ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ

Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών

Λογικός Προγραμματισμός

Ασκήσεις

Παναγιώτης Σταματόπουλος

Τμήμα Πληροφορικής και Τηλεπικοινωνιών

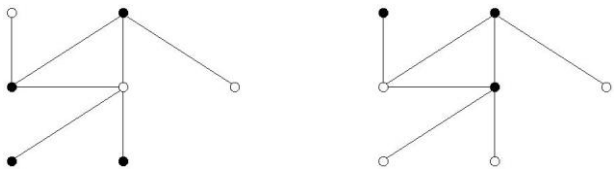
Περιεχόμενα

1. Β΄ Ομάδα Ασκήσεων "Λογικού Προγραμματισμού" Ακαδημαϊκού Έτους 2011-12

Οι ασκήσεις της ομάδας αυτής πρέπει να αντιμετωπισθούν με τη βοήθεια της τεχνολογίας του **προγραμματισμού με περιορισμούς**. Ένα σύστημα λογικού προγραμματισμού που υποστηρίζει την τεχνολογία αυτή είναι η **ECL^{i}PS^e**. Μπορείτε να χρησιμοποιήσετε είτε την παλιότερη βιβλιοθήκη **`fd`** είτε την νεώτερη **`ic`**. Η βιβλιοθήκη **`fd`** τεκμηριώνεται στο κεφάλαιο 2 του “Obsolete Libraries Manual” και η **`ic`** στα κεφάλαια 3 και 4 του “Constraint Library Manual”. Αν χρησιμοποιήσετε την **`ic`**, θα σας χρειαστεί και η βιβλιοθήκη **branch_and_bound**, ιδιαιτέρως το κατηγόρημα bb_min/3, το οποίο τεκμηριώνεται, όπως και όλα τα κατηγορήματα που παρέχει η **ECL^{i}PS^e**, στο “Alphabetical Predicate Index”.

1.1 Άσκηση 4

Στη Θεωρητική Πληροφορική, το πρόβλημα της βέλτιστης κάλυψης κορυφών (optimal vertex cover) ενός γράφου συνίσταται στην εύρεση ενός συνόλου κορυφών, ελαχίστου πλήθους, τέτοιου ώστε για κάθε ακμή του γράφου, τουλάχιστον μία από τις κορυφές-άκρα της να ανήκει στο σύνολο αυτό. Στο σχήμα αριστερά φαίνεται μία κάλυψη κορυφών του δεδομένου γράφου (μαύροι κόμβοι), η οποία όμως δεν είναι βέλτιστη. Περιλαμβάνει τέσσερις κόμβους, ενώ στο δεξιό σχήμα η κάλυψη κορυφών είναι βέλτιστη (με τρεις κόμβους), αφού δεν είναι δυνατόν να υπάρξει κάποια με δύο κόμβους.



Για την αντιμετώπιση του προβλήματος αυτού, πρέπει να χρησιμοποιήσετε γράφους που κατασκευάζονται μέσω του κατηγορήματος `create_graph(N, D, G)`, όπως αυτό ορίζεται στο αρχείο http://www.di.uoa.gr/~takis/graph.pl. Κατά την κλήση του κατηγορήματος δίνεται το πλήθος N των κόμβων του γράφου και η πυκνότητά του D (σαν ποσοστό των ακμών που υπάρχουν στον γράφο σε σχέση με όλες τις δυνατές ακμές) και επιστρέφεται ο γράφος G σαν μία λίστα από ακμές της μορφής N1-N2, όπου N1 και N2 είναι οι δύο κόμβοι της ακμής (οι κόμβοι του γράφου παριστάνονται σαν ακέραιοι από το 1 έως το N). Ένα παράδειγμα εκτέλεσης του κατηγορήματος αυτού είναι το εξής:[1]

```
?- seed(1), create_graph(9, 30, G).

G = [1 - 5, 2 - 4, 2 - 6, 3 - 4, 3 - 6, 3 - 9, 4 - 7, 5 - 7,
     6 - 7, 6 - 8, 6 - 9]
```

Για την άσκηση αυτή θα πρέπει να ορίσετε ένα κατηγόρημα `vertexcover/3`, το οποίο όταν καλείται σαν `vertexcover(N, D, C)`, αφού δημιουργήσει ένα γράφο N κόμβων και πυκνότητας D,

[1] Το κατηγόρημα seed/1 αρχικοποιεί τη γεννήτρια τυχαίων αριθμών. Η συγκεκριμένη εκτέλεση έγινε σε μηχάνημα Linux του εργαστηρίου του Τμήματος.

καλώντας το κατηγόρημα `create_graph/3`, να επιστρέφει στο C μία λίστα από κόμβους που αποτελούν μία βέλτιστη κάλυψη κορυφών του γράφου. Κάποια παραδείγματα εκτέλεσης είναι τα εξής:[2]

```
?- seed(2012), vertexcover(9, 30, C).
C = [1, 6, 9]

?- seed(1000), vertexcover(17, 50, C).

C = [1, 2, 3, 4, 5, 6, 8, 9, 11, 15, 16, 17]

?- seed(12345), vertexcover(33, 65, C).

C = [1, 2, 3, 4, 5, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17, 18,
     19, 20, 21, 23, 26, 27, 28, 29, 30, 31, 32, 33]

?- seed(1), vertexcover(100, 70, C), length(C, L).

C = [2, 3, 5, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19,
     20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, .....
L = 94

?- seed(100), vertexcover(100, 35, C), length(C, L).

C = [1, 3, 4, 5, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18,
     19, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, .....
L = 87
```

Παραδοτέο για την άσκηση είναι **ένα πηγαίο αρχείο Prolog**.

1.2 Άσκηση 5

[2] Σε μηχάνημα Linux του εργαστηρίου του Τμήματος.

Στην άσκηση αυτή, θα αντιμετωπίσετε το *ετερογενές πρόβλημα δρομολόγησης οχημάτων με χωρητικότητες* (heterogeneous capacitated vehicle routing problem). Στο πρόβλημα αυτό, υπάρχει μία εταιρεία, η οποία πρόκειται να διανείμει συγκεκριμένες ποσότητες από το προϊόν που παράγει σε συγκεκριμένους πελάτες. Όλο το προϊόν βρίσκεται αρχικά στην αποθήκη της εταιρείας. Για τη διανομή των παραγγελιών στους πελάτες, θα χρησιμοποιηθεί ένας στόλος από οχήματα, πιθανώς διαφορετικών χωρητικοτήτων το καθένα. Δεδομένα ενός στιγμιότυπου του προβλήματος για 8 οχήματα και 20 πελάτες δίνονται στη μορφή των γεγονότων Prolog που φαίνονται στη συνέχεια. Τα δεδομένα αυτά μπορείτε να τα πάρετε από το αρχείο http://www.di.uoa.gr/~takis/hcvrp_data.pl.

```
vehicles([35, 40, 55, 15, 45, 25, 85, 55]).


clients([c(15,  77,  97), c(23, -28,  64), c(14,  77, -39),
         c(13,  32,  33), c(18,  32,   8), c(18, -42,  92),
         c(19,  -8,  -3), c(10,   7,  14), c(18,  82, -17),
         c(20, -48, -13), c(15,  53,  82), c(19,  39, -27),
         c(17, -48, -13), c(12,  53,  82), c(11,  39, -27),
         c(15, -48, -13), c(25,  53,  82), c(14, -39,   7),
         c(22,  17,   8), c(23, -38,  -7)]).
```

Η λίστα που δίνεται σαν όρισμα στο κατηγόρημα `vehicles/1` αντιστοιχεί στα φορτηγά της εταιρείας. Κάθε στοιχείο της λίστας είναι η χωρητικότητα του αντίστοιχου φορτηγού, για το προϊόν που παράγει η εταιρεία. Η λίστα στο κατηγόρημα `clients/1` αναπαριστά τα δεδομένα των πελατών της εταιρείας. Τα στοιχεία της λίστας είναι δομές της μορφής `c(D, X, Y)`, που κάθε μία αντιστοιχεί σε έναν πελάτη, όπου D είναι η ποσότητα του προϊόντος που έχει παραγγείλει ο πελάτης και τα `X` και `Y` είναι οι συντεταγμένες του.

Το ζητούμενο είναι να διανεμηθεί σε κάθε πελάτη η ποσότητα του προϊόντος που έχει παραγγείλει, με μία αποστολή. Κάθε φορτηγό, εφ' όσον χρησιμοποιηθεί για τη διανομή, θα πρέπει να κάνει ένα μόνο δρομολόγιο, αναλαμβάνοντας να εξυπηρετήσει συγκεκριμένους πελάτες. Θα ξεκινήσει από την αποθήκη, έχοντας φορτώσει ποσότητα του προϊόντος ίση με το σύνολο των παραγγελιών των πελατών που θα εξυπηρετήσει, η οποία δεν πρέπει να υπερβαίνει τη χωρητικότητά του, θα επισκεφθεί τους πελάτες με κάποια σειρά, για να τους παραδώσει τις παραγγελίες τους, και θα επιστρέψει στην αποθήκη. Η αποθήκη βρίσκεται στη θέση (0,0). Η ικανοποίηση των παραγγελιών πρέπει να γίνει με τον βέλτιστο για την εταιρεία τρόπο, που συνίσταται στην ελαχιστοποίηση της συνολικής απόστασης που θα διανύσουν τα φορτηγά. Η αποθήκη και οι πελάτες συνδέονται ανά δύο μεταξύ τους με δρόμους που είναι ευθείες γραμμές. Δηλαδή, σαν απόσταση μεταξύ δύο πελατών, ή της αποθήκης και ενός πελάτη, θεωρείται η ευκλείδεια απόστασή τους. Για λόγους επαλήθευσης των αποτελεσμάτων που θα δοθούν στη συνέχεια, μπορεί να θεωρηθεί ότι η απόσταση, αφού πολλαπλασιαστεί με το 1000, στρογγυλεύεται στον πλησιέστερο ακέραιο (ουσιαστικά, πρόκειται για στρογγύλευση στο τρίτο δεκαδικό ψηφίο).

Ορίστε ένα κατηγόρημα `hcvrp/6`, το οποίο όταν καλείται σαν `hcvrp(NCl, NVe, Timeout, Solution, Cost, Time)`, να επιλύει το πρόβλημα, λαμβάνοντας ως δεδομένα τους πρώτους NCl πελάτες από τη λίστα του `clients/1` και τα πρώτα `NVe` οχήματα από τη λίστα του `vehicles/1`. Το κατηγόρημα να επιστρέφει στο Solution μία λίστα κάθε στοιχείο της οποίας

αντιστοιχεί σε ένα φορτηγό και είναι επίσης μία λίστα με τους αύξοντες αριθμούς των πελατών που θα εξυπηρετήσει το εν λόγω φορτηγό, και μάλιστα με τη σειρά που θα τους επισκεφθεί. Το Cost είναι το κόστος της λύσης (συνολική διανυθείσα απόσταση από τα φορτηγά). Ιδανικά, πρέπει να βρίσκεται η βέλτιστη λύση. Όμως αυτό δεν είναι εφικτό για τις μεγαλύτερες εισόδους, οπότε μπορεί να δοθεί κατά την κλήση του κατηγορήματος το Timeout, που είναι ο χρόνος CPU (σε δευτερόλεπτα) στον οποίο θα πρέπει να τερματισθεί η αναζήτηση, αν δεν έχει βρεθεί η βέλτιστη λύση, και να επιστραφεί η καλύτερη που έχει βρεθεί μέχρι εκείνη τη στιγμή. Για Timeout ίσο με 0, δεν θα πρέπει να διακοπεί η αναζήτηση μέχρι να βρεθεί η βέλτιστη λύση. Στο Time να επιστρέφεται ο χρόνος εκτέλεσης (σε CPU seconds). Κάποια παραδείγματα εκτέλεσης είναι τα εξής:[3]

```
?- hcvrp(1, 1, 0, Solution, Cost, Time).
Found a solution with cost 247694

Solution = [[1]]
Cost = 247694
Time = 0.0

?- hcvrp(2, 1, 0, Solution, Cost, Time).
Found no solution with cost 0.0 .. 371541.0

?- hcvrp(2, 2, 0, Solution, Cost, Time).
Found a solution with cost 387408
Found a solution with cost 303768
Found no solution with cost 0.0 .. 303767.0

Solution = [[], [2, 1]]
Cost = 303768
Time = 0.0

?- hcvrp(3, 2, 0, Solution, Cost, Time).
Found a solution with cost 485874
```

[3] Στο μηχάνημα linux27 του εργαστηρίου του Τμήματος.

```
Found a solution with cost 476394
Found no solution with cost 0.0 .. 476393.0

Solution = [[3], [2, 1]]
Cost = 476394
Time = 0.01

?- hcvrp(4, 2, 0, Solution, Cost, Time).
Found a solution with cost 520954
Found no solution with cost 0.0 .. 520953.0

Solution = [[4, 3], [2, 1]]
Cost = 520954
Time = 0.02

?- hcvrp(5, 2, 0, Solution, Cost, Time).
Found no solution with cost 0.0 .. 1718544.0

?- hcvrp(5, 3, 0, Solution, Cost, Time).
Found a solution with cost 654901
Found a solution with cost 589826
Found a solution with cost 580346
Found a solution with cost 571261
Found a solution with cost 506186
Found a solution with cost 488492
Found no solution with cost 0.0 .. 488491.0

Solution = [[5, 3], [], [4, 1, 2]]
Cost = 488492
Time = 0.31
```

```
?- hcvrp(6, 3, 0, Solution, Cost, Time).
Found a solution with cost 729877
Found a solution with cost 709946
Found a solution with cost 706149
Found a solution with cost 690760
Found a solution with cost 652408
Found a solution with cost 634714
Found no solution with cost 0.0 .. 634713.0

Solution = [[5, 3], [4, 1], [6, 2]]
Cost = 634714
Time = 1.44

?- hcvrp(7, 3, 0, Solution, Cost, Time).
Found a solution with cost 821699
Found a solution with cost 756624
Found a solution with cost 755723
. . . . . .
Found a solution with cost 685498
Found a solution with cost 675115
Found a solution with cost 670944
Found no solution with cost 0.0 .. 670943.0

Solution = [[3, 1], [7, 5], [4, 6, 2]]
Cost = 670944
Time = 5.69

?- hcvrp(8, 3, 0, Solution, Cost, Time).
Found no solution with cost 0.0 .. 4619061.0
```

```
?- hcvrp(8, 4, 0, Solution, Cost, Time).
Found a solution with cost 852176
Found a solution with cost 787101
Found a solution with cost 786200
. . . . . .
Found a solution with cost 712619
Found a solution with cost 706419
Found a solution with cost 702248
Found no solution with cost 0.0 .. 702247.0

Solution = [[3, 1], [7, 5], [4, 6, 2], [8]]
Cost = 702248
Time = 65.77

?- hcvrp(9, 4, 0, Solution, Cost, Time).
Found no solution with cost 0.0 .. 6866668.0

?- hcvrp(9, 5, 300, Solution, Cost, Time).
Found a solution with cost 1004995
Found a solution with cost 939920
Found a solution with cost 930440
. . . . . .
Found a solution with cost 743513
Found a solution with cost 700225
Found a solution with cost 696500
Branch-and-bound timeout!

Solution = [[7], [4, 1], [5, 9, 3], [8], [6, 2]]
Cost = 696500
```

```
Time = 300.03

?- hcvrp(9, 5, 0, Solution, Cost, Time).
Found a solution with cost 1004995
. . . . . .
Found a solution with cost 694976
Found a solution with cost 672176
Found a solution with cost 665578
Found no solution with cost 0.0 .. 665577.0

Solution = [[7], [8, 1, 4], [5, 9, 3], [], [6, 2]]
Cost = 665578
Time = 2236.39

?- hcvrp(10, 5, 300, Solution, Cost, Time).
Found a solution with cost 1070882
Found a solution with cost 1025051
Found a solution with cost 959976
. . . . . .
Found a solution with cost 851423
Found a solution with cost 825929
Found a solution with cost 782641
Branch-and-bound timeout!

Solution = [[9, 3], [10, 7], [5, 1, 4], [8], [6, 2]]
Cost = 782641
Time = 300.03

?- hcvrp(10, 5, 0, Solution, Cost, Time).
Found a solution with cost 1070882
```

```
. . . . . .
Found a solution with cost 782641
Found a solution with cost 778916
Found no solution with cost 0.0 .. 778915.0

Solution = [[4, 1], [10, 7], [5, 9, 3], [8], [6, 2]]
Cost = 778916
Time = 12281.82

?- hcvrp(11, 5, 600, Solution, Cost, Time).
Found a solution with cost 1227161
Found a solution with cost 1226884
Found a solution with cost 1220306
. . . . . .
Found a solution with cost 1121383
Found a solution with cost 1109757
Found a solution with cost 1101024
Branch-and-bound timeout!

Solution = [[11, 10], [2, 1], [7, 6, 5], [8], [4, 9, 3]]
Cost = 1101024
Time = 599.68

?- hcvrp(11, 5, 3600, Solution, Cost, Time).
Found a solution with cost 1227161
. . . . . .
Found a solution with cost 1055385
Found a solution with cost 1047698
Found a solution with cost 1043527
Branch-and-bound timeout!
```

```
Solution = [[11, 5], [10, 7], [4, 6, 2], [3], [9, 1, 8]]
Cost = 1043527
Time = 3599.61

?- hcvrp(11, 5, 5400, Solution, Cost, Time).
Found a solution with cost 1227161
. . . . . .
Found a solution with cost 1028512
Found a solution with cost 995302
Found a solution with cost 983676
Branch-and-bound timeout!

Solution = [[11, 5], [10, 7], [8, 4, 9, 3], [1], [6, 2]]
Cost = 983676
Time = 5399.61

?- hcvrp(11, 5, 14400, Solution, Cost, Time).
Found a solution with cost 1227161
. . . . . .
Found a solution with cost 983676
Branch-and-bound timeout!

Solution = [[11, 5], [10, 7], [8, 4, 9, 3], [1], [6, 2]]
Cost = 983676
Time = 14398.19

?- hcvrp(11, 5, 0, Solution, Cost, Time).
Found a solution with cost 1227161
. . . . . .
```

```
Found a solution with cost 983676
Found a solution with cost 980274
Found no solution with cost 0.0 .. 980273.0

Solution = [[5, 1], [10, 7], [8, 4, 9, 3], [11], [6, 2]]
Cost = 980274
Time = 83603.02

?- hcvrp(12, 5, 900, Solution, Cost, Time).
Branch-and-bound timeout!

?- hcvrp(12, 6, 900, Solution, Cost, Time).
Found a solution with cost 1358715
Found a solution with cost 1314512
Found a solution with cost 1295220
. . . . . .
Found a solution with cost 1190030
Found a solution with cost 1139720
Found a solution with cost 1128094
Branch-and-bound timeout!

Solution = [[12, 8], [10, 7], [6, 1, 5], [11], [4, 9, 3], [2]]
Cost = 1128094
Time = 899.99

?- hcvrp(12, 6, 14400, Solution, Cost, Time).
Found a solution with cost 1358715
. . . . . .
Found a solution with cost 1094260
Found a solution with cost 1090535
```

```
Found a solution with cost 1088828
Branch-and-bound timeout!

Solution = [[12, 1], [10, 7], [5, 9, 3], [11], [6, 2], [8, 4]]
Cost = 1088828
Time = 14398.64

?- hcvrp(13, 6, 900, Solution, Cost, Time).
Branch-and-bound timeout!

?- hcvrp(13, 7, 900, Solution, Cost, Time).
Found a solution with cost 1501062
Found a solution with cost 1417422
Found a solution with cost 1390636
. . . . . .
Found a solution with cost 1205205
Found a solution with cost 1198394
Found a solution with cost 1188555
Branch-and-bound timeout!

Solution = [[13, 9], [12, 10], [7, 3, 5], [11], [], [],
            [8, 4, 1, 6, 2]]
Cost = 1188555
Time = 900.0

?- hcvrp(13, 7, 14400, Solution, Cost, Time).
Found a solution with cost 1501062
. . . . . .
Found a solution with cost 1119426
Found a solution with cost 1116990
```

```
Found a solution with cost 1089208
Branch-and-bound timeout!

Solution = [[13, 9], [10, 7], [12, 3, 5], [11], [], [],
            [8, 4, 1, 6, 2]]
Cost = 1089208
Time = 14398.3

?- hcvrp(14, 7, 900, Solution, Cost, Time).
Found a solution with cost 1668002
Found a solution with cost 1602927
Found a solution with cost 1585510
. . . . . .
Found a solution with cost 1262278
Found a solution with cost 1261366
Found a solution with cost 1233584
Branch-and-bound timeout!

Solution = [[14, 13], [12, 10], [9, 3, 5], [11], [7], [],
            [8, 4, 1, 6, 2]]
Cost = 1233584
Time = 900.03

?- hcvrp(14, 7, 14400, Solution, Cost, Time).
Found a solution with cost 1668002
. . . . . .
Found a solution with cost 1261366
Found a solution with cost 1233584
Found a solution with cost 1226986
Branch-and-bound timeout!
```

```
Solution = [[14, 13], [12, 10], [5, 9, 3], [11], [7], [],
            [8, 4, 1, 6, 2]]
Cost = 1226986
Time = 14385.58

?- hcvrp(15, 7, 900, Solution, Cost, Time).
Found a solution with cost 1675962
Found a solution with cost 1610887
Found a solution with cost 1556658
. . . . . .
Found a solution with cost 1309087
Found a solution with cost 1296364
Found a solution with cost 1283969
Branch-and-bound timeout!

Solution = [[15, 14, 8], [13, 12], [9, 3, 5], [11], [10, 7],
            [], [4, 1, 6, 2]]
Cost = 1283969
Time = 897.84

?- hcvrp(15, 7, 14400, Solution, Cost, Time).
Found a solution with cost 1675962
. . . . . .
Found a solution with cost 1170885
Found a solution with cost 1169510
Found a solution with cost 1165339
Branch-and-bound timeout!

Solution = [[15, 14, 8], [13, 10], [4, 6, 2], [11], [7], [],
```

```
            [12, 3, 9, 1, 5]]
Cost = 1165339
Time = 14357.16

?- hcvrp(16, 7, 900, Solution, Cost, Time).
Found a solution with cost 1832263
Found a solution with cost 1767188
Found a solution with cost 1756286
. . . . . .
Found a solution with cost 1443468
Found a solution with cost 1426409
Found a solution with cost 1404173
Branch-and-bound timeout!

Solution = [[16, 15], [14, 13, 8], [12, 9, 3], [11], [10, 7],
            [5], [4, 1, 6, 2]]
Cost = 1404173
Time = 899.15

?- hcvrp(16, 7, 14400, Solution, Cost, Time).
Found a solution with cost 1832263
. . . . . .
Found a solution with cost 1397307
Found a solution with cost 1395888
Found a solution with cost 1391717
Branch-and-bound timeout!

Solution = [[16, 15], [14, 13, 8], [4, 6, 2], [11], [10, 7],
            [], [12, 3, 9, 1, 5]]
Cost = 1391717
```

```
Time = 14395.24

?- hcvrp(17, 7, 900, Solution, Cost, Time).
Found a solution with cost 1895802
Found a solution with cost 1830727
Found a solution with cost 1829812
. . . . . .
Found a solution with cost 1577313
Found a solution with cost 1564590
Found a solution with cost 1553978
Branch-and-bound timeout!

Solution = [[17, 8], [15, 14, 13], [12, 11, 5], [16],
            [4, 9, 3], [7], [10, 2, 6, 1]]
Cost = 1553978
Time = 900.1

?- hcvrp(17, 7, 14400, Solution, Cost, Time).
Found a solution with cost 1895802
. . . . . .
Found a solution with cost 1374411
Found a solution with cost 1354336
Found a solution with cost 1347470
Branch-and-bound timeout!

Solution = [[17, 8], [15, 14, 13], [12, 3, 9], [16], [10, 7],
            [5], [4, 11, 1, 6, 2]]
Cost = 1347470
Time = 14396.54
```

```
?- hcvrp(18, 7, 900, Solution, Cost, Time).
Branch-and-bound timeout!

?- hcvrp(18, 8, 900, Solution, Cost, Time).
Found a solution with cost 1789382
Found a solution with cost 1779902
Found a solution with cost 1759971
. . . . . .
Found a solution with cost 1492989
Found a solution with cost 1492254
Found a solution with cost 1485656
Branch-and-bound timeout!

Solution = [[18, 16], [17, 14], [13, 12, 11], [15],
            [10, 8, 4], [], [7, 2, 6, 1], [5, 9, 3]]
Cost = 1485656
Time = 900.15

?- hcvrp(18, 8, 14400, Solution, Cost, Time).
Found a solution with cost 1789382
. . . . . .
Found a solution with cost 1401968
Found a solution with cost 1395370
Found a solution with cost 1389628
Branch-and-bound timeout!

Solution = [[18, 16], [17, 14], [13, 12, 11], [15], [10, 7],
            [], [8, 4, 1, 6, 2], [5, 9, 3]]
Cost = 1389628
Time = 14398.17
```

```
?- hcvrp(19, 8, 900, Solution, Cost, Time).
Found a solution with cost 2046763
Found a solution with cost 2019977
Found a solution with cost 2002283
. . . . . .
Found a solution with cost 1736210
Found a solution with cost 1723815
Found a solution with cost 1717217
Branch-and-bound timeout!

Solution = [[19, 15], [17, 16], [14, 13, 12], [18],
            [11, 10, 8], [7], [4, 1, 6, 2], [5, 9, 3]]
Cost = 1717217
Time = 900.11

?- hcvrp(19, 8, 14400, Solution, Cost, Time).
Found a solution with cost 2046763
. . . . . .
Found a solution with cost 1703450
Found a solution with cost 1696852
Found a solution with cost 1691110
Branch-and-bound timeout!

Solution = [[19, 15], [17, 16], [14, 13, 12], [18], [11, 10],
            [7], [8, 4, 1, 6, 2], [5, 9, 3]]
Cost = 1691110
Time = 14398.24

?- hcvrp(20, 8, 900, Solution, Cost, Time).
```

```
Found a solution with cost 1909708

Found a solution with cost 1898024

Found a solution with cost 1856292

. . . . . .

Found a solution with cost 1634269

Found a solution with cost 1632256

Found a solution with cost 1628450

Branch-and-bound timeout!


Solution = [[20, 15], [19, 16], [17, 14, 13], [18],

            [12, 11, 8], [7], [10, 3, 9, 1, 5], [4, 6, 2]]

Cost = 1628450

Time = 900.22


?- hcvrp(20, 8, 14400, Solution, Cost, Time).

Found a solution with cost 1909708

. . . . . .

Found a solution with cost 1628450

Branch-and-bound timeout!


Solution = [[20, 15], [19, 16], [17, 14, 13], [18],

            [12, 11, 8], [7], [10, 3, 9, 1, 5], [4, 6, 2]]

Cost = 1628450

Time = 14380.4


?-
```

Παραδοτέο για την άσκηση είναι **ένα πηγαίο αρχείο Prolog**.

1.3 Άσκηση 6

Αντιμετωπίστε τις προηγούμενες ασκήσεις της Β΄ ομάδας (4 και 5) με τη βοήθεια της βιβλιοθήκης C++ για προγραμματισμό με περιορισμούς, τον Naxos Solver (http://www.di.uoa.gr/~pothitos/naxos/),

που έχει σχεδιάσει και αναπτύξει ο υποψήφιος διδάκτωρ του Τμήματός μας **Νίκος Ποθητός** (pothitos@di.uoa.gr).

Παραδοτέο για την άσκηση είναι **ένα αρχείο zip**, που θα περιλαμβάνει όλη τη δουλειά σας για τα δύο προβλήματα, κατάλληλα οργανωμένη σε καταλόγους ανά πρόβλημα. Μην περιλάβετε μέσα στο αρχείο zip εκτελέσιμα ή αντικειμενικά προγράμματα.

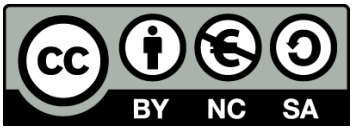
Σημειώματα

Σημείωμα Ιστορικού Εκδόσεων Έργου

Το παρόν έργο αποτελεί την έκδοση 1.0.

Σημείωμα Αναφοράς

Σημείωμα Αδειοδότησης



[1] http://creativecommons.org/licenses/by-nc-sa/4.0/

Ως **Μη Εμπορική** ορίζεται η χρήση:

- που δεν περιλαμβάνει άμεσο ή έμμεσο οικονομικό όφελος από την χρήση του έργου, για το διανομέα του έργου και αδειοδόχο
- που δεν περιλαμβάνει οικονομική συναλλαγή ως προϋπόθεση για τη χρήση ή πρόσβαση στο έργο
- που δεν προσπορίζει στο διανομέα του έργου και αδειοδόχο έμμεσο οικονομικό όφελος (π.χ. διαφημίσεις) από την προβολή του έργου σε διαδικτυακό τόπο

Ο δικαιούχος μπορεί να παρέχει στον αδειοδόχο ξεχωριστή άδεια να χρησιμοποιεί το έργο για εμπορική χρήση, εφόσον αυτό του ζητηθεί.

Διατήρηση Σημειωμάτων

- Οποιαδήποτε αναπαραγωγή ή διασκευή του υλικού θα πρέπει να συμπεριλαμβάνει:
- το Σημείωμα Αναφοράς
- το Σημείωμα Αδειοδότησης
- τη δήλωση Διατήρησης Σημειωμάτων

- το Σημείωμα Χρήσης Έργων Τρίτων (εφόσον υπάρχει)

μαζί με τους συνοδευόμενους υπερσυνδέσμους.

Σημείωμα Χρήσης Έργων Τρίτων

Το Έργο αυτό κάνει χρήση των ακόλουθων έργων:

Εικόνες/Σχήματα/Διαγράμματα/Φωτογραφίες

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στο πλαίσιο του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Αθηνών**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.