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 2009-10

Οι ασκήσεις της ομάδας αυτής πρέπει να αντιμετωπισθούν με τη βοήθεια της τεχνολογίας του **λογικού 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 5

Το πρόβλημα του χρωματισμού γράφου συνίσταται στην εύρεση του ελάχιστου αριθμού χρωμάτων που πρέπει να χρησιμοποιήσουμε για να χρωματίσουμε τους κόμβους δεδομένου γράφου, έτσι ώστε να μην υπάρχει ζευγάρι γειτονικών κόμβων που να έχουν το ίδιο χρώμα. Ο ελάχιστος αριθμός απαιτούμενων χρωμάτων ονομάζεται χρωματικός αριθμός του γράφου.

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 `color_graph/4`, το οποίο όταν καλείται σαν `color_graph(N, D, Col, C)`, αφού δημιουργήσει ένα γράφο `N` κόμβων και πυκνότητας `D`, καλώντας το κατηγόρημα `create_graph/3`, να βρίσκει ένα βέλτιστο χρωματισμό του γράφου, επιστρέφοντας στο `Col` τη λίστα με τα χρώματα (ακέραιοι αριθμοί) των κόμβων και στο `C` τον χρωματικό αριθμό του γράφου. Κάποια παραδείγματα εκτέλεσης είναι τα εξής:[2]

```
?- seed(1), color_graph(10, 80, Col, C).


Col = [1, 1, 2, 3, 3, 1, 4, 5, 6, 6]
```

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

```
C = 6

?- seed(2010), color_graph(15, 60, Col, C).

Col = [1, 1, 2, 3, 2, 2, 4, 5, 4, 3, 6, 3, 6, 7, 5]
C = 7

?- seed(12345), color_graph(20, 70, Col, C).

Col = [1, 1, 2, 3, 4, 2, 5, 4, 3, 6, 5, 5, 3, 7, 6, 7, 2, 6,
       8, 7]
C = 8

?- seed(1000), color_graph(25, 50, Col, C).

Col = [1, 1, 2, 3, 4, 4, 4, 5, 1, 6, 4, 3, 7, 1, 2, 5, 1,
       5, 3, 6, 1, 2, 6, 7, 6]
C = 7

?- seed(9876), color_graph(50, 20, Col, C).

Col = [1, 2, 1, 3, 2, 4, 2, 3, 5, 2, 1, 5, 1, 1, 1, 4, 1, 1,
       2, 5, 5, 2, 2, 4, 3, 4, 3, 2, 3, 2, 1, 3, 4, 5, 1, 3,
       1, 1, 3, 1, 1, 4, 4, 4, 5, 5, 5, 1, 3, 2]
C = 5

?- seed(10), color_graph(40, 34, Col, C).

Col = [1, 2, 3, 1, 4, 5, 3, 5, 4, 2, 2, 4, 6, 2, 6, 6, 2, 4,
       3, 5, 1, 6, 5, 3, 3, 4, 5, 2, 6, 6, 1, 6, 1, 6, 3, 4, 1, 1, 1, 2]
```

```
C = 6
```

Παραδοτέο για την άσκηση είναι ένα αρχείο που περιλαμβάνει τον πηγαίο κώδικα Prolog που θα γράψετε.

1.2 Άσκηση 6

Μία αεροπορική εταιρεία έχει προγραμματίσει να εκτελέσει για μία προκαθορισμένη χρονική περίοδο N πτήσεις, στις οποίες μπορεί να αναφέρεται κανείς με τους κωδικούς 1, 2, 3, …, N. Επιπλέον, μέσω κάποιας μεθόδου που εφάρμοσε, έχει δημιουργήσει M συνδυασμούς πτήσεων (pairings) P_i ($1 \le i \le M$). Δηλαδή, κάθε P_i περιλαμβάνει κάποιες από τις πτήσεις 1, 2, 3, …, N και, επίσης, τα P_i δεν είναι κατ' ανάγκη ξένα μεταξύ τους. Οι συνδυασμοί αυτοί είναι έτσι κατασκευασμένοι ώστε να είναι δυνατόν λόγω κανονισμών, συμβάσεων κλπ. να πραγματοποιηθούν ο καθένας, δηλαδή όλες οι πτήσεις που περιλαμβάνει, με έναν από τους διαθέσιμους κυβερνήτες της εταιρείας. Το ζητούμενο είναι να επιλεγούν κάποιοι συνδυασμοί που καλύπτουν ακριβώς τις πτήσεις της εταιρείας, με σκοπό να ανατεθούν σε συγκεκριμένους κυβερνήτες. Δηλαδή, δεν πρέπει ούτε να μείνει πτήση χωρίς κυβερνήτη, ούτε κάποια πτήση να έχει δύο ή περισσότερους κυβερνήτες. Τέλος, αν ένας συνδυασμός πτήσεων P_i έχει ένα κόστος (έξοδα διανυκτερεύσεων, αποζημιώσεις εκτός έδρας, πληρωμή υπερωριών κλπ.) στην εταιρεία C_i, οι συνδυασμοί που θα επιλεγούν πρέπει να είναι αυτοί που προκαλούν το ελάχιστο συνολικό κόστος. Εφαρμόστε τη μέθοδό σας για εισόδους που προκύπτουν από το `get_flight_data(I,N,P,C)` του προγράμματος http://www.di.uoa.gr/~takis/flight_data.pl, δίνοντας έναν αύξοντα αριθμό εισόδου (1, 2, …) στο I και παίρνοντας το πλήθος των πτήσεων στο N, μία λίστα συνδυασμών πτήσεων στο P και τη λίστα κοστών των συνδυασμών αυτών στο C. Αν θελήσετε να δουλέψετε και με δεδομένα σημαντικού μεγέθους, θα σας χρειαστεί και το αρχείο http://www.di.uoa.gr/~takis/acsdata.zip.

Για την άσκηση αυτή, θα πρέπει να ορίσετε ένα κατηγόρημα `flights/3`, το οποίο όταν καλείται σαν `flights(I, Pairings, Cost)`, θα πρέπει, για το σύνολο δεδομένων με αύξοντα αριθμό `I`, να βρίσκει τη βέλτιστη λύση του προβλήματος (Pairings – ζευγάρια επιλεγέντων συνδυασμών πτήσεων με τα κόστη τους) κόστους Cost.

Μία ενδεικτική εκτέλεση, για όλα τα σύνολα δεδομένων που σας διατέθηκαν, φαίνεται στη συνέχεια:

```
?- member(I, [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12,
              13, 14, 15, 16]),
   write('I = '), writeln(I),
   flights(I, Pairings, Cost),
   write('Pairings = '), writeln(Pairings),
   write('Cost = '), writeln(Cost),
   nl, fail.


I = 1
Pairings = [[1, 2, 3, 7] / 10,
            [5, 8] / 12,
            [4, 9, 10] / 34,
```

```
            [6] / 34]
Cost = 90

I = 2
Pairings = [[1, 2, 5, 8] / 10,
            [3, 6, 9] / 25,
            [4, 7, 10] / 20]
Cost = 55

I = 3
Pairings = [[6, 9] / 32,
            [2, 5, 8] / 10,
            [1, 3, 4, 7, 10] / 28]
Cost = 70

I = 4
Pairings = [[1, 5, 11] / 2,
            [7, 10, 12] / 3,
            [2, 9, 15, 16] / 1,
            [4, 8, 14] / 2,
            [3, 6, 13] / 1]
Cost = 9

I = 5
Pairings = [[3, 4, 5] / 4,
            [1] / 1,
            [6, 8, 11] / 5,
            [10, 12, 13] / 6,
            [2, 7, 9] / 2]
Cost = 18
```

```
I = 6
Pairings = [[1, 3, 5, 7, 9, 11, 13, 15] / 1,
            [17, 19, 21, 23, 25, 27, 29] / 10,
            [16, 18, 20, 22, 24, 26, 28, 30] / 4,
            [2, 4, 6, 8, 10, 12, 14] / 5]
Cost = 20

I = 7
Pairings = [[1, 3, 4, 8, 10] / 2259,
            [2, 7, 11] / 2112,
            [5, 16, 17] / 1158,
            [6, 12, 13] / 2445,
            [9, 14, 15] / 3333]
Cost = 11307

I = 8
Pairings = [[1] / 3384,
            [2, 9] / 2793,
            [3] / 630,
            [4, 5, 16] / 2112,
            [6, 7, 11, 15] / 2109,
            [8, 14] / 1047,
            [10, 12, 13, 17, 18, 19] / 2802]
Cost = 14877

I = 9
Pairings = [[1, 7, 13, 18] / 3333,
            [2, 3, 5, 10] / 3093,
            [4, 6, 11, 12, 14, 15] / 2826,
```

```
            [8, 9, 16, 17, 19] / 1557]
Cost = 10809

I = 10
Pairings = [[1, 2, 3, 9, 14, 24] / 2750,
            [4, 5] / 1622,
            [6, 13, 21] / 1144,
            [7] / 76,
            [8] / 9790,
            [10, 18] / 120,
            [11, 20] / 300,
            [12] / 70,
            [15] / 9510,
            [16, 19, 23] / 352,
            [17] / 720,
            [22] / 9440]
Cost = 35894

I = 11
Pairings = [[1] / 1875,
            [2] / 1800,
            [3, 5, 12, 14, 22, 26, 29] / 8334,
            [4, 8, 16, 23] / 4725,
            [6, 7, 9, 10] / 9900,
            [11, 13, 18, 19, 24] / 28428,
            [15, 17, 20, 21, 25, 27, 28, 30, 31] / 12681]
Cost = 67743

I = 12
Pairings = [[1] / 1156,
```

```
            [2, 12] / 1032,
            [3, 5] / 804,
            [4, 6, 8, 9, 20, 23, 24] / 822,
            [7, 10, 13, 17, 19, 21, 22] / 1874,
            [11, 18, 25] / 1226,
            [14, 15, 16] / 494]
Cost = 7408

I = 13
Pairings = [[1, 13, 15, 16] / 712,
            [2, 10, 17, 18, 19, 20] / 2668,
            [3] / 240,
            [4, 5] / 190,
            [6, 9] / 192,
            [7, 8] / 1032,
            [11, 12, 14, 21, 22, 23] / 1950]
Cost = 6984

I = 14
Pairings = [[1] / 156,
            [2, 6] / 2504,
            [3, 7] / 2396,
            [4, 5] / 2394,
            [8] / 1218,
            [9] / 1208,
            [10] / 688,
            [11, 17, 20] / 2316,
            [12] / 70,
            [13, 16] / 216,
            [14, 15] / 352,
```

```
            [18, 19] / 112,
            [21, 25, 26, 27] / 270,
            [22, 23] / 152,
            [24] / 66]
Cost = 14118



I = 15
Pairings = [[1] / 908,
            [2, 3, 4] / 936,
            [5] / 438,
            [6, 8, 9, 19] / 3178,
            [7] / 406,
            [10] / 548,
            [11, 12, 16, 18] / 3992,
            [13, 14, 15, 17] / 2128]
Cost = 12534


I = 16
Pairings = [[1] / 4797,
            [2, 4, 5, 7, 8, 12, 15] / 3015,
            [3] / 2466,
            [6, 9, 13, 14, 16, 18, 21] / 4236,
            [10, 11, 17, 19, 20, 22] / 2298]
Cost = 16812
```

Παραδοτέο για την άσκηση είναι ένα αρχείο που περιλαμβάνει τον πηγαίο κώδικα Prolog που θα γράψετε.

1.3 Άσκηση 7

Στην άσκηση 4, αντιμετωπίσατε μία εκδοχή του job-shop προβλήματος, και μάλιστα σε δύο παραλλαγές του. Η δεύτερη παραλλαγή ήταν μία επέκταση της πρώτης, ως προς το ότι είχε προστεθεί και περιορισμός σχετικά με το διαθέσιμο προσωπικό. Μελετήστε πάλι προσεκτικά την εκφώνηση της άσκησης 4, ώστε να θυμηθείτε το συγκεκριμένο πρόβλημα.

Στην άσκηση αυτή, πρέπει να αντιμετωπίσετε τη δεύτερη παραλλαγή του προβλήματος της άσκησης 4, σαν πρόβλημα ικανοποίησης περιορισμών, αλλά, ταυτόχρονα, και βελτιστοποίησης. Δηλαδή, σας ζητείται να ορίσετε ένα κατηγόρημα `job_shop_opt/2`, το οποίο όταν καλείται σαν `job_shop_opt(Schedule, Cost)`, να επιστρέφει στο Schedule ένα βέλτιστο πρόγραμμα εκτέλεσης των εργασιών στις μηχανές, το οποίο έχει κόστος `Cost`. Ως κόστος ενός προγράμματος ορίζεται ο απαιτούμενος χρόνος για την εκτέλεση όλων των εργασιών, οπότε ένα πρόγραμμα είναι βέλτιστο όταν ο χρόνος εκτέλεσής του είναι ο ελάχιστος δυνατός. Προσπαθήστε να βρείτε ένα βέλτιστο πρόγραμμα για τα παρακάτω δεδομένα:

```
job(j1, [t11,t12,t13,t14]).
job(j2, [t21,t22,t23,t24]).
job(j3, [t31,t32,t33]).
job(j4, [t41,t42,t43]).
job(j5, [t51]).

task(t11, m1, 3, 3).
task(t12, m2, 2, 3).
task(t13, m1, 2, 3).
task(t14, m2, 3, 1).
task(t21, m1, 2, 2).
task(t22, m2, 3, 2).
task(t23, m1, 3, 1).
task(t24, m1, 4, 2).
task(t31, m1, 3, 1).
task(t32, m2, 1, 3).
task(t33, m2, 4, 3).
task(t41, m1, 1, 1).
task(t42, m1, 3, 2).
task(t43, m1, 2, 3).
task(t51, m2, 3, 2).

machine(m1, 2).
machine(m2, 2).

staff(10).
```

Η σημασία των κατηγορημάτων `job/2`, `task/4`, `machine/2` και `staff/1` είναι ακριβώς η ίδια όπως και στην άσκηση 4. Παρατηρήστε, ότι πλέον δεν χρειάζεται κατηγόρημα `deadline/1`, αφού μας ενδιαφέρει η εύρεση της βέλτιστης λύσης και όχι οποιαδήποτε που ολοκληρώνεται μέχρι κάποια δεδομένη προθεσμία. Ένα παράδειγμα εκτέλεσης για τα παραπάνω δεδομένα είναι το εξής:

```
?- job_shop_opt(Schedule, Cost).
Schedule = [execs(m1,[t(t11,0,3),t(t41,3,4),t(t42,4,7),
                      t(t13,7,9),t(t43,9,11)]),
            execs(m1,[t(t21,0,2),t(t31,2,5),t(t23,5,8),
                      t(t24,8,12)]),
            execs(m2,[t(t51,0,3),t(t12,3,5),t(t32,5,6),
                      t(t14,9,12)]),
            execs(m2,[t(t22,2,5),t(t33,6,10)])]
Cost = 12
```

Τα δεδομένα του προβλήματος μπορείτε να τα βρείτε, στη μορφή που φαίνονται παραπάνω, στο http://www.di.uoa.gr/~takis/jobshop_opt_data.pl. Αν θέλετε να δημιουργήσετε και άλλα

δεδομένα για το πρόβλημα, μπορείτε να χρησιμοποιήσετε το πρόγραμμα `http://www.di.uoa.gr/~takis/rand_js_data.c`.[3]

Αν σε κάποιες περιπτώσεις δεν μπορείτε να βρείτε βέλτιστη λύση στο πρόβλημα, μπορείτε να χρησιμοποιήσετε και τις δυνατότητες που παρέχονται από τις βιβλιοθήκες **`fd`** και **`ic`** της ECL^{i}PS^e, για να βρείτε λύση περίπου βέλτιστη ή/και για να βάλετε κάποιο χρονικό όριο στην εκτέλεση του προγράμματός σας και να επιστρέψετε την καλύτερη λύση που βρέθηκε μέχρι το όριο αυτό. Αν ακολουθήσετε αυτή την προσέγγιση, τεκμηριώστε με σαφήνεια τις επιλογές σας.

Παραδοτέο για την άσκηση είναι ένα αρχείο που περιλαμβάνει τον πηγαίο κώδικα Prolog που θα γράψετε.

1.4 Άσκηση 8 (προαιρετική: bonus 1 μονάδα)

Αντιμετωπίστε τις προηγούμενες ασκήσεις της Β' ομάδας (5, 6 και 7) 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 και για τα τρία προβλήματα (ή για όσα ασχοληθείτε και παραδώσετε τα αποτελέσματά σας) κατάλληλα οργανωμένη σε καταλόγους ανά πρόβλημα. Μην περιλάβετε μέσα στο αρχείο zip εκτελέσιμα ή αντικειμενικά προγράμματα.

[3] Τα δεδομένα που δίνονται στην εκφώνηση έχουν παραχθεί από το πρόγραμμα αυτό, σε μηχάνημα Linux του εργαστηρίου του Τμήματος, καλώντας το με πρώτο όρισμα το 400.

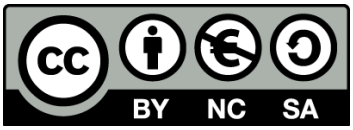
Σημειώματα

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.