ΥΛΟΠΟΙΗΣΗ ΑΛΓΟΡΙΘΜΟΥ SIMPLEX ΓΙΑ ΤΥΧΑΙΑ ΓΡΑΜΜΙΚΑ ΠΡΟΒΛΗΜΑΤΑ ΣΕ PYTHON

Λάμπρογλου Αικατερίνη

Επιβλέπων Καθηγητής: Σαμαράς Νικόλαος

Πανεπιστήμιο Μακεδονίας

Ιούνιος 2019

Εφαρμογές

- Υλοποίηση αναθεωρημένου αλγόριθμου simplex σε γλώσσα Python
- Δημιουργία τυχαίων προβλημάτων
- Επίλυση των προβλημάτων με τον υλοποιημένο αλγόριθμο

Γενική μορφή προβλημάτων γραμμικού προγραμματισμού

$$\min\backslash\max z = c_1x_1 + c_2x_2 + \ldots\ldots\ldots + c_nx_n$$

$$\begin{aligned}
\text{μ.π.} \quad & \alpha_{11}x_1 + \alpha_{12}x_2 + \ldots\ldots\ldots + \alpha_1x_n \quad \otimes \quad b_1 \\
& \alpha_{21}x_1 + \alpha_{22}x_2 + \ldots\ldots\ldots + \alpha_{2n}x_n \quad \otimes \quad b_2 \\
& \ldots\ldots \quad \ldots\ldots\ldots \ldots\ldots \quad \ldots\ldots \ldots\ldots \quad \ldots\ldots \ldots\ldots \\
& \alpha_{m1} + \alpha_{m2}x_2 + \ldots\ldots\ldots + \alpha_{mn}x_n \quad \otimes \quad b_m
\end{aligned}$$

$$x_j \geq 0 \quad (j=1,2,\ldots,n)$$

Αναθεωρημένος αλγόριθμος simplex δύο φάσεων

<u>Βήμα 0: Αρχικοποίηση</u>

Επιλογή μιας βασικής διαμέρισης (B, N). Υπολογισμός της μήτρας B^{-1} και των διανυσμάτων x_B , s_N και w.

Αν υπάρχει $x_{Bi} < 0$, η βασική διαμέριση δεν είναι εφικτή, οπότε ξεκινάει η φάση I. Διαφορετικά, επιλέγεται το Βήμα 1.

Αναθεωρημένος αλγόριθμος simplex δύο φάσεων

Βήμα 1: Έλεγχος βελτιστότητας

Αν $s_N \geq 0$, STOP το πρόβλημα είναι βέλτιστο

Βήμα 2: Επιλογή εισερχόμενης και εξερχόμενης μεταβλητής

α)Επιλογή εισερχόμενης x_l με τον κανόνα ελαχίστου στοιχείου, όπου l= N(t).

Αναθεωρημένος αλγόριθμος simplex δύο φάσεων

<u>Βήμα 2: Επιλογή εισερχόμενης και εξερχόμενης μεταβλητής</u>

β)Υπολογισμός στήλης περιστροφής h_l

Αν $h_l < 0$, STOP, το πρόβλημα είναι απεριόριστο. Διαφορετικά επιλέγεται η εξερχόμενη μεταβλητή x_k με τον κανόνα ελαχίστου λόγου όπου $x_{B[r]} = x_k$

Αναθεωρημένος αλγόριθμος simplex δύο φάσεων

<u>Βήμα 3: Περιστροφή</u>

Ανανέωση των συνόλων δεικτών B και N. Τίθεται N(t) = k και B(r) = l.

Υπολογισμός της νέας αντιστροφής της βάσης B^{-1} και των διανυσμάτων x_B , s_N

Επιστροφή στο Βήμα 1.

Αναθεωρημένος αλγόριθμος simplex δύο φάσεων

Φάση Ι

<u>Βήμα 0: Αρχικοποίηση</u>

Εισαγωγή μιας νέας μεταβλητής απόφασης x_{n+1} με διάνυσμα συντελεστών $d= - Be$ όπου $e_i = 1$ για κάθε $i=1,2,\ldots,m$. Η μεταβλητή αυτή ονομάζεται τεχνητή.

Τίθεται $f_i =0$, $i=1,2,\ldots,n$ και $f_{n+1} =1$ για τους συντελεστές της νέας γραμμής κόστους

Αναθεωρημένος αλγόριθμος simplex δύο φάσεων

Φάση Ι

<u>Βήμα 1: Περιστροφή</u>

Επιλογή της x_{n+1} ως εισερχόμενης και αυτής με το μικρότερο δεξιό μέρος ως εξερχόμενης

Ανανέωση των συνόλων δεικτών B και N

Υπολογισμός της στήλης περιστροφής , της μήτρας B^{-1} και των διανυσμάτων x_B , s_N

Αναθεωρημένος αλγόριθμος simplex δύο φάσεων

Φάση Ι

<u>Βήμα 2: Φάση Ι</u>

Εφαρμογή του αναθεωρημένου αλγόριθμου simplex για τη βασική διαμέριση (B,N). Αν $x_{n+1} \in B$ και $x_{n+1} > 0$, STOP, το πρόβλημα είναι αδύνατο. Αν $x_{n+1} \in B$ και $x_{n+1}=0$, γίνεται μια ακόμα περιστροφή. Διαφορετικά μετακινείται απευθείας στο Βήμα 3

Αναθεωρημένος αλγόριθμος simplex δύο φάσεων

Βήμα 3: Φάση ΙΙ

Με τη βασική διαμέριση που προέκυψε από τη Φάση Ι εκτελείται ο αναθεωρημένος αλγόριθμος simplex για το αρχικό γραμμικό πρόβλημα

Δεσμοί (1)

Περισσότεροι του ενός επιλέξιμοι δείκτες

- Εισερχόμενη
  Τυχαία επιλογή

Δεσμοί (2)

- Εξερχόμενη

  Φάση ΙΙ

  Επιλογή δείκτη με τη μικρότερη τιμή

  Φάση Ι

  Προτεραιότητα έχει η τεχνητή μεταβλητή

Αντιστροφή βάσης

- Χρήση μορφής γινομένου του αντιστρόφου
- Κάθε 100 επαναλήψεις, υπολογισμός αντιστροφής βάσης από την αρχή με χρήση της built-in συνάρτησης linalg.inv

Αντιστροφή βάσης- Ο κώδικας

```
def create_eta_inv(B,r,h_l):
    '''creates the inverse eta matrix'''
    I=np.eye(len(B))
    E=np.matrix(I)
    a=-h_l.values/h_l[r]
    a[r]=1/h_l[r]
    a_rev=a.reshape(len(a),1)
    E[:,r]=a_rev
    eta_inv=E
    return eta_inv

if Iter%100==0:
    B_inv=pd.DataFrame(np.linalg.inv(B),columns=B.columns.tolist())
 else:
    E_inv=create_eta_inv(B,r,h_l)
    B_inv=pd.DataFrame(E_inv @ B_inv)
```

Τυχαία προβλήματα

- Μορφή προβλημάτων :

  $\min\backslash\max\{ c^T x : Ax \geq \text{ή} \leq b, x \geq 0\}$
- Πυκνότητα 100%
- Μέγεθος 50:850 με βήμα 50
- 10 βέλτιστα σε κάθε κλάση
- Συνολικά 170 προβλήματα

Εύρος τιμών τυχαίων προβλημάτων

A	-1000	1000
c	-400	900
b	10	100

Η συνάρτηση linprog

- Είναι η συνάρτηση επίλυσης γραμμικών προβλημάτων της Python
- Περιλαμβάνεται στη βιβλιοθήκη scipy
- Επιλύει γραμμικά προβλήματα της μορφής :

  $\min \{ c^T x : Ax \leq \text{ ή } = b, x \geq 0 \}$
- Μέθοδοι επίλυσης:

  αλγόριθμος simplex (εξ ορισμού)

  αλγόριθμος εσωτερικών σημείων

Εφαρμογή σε τυχαία γραμμικά προβλήματα (1)

Python 3.5.2 Shell

File Edit Shell Debug Options Window Help

```
>>> from scipy.optimize import linprog
>>> res=linprog(c, A_ub=A, b_ub=b, method='simplex')
>>> res
     fun: -9176.341204830896
 message: 'Optimization terminated successfully.'
     nit: 48
   slack: array([12816.77111914,     0.        ,     0.        ,  4680.25684308,
          0.        , 25112.90871744,  8867.17081438,     0.        ,
       11803.72814357,     0.        ,     0.        ,     0.        ,
        6945.59860486,     0.        , 23066.2716491 ,  2846.95391589,
       39255.0555399 ,     0.        , 43050.398972  , 27846.24062123,
       24766.64944423,  1930.40640216,     0.        ,     0.        ,
       21938.35741435,     0.        ,  7602.04997248,  4828.94873192,
          0.        , 18108.76612284,     0.        , 11448.2527794 ,
        7780.40242999,     0.        ,  8513.08556799,     0.        ,
          0.        ,     0.        ,     0.        ,  5769.22415745,
       21461.01160747,  9060.69527587,     0.        , 23708.66458735,
        1365.69690962, 13699.45174518,  4621.43335477,  7385.46203844,
       22135.67005684, 16171.69789628])
  status: 0
 success: True
       x: array([ 0.        ,  0.        ,  0.        ,  0.        ,  1.25412532,
        0.        ,  0.        ,  3.73069835, 12.79145763,  9.08357823,
        0.57326383,  0.        ,  0.        ,  0.        ,  0.        ,
        1.88273595,  0.        ,  0.        ,  7.76880696, 11.22095638,
        0.        ,  0.        ,  0.        ,  9.63930215,  1.52671194,
        0.        ,  9.98400732,  1.15903511,  0.        ,  0.        ,
        0.        ,  1.51333993,  0.        ,  2.52926035,  0.        ,
        2.69580828,  2.75265808,  8.2001877 ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
        3.00408382,  0.25734689,  0.        ,  0.        ,  4.0134304 ])
```

Εφαρμογή σε τυχαία γραμμικά προβλήματα (2)

Python 3.5.2 Shell

File Edit Shell Debug Options Window Help

```
>>> res=linprog(c, A_ub=A, b_ub=b, method='interior-point')
>>> res
     con: array([], dtype=float64)
     fun: -9176.341130145767
 message: 'Optimization terminated successfully.'
     nit: 12
   slack: array([1.28167708e+04, 3.34526071e-05, 3.72146924e-05, 4.68025676e+03,
       1.07009082e-05, 2.51129089e+04, 8.86717073e+03, 3.03008892e-06,
       1.18037280e+04, 2.74672834e-06, 3.09229881e-06, 5.64743641e-06,
       6.94559863e+03, 2.07238772e-05, 2.30662715e+04, 2.84695297e+03,
       3.92550556e+04, 1.12152156e-05, 4.30503995e+04, 2.78462409e+04,
       2.47666494e+04, 1.93040648e+03, 6.54152518e-06, 8.54075188e-06,
       2.19383573e+04, 2.24016334e-04, 7.60205020e+03, 4.82894863e+03,
       2.30985474e-05, 1.81087660e+04, 5.58724736e-04, 1.14482524e+04,
       7.78040201e+03, 5.47991494e-05, 8.51308565e+03, 1.35231613e-04,
       1.38365079e-04, 6.65062180e-06, 2.46365607e-06, 5.76922431e+03,
       2.14610116e+04, 9.06069468e+03, 4.69737211e-06, 2.37086650e+04,
       1.36569701e+03, 1.36994520e+04, 4.62143343e+03, 7.38546249e+03,
       2.21356705e+04, 1.61716981e+04])
  status: 0
 success: True
       x: array([1.14241558e-08, 1.25974558e-09, 3.54504222e-09, 1.31254842e-09,
       1.25412545e+00, 3.22247801e-09, 1.58115561e-09, 3.73069835e+00,
       1.27914576e+01, 9.08357827e+00, 5.73263741e-01, 7.09308501e-09,
       1.57898036e-09, 3.09222325e-09, 1.08094758e-09, 1.88273587e+00,
       1.61449181e-09, 1.85940090e-09, 7.76880700e+00, 1.12209565e+01,
       2.05625870e-09, 2.15469356e-09, 8.25326676e-10, 9.63930198e+00,
       1.52671209e+00, 5.85539021e-09, 9.98400736e+00, 1.15903520e+00,
       5.65056084e-10, 8.01449183e-10, 1.84248991e-08, 1.51334004e+00,
       8.92934298e-09, 2.52926051e+00, 1.34747129e-09, 2.69580823e+00,
       2.75265800e+00, 8.20018780e+00, 1.07793616e-09, 1.12998218e-08,
       4.32468496e-09, 2.20741891e-09, 3.78911249e-10, 9.09704136e-10,
       1.76281039e-08, 3.00408368e+00, 2.57346792e-01, 1.61633379e-09,
       3.46222486e-09, 4.01343060e+00])
>>>
```

Εφαρμογή σε τυχαία γραμμικά προβλήματα (3)

Python 3.5.2 Shell

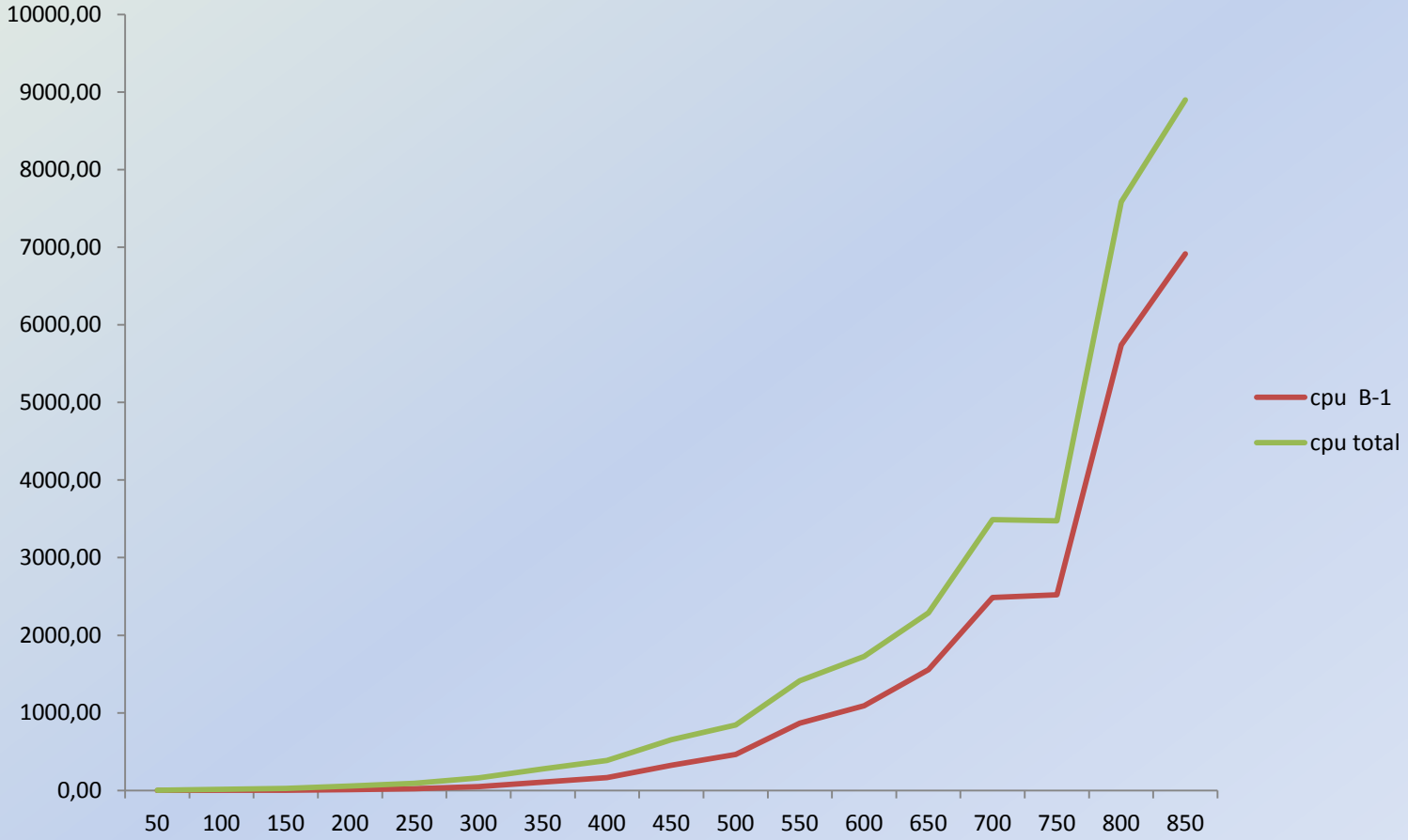
File Edit Shell Debug Options Window Help

```
The problem is feasible
Number of iterations: 49
The optimal solution is:
 ['x51' 'x28' 'x11' 'x54' 'x25' 'x56' 'x57' 'x9' 'x59' 'x46' 'x19' 'x27'
 'x36' 'x24' 'x65' 'x66' 'x67' 'x5' 'x69' 'x70' 'x71' 'x37' 'x95' 'x34'
 'x75' 'x50' 'x77' 'x78' 'x10' 'x80' 'x97' 'x82' 'x83' 'x16' 'x85' 'x8'
 'x32' 'x72' 'x47' 'x90' 'x91' 'x92' 'x20' 'x94' 'x63' 'x96' 'x38' 'x98'
 'x99' 'x100'] =
 [1.28167711e+04 1.15903511e+00 5.73263834e-01 4.68025684e+03
 1.52671194e+00 2.51129087e+04 8.86717081e+03 1.27914576e+01
 1.18037281e+04 3.00408382e+00 7.76880696e+00 9.98400732e+00
 2.69580828e+00 9.63930215e+00 2.30662716e+04 2.84695392e+03
 3.92550555e+04 1.25412532e+00 4.30503990e+04 2.78462406e+04
 2.47666494e+04 2.75265808e+00 1.36569691e+03 2.52926035e+00
 2.19383574e+04 4.01343040e+00 7.60204997e+03 4.82894873e+03
 9.08357823e+00 1.81087661e+04 4.62143335e+03 1.14482528e+04
 7.78040243e+03 1.88273595e+00 8.51308557e+03 3.73069835e+00
 1.51333993e+00 1.93040640e+03 2.57346891e-01 5.76922416e+03
 2.14610116e+04 9.06069528e+03 1.12209564e+01 2.37086646e+04
 6.94559860e+03 1.36994517e+04 8.20018770e+00 7.38546204e+03
 2.21356701e+04 1.61716979e+04]
The optimal solution of the objective function is z= [[-9176.34120483]]
>>>
```

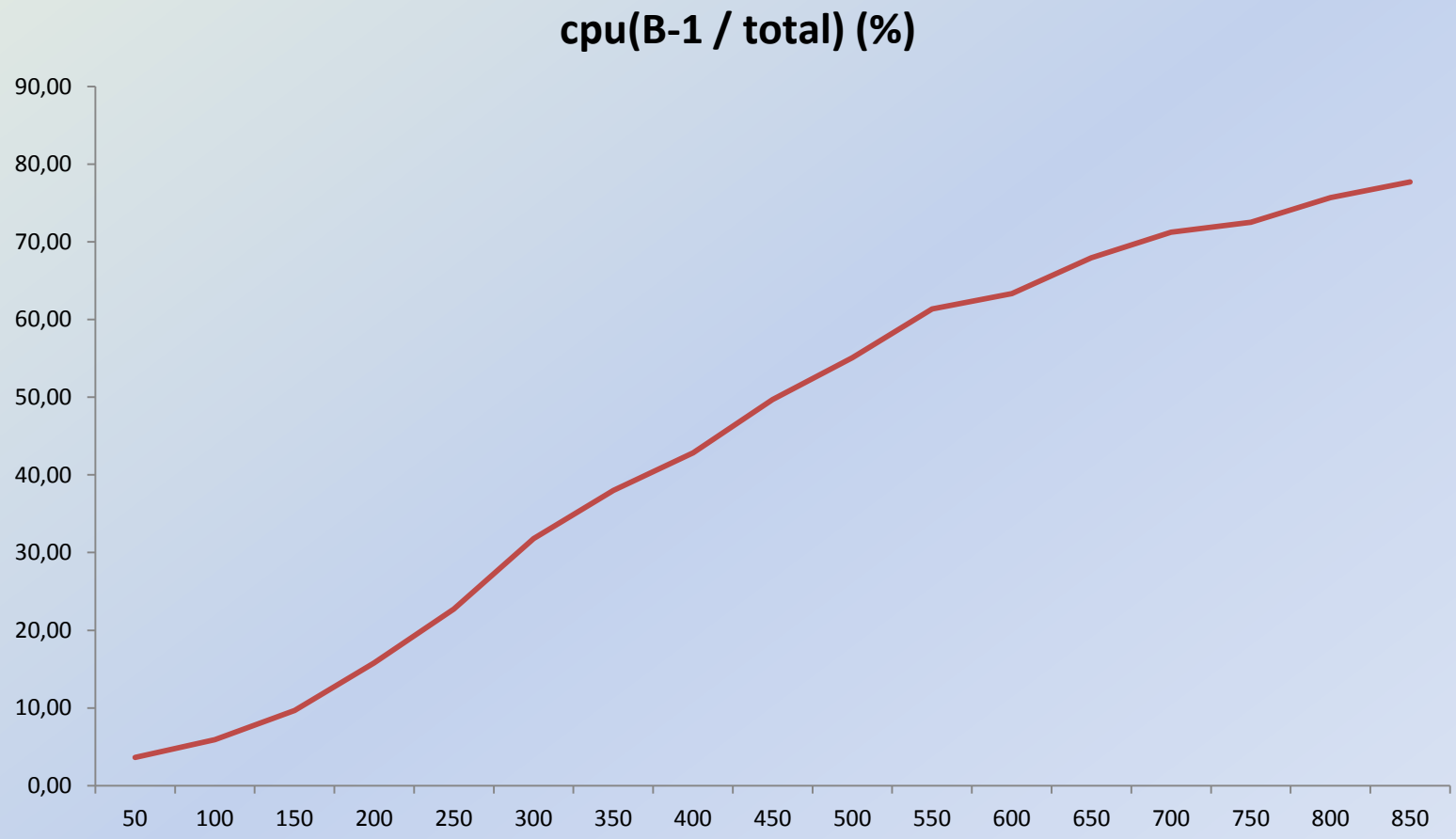
Υπολογιστική Μελέτη (1)

nxn	iter	cpu B^{-1}	cpu total	cpu(B^{-1} / total) (%)	Αύξηση cpu
50x50	107,7	0,13	3,48	3,64	1,00
100x100	356,2	0,85	14,32	5,91	4,12
150x150	703,4	2,53	26,10	9,69	7,51
200x200	1321,0	8,73	55,25	15,79	15,90
250x250	1886,5	20,44	89,96	22,73	25,88
300x300	2676,7	50,95	160,20	31,80	46,09
350x350	3954,3	106,25	279,53	38,01	80,42
400x400	4626,8	166,03	387,47	42,85	111,48
450x450	6275,3	325,17	654,24	49,70	188,23
500x500	6727,3	464,68	843,66	55,08	242,73
550x550	9302,7	868,28	1414,69	61,38	407,03
600x600	9836,8	1093,08	1725,37	63,35	496,41
650x650	10026,5	1555,97	2289,74	67,95	658,79
700x700	12888,1	2486,32	3489,20	71,26	1003,89
750x750	11795,4	2519,64	3474,88	72,50	999,77
800x800	19843,1	5742,14	7585,11	75,70	2182,34
850x850	19471,6	6916,08	8899,50	77,71	2560,50

Υπολογιστική Μελέτη (2)



Υπολογιστική Μελέτη (3)



Συμπεράσματα - Σχόλια

Για τυχαία πυκνά προβλήματα

- Αύξηση των διαστάσεων συνεπάγεται μεγάλη αύξηση του χρόνου επίλυσης
- Άρα μη ικανοποιητική απόδοση του αλγόριθμου

Περαιτέρω μελέτη

- Σε προβλήματα μεγαλυτέρων διαστάσεων

Βελτίωση

- Παράλληλη υλοποίηση

Σχετικά με την Python

- Εύκολη γραφή κώδικα
- Μεγάλη γκάμα βιβλιοθηκών που συνεχώς ανανεώνονται
- Μετατροπή τύπων δεδομένων (πχ list, DataFrame σε array, matrix) για διεκπεραίωση επιστημονικών υπολογισμών

Ευχαριστώ!