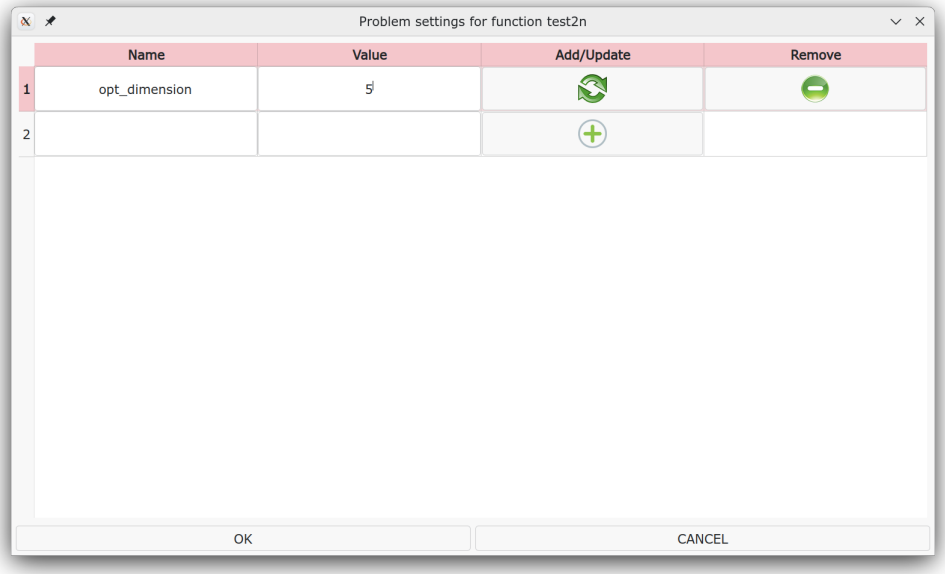
$$f(x) = \frac{1}{2}\sum_{i=1}^{n} x_i^4 - 16x_i^2 + 5x_i, \quad x_i \in [-5, 5]$$

Σε αυτήν την συνάρτηση η παράμετρος n είναι η διάσταση του προβλήματος και η συνάρτηση έχει 2^n τοπικά ελάχιστα στο πεδίο ορισμού της. Η χρήση της παραμέτρου της διάστασης μέσα στο πρόβλημα test2n γίνεται με την εξής γραμμή κώδικα:

```
int n = params["opt_dimension"].toString().toInt();
```

Για να μπορέσει ο χρήστης να αλλάξει τις τιμές των παραμέτρων στα προβλήματα μπορεί να το κάνει με τον γραφικό τρόπο του διαλόγου που παρουσιάζεται στο σχήμα 2.1. Αυτός ο διάλογος ανοίγει όταν ο χρήστης επιλέξει από το μενού SETTINGS την επιλογή PROBLEM.

Σχήμα 2.1: Αλλαγή παραμέτρων προβλήματος.



Μετά την αλλαγή της παραμέτρου ο χρήστης πατάει το πλήκτρο Add/Update και η παράμετρος αλλάζει στην νέα τιμή. Στην κενή γραμμή ο χρήστης μπορεί να προσθέσει όποια παράμετρο επιθυμεί.

2.2 Η συνάρτηση δημιουργίας και τα όρια της συνάρτησης

Στην συνάρτηση δημιουργίας ο χρήστης θα πρέπει να ορίσει την διάσταση και τα όρια της αντικειμενικής συνάρτησης, αν και αυτά μπορούν να αλλάξουν στην συνέχεια με την χρήση διαφορετικών συναρτήσεων πρόσβασης. Σαν ένα παράδειγμα συνάρτησης δημιουργίας θεωρήστε την ακόλουθη για το πρόβλημα Rastrigin το οποίο ορίζεται ως εξής:

$$f(x) = x_1^2 + x_2^2 - \cos(18x_1) - \cos(18x_2), \quad x \in [-1, 1]^2$$

Ο κώδικας για την συνάρτηση δημιουργίας είναι ο ακόλουθος:

```
RastriginProblem::RastriginProblem()
    : Problem(2)
{
    Data l, r;
    l.resize(2);
    r.resize(2);
```

```
    for (int i = 0; i < 2; i++)
    {
        l[i] = -1.0;
        r[i] = 1.0;
    }
    setLeftMargin(l);
    setRightMargin(r);
}
```

Η μέθοδος setLeftMargin() ορίζει τα αριστέρα άκρα του πεδίου ορισμού της συνάρτησης και η μέθοδος setRightMargin() τα δεξιά άκρα. Αν ο προγραμματιστής θέλει να αλλάξει την διάσταση του προβλήματος δυναμικά μπορεί να χρησιμοποιήσει την μέθοδο setDimension(n), οπου ν είναι η νέα διάσταση.

2.3 Η συνάρτηση init()

Η συνάρτηση init() χρησιμοποιείται για να μπορέσει η συνάρτηση να λάβει τις παραμέτρους που εισάγονται από το σύστημα του Optimus (εικόνα 2.1). Σαν ένα τέτοιο παράδειγμα ας θεωρήσουμε και πάλι την συνάρτηση test2n, όπου η διάσταση του προβλήματος μπορεί να αλλάξει δυναμικά. Ο απαιτούμενος κώδικας για την συνάρτηση init() για αυτό το αντικειμενικό πρόβλημα παρουσιάζεται στην συνέχεια:

```
void Test2nProblem::init(QJsonObject &params)
{
    int n = params["opt_dimension"].toString().toInt();
    setDimension(n);
    Data l, r;
    l.resize(n);
    r.resize(n);
    for (int i = 0; i < n; i++)
    {
        l[i] = -5.0;
        r[i] = 5.0;
    }
    setLeftMargin(l);
    setRightMargin(r);
}
```

Οι παράμετροι στην αντικειμενική συνάρτηση παρέχονται από το σύστημα σε μορφή Json Object. Η συνάρτηση αυτή θα κληθεί από το σύστημα μόνον μια φορά όταν φορτωθεί στην μνήμη η αντικειμενική συνάρτηση και ο σκοπός της είναι να γίνει πέρασμα παραμέτρων στο αντικειμενικό πρόβλημα. Για παράδειγμα στην περίπτωση των τεχνητών νευρωνικών δικτύων σε αυτήν την συνάρτηση θα μπορούσε ο χρήστης να δώσει τα πρότυπα για την εκπαίδευση του μοντέλου μηχανικής μάθησης.

2.4 Η συνάρτηση funmin

Η συνάρτηση funmin() χρησιμοποιείται για την υλοποίηση της τιμής της αντικειμενικής συνάρτησης, όπως αυτή αποτιμάται σε ένα συγκεκριμένο σημείο x εντός του πεδίου ορισμού της συνάρτησης. Ένα ενδεικτικό παράδειγμα υλοποίησης παρουσιάζεται στην συνέχεια για την συνάρτηση rastrigin:

```
double RastriginProblem::funmin(Data &x)
{
    return x[0] * x[0] + x[1] * x[1] - cos(18.0 * x[0]) - cos(18.0 * x[1]);
}
```

2.5 Η συνάρτηση gradient

Η συνάρτηση gradient() χρησιμοποιείται για την επιστροφή της τιμής της παραγώγου σε ένα δεδομένο σημείο, μιας και αρκετές τεχνικές βελτιστοποίησης (και ειδικά οι μέθοδοι τοπικής βελτιστοποίησης) κάνουν συχνή χρήση αυτής της πληροφορίας. Για την συνάρτηση rastrigin γίνεται χρήση της επόμενης συνάρτησης gradient:

```
Data RastriginProblem::gradient(Data &x)
{
    Data g;
    g.resize(2);
    g[0] = 2.0 * x[0] + 18.0 * sin(18.0 * x[0]);
    g[1] = 2.0 * x[1] + 18.0 * sin(18.0 * x[1]);
    return g;
}
```

Σε πολλές περιπτώσεις η τιμή της παραγώγου είτε δεν είναι γνωστή είτε δεν μπορεί να υπολογιστεί αναλυτικά. Επομένως θα πρέπει να χρησιμοποιηθεί διαφορετική προσέγγιση για την ανάκτηση της παραπάνω τιμής. Σαν ένα τέτοιο παράδειγμα ας θεωρήσουμε τον υπολογισμό της παραγώγου με την χρήση διαφορών σύμφωνα με το σχήμα:

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \tag{2.1}$$

Μια ενδεικτική υλοποίηση της παραπάνω προσέγγισης σε C++ παρουσιάζεται στην συνέχεια:

```
Data      TestProblem::gradient(vector<double> &x)
{
          Data g;
    g.resize(x.size());
          for(int i=0;i<getdimension();i++)
          {
               double amax = 1.0>fabs(x[i])?1.0:fabs(x[i]);
                    double eps=pow(1e-18,1.0/3.0)*amax;
```

```
                x[i]+=eps;
                double v1=funmin(x);
                x[i]-=2.0*eps;
                double v2=funmin(x);
                g[i]=(v1-v2)/(2.0 * eps);
                x[i]+=eps;
            }
    return g;
}
```

2.6 Η συνάρτηση done()

Η συνάρτηση δονε() με την ακόλουθη σύνταξη

```
QJsonObject UserProblem::done(Data &x)
{
    QJsonObject t;
    return t;
}
```

καλείται όταν τελειώσει η ζωή του αντικειμενικού προβλήματος και θα αποφορτωθεί από την μνήμη του υπολογιστή. Στο Json Object με το συμβολικό όνομα x ο χρήστης μπορεί να επιστρέψει τιμές από την εκτέλεση της βελτιστοποίησης, όπως για παράδειγμα στα τεχνητά νευρωνικά δίκτυα ο χρήστης μπορεί να επιστρέψει το σφάλμα του μοντέλου μηχανικής μάθησης στα δεδομένα ελέγχου (test set).

2.7 Η κατηγορία UserProblem

Η κατηγορία ΥσερΠροβλεμ μπορεί να χρησιμοποιηθεί για την υλοποίηση προβλημάτων βελτιστοποίησης που θέλει να ορίσει ο χρήστης και δεν συμπεριλαμβάνονται στην λίστα των προβλημάτων που έχουν υλοποιηθεί στο Optimus.

Κεφάλαιο 3

Μέθοδοι βελτιστοποίησης

3.1 Μέθοδοι τοπικής ελαχιστοποίησης

Οι μέθοδοι τοπικής βελτιστοποίησης χρησιμοποιούνται για την εύρεση ενός τοπικού ελαχίστου μιας συνάρτησης $f(x)$ και σε πολλές περιπτώσεις κάνουν και χρήση παραγώγων για την αποτελεσματική εύρεσή του. Επιπλέον, σχεδόν όλες οι τεχνικές καθολικής βελτιστοποίησης μπορούν να χρησιμοποιήσουν και μεθόδους τοπικής βελτιστοποίησης για την αποτελεσματικότερη εύρεση του ολικού ελαχίστου των αντικειμενικών συναρτήσεων. Σε αυτές τις τεχνικές γίνεται αναζήτηση $x^* \in S$, με την ιδιότητα

$$f(x^*) < f(x), \forall x \in S, \|x - x^*\| \leq e$$

όπου ε ένας μικρός θετικός αριθμός. Στο λογισμικό Optimus έχουν υλοποιηθεί οι παρακάτω τεχνικές τοπικής βελτιστοποίησης:

1. **GradientDescent**. Με την μέθοδο αυτή γίνεται χρήση της τεχνικής Γραδιεντ Δεσςεντ[1] για την εύρεση ενός τοπικού ελαχίστου, σύμφωνα με το σχήμα:

   $$x' = x - n\frac{\partial f(x)}{\partial x}$$

   όπου n είναι ο ρυθμός μάθησης. Σε πολλές περιπτώσεις η παράμετρος n δεν είναι σταθερά αλλά μπορεί να υπολογιστεί με τεχνικές γραμμικής αναζήτησης σύμφωνα με τον αλγόριθμου 3.1. Οι βασικές παράμετροι της GradientDescent είναι οι ακόλουθες:

   (a) **gd_maxiters**. Ορίζει τον μέγιστο αριθμό επαναλήψεων της τεχνικής.

   (b) **gd_linesearch**. Ορίζει την τεχνική που θα χρησιμοποιηθεί για Line Search. Επιτρεπτές τιμές είναι: none, golden [2], fibonacci [3], armijo [4].

   (c) **gd_epsilon**. Είναι μια πολύ μικρή τιμή (πχ 10^{-5}) που θα χρησιμοποιηθεί σαν κριτήριο τερματισμού της τεχνικής όταν η διαφορά μεταξύ των σημείων x και x' πέσει κάτω από αυτό το όριο.

(d) **gd_rate**. Είναι ο ρυθμός μάθησης n που θα χρησιμοποιηθεί στην Gradient Descent αν έχει επιλεχθεί σαν μέθοδος line search η επιλογή none.

2. **Bfgs**. Υλοποιεί μια τεχνική τοπικής βελτιστοποίησης από την οικογένεια των μεθόδων Broyden–Fletcher–Goldfarb–Shanno (BFGS) [5] που έχει προταθεί από τον Powell [6] και αξιοποιεί τα όρια των αντικειμενικών συναρτήσεων. Η μόνη παράμετρος που έχει αυτή η τεχνική είναι η ακέραια παράμετρος bfgs_iters που ορίζει τον μέγιστο αριθμό επαναλήψεων και έχει τεθεί στην τιμή 2001.

3. **LBfgs**. Υλοποιεί την τεχνική Limited Memory BFGS (LBFGS) [7], που χρησιμοποιείται κυρίως σε προβλήματα μεγάλης κλίμακας (μεγάλος αριθμός n). Η μόνη παράμετρος που διαθέτει αυτή η μέθοδος είναι η παράμετρος lbfgs_iters που ορίζει τον μέγιστο αριθμό επαναλήψεων.

4. **NelderMead**. Υλοποιεί την τεχνική Nelder Mead [8] για τοπική ελαχιστοποίηση.

5. **Adam**. Υλοποιεί την τεχνική Adam [9] για τοπική ελαχιστοποίηση, που αποτελεί μια βελτίωση της τεχνικής Gradient Descent και έχει εφαρμοσθεί αποτελεσματικά σε μοντέλα βαθιάς μηχανικής μάθησης [10].

Αλγόριθμος 3.1 Το γενικό σχήμα της γραμμικής αναζήτησης.

---

1. Δίνεται ένα σημείο $x^{(k)}$
   (a) Υπολογίζεται μια φθίνουσα κατεύθυνση $s^{(k)}$
   (b) Ελαχιστοποιείται ως προς λ ($\lambda > 0$) η συνάρτηση

$$\phi(\lambda) = f\left(x^{(k)} + \lambda s^{(k)}\right)$$

   και εντοπίζεται η βέλτιστη τιμή $\lambda^{(k)}$
   (c) Τίθεται $x^{(k+1)} = x^{(k)} + \lambda^{(k)} s^{(k)}$

Στο σημείο 3 γίνεται η λεγόμενη **γραμμική αναζήτηση**.

3.2 Μέθοδοι καθολικής ελαχιστοποίησης

Στην περίπτωση των τεχνικών καθολικής βελτιστοποίησης, στόχος είναι η εύρεση του ολικού ελαχίστου της αντικειμενικής συνάρτησης, δηλαδή

$$\min_x f(x),\ x \in S \subset R^n$$

Στο λογισμικό έχουν υλοποιηθεί μια σειρά από τεχνικές καθολικής βελτιστοποίησης που παροσιάζονται στην συνέχεια.

3.2.1 Multistart (πολλαπλών εκκινήσεων)

Υλοποιεί μια μέθοδο πολλαπλών εκκινήσεων [11] για την εύρεση του ολικού ελαχίστου. Το γενικό σχήμα αυτής της μεθόδου παρουσιάζεται στον αλγόριθμο 3.2.

Αλγόριθμος 3.2 Το γενικό σχήμα της τεχνικής πολλαπλών εκκινήσεων.

1. Λήψη N δειγμάτων από την συνάρτηση $f(x)$
2. Επεξεργασία των δειγμάτων, πχ εκτέλεση μεθόδου ελαχιστοποίησης
3. Έλεγχος για τερματισμό. Αν όχι μετάβαση στο 1
4. Το δείγμα με την χαμηλότερη τιμή είναι και το τελικό αποτέλεσμα.

Οι παράμετροι της τεχνικής είναι οι ακόλουθες:

1. **ms_samples**. Η παράμετρος ορίζει αυτή πόσα δείγματα θα λαμβάνονται σε κάθε εκτέλεση της μεθόδου Μυλτισταρτ.
2. **ms_maxiters**. Η παράμετρος αυτή ορίζει τον μέγιστο αριθμό επαναλήψεων της τεχνικής.

3.2.2 Genetic (Γενετικός αλγόριθμος)

Genetic. Αυτή η επιλογή υλοποιεί έναν Γενετικό Αλγόριθμο [12] για την βελτιστοποίηση πολυδιάστατων συναρτήσεων. Τα χρωμοσώματα στον αλγόριθμο είναι διανύσματα δεκαδικών τιμών. Ένα βασικό σχήμα ενός γενετικού αλγόριθμου (χωρίς να είναι το μόνο) παρουσιάζεται στον αλγόριθμο 3.1.

Αλγόριθμος 3.3 Γενικό σχήμα γενετικού αλγορίθμου.

```
Procedure Genetic
   t=0
   Initialize ( P(t) )
   Evaluate ( P(t) )
   while (not terminated (P(t)) )do
     t=t+1
     subP(t)=Select( P(t) )
     Crossover( subP(t) )
     Mutate ( subP(t) )
     P(t+1) = Recombine (P(t), subP(t))
   end while
End Genetic
```

Οι βασικές παράμετροι του γενετικού αλγόριθμου είναι:

1. **gen_count**. Αυτή η ακέραια παράμετρος ορίζει το πλήθος των χρωμοσωμάτων στον γενετικό αλγόριθμο.

2. **gen_maxiters**. Αυτή η ακέραια παράμετρος ορίζει το μέγιστο αριθμό επαναλήψεων (γενιών) για τον γενετικό αλγόριθμο.

3. **gen_srate**. Αυτή η δεκαδική τιμή λαμβάνει τιμές στο διάστημα $[0, 1]$ και ορίζει το ρυθμό επιλογής (selection rate) για τον γενετικό αλγόριθμο.

4. **gen_mrate**. Αυτή η δεκαδική τιμή λαμβάνει τιμές στο διάστημα $[0, 1]$ και ορίζει τον ρυθμό μετάλλαξης (mutation rate) για τον γενετικό αλγόριθμο.

5. **gen_lrate**. Αυτή η δεκαδική τιμή λαμβάνει τιμές στο διάστημα $[0, 1]$ και ορίζει τον ρυθμό εφαρμογής μεθόδου τοπικής ελαχιστοποίησης στα χρωμοσώματα. Για λόγους ταχύτητας η προκαθορισμένη τιμή είναι 0.0

6. **gen_tsize**. Αυτή η ακέραια παράμετρος ορίζει τον αριθμό των χρωμοσωμάτων που θα συμπεριλαμβάνονται στην λίστα επιλογής χρωμοσώματος με την μέθοδο του tournament selection [13]. Κατά την διάρκεια εκτέλεσης αυτής της τεχνικής N τυχαία χρωμοσώματα επιλέγονται και το χρωμόσωμα με την χαμηλότερη συναρτησιακή τιμή είναι αυτό το οποίο θα επιλεχθεί.

7. **gen_selection**. Η επιλογή αυτή λαμβάνει δύο τιμές: roulette, tournament και ορίζει τον τρόπο με τον οποίο θα γίνεται η επιλογή γονέων. Η πρώτη τιμή χρησιμοποιεί την τεχνική της ρουλέτας [14], της οποίας ένα ενδεικτικό σχήμα παρουσιάζεται στην εικόνα 3.1 και η δεύτερη επιλογή χρησιμοποιεί την επιλογή του tournament selection.

8. **gen_crossover**. Η επιλογή αυτή ορίζει την τεχνική που θα χρησιμοποιηθεί κατά την διαδικασία της διασταύρωσης και οι αποδεκτές τιμές είναι: double, uniform, onepoint. Στην πρώτη περίπτωση η δημιουργία των απογόνων $(\tilde{z}, \tilde{w})$ γίνεται με την διαδικασία

$$\begin{aligned} \tilde{z}_i &= a_i z_i + (1 - a_i) w_i \\ \tilde{w}_i &= a_i w_i + (1 - a_i) z_i \end{aligned} \tag{3.1}$$

   Στο σχήμα αυτό z και w είναι οι επιλεχθέντες γονείς και $a_i \in [-0.5, 1.5]$ [15]. Στην δεύτερη περίπτωση (ομοιόμορφη διασταύρωση) τα χρωμοσώματα δημιουργούνται με τυχαίο τρόπο λαμβάνοντας τιμές είτε από τον γονέα z είτε από τον γονέα w. Στην τρίτη περίπτωση, που χρησιμοποιείται περισσότερο σε ακέραια χρωμοσώματα, δημιουργείται ένα σημείο τομής και γίνεται ανταλλαγή γενετικού υλικού όπως παρουσιάζεται και στο σχήμα 3.1.

9. **gen_mutation**. Η παράμετρος αυτή αποδέχεται δύο δυνατές τιμές: double, random. Στην πρώτη περίπτωση για κάθε χρωμόσωμα g_i και για κάθε στοιχείο g_{ij} του παράγεται ένας τυχαίος αριθμός $r \in [0, 1]$. Αν αυτός ο αριθμός είναι μικρότερος της πιθανότητας μετάλλαξης, τότε

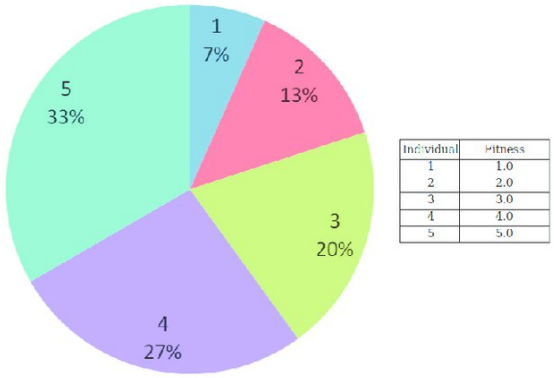
$$g_{ij} = \begin{cases} g_{ij} + \Delta\left(\mathrm{t}, b_{g,i} - g_{ij}\right) & t = 0 \\ g_{ij} - \Delta\left(\mathrm{t}, g_{ij} - a_{g,i}\right) & t = 1 \end{cases} \tag{3.2}$$

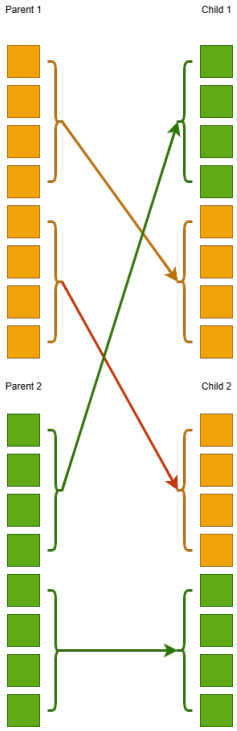
Ο αριθμός t είναι ένας τυχαίος αριθμός που λαμβάνει τις τιμές 0 και 1 και η συνάρτηση $\Delta(\tau, y)$ ορίζεται ως:

$$\Delta(t, y) = y\left(1 - \omega^{(1-\frac{t}{N_t})z}\right) \tag{3.3}$$

Όπου $\omega \in [0, 1]$ είναι ένας ακόμα τυχαίος αριθμός και η μεταβλητή z ορίζεται από τον χρήστη. Στην περίπτωση της επιλογής ρανδομ, το χρωμόσωμα διαταράσσεται τυχαία γύρω από την τρέχουσα τιμή του.

Σχήμα 3.1: Σχηματικό παράδειγμα της επιλογής ρουλέτας.





Σχήμα 3.2: Παράδειγμα διασταύρωσης ενός σημείου.

3.2.3 DifferentialEvolution (διαφοροεξελικτική τεχνική)

Είναι μια από τις βασικότερες εξελικτικές τεχνικές [16] και όπως και ο γενετικός χρησιμοποιεί μια ομάδα υποψηφίων λύσεων (πράκτορες) που εξελίσσονται με την πάροδο του χρόνου για την εύρεση του ολικού ελαχίστου μια συνάρτησης. Το βασικό σχήμα της τεχνικής αυτής παρουσιάζεται στον αλγόριθμο 3.4.

Αλγόριθμος 3.4 Ο αλγόριθμος Differential Evolution.

1. **Αρχικοποίηση** όλων των πρακτόρων.
   (a) **Θέσε** k=0
   (b) **Για** κάθε πράκτορα $x_i,\ i = 1, .., NP$ **κάνε**
      i. **Επιλογή** τριών πρακτόρων a,b,c από το σύνολο των πρακτόρων. Οι επιλεχθέντες πράκτορες πρέπει να είναι διαφορετικοί μεταξύ τους.
      ii. **Επιλογή** ενός τυχαίου ακεραίου $R \in [1, n]$, n είναι διάσταση του προβλήματος.
      iii. **Υπολογισμός** του νέου πράκτορα $y = (y_1, y_2, ..., y_n)$ ως εξής:
         A. Επιλογή τυχαίου αριθμού $r_i \in (0, 1),\ i = 1, ..., n$
         B. Αν $r_j < CR$ ή $j = R$, $y_j = a_j + F \times (b_j - c_j)$, αλλιώς $y_j = x_{i,j}$
      iv. **Αν** $f(y) \leq f(x_i)$ **Θέσε** $x_i = y$
   (c) **Τέλος Επανάληψης**
   (d) Θέσε k=k+1
   (e) **Αν** ισχύει το κριτήριο τερματισμού **τερματισμός**, **αλλιώς** μετάβαση στο βήμα 2.

Οι βασικές παράμετροι της τεχνικής είναι:

1. **de_np**. Ακέραια παράμετρος που ορίζει τον αριθμό των πρακτόρων.
2. **de_f**. Ορίζει την τιμή του συντελεστή κλιμάκωσης F.
3. **de_cr**. Ορίζει την τιμή του συντελεστή διασταύρωσης CR.
4. **de_tsize**. Ορίζει το πλήθος των πρακτόρων που θα συμμετάσχουν στην επιλογή tournament, αν επιλεχθεί αυτή τη τιμή για την παράμετρο **de_selection**.
5. **de_maxiters**. Ορίζει το μέγιστο αριθμό επαναλήψεων.
6. **de_fselection**. Η παράμετρος αυτή χρησιμοποιείται για τον υπολογισμό του συντελεστή κλιμάκωσης F και έχει τις παρακάτω τιμές:
   (a) **number**. Σε αυτήν την περίπτωση η τιμή της παραμέτρου F καθορίζεται από την παράμετρο **de_f**.
   (b) **ali**. Στην σχετική εργασία [17] προτάθηκε ο ακόλουθος μηχανισμός υπολογισμού της παραμέτρου:

$$F = \begin{cases} \max\left(l_{\min}, 1 - \left|\frac{f\max}{f\min}\right|\right) & , \quad \text{if} \quad \left|\frac{f\max}{f\min}\right| \leq 1 \\ \max\left(l_{\min}, 1 - \left|\frac{f\min}{f\max}\right|\right) & , \quad \text{otherwise} \end{cases} \tag{3.4}$$

(a) **adaptive**. Σε αυτήν την περίπτωση ο υπολογισμός γίνεται με την φόρμουλα:

$$F = -\frac{1}{2} + 2 \times R \tag{3.5}$$

όπως συστήνεται και στην σχετική εργασία [18].

(b) **migrant**. Σε αυτήν την περίπτωση ο υπολογισμός πραγματοποιείται όπως συστήνεται και στην εργασία [19].

7. **de_lrate**. Αυτή η δεκαδική τιμή λαμβάνει τιμές στο διάστημα $[0, 1]$ και ορίζει τον ρυθμό εφαρμογής μεθόδου τοπικής ελαχιστοποίησης στους πράκτορες.

8. **de_selection**. Η παράμετρος αυτή δέχεται δύο τιμές random και tournament και ορίζει τον τρόπο με το οποίο θα επιλέγονται οι πράκτορες $a, b,\ c$ στον αλγόριθμο.

3.2.4 iPso (βελτιωμένη τεχνική PSO)

Με την επιλογή αυτή προτείνεται μια βελτιωμένη εκδοχή της μεθόδου Particle Swarm Optimization (PSO) [20] όπως έχει προταθεί και στην αντίστοιχη δημοσίευση [21]. Ένα γενικό σχήμα του αλγορίθμου ΠΣΟ παρουσιάζεται στον αλγόριθμο 3.5.

Αλγόριθμος 3.5 Ένας τυπικός αλγόριθμος ΠΣΟ.

1. Για κάθε σωμάτιο i = 1, ..., S
   (a) Αρχικοποίηση της θέσης x_i κάθε σωματιού
      i. Αρχικοποίηση της ταχύτητας u_i κάθε σωματίου
      ii. $x^b = \text{argmin}\left(f(x)\right), y_i = x_i$
2. Μέχρι να ισχύσει ένα κριτήριο τερματισμού
   (a) Για κάθε σωμάτιο i = 1, ..., S
      i. Για κάθε διάσταση d = 1, ..., n
         A. Ενημέρωση της ταχύτητας $u_{i,d} = \omega u_{i,d} + r_1\left(x_{i,d} - y_{i,d}\right) + r_2\left(x_{i,d} - x_d^b\right)$
         B. Ενημέρωση θέσης $x_{i,d} = x_{i,d} + u_{i,d}$
      ii. Αν $f\left(x_i\right) < f((y_i), y_i = x_i$
      iii. Αν $f\left(x_i\right) < f\left(x^b\right), x^b = x_i$

Οι βασικές παράμετροι του αλγορίθμου που διαθέτει η τεχνική που υλοποιήθηκε είναι:

1. **ipso_particles.** Αυτή η ακέραια παράμετρος ορίζει τον αριθμό των σωματιδίων στον αλγόριθμο.

2. **ipso_generations**. Αυτή η ακέραια παράμετρος ορίζει τον μέγιστο αριθμό γενιών για τον αλγόριθμο PSO.

3. **ipso_c1**. Είναι μια δεκαδική παράμετρος που ορίζει την τιμή του c_1 του αλγορίθμου.

4. **ipso_c2**. Είναι μια δεκαδική παράμετρος που ορίζει την τιμή της παραμέτρου c_2.

5. **ipso_inertia_start**. Ορίζει την αρχική τιμή για τον συντελεστή inertia (ω).

6. **ipso_inertia_end**. Ορίζει την τελική τιμή για τον συντελεστή inertia (ω).

7. **ipso_localsearch_rate**. Αυτή η δεκαδική τιμή λαμβάνει τιμές στο διάστημα $[0, 1]$ και ορίζει τον ρυθμό εφαρμογής μεθόδου τοπικής ελαχιστοποίησης στα σωματίδια του αλγορίθμου.

8. **ipso_gradientcheck**. Αυτή η παράμετρος παίρνει τις τιμές true/false και χρησιμοποιείται για να ενεργοποιήσει/απενεργοποιήσει την χρήση του μηχανισμού απόρριψης σημείων, όπως προτείνεται και στην σχετική δημοσίευση [21].

9. **ipso_inertiatype**. Ορίζει τον τρόπο υπολογισμού της παραμέτρου inertia (ω). Μια σειρά από ενδεικτικές τιμές που λαμβάνει αυτή η παράμετρος είναι:

   (a) Random Inertia (**τιμή 4**). Η παράμετρος ω υπολογίζεται σύμφωνα με όσα προτείνονται στην σχετική δημοσίευση [22] και ο τύπος είναι :

   $$\omega_{\text{iter}} = 0.5 + \frac{r}{2} \tag{3.6}$$

   Όπου r ένας τυχαίος αριθμός με την ιδιότητα $r \in [0, 1]$.

   (b) Linear time varying inertia (min version) (**τιμή 5**). Αυτό το σχήμα προτείνεται σε μια σειρά από δημοσιεύσεις [22, 23, 24] και ορίζεται ως εξής:

   $$\omega_{\text{iter}} = \frac{\text{iter}_{\max} - \text{iter}}{\text{iter}_{\max}} \left(\omega_{\max} - \omega_{\min}\right) + \omega_{\min} \tag{3.7}$$

   (c) Linear time varying inertia (max version) (**τιμή 8**). Αυτή η τεχνική προτείνεται σε μια σειρά από εργασίες [25, 26] και ορίζεται ως:

   $$\omega_{\text{iter}} = \frac{\text{iter}_{\max} - \text{iter}}{\text{iter}_{\max}} \left(\omega_{\min} - \omega_{\max}\right) + \omega_{\max} \tag{3.8}$$

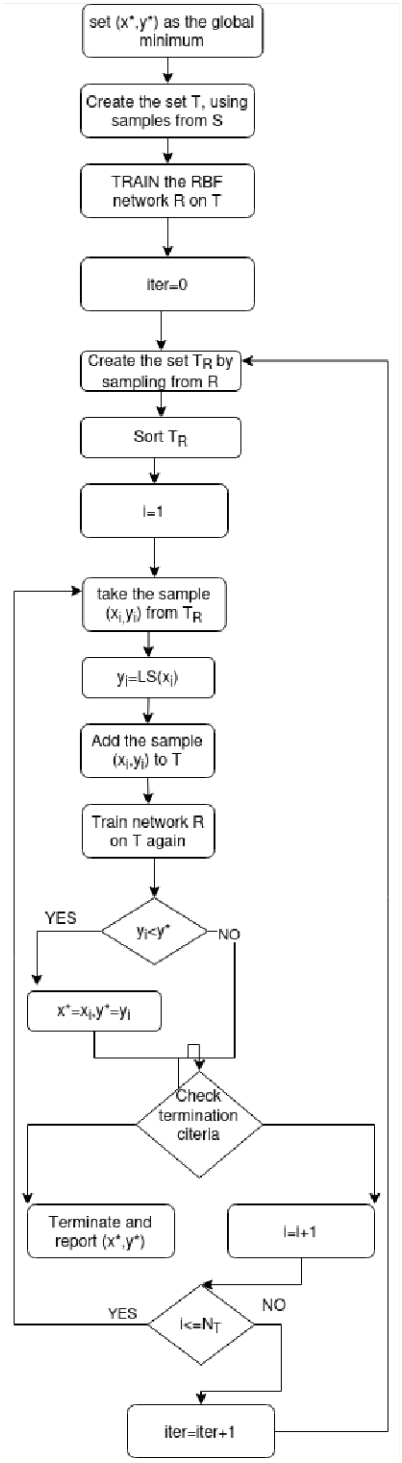
   (d) Ipso variant (**τιμή 0**). Σε αυτήν την περίπτωση η αδράνεια υπολογίζεται ως εξής:

   $$\omega_{\text{ιτερ}} = \frac{1}{4 + \frac{R}{2}}$$

   όπου P τυχαίος αριθμός στο διάστημα $[0, 1]$.

3.2.5 NeuralMinimizer

Σε αυτήν την τεχνική γίνεται υλοποίηση της μεθόδου NeuralMinimizer, όπως έχει προταθεί και στην σχετική δημοσίευση [27]. Σε αυτήν την τεχνική λαμβάνονται μερικά δείγματα από την αντικειμενική συνάρτηση και στην συνέχεια γίνεται εκπαίδευση ενός τεχνητού νευρωνικού δικτύου ΡΒΦ [28] πάνω σε αυτά για να δημιουργηθεί μια προσέγγιση της αντικειμενικής συνάρτησης. Στην συνέχεια η δειγματοληψία γίνεται αποκλειστικά από την προσεγγιστική συνάρτηση και όχι από την πραγματική. Το διάγραμμα ροής για αυτήν την τεχνική παρουσιάζεται στο σχήμα 3.3.



Σχήμα 3.3: Το διάγραμμα ροής για την μέθοδο NeuralMinimizer.

Οι βασικές παράμετροι αυτής της τεχνικής είναι:

1. **neural_weights**. Αυτή η ακέραια παράμετρος χρησιμοποιείται για τον καθορισμό των βαρών του δικτύου RBF που θα χρησιμοποιηθεί.
2. **neural_start_samples**. Η παράμετρος αυτή πόσα σημεία θα λάβει αρχικά η μέθοδος για την κατασκευή του προσεγγιστικού μοντέλου της αντικειμενικής συνάρτησης.
3. **neural_samples**. Η παράμετρος αυτή ορίζει πόσα δείγματα θα λαμβάνονται κάθε φορά από το προσεγγιστικό μοντέλο σε κάθε επανάληψη της μεθόδου.
4. **neural_iterations**. Αυτή η ακέραια παράμετρος ορίζει πόσες επαναλήψεις θα κάνει η μέθοδος βελτιστοποίησης.

3.2.6 ParallelDe

Η μέθοδος αυτή υλοποιεί την παράλληλη εκδοχή της Differential Evolution από την σχετική δημοσίευση [29]. Για την υλοποίηση χρησιμοποιήθηκε μια τεχνική παρόμοια με τους παράλληλους γενετικούς αλγόριθμους νησιών [30], όπου ο συνολικός πληθυσμός επιμερίζεται σε ανεξάρτητες υπολογιστικές μονάδες και σε κάθε μονάδα εκτελείται ένας διαφορετικός αλγόριθμος. Οι βασικές παράμετροι αυτής της τεχνικής είναι:

1. **parde_agents**. Είναι το πλήθος των πρακτόρων σε κάθε νησί.
2. **parde_generations**. Είναι ο μέγιστος αριθμός των επιτρεπόμενων επαναλήψεων του αλγορίθμου.
3. **parde_cr**. Ορίζει την τιμή του συντελεστή διασταύρωσης CR.
4. **parde_weight_method**. Είναι η μέθοδος που χρησιμοποιείται για τον υπολογισμό της παραμέτρου F με παρόμοιες τιμές όπως η παράμετρος **de_fselection** της Differential Evolution.
5. **parde_f**. Ορίζει την τιμή του συντελεστή κλιμάκωσης F.
6. **parde_propagate_method**. Ορίζει τον τρόπο με τον οποίο θα γίνεται η διάδοση των καλύτερων τιμών μεταξύ των διακριτών νησιών. Οι αποδεκτές τιμές είναι:
   (a) 1to1. Σε αυτήν την περίπτωση κάθε νησί επιλέγει τυχαία ένα από την λίστα και στέλνει σε αυτό τον καλύτερο πράκτορα του.
   (b) 1toN. Σε αυτήν την περίπτωση κάθε νησί ενημερώνει για τον καλύτερο του πράκτορα όλα τα υπόλοιπα νησιά.
   (c) Nto1. Σε αυτήν την περίπτωση όλα τα νησιά επιλέγουν το ίδιο νησί προς ενημέρωση.

(d) NtoN. Σε αυτήν την περίπτωση όλα τα νησιά επιλέγουν όλα τα υπόλοιπα προς ενημέρωση.

7. **parde_propagate_rate**. Είναι μια ακέραια παράμετρος που ορίζει τον αριθμό των γενιών που θα πρέπει να περάσουν πριν γίνει διάδοση των καλύτερων τιμών μεταξύ των νησιών.

8. **parde_selection_method**. Η παράμετρος αυτή δέχεται δύο τιμές random και tournament και ορίζει τον τρόπο με το οποίο θα επιλέγονται οι πράκτορες $a, b,\ c$ στον αλγόριθμο.

9. **parde_islands**. Ορίζει το πλήθος των διακριτών νησιών του αλγορίθμου.

10. **parde_islands_enable**. Ορίζει το πλήθος των νησιών που θα συμμετέχουν σε απόφαση για τερματισμό ή όχι της συνολικής διαδικασίας. Προφανώς θα πρέπει να ισχύει **parde_islands_enable<=parde_islands.**

3.2.7 ParallelPso

Η επιλογή αυτή υλοποιεί την τεχνική της παράλληλης PSO όπως έχει δημοσιευθεί και στην αντίστοιχη εργασία [31]. Οι βασικές παράμετροι αυτής της τεχνικής είναι:

1. **parallelPso_particles**. Ο αριθμός των σωματιδίων σε κάθε νησί.

2. **parallelPso_generations**. Ο μέγιστος αριθμός των επιτρεπόμενων επαναλήψεων.

3. **parallelPso_c1**. Η παράμετρος c_1 του αλγορίθμου PSO.

4. **parallelPso_c2**. Η παράμετρος c_2 του αλγορίθμου PSO.

5. **parallelPso_propagateRate**. Είναι μια ακέραια παράμετρος που ορίζει τον αριθμό των γενιών που θα πρέπει να περάσουν πριν γίνει διάδοση των καλύτερων τιμών μεταξύ των νησιών.

6. **parallelPso_propagateMethod**. Ορίζει τον τρόπο με τον οποίο θα γίνεται η διάδοση των καλύτερων τιμών μεταξύ των διακριτών νησιών. Οι αποδεκτές τιμές είναι:

   (a) 1to1. Σε αυτήν την περίπτωση κάθε νησί επιλέγει τυχαία ένα από την λίστα και στέλνει σε αυτό τον καλύτερο πράκτορα του.

   (b) 1toN. Σε αυτήν την περίπτωση κάθε νησί ενημερώνει για τον καλύτερο του πράκτορα όλα τα υπόλοιπα νησιά.

   (c) Nto1. Σε αυτήν την περίπτωση όλα τα νησιά επιλέγουν το ίδιο νησί προς ενημέρωση.

   (d) NtoN. Σε αυτήν την περίπτωση όλα τα νησιά επιλέγουν όλα τα υπόλοιπα προς ενημέρωση.

7. **parallelPso_subCluster**. Ο αριθμός των διακριτών νησιών της τεχνικής.
8. **parallelPso_subClusterEnable**. Ορίζει το πλήθος των νησιών που θα συμμετέχουν σε απόφαση για τερματισμό ή όχι της συνολικής διαδικασίας.
9. **parallelPso_pNumber**. Ορίζει τον αριθμό (ελάχιστο 1) των χρωμοσωμάτων που θα ανταλλάσσονται κατά την διαδικασία της διάδοσης χρωμοσωμάτων μεταξύ των διακριτών νησιών.

3.2.8 Simman

Η μέθοδος Simulated Annealing αναπτύχθηκε το 1983 από τον Κιρκπατρικ [32] κυρίως για συνδυαστικά προβλήματα και μιμείται την φυσική διαδικασία της Ανόπτυσης, όπου ζεσταίνεται ένα μέταλλο σε υψηλή θερμοκρασία και εν συνεχεία μειώνεται η θερμοκρασία μέχρι να φτάσει στο 0. Στην άνοδο τα μόρια του μετάλλου κινούνται γρήγορα (αναζήτηση λύσεων) αλλά στην συνέχεια όσο αυτό ψύχεται μειώνεται η ταχύτητά τους. Ο τρόπος που πέφτει η θερμοκρασία ονομάζεται cooling schedule. Ένα γενικό σχήμα αυτής της τεχνικής παρουσιάζεται στον αλγόριθμο 3.6.

Αλγόριθμος 3.6 Το γενικό σχήμα του αλγορίθμου Simulated Annealing.

1. **Θέτουμε** k = 0, $T_0 > 0$
   (a) **Έστω** x_0 το αρχικό σημείο.
   (b) **Έστω** $N_{\text{eps}} > 0$ ένας θετικός ακέραιος.
   (c) **Έστω** e> 0, ένας μικρός θετικός δεκαδικός αριθμός.
   (d) **Για** $i = 1, ..., N_{\text{eps}}$ **επανέλαβε**
      i. **Έστω** y ένα νέο δείγμα.
      ii. **Αν** $f(y) \leq f(x_k)$ $x_{k+1} = y$
      iii. **Διαφορετικά** $x_{k+1} = y$ με πιθανότητα $\min\left\{1, \exp\left(-\frac{f(y)-f(x_k)}{T_k}\right)\right\}$
   (e) **Τέλος Για**
   (f) **Ενημέρωση** θερμοκρασίας T_k σύμφωνα με τον μηχανισμό ψύξης.
   (g) k=k+1
   (h) **Αν** $T_k \leq e$ **τερματισμός**
   (i) **Αλλιώς** μετάβαση στο βήμα 5.

Στο πακέτο λογισμικού Optimus έχει υλοποιηθεί με το όνομα Simman και οι βασικές παράμετροι της τεχνικής είναι:

1. **siman_t0**. Ορίζει την αρχική θερμοκρασία του αλγορίθμου.

2. **siman_neps**. Ορίζει τον αριθμό των σημείων που θα δημιουργεί σε κάθε επανάληψη ο αλγόριθμος.

3. **siman_eps**. Ορίζει έναν πολύ μικρό αριθμό που θα χρησιμοποιηθεί στον κριτήριο τερματισμού του αλγορίθμου.

4. **siman_coolmethod**. Ορίζει τον μηχανισμό ψύξης της τεχνικής. Οι παρακάτω μηχανισμοί υποστηρίζονται:

   (a) exp. Η μείωση γίνεται σύμφωνα με τον τύπο:

   $$T_k = T_0 a^k,\ 0.8 \leq a \leq 0.9$$

   (b) log. Η μείωση γίνεται σύμφωνα με τον τύπο:

   $$T_k = \frac{T_0}{1 + a \log(1 + k)}$$

   (c) linear. Η μείωση γίνεται σύμφωνα με τον τύπο:

   $$T_k = \frac{T_0}{1 + ak}$$

   (d) quad. Η μείωση γίνεται σύμφωνα με τον τύπο:

   $$T_k = \frac{T_0}{1 + ak^2}$$

3.3 Παράμετροι μεθόδου

Οι παράμετροι είναι βασικό στοιχείων των τεχνικών ελαχιστοποίησης για την αποτελεσματική προσαρμογή των τεχνικών αυτών στις απαιτήσεις του χρήστη. Για την υλοποίηση τους γίνεται χρήση της κατηγορίας Parameter που είναι στον φάκελο OPTIMUS. Εκτός από τις παραμέτρους κάθε τεχνικής υπάρχουν και μια σειρά από καθολικές παράμετροι που επηρεάζουν όλες τις τεχνικές ελαχιστοποίησης.

3.3.1 Καθολικές παράμετροι

Οι καθολικές παράμετροι συνήθως έχουν το πρόθεμα opt_ και οι βασικότερες από αυτές είναι:

1. **opt_debug**. Αυτή η παράμετρος λαμβάνει τις τιμές yes/no και μπορεί να χρησιμοποιηθεί για να ενεργοποιεί/απενεργοποιεί την εμφάνιση μηνυμάτων από τις μεθόδους βελτιστοποίησης.

2. **opt_localsearch**. Αυτή η παράμετρος ορίζει την τεχνική τοπικής βελτιστοποίησης που θα χρησιμοποιηθεί από την μέθοδο καθολικής βελτιστοποίησης είτε κατά την διάρκεια εκτέλεσης της είτε στο τέλος αυτής. Οι επιτρεπόμενες τιμές είναι:

(a) bfgs.

(b) lbfgs.

(c) nelderMead.

(d) gradient.

(e) adam.

(f) none.

3. **opt_sampler**. Ορίζει την τεχνική δειγματοληψίας που θα χρησιμοποιηθεί κατά την εκκίνηση της μεθόδου τεχνικής καθολικής ελαχιστοποίησης. Οι αποδεκτές τιμές αυτής της παραμέτρου είναι:

   (a) uniform. Με αυτήν την επιλογή ομοιόμορφη κατανομή θα χρησιμοποιηθεί για την δειγματοληψία από την αντικειμενική συνάρτηση. Ως παράδειγμα ομοιόμορφης κατανομής θεωρήστε τον επόμενο τύπο:

   $$x = a + r \times (b - a)$$

   όπου $r \in [0, 1]$ ένας τυχαίος αριθμός. Ο παραπάνω τύπος παράγει τυχαία σημεία με ομοιόμορφο τρόπο στο διάστημα $[a, b]$.

   (b) triangular. Χρησιμοποιείται η τριγωνική κατανομή [33] για την λήψη δειγμάτων από την αντικειμενική συνάρτηση.

   (c) maxwell. Χρησιμοποιείται η κατανομή μαξωελλ [34] για την λήψη αρχικών δειγμάτων.

   (d) mlp. Με αυτήν την επιλογή παράγονται αρχικά δείγματα με την χρήση ενός νευρωνικού δικτύου MLP [35].

   (e) rbf. Με την χρήση αυτής της τιμής τα αρχικά δείγματα λαμβάνονται με την χρήση ενός RBF δικτύου.

   (f) kmeans. Σε αυτήν την περίπτωση η τεχνική των κ-μέσων [36] χρησιμοποιείται για την παραγωγή αρχικών σημείων.

   (g) user. Σε αυτήν την περίπτωση θα χρησιμοποιηθεί δειγματοληψία που ορίζεται από τον χρήστη. Για αυτόν τον λόγο ο προγραμματιστής θα πρέπει να αλλάξει στο επιθυμητό τον κώδικα της μεθόδου sampleFromProblem που βρίσκεται στην κατηγορία UserSampler στον φάκελο SAMPLER του λογισμικού.

4. **opt_termination**. Η παράμετρος αυτή ορίζει τον τρόπο τερματισμού της μεθόδου καθολικής βελτιστοποίησης με αποδεκτές τιμές:

   (a) doublebox. Το κριτήριο του διπλού κουτιού προσπαθεί να τερματίσει την τεχνική, όταν με κάποια βεβαιότητα δεν έχει βρεθεί νέο ολικό ελάχιστο για μια σειρά από επαναλήψεις. Τα βασικά σημεία αυτής της τεχνικής τερματισμού παρουσιάζονται στον αλγόριθμο 3.7.

Αλγόριθμος 3.7 Τα βασικά σημεία του αλγορίθμου Doublebox.

- Η διακύμανση $\sigma^{(iter)}$ της ποσότητας $f_{\min}$ υπολογίζεται σε κάθε επανάληψη iter.
  - Αν ο αλγόριθμος δεν μπορεί να εντοπίσει μια νέα καλύτερη τιμή για την $f_{\min}$ για μια σειρά από επαναλήψεις, τότε η μέθοδος θα πρέπει να τερματιστεί.
  - Τερματισμός αν:

$$\sigma^{(\text{iter})} \leq \frac{\sigma^{(\text{iter}_{\text{last}})}}{2} \tag{3.9}$$

  όπου $\text{iter}_{\text{last}}$ είναι η τελευταία επανάληψησ στην οποία βρέθηκε μια νέα καλύτερη τιμή για το $f_{\min}$

(b) similarity. Σε κάθε επανάληψη k γίνεται υπολογισμός της διαφοράς $\left|f_{\min}^{(k)} - f_{\min}^{(k-1)}\right|$ και ο αλγόριθμος τερματίζει αν $\left|f_{\min}^{(k)} - f_{\min}^{(k-1)}\right| \leq e$ για ένα αριθμό συνεχόμενων επαναλήψεων $k_{\max}$.

(c) maxiters. Με τον αυτόν τον κανόνα τερματισμού η μέθοδος βελτιστοποίησης θα τερματιστεί όταν θα συμπληρωθεί ένας προκαθορισμένος αριθμός επαναλήψεων.

3.3.2 Δημιουργία παραμέτρων

Για τον χειρισμό των παραμέτρων αρμόδια κατηγορία είναι η κατηγορία Parameter στον φάκελο OPTIMUS. Τα βασικά πεδία κάθε παραμέτρου είναι:

1. **QString** name. Είναι το όνομα της συγκεκριμένης παραμέτρου.
2. **QString** value. Είναι η τιμή της συγκεκριμένης παραμέτρου.
3. **QString** help. Μπορεί να χρησιμοποιηθεί για την εμφάνιση βοήθειας στον χρήστη.

Επιπλέον, για τον αποτελεσματικότερο χειρισμό των παραμέτρων προσφέρονται μια σειρά από συναρτήσεις δημιουργίας, όπως αναφέρονται στην συνέχεια:

1. Parameter(**QString** n, **QString** v, **QString** h)· Είναι η προκαθορισμένη συνάρτηση δημιουργίας.
2. Parameter(**QString** n, **int** v, **int** low, **int** upper, **QString** h); Χρησιμοποιείται για να ορίσει μια ακέραια παράμετρο με όρια [low,upper]. Η παράμετρος v είναι η προκαθορισμένη τιμή της παραμέτρου.

3. Parameter(**QString** n, **double** v, **double** low, **double** upper, **QString** h); Χρησιμοποιείται για να ορίσει μια δεκαδική παράμετρο με όρια [low,upper]. Η παράμετρος v είναι η προκαθορισμένη τιμή της παραμέτρου.

4. Parameter(**QString** n, **QString** v, **QStringList** list, **QString** h); Με αυτήν την συνάρτηση δημιουργίας ο χρήστης μπορεί να ορίσει μια αλφαριθμητική παράμετρο της οποίας οι πιθανές τιμές είναι στην λίστα list. Η παράμετρος v είναι η προκαθορισμένη τιμή της παραμέτρου.

Η προσθήκη παραμέτρων σε μια μέθοδο βελτιστοποίησης μπορεί να γίνει με την μέθοδο addParam(). Μια ενδεικτική χρήση της παρουσιάζεται στην συνέχεια από κώδικα που προέρχεται από την υλοποίηση του Γενετικού αλγορίθμου (αρχείο genetic.cpp στον φάκελο METHODS).

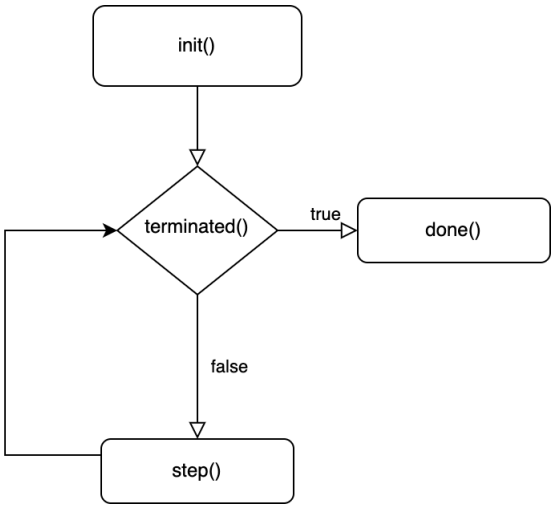
```
addParam(Parameter("gen_count",200,10,2000,"Number of chromosomes"));
```

Η ανάκτηση της συγκεκριμένης παραμέτρου από τον χρήστη γίνεται με τον ακόλουθο κώδικα:

```
chromosomeCount=getParam("gen_count").getValue().toInt();
```

3.4 Ο κύκλος εκτέλεσης

Όλες οι μέθοδοι βελτιστοποίησης ακολουθούν έναν κύκλο εκτέλεσης των μεθόδων τους, όπως αυτός παρουσιάζεται στο σχήμα 3.4.



Σχήμα 3.4: Ο κύκλος εκτέλεσης των τεχνικών βελτιστοποίησης.

3.4.1 Η μέθοδος init()

Στην μέθοδο αυτή λαμβάνει χώρα η αρχικοποίηση των παραμέτρων της μεθόδου και πιθανώς και δειγματοληψία από την αντικειμενική συνάρτηση αν αυτό απαιτείται. Ένα παράδειγμα τέτοιας συνάρτησης παρουσιάζεται στον αλγόριθμο 3.8 για την μέθοδο του Γενετικού Αλγορίθμου. Η συνάρτηση sampleFromProblem() εκτελεί δειγματοληψία από την αντικειμενική συνάρτηση χρησιμοποιώντας την μέθοδο δειγματοληψίας που ορίζει η καθολική παράμετρος **opt_sampler**.

Αλγόριθμος 3.8 Η μέθοδος init() του Γενετικού Αλγορίθμου.

```
void     Genetic::init()
{
    chromosomeCount=getParam("gen_count").getValue().toInt();
    maxGenerations=getParam("gen_maxiters").getValue().toInt();
    selectionRate=getParam("gen_srate").getValue().toDouble();
    mutationRate=getParam("gen_mrate").getValue().toDouble();
    tournamentSize = getParam("gen_tsize").getValue().toInt();
    selectionMethod=getParam("gen_selection").getValue();
    crossoverMethod=getParam("gen_crossover").getValue();
    mutationMethod=getParam("gen_mutation").getValue();
    localsearchRate=getParam("gen_lrate").getValue().toDouble();
    lsearchGens=getParam("gen_lsearchgens").getValue().toInt();
    lsearchItems=getParam("gen_lsearchitems").getValue().toInt();
    localSearchMethod = getParam("gen_lsearchmethod").getValue();

    generation = 0;
    //init process
    population.resize(chromosomeCount);
    fitnessArray.resize(chromosomeCount);

    sampleFromProblem(chromosomeCount,population,fitnessArray);
     childrenArray.resize(chromosomeCount);
    for(int i=0;i<chromosomeCount;i++)
    {
        childrenArray[i].resize(myProblem->getDimension());
    }
}
```

3.4.2 Η μέθοδος step()

Η συνάρτηση step() εκτελεί το βασικό βήμα της μεθόδου βελτιστοποίησης και επαναλαμβάνεται διαρκώς μέχρι να ισχύσει το κριτήριο τερματισμού. Ένα παράδειγμα εκτέλεσης για την μέθοδο του Γενετικού Αλγορίθμου παρουσιάζεται στον αλγόριθμο 3.9.

Αλγόριθμος 3.9 Η μέθοδος step() του Γενετικού Αλγορίθμου.

```
void      Genetic::step()
{
    ++generation;
    CalcFitnessArray();
    Selection();
    Crossover();
    Mutate();
    if(generation%lsearchGens==0 &&
localSearchMethod!=LOCAL_NONE)
    {
        LocalSearch(0);
        for(int i=0;i<lsearchItems;i++)
        {
            int pos = rand()  % population.size();
            LocalSearch(pos);
        }
        Selection();
    }
}
```

3.4.3 Η μέθοδος terminated()

Η μέθοδος αυτή ελέγχει το κριτήριο τερματισμού μετά την εκτέλεση της step() και αν είναι αληθές επιστρέφει true, διαφορετικά επιστρέφει false. Σαν παράδειγμα υλοποίησης στον αλγόριθμο 3.10 παρουσιάζεται ο κώδικας της μεθόδου για τον Γενετικό Αλγόριθμο.

Αλγόριθμος 3.10 Η μέθοδος terminated() του Γενετικού Αλγορίθμου.

```
bool      Genetic::terminated()
{
    double besty;
    besty = fitnessArray[0];
    if(generation>=maxGenerations) return true;
    if(terminationMethod=="doublebox")
        return doubleBox.terminate(besty);
    else
        if(terminationMethod=="similarity")
        return similarity.terminate(besty);
    return false;
}
```

3.4.4 Η μέθοδος done()

Η μέθοδος done() εκτελείται μόνον μια φορά και στο τέλος της μεθόδου βελτιστοποίησης. Σε αυτήν συνήθως γίνεται η εκτέλεση μιας τεχνικής τοπικής βελτιστοποίησης πάνω στο καλύτερο στοιχείο που έχει παράγει η μέθοδος καθολικής βελτιστοποίησης προκειμένου να ανακτηθεί μια καλύτερη τιμή για το ολικό ελάχιστο της αντικειμενικής συνάρτησης. Ένα ενδεικτικό παράδειγμα υλοποίησης παρουσιάζεται στον αλγόριθμο 3.11 για την περίπτωση του Γενετικού Αλγορίθμου. Η μέθοδος localSearch() καλεί την επιλεγμένη τεχνική τοπικής βελτιστοποίησης που έχει οριστεί από την καθολική παράμετρο **opt_localsearch**.

Αλγόριθμος 3.11 Η μέθοδος done() του Γενετικού Αλγορίθμου.

```
void      Genetic::done()
{
    fitnessArray[0]=localSearch(population[0]);
}
```

3.5 Η κατηγορία UserMethod

Η κατηγορία UserMethod που υπάρχει στον φάκελο METHODS μπορεί να χρησιμοποιηθεί για την δημιουργία μιας μεθόδου ελαχιστοποίησης που επιθυμεί ο χρήστης πέρα από αυτές που έχουν υλοποιηθεί στην διανομή. Ο χρήστης θα πρέπει να υλοποιήσει τις παρακάτω μεθόδους (τουλάχιστον)

1. Συνάρτηση δημιουργίας.
2. Μέθοδος init().
3. Μέθοδος step().
4. Μέθοδος terminated().
5. Μέθοδος done().

Στην συνέχεια θα πρέπει να κάνει και πάλι μεταγλώττιση το λογισμικό μέσα από το περιβάλλον του Qt Creator.

Βιβλιογραφία

[1] Ruder, S. (2016). An overview of gradient descent optimization algorithms. arXiv preprint arXiv:1609.04747.

[2] Pronzato, L., Wynn, H. P., & Zhigljavsky, A. A. (1998). A generalized golden-section algorithm for line search. IMA Journal of Mathematical Control and Information, 15(2), 185-214.

[3] Hassin, R. (1981). On maximizing functions by Fibonacci search. The Fibonacci Quarterly, 19(4), 347-351.

[4] Shi, Z., & Wang, S. (2011). Modified nonmonotone Armijo line search for descent method. Numerical Algorithms, 57(1), 1-25.

[5] Dai, Y. H. (2002). Convergence properties of the BFGS algoritm. SIAM Journal on Optimization, 13(3), 693-701.

[6] M.J.D Powell, A Tolerant Algorithm for Linearly Constrained Optimization Calculations, Mathematical Programming **45**, pp. 547-566, 1989.

[7] Zhu, C., Byrd, R. H., Lu, P., & Nocedal, J. (1997). Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. ACM Transactions on mathematical software (TOMS), 23(4), 550-560.

[8] Lagarias, J. C., Reeds, J. A., Wright, M. H., & Wright, P. E. (1998). Convergence properties of the Nelder--Mead simplex method in low dimensions. SIAM Journal on optimization, 9(1), 112-147.

[9] Kingma, D. P. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

[10] Liu, R., Wu, T., & Mozafari, B. (2020). Adam with bandit sampling for deep learning. Advances in Neural Information Processing Systems, 33, 5393-5404.

[11] Martí, R., Moreno-Vega, J. M., & Duarte, A. (2010). Advanced multi-start methods. In Handbook of metaheuristics (pp. 265-281). Springer, Boston, MA.

[12] Lambora, A., Gupta, K., & Chopra, K. (2019, February). Genetic algorithm-A literature review. In 2019 international conference on machine learning, big data, cloud and parallel computing (COMITCon) (pp. 380-384). IEEE.

[13] Blickle, T. (2000). Tournament selection. Evolutionary computation, 1(181-186), 188.

[14] Lipowski, A., & Lipowska, D. (2012). Roulette-wheel selection via stochastic acceptance. Physica A: Statistical Mechanics and its Applications, 391(6), 2193-2196.

[15] P. Kaelo, M.M. Ali, Integrated crossover rules in real coded genetic algorithms, European Journal of Operational Research **176**, pp. 60-76, 2007.

[16] Pant, M., Zaheer, H., Garcia-Hernandez, L., & Abraham, A. (2020). Differential Evolution: A review of more than two decades of research. Engineering Applications of Artificial Intelligence, 90, 103479.

[17] M.M. Ali and A. Törn, Population set-based global optimization algorithms: some modifications and numerical studies, Computers & Operations Research **31**, pp. 1703-1725, 2004.

[18] Charilogis, V., Tsoulos, I. G., Tzallas, A., & Karvounis, E. (2022). Modifications for the differential evolution algorithm. Symmetry, 14(3), 447.

[19] Yang, Q., Yuan, S., Gao, H., & Zhang, W. (2024). Differential evolution with migration mechanism and information reutilization for global optimization. Expert Systems with Applications, 238, 122076.

[20] Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the ICNN'95—International Conference on Neural Networks, Perth, Australia, 1 December 1995; Volume 4, pp. 1942–1948.

[21] Charilogis, V., & Tsoulos, I. G. (2022). Toward an ideal particle swarm optimizer for multidimensional functions. Information, 13(5), 217.

[22] R.C. Eberhart, Y.H. Shi, Tracking and optimizing dynamic systems with particle swarms, in: Congress on Evolutionary Computation, Korea, 2001.

[23] Y.H. Shi, R.C. Eberhart, Empirical study of particle swarm optimization, in: Congress on Evolutionary Computation, Washington DC, USA, 1999.

[24] Y.H. Shi, R.C. Eberhart, Experimental study of particle swarm optimization, in: SCI2000 Conference, Orlando, 2000.

[25] Y. Zheng, L. Ma, L. Zhang, J. Qian, Empirical study of particle swarm optimizer with an increasing inertia weight, in: IEEE Congress on Evolutionary Computation, 2003.

[26] Y. Zheng, L. Ma, L. Zhang, J. Qian, On the convergence analysis and param- eter selection in particle swarm optimization, in: Proceedings of the Second International Conference on Machine Learning and Cybernetics, 2003.

[27] Tsoulos, I. G., Tzallas, A., Karvounis, E., & Tsalikakis, D. (2023). NeuralMinimizer: A novel method for global optimization. Information, 14(2), 66.

[28] J. Park, I.W. Sandberg, Approximation and Radial-Basis-Function Networks, Neural Computation **5**, pp. 305-316, 1993.

[29] Charilogis, V., & Tsoulos, I. G. (2023). A parallel implementation of the differential evolution method. Analytics, 2(1), 17-30.

[30] Tosun, U., Dokeroglu, T., & Cosar, A. (2013). A robust island parallel genetic algorithm for the quadratic assignment problem. International Journal of Production Research, 51(14), 4117-4133.

[31] Charilogis, V., Tsoulos, I. G., & Tzallas, A. (2023). An improved parallel particle swarm optimization. SN Computer Science, 4(6), 766.

[32] Kirkpatrick, S., Gelatt Jr, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. science, 220(4598), 671-680.

[33] Fairchild, K. W., Misra, L., & Shi, Y. (2016). Using triangular distribution for business and finance simulations in Excel. Journal of Financial Education, 42(3-4), 313-336.

[34] Bekker, A. J. J. J., & Roux, J. J. J. (2005). Reliability characteristics of the Maxwell distribution: A Bayes estimation study. Communications in Statistics-Theory and Methods, 34(11), 2169-2178.

[35] Abiodun, O. I., Jantan, A., Omolara, A. E., Dada, K. V., Mohamed, N. A., & Arshad, H. (2018). State-of-the-art in artificial neural network applications: A survey. Heliyon, 4(11).

[36] Ahmed, M., Seraj, R., & Islam, S. M. S. (2020). The k-means algorithm: A comprehensive survey and performance evaluation. Electronics, 9(8), 1295.

Περιεχόμενα