Βελτιστοποίηση

Ιωάννης Γ. Τσούλος

Τμήμα Πληροφορικής και τηλεπικοινωνιών
Πανεπιστήμιο Ιωαννίνων

2022

Περίληψη

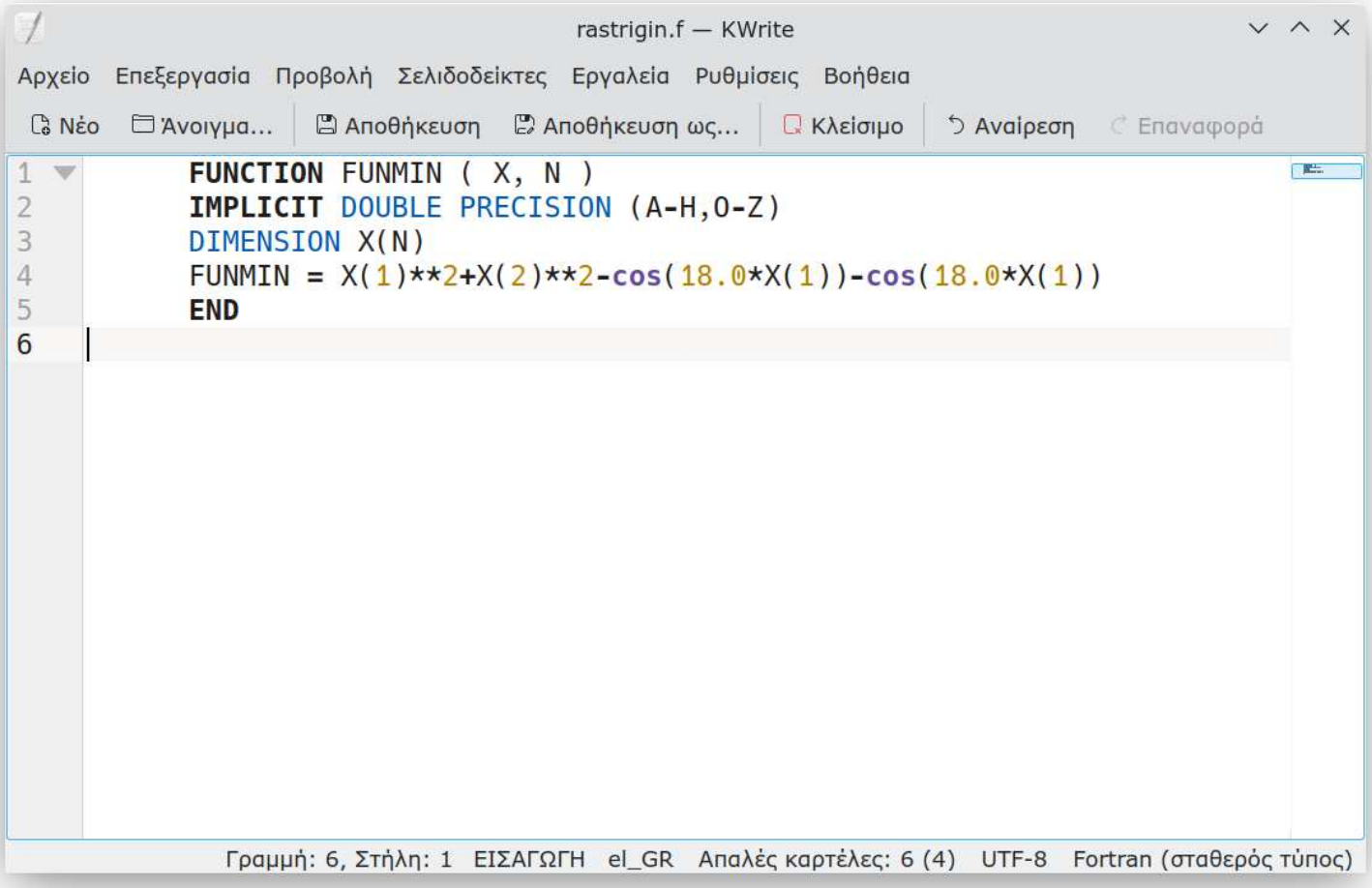
Διαθεσιμότητα

1. Διατίθεται δωρεάν από τον δικτυακό τόπο `http://merlin.cs.uoi.gr/`.
2. Είναι γραμμένο σε Fortran 77.
3. Μπορεί να εγκατασταθεί σε κάθε λειτουργικό σύστημα (Windows, Macos, Linux).
4. Όλο το πακέτο λογισμικού είναι ένα αρχείο σε Fortran 77.

Εγκατάσταση Merlin

1. Η τελευταία έκδοση είναι σε συμπιεσμένη μορφή στο `http://merlin.cs.uoi.gr/files/merlin-3.1.4.tgz`
2. Αποσυμπίεση σε ένα φάκελο με tar zxfv merlin-3.1.4.tgz
3. cd merlin-3.1.4
4. Αλλάζουμε τις προτιμήσεις εγκατάστασης στο αρχείο Makefile.inc
5. Δίνουμε make
6. Δίνουμε make install

Παράδειγμα συνάρτησης (rastrigin)



Μεταγλώττιση παραδείγματος και εκτέλεση

1. Δίνουμε /usr/local/merlin/bin/run-merlin rastrigin.f
2. Στην προτροπή δίνουμε 2 (αριθμός μεταβλητων) και 0 (αριθμός όρων για άθροισμα τετραγώνων)

Εντολές στο περιβάλλον Merlin

1. Το πακέτο δέχεται μια σειρά από εντολές που δίνονται είτε με κεφαλαία είτε με μικρά γράμματα
2. Για να εμφανιστούν οι εντολές ο χρήστης μπορεί να δώσει την εντολή **list**
3. Για βοήθεια πάνω σε μια εντολή ο χρήστης δίνει help command, πχ **help bfgs**
4. Για την εμφάνιση της τρέχουσας κατάστασης της βελτιστοποίησης υπάρχει η εντολής **shortdis** (δείχνει το τρέχον ελάχιστο)

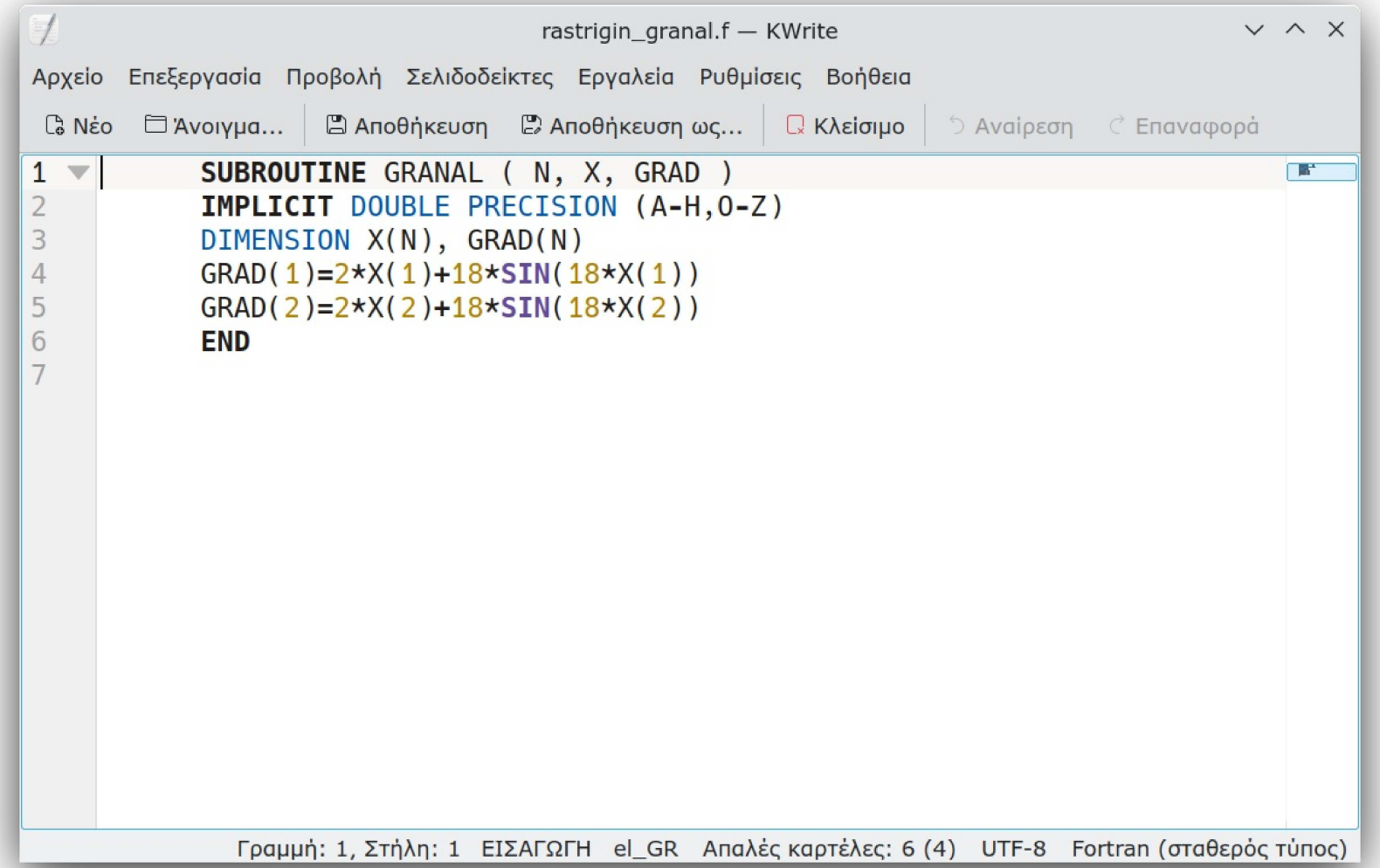
Αρχικοποίηση συνάρτησης

1. Η εντολή **lmargin 1 0.5** βάζει σαν αριστερό όριο στην πρώτη διάσταση την τιμή 0.5
2. Η εντολή **lmargin 1- 0.5** βάζει σαν αριστερό όριο σε όλες τις διαστάσεις την τιμή 0.5
3. Η εντολή **rmargin 2 0.2** βάζει σαν δεξί όριο στην δεύτερη διάσταση την τιμή 0.2
4. Η εντολή **point 1 0.2 2 0.9** αρχικοποιεί την πρώτη διάσταση στην τιμή 0.2 και την δεύτερη διάσταση στην τιμή 0.9

Εκτέλεση μεθόδων ελαχιστοποίησης (bfgs, simplex)

1. Για την εκτέλεση μιας εντολής ελαχιστοποίησης συνήθως αρκεί το όνομα της.
2. Πχ η εντολή **bfgs noc 2000** θα τρέξει την μέθοδο ελαχιστοποίησης με 2000 μέγιστο αριθμό επαναλήψεων.
3. Η εντολή **simplex** θα τρέξει την μέθοδο simplex.
4. Η εντολή **bfgs** θα εμφανίσει το πάνελ με τις διαθέσιμες επιλογές της bfgs
5. Για να ελέξουμε αν όντως οι μέθοδοι έφτασαν σε ελάχιστο υπάρχει η εντολή **graddis**.

Χρήση παραγώγου για καλύτερα αποτελέσματα



Χρήση παραγώγων

1. Για την χρήση αναλυτικής παραγώγου χρειάζεται η εντολή **anal**
2. Για χρήση άλλων τεχνικών υπάρχουν οι εντολές **fast**, **quad**
3. Για εμφάνιση της τρέχουσας τιμής της παραγώγου υπάρχει η εντολή **graddis**
4. Για τον έλεγχο ορθότητας των παραγώγων διατίθεται η εντολή **gradcheck**.

Βασικές αρχές

1. Είναι παράλληλη τεχνική βελτιστοποίησης.
2. Χρησιμοποιεί την βιβλιοθήκη OpenMPI για παραλληλισμό της εκτέλεσης.
3. Ο χρήστης μπορεί να γράψει τον κώδικά του είτε σε C++ είτε σε Fortran.

Τεχνικές παράλληλων γενετικών

1. Στην βιβλιογραφία έχουν αναπτυχθεί διάφορες αρχιτεκτονικές για παράλληλους γενετικούς που εκμεταλλεύονται τις σύγχρονες παράλληλες υπολογιστικές δομές.
2. Οι κυριότερες από αυτές είναι οι αλγόριθμοι
   1. ISLAND
   2. CELLULAR.

Ενδεικτικός αλγόριθμος

1. **Αρχικοποίηση πληθυσμού**.
2. **Υπολογισμός καταλληλότητας**.
   1. Ο master node αποστέλλει σε κάθε slave node το χρωμόσωμα που του αντιστοιχεί
   2. Υπολογισμός της αντίστοιχης καταλληλότητας.
   3. Επιστροφή στον μάστερ της τιμής.
3. **Εκτέλεση γενετικών τελεστών**., στον κεντρικό κόμβο.
4. **Έλεγχος κριτηρίου τερματισμού**, στον κεντρικό κόμβο.
5. **Μετάβαση** στο βήμα 2.

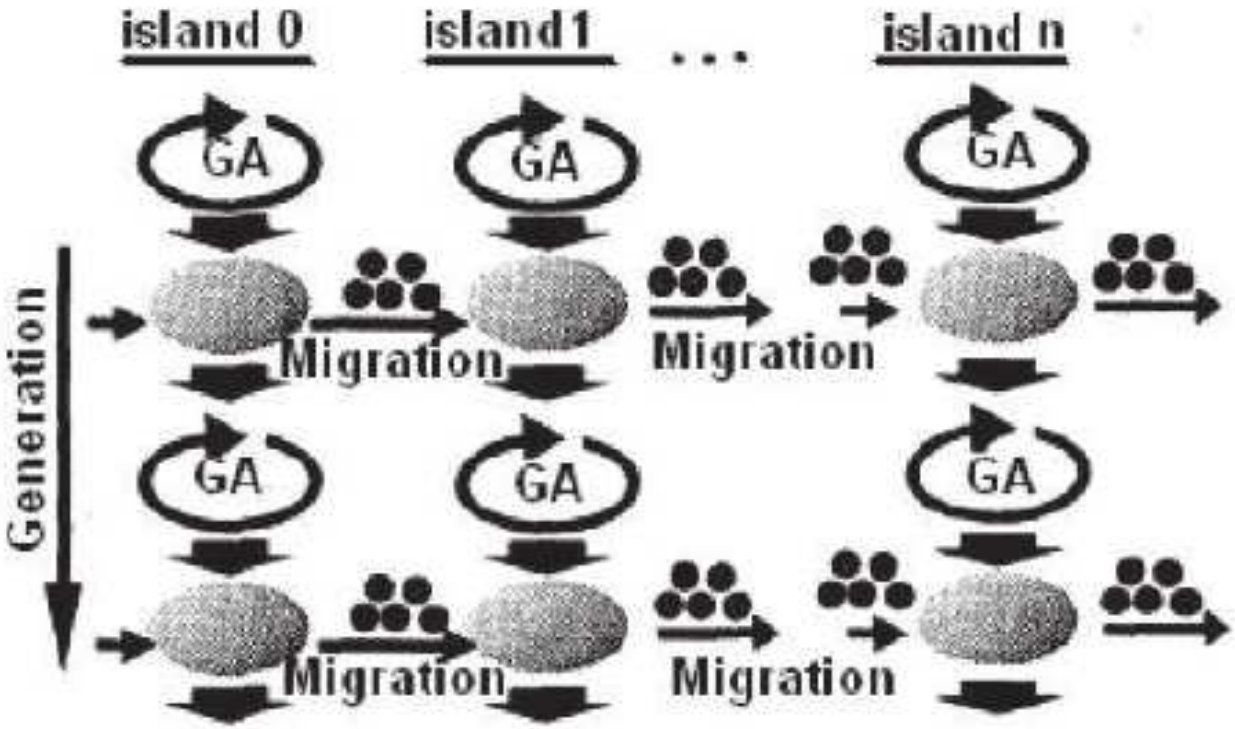
Προβλήματα

1. Θα πρέπει κάθε κόμβος να ξέρει όλο το πρόβλημα για να υπολογίσει την καταλληλότητα.
2. Οι κόμβοι επεξεργασίας θα πρέπει να είναι υπολογιστικά όμοιοι.
3. Αν τα χρωμοσώματα είναι πάρα πολλά ή μεγάλα σε μέγεθος , τότε θα δαπανηθεί αρκετός χρόνος για την επικοινωνία μεταξύ των κόμβων.
4. Αν τα χρωμοσώματα είναι περισσότερα από τους κόμβους επεξεργασίας , τότε κάθε κόμβος επεξεργασίας θα αναλάβει περισσότερους από έναν υπολογισμούς.
5. Συγχρονισμός τυχαίων αριθμών.
6. Θα πρέπει ο υπολογισμός της καταλληλότητας να έχει νόημα και να είναι πολύ δαπανηρός χρονικά.

Αλγόριθμοι ISLAND

1. Στα μοντέλα αυτά ο συνολικός πληθυσμός χωρίζεται σε ανεξάρτητα τμήματα που ονομάζονται νησιά.
2. Το κάθε τμήμα εξελίσσεται ανεξάρτητα από κάθε άλλο μέλος του πληθυσμού και η εκτέλεση γίνεται σε διαφορετικό επεξεργαστή.
3. Ανά τακτά χρονικά διαστήματα τα νησιά ανταλάσσουν μηνύματα στα οποία συνήθως μπαίνει το καλύτερα ή τα καλύτερα χρωμοσώματα κάθε νησιού.
4. Αυτά τα μηνύματα είτε στέλνονται απευθείας από νησί σε νησί είτε στέλνονται πρώτα σε ένα κεντρικό κόμβο επεξεργασίας και στην συνέχεια από αυτό στέλνονται στα νησιά.

Αλγόριθμοι ISLAND (σχήμα)



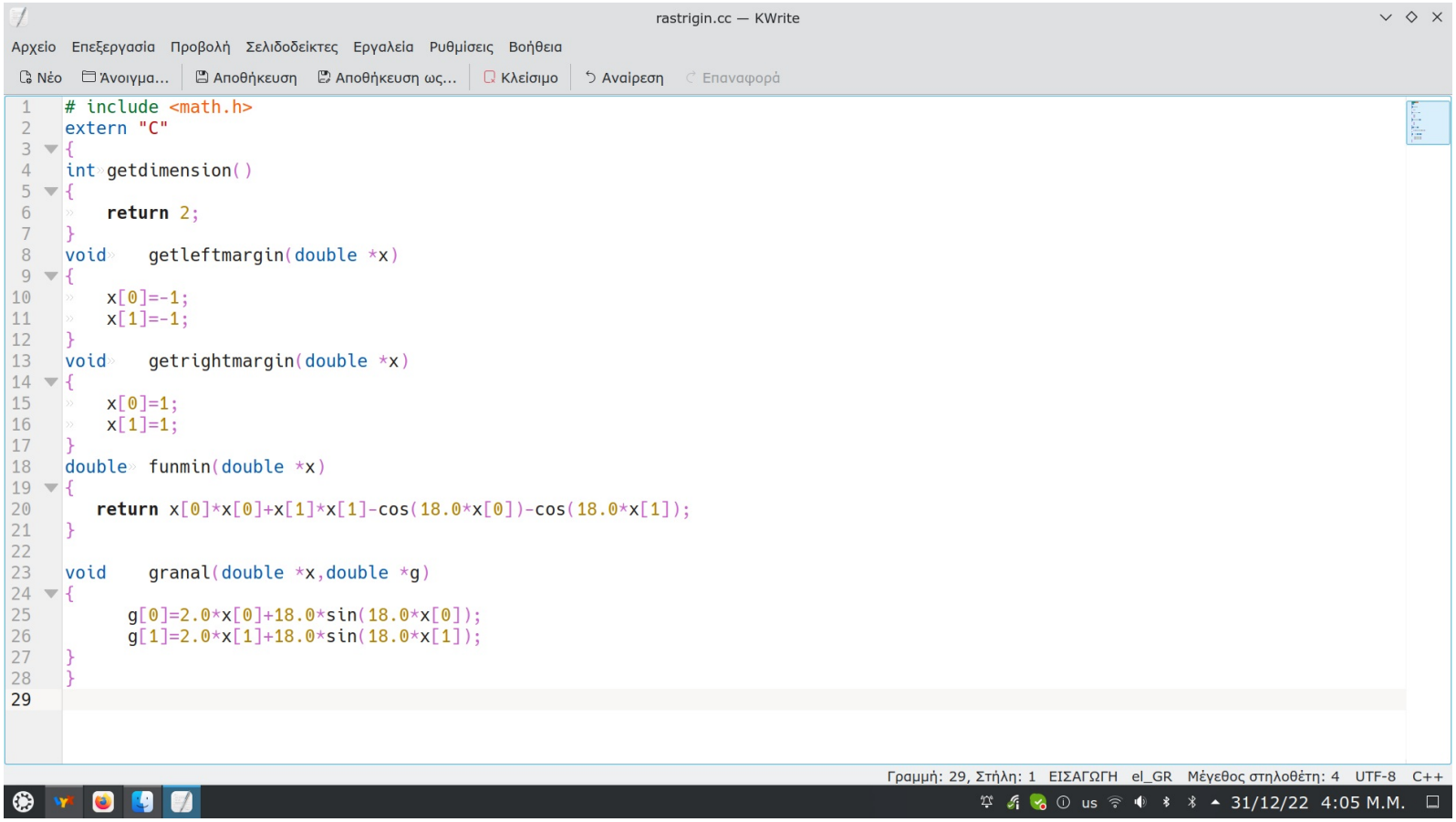
Αλγόριθμοι ISLAND (μέγεθος πληθυσμού)

1. Θα πρέπει να καθοριστεί το μέγεθος κάθε υποπληθυσμού.
2. Αν αυτό το μέγεθος είναι μικρό, τότε τα νησιά μπορεί να εγκλωβιστούν σε τοπικά ελάχιστα και να μην υπάρχει εξέλιξη.
3. Αν είναι πολύ μεγάλο κάθε νησί θα αργεί να προχωρήσει στην εξέλιξη του συνολικού αποτελέσματος.

Αλγόριθμοι ISLAND (μηνύματα)

1. Θα πρέπει να καθοριστεί κάθε πότε θα γίνεται ανταλλαγή μηνυμάτων.
2. Αν η ανταλλαγή είναι πολύ συχνή, τότε θα χάνεται πολύτιμος χρόνος για την μετάδοση δεδομένων μέσω του διαδικτύου.
3. Αν γίνεται αραιά, τότε δεν θα υπάρχει σχεδόν κανένα ώφελος από την χρήση παράλληλων τεχνικών, αφού κάθε νησί θα είναι σχεδόν ανεξάρτητο.
4. Θα πρέπει να καθοριστεί αν στα μηνύματα θα υπάρχει μόνον το καλύτερο χρωμόσωμα ή τα N καλύτερα χρωμοσώματα.
5. Θα πρέπει να καθοριστεί αν τα χρωμοσώματα που έρχονται με ανταλλαγή μηνυμάτων θα αντικαθιστούν κάποιο χρωμόσωμα στον τοπικό πληθυσμό.

Παράδειγμα συνάρτησης



Προαπαιτούμενα

1. Η διάσταση του προβλήματος
2. Τα όρια του προβλήματος
3. Η αντικειμενική συνάρτηση
4. Η παράγωγος της συνάρτησης

Μεταγλώττιση του πακέτου

1. Αποσυμπίεση πακέτου: tar zxfv PDoublePop.tar.gz
2. cd PDoublePop
3. Επεξεργασία του αρχείου Makefile.inc για αλλαγή ρυθμίσεων
4. Μεταγλώττιση με το make
5. Το τελικό προιόν είναι το πρόγραμμα *make_doublepop*

Μεταγλώττιση αντικειμενικού προβλήματος

1. Γίνεται χρήση του make_doublepop
2. Για παράδειγμα make_doublepop -p rastrigin.cc -o test
3. Το τελικό παράγωγο είναι το εκτελέσιμο test

Εκτέλεση προγράμματος

1. Ο χρήστης δίνει mpirun -np N ./test όπου N το πλήθος των επεξεργαστών.
2. Το πλήθος των επεξεργαστών πρέπει να ειναι N>=2, αφού ένας επεξεργαστής θα είναι ο master node.
3. Η μέθοδος τερματίζει όταν τελειώσουν όλα τα slave nodes.

Σύγκριση με Galib(αριθμός γενιών)

FUNCTION	GALib	PDoublePop
Rastrigin	1608.27(0.97)	19.42
Griewank2	257.13(0.67)	20.34
Griewank10	694.40(0.87)	35.38
Hansen	281.43	29.38
Rosenbrock32	881.67	31.69
Sinu16	1335.30	99.85
Shubert	239.01	17.80
Gkls350	287.53	24.97
Test2N5	840.83(0.87)	43.78
Test2N6	1067.77(0.80)	55.19
Elp16	1061.73	286.99
Potential6	863.37(0.70)	418.87
Potential7	339.03	224.73

Επιτάχυνση

PROBLEM	SERIAL	PARALLEL-2	PARALLEL-5
Elp4	0.57	0.21	0.10
Elp8	0.93	0.33	0.17
Elp16	1.29	0.53	0.14
Elp32	1.46	0.42	0.23
Elp64	1.55	0.49	0.22

Διαθεσιμότητα OPTIMUS

- Είναι γραμμένο σε C++
- Ο χρήστης μπορεί να γράψει τις συναρτήσεις σε C++
- Διαθέτει μια σειρά από συναρτήσεις τοπικής και καθολικής βελτιστοποίησης.
- Διατίθεται από `https://github.com/itsoulos/OPTIMUS.git`
- Για την ώρα εγκαθίσταται μόνο σε Linux

Εγκατάσταση

- Εγκατάσταση της βιβλιοθήκης QT από την σελίδα https://qt.io
- Ορίζουμε την μεταβλητή περιβάλλοντος OPTIMUSPATH ώστε να δείχνει στον φάκελο εγκατάστασης του OPTIMUS, πχ. OPTIMUSPATH=/home/user/OPTIMUS/
- Θέτουμε την μεταβλητή περιβάλλοντος LD_LIBRAPY_PATH LD_LIBRAPY_PATH=$LD_LIBRAPY_PATH:$OPTIMUSPATH/li
- Εκτέλεση της εντολής cd $OPTIMUSPATH
- Εκτέλεση του σεναρίου εγκατάστασης ./compile.sh

Υλοποίηση αντικειμενικού προβλήματος

- Χρειάζεται η διάσταση του προβλήματος
- Απαιτούνται τα όρια του προβλήματος
- Ο χρήστης ορίζει και την αντικειμενική συνάρτηση
- Ο χρήστης πρέπει να ορίσει και την παράγωγο του προβλήματος
- Μπορούμε να αρχικοποιήσουμε στοιχεία του προβλήματος στην μέθοδο init()
- Μπορούμε να ορίσουμε ενέργειες για το τέλος της βελτιστοποίησης στην μέθοδο done()
- Τα προβλήματα μεταγλωττίζονται σαν shared libraries.

Υλοποίηση αντικειμενικού προβλήματος

rastrigin.cc — KWrite

Αρχείο Επεξεργασία Selection Προβολή Μετάβαση Εργαλεία Ρυθμίσεις Βοήθεια

Νέο Άνοιγμα Αποθήκευση Αποθήκευση ως Αναίρεση Επαναφορά

```
13
14  void    init(QJsonObject data)
15  {}
16
17  int»getdimension()
18  {
19  »   return 2;
20  }
21  void    getmargins(vector<Interval> &x)
22  {
23          for(int i=0;i<x.size();i++)
24                  x[i]=Interval(-1,1);
25  }
26
27  double» funmin(vector<double> &x)
28  {
29     return (x[0]*x[0])+(x[1]*x[1])-cos(18.0*x[0])-cos(18.0*x[1]);
30  }
31
32  void    granal(vector<double> &x,vector<double> &g)
33  {
34        g[0]=2.0*x[0]+18.0*sin(18.0*x[0]);
35        g[1]=2.0*x[1]+18.0*sin(18.0*x[1]);
36  }
37
38  QJsonObject    done(vector<double> &x)
39  {
40      return QJsonObject();
41  }
42
```

42:1 ΕΙΣΑΓΩΓΗ el_GR Απαλές καρτέλες: 4 UTF-8 C++

Παράδειγμα βελτιστοποίησης

1. Για την βελτιστοποίηση συναρτήσεων διατίθεται η εφαρμογή OptimusApp στον φάκελο bin.
2. Παράδειγμα εκτέλεσης: ./OptimusApp --filename=../PROBLEMS/librastrigin.so --opt_method=Pso --iterations=1

Σύνοψη

- Το πακέτο λογισμικού Merlin
- Το πακέτο λογισμικού PDoublePop
- Το πακέτο λογισμικού OPTIMUS

Βιβλιογραφία I


Rosenblatt, Frank (1958), The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain, Cornell Aeronautical Laboratory, Psychological Review, v65, No. 6, pp. 386–408.


Freund, Y. and Schapire, R. E. 1998. Large margin classification using the perceptron algorithm. In Proceedings of the 11th Annual Conference on Computational Learning Theory (COLT' 98). ACM Press.