ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

Β΄ Ομάδα Ασκήσεων "Λογικού Προγραμματισμού" Ακαδημαϊκού Έτους 2019-20

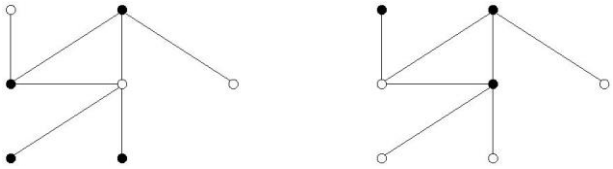
Οι ασκήσεις της ομάδας αυτής πρέπει να αντιμετωπισθούν με τη βοήθεια της τεχνολογίας του **προγραμματισμού με περιορισμούς**. Ένα σύστημα λογικού προγραμματισμού που υποστηρίζει την τεχνολογία αυτή είναι η **ECL^{i}PS^e** (http://www.eclipseclp.org), μέσω της βιβλιοθήκης **`ic`**, η οποία τεκμηριώνεται στα κεφάλαια 3 και 4 του "Constraint Library Manual". 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". Εναλλακτικά, μπορείτε να χρησιμοποιήσετε είτε την βιβλιοθήκη **`gfd`**, που τεκμηριώνεται στο κεφάλαιο 7 του "Constraint Library Manual", και αποτελεί τη διεπαφή της **ECL^{i}PS^e** με τον επιλυτή περιορισμών Gecode, είτε την παλαιότερη βιβλιοθήκη **`fd`**, που τεκμηριώνεται στο κεφάλαιο 2 του "Obsolete Libraries Manual" (http://www.di.uoa.gr/~takis/obsman.pdf). Ενδεχομένως να σας είναι χρήσιμη (για την άσκηση 4) και η βιβλιοθήκη συμβολικών πεδίων της **ECL^{i}PS^e**, η οποία τεκμηριώνεται στο κεφάλαιο 6 του "Constraint Library Manual", αν και για την άσκηση αυτή αρκεί μόνο η βιβλιοθήκη **`fd`**.

Άλλα συστήματα λογικού προγραμματισμού με περιορισμούς που μπορείτε να χρησιμοποιήσετε για τις ασκήσεις αυτής της ομάδας είναι η **GNU Prolog** (http://www.gprolog.org) και η **SWI-Prolog** (http://www.swi-prolog.org), αλλά δεν προτείνονται, διότι οι βιβλιοθήκες περιορισμών τους είναι περιορισμένης λειτουργικότητας και όχι ιδιαίτερα αποδοτικές.

Σε κάθε περίπτωση, στα αρχεία που θα παραδώσετε, θα πρέπει να αναφέρεται στην αρχή τους, σαν σχόλιο, για ποιο σύστημα Prolog έχουν υλοποιηθεί τα αντίστοιχα προγράμματα, εάν αυτό είναι διαφορετικό από την **ECL^{i}PS^e**.

Άσκηση 3

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`, 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(2020), vertexcover(9, 30, C).
Found a solution with cost 6
Found a solution with cost 5
Found no solution with cost 0.0 .. 4.0

C = [1, 5, 6, 8, 9]

?- seed(1000), vertexcover(17, 50, C).
Found a solution with cost 13
```

[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.

```
Found a solution with cost 12
Found no solution with cost 0.0 .. 11.0

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

?- seed(12345), vertexcover(33, 65, C).
Found a solution with cost 29
Found a solution with cost 28
Found no solution with cost 0.0 .. 27.0

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).
Found a solution with cost 96
Found a solution with cost 95
Found a solution with cost 94
Found no solution with cost 0.0 .. 93.0

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).
Found a solution with cost 92
Found a solution with cost 91
Found a solution with cost 90
Found a solution with cost 89
Found a solution with cost 88
Found a solution with cost 87
Found no solution with cost 0.0 .. 86.0

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

?- seed(1999), vertexcover(150, 50, C), length(C, L).
Found a solution with cost 143
Found a solution with cost 142
Found a solution with cost 141
Found a solution with cost 140
Found no solution with cost 0.0 .. 139.0

C = [1, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 16, 17, 18, 19,
20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, .....
L = 140
```

Παραδοτέο για την άσκηση είναι **ένα πηγαίο αρχείο Prolog** με όνομα **`vertexcover.pl`**.

Άσκηση 4

Έστω ότι έχουμε Ν άντρες και Ν γυναίκες και το ζητούμενο είναι να τους παντρέψουμε με τέτοιο τρόπο ώστε να προκύψουν Ν σταθεροί γάμοι. Κάθε άντρας προτιμά σε διαφορετικό βαθμό τις υποψήφιες συζύγους του και κάθε γυναίκα επίσης έχει διαβαθμίσει τους υποψήφιους συζύγους της. Ένας γάμος Α-Γ είναι σταθερός όταν για κάθε γυναίκα Γ1 που ο άντρας Α προτιμά περισσότερο από τη σύζυγό του Γ αυτή προτιμά το σύζυγό της Α1 περισσότερο από τον Α και για κάθε άντρα Α2 που η γυναίκα Γ προτιμά περισσότερο από το σύζυγό της Α αυτός προτιμά τη σύζυγό του Γ2 περισσότερο από τη Γ. Αν δουλέψετε με τη βιβλιοθήκη `fd` της ECL^{i}PS^e, δεδομένα θα βρείτε στο http://www.di.uoa.gr/~takis/stablefd_data.pl (για την `ic`, στο http://www.di.uoa.gr/~takis/stableic_data.pl). Ορίστε ένα κατηγόρημα `stable/1`, το οποίο να επιστρέφει μία λύση του προβλήματος, και μέσω οπισθοδρόμησης όλες τις λύσεις τελικά, σαν μία λίστα από γάμους Α-Γ. Έστω ότι τα δεδομένα (από το http://www.di.uoa.gr/~takis/stablefd_data.pl) για το πρόβλημα είναι τα εξής:

```
men([rick,jack,john,hugh,greg,nick,bill,andy,alan,dick]).

women([helen,tracy,linda,sally,wanda,maria,diana,patty,jenny,lilly]).

prefers(rick,[tracy,linda,jenny,maria,wanda,sally,diana,patty,helen,lilly]).
prefers(jack,[tracy,lilly,patty,sally,diana,linda,jenny,helen,maria,wanda]).
prefers(john,[diana,jenny,wanda,linda,patty,tracy,sally,helen,lilly,maria]).
prefers(hugh,[helen,wanda,diana,maria,sally,patty,linda,lilly,jenny,tracy]).
prefers(greg,[jenny,maria,lilly,patty,sally,linda,tracy,diana,helen,wanda]).
prefers(nick,[sally,linda,diana,maria,jenny,lilly,wanda,helen,patty,tracy]).
prefers(bill,[linda,tracy,diana,patty,lilly,sally,jenny,helen,wanda,maria]).
prefers(andy,[helen,jenny,lilly,maria,sally,patty,wanda,tracy,diana,linda]).
prefers(alan,[patty,sally,maria,jenny,linda,diana,helen,lilly,tracy,wanda]).
prefers(dick,[sally,wanda,helen,maria,lilly,diana,jenny,tracy,patty,linda]).
prefers(helen,[alan,rick,nick,bill,jack,hugh,dick,john,andy,greg]).
prefers(tracy,[andy,john,alan,hugh,bill,rick,greg,dick,jack,nick]).
prefers(linda,[dick,hugh,john,nick,andy,greg,alan,jack,bill,rick]).
prefers(sally,[nick,dick,bill,rick,greg,andy,jack,hugh,alan,john]).
prefers(wanda,[greg,bill,dick,jack,john,nick,alan,andy,rick,hugh]).
prefers(maria,[john,alan,nick,rick,greg,hugh,jack,bill,andy,dick]).
prefers(diana,[greg,hugh,andy,nick,john,bill,alan,dick,jack,rick]).
prefers(patty,[alan,bill,jack,hugh,greg,dick,rick,nick,andy,john]).
prefers(jenny,[greg,dick,jack,bill,alan,john,andy,hugh,rick,nick]).
prefers(lilly,[hugh,jack,dick,nick,andy,alan,greg,rick,bill,john]).
```

Τότε, μία ενδεικτική εκτέλεση του ζητούμενου κατηγορήματος είναι:

```
?- stable(M).

M = [rick-helen, jack-lilly, john-maria, hugh-diana, greg-jenny,
     nick-sally, bill-wanda, andy-tracy, alan-patty, dick-linda]
More --> ;

M = [rick-maria, jack-lilly, john-diana, hugh-helen, greg-jenny,
     nick-sally, bill-linda, andy-tracy, alan-patty, dick-wanda]
More --> ;

M = [rick-maria, jack-lilly, john-linda, hugh-diana, greg-jenny,
     nick-sally, bill-helen, andy-tracy, alan-patty, dick-wanda]
More --> ;

M = [rick-tracy, jack-lilly, john-diana, hugh-helen, greg-jenny,
     nick-sally, bill-linda, andy-maria, alan-patty, dick-wanda]
```

Βρείτε μέχρι ποιο μέγεθος δεδομένων εισόδου το πρόγραμμά σας βρίσκει όλες τις λύσεις για το πρόβλημα σε εύλογο χρόνο. Χρησιμοποιήστε το C πρόγραμμα που θα βρείτε στο http://www.di.uoa.gr/~takis/randstablefddata.c, το οποίο παράγει με τυχαίο τρόπο δεδομένα για το πρόβλημα (τα δεδομένα αυτά είναι κατάλληλα για την βιβλιοθήκη `fd`, αλλά πολύ εύκολα μπορείτε να επεκτείνετε το πρόγραμμα για να καλυφθεί και η `ic`).

Παραδοτέο για την άσκηση είναι **ένα πηγαίο αρχείο Prolog** με όνομα **stable.pl**.

Άσκηση 5

Στην άσκηση 2, 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 2, ώστε να θυμηθείτε το συγκεκριμένο πρόβλημα.

Στην άσκηση αυτή, πρέπει να αντιμετωπίσετε τη δεύτερη παραλλαγή του προβλήματος της άσκησης 2, σαν πρόβλημα ικανοποίησης περιορισμών, αλλά, ταυτόχρονα, και βελτιστοποίησης. Δηλαδή, σας ζητείται να ορίσετε ένα κατηγόρημα `jobshop_opt/6`, το οποίο όταν καλείται σαν `jobshop_opt(Jobs, Staff, Schedule, Cost, Delta, Timeout)`, να επιστρέφει στο `Schedule` ένα βέλτιστο πρόγραμμα εκτέλεσης, με κόστος `Cost`, των εργασιών που ανήκουν στα έργα της λίστας `Jobs`, με διαθέσιμο προσωπικό `Staff`. 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. Η ερμηνεία της παραμέτρου `Delta` είναι ότι σε κάθε νέα επανάληψη της μεθόδου «διακλάδωσε-και-φράξε» αναζητείται λύση με κόστος το πολύ `C-Delta`, όπου `C` είναι το κόστος της λύσης που βρέθηκε στην προηγούμενη επανάληψη, οπότε η λύση που θα βρεθεί δεν είναι εγγυημένα η βέλτιστη, αλλά σίγουρα το κόστος της βέλτιστης βρίσκεται μέσα στο διάστημα `(C-Delta,C]`, όπου `C` είναι το κόστος της τελευταίας λύσης που βρέθηκε. Σε προβλήματα που το κόστος των λύσεων είναι ακέραιος αριθμός, όπως αυτό της συγκεκριμένης άσκησης, θέτοντας το `Delta` ίσο με `1.0`, η μεθοδος «διακλάδωσε-και-φράξε» βρίσκει εγγυημένα τη βέλτιστη λύση. Η παράμετρος `Timeout` είναι ο χρόνος (σε CPU secs) που διατίθεται για την εύρεση λύσης. Αν παρέλθει ο χρόνος αυτός, επιστρέφεται η καλύτερη λύση που έχει βρεθεί μέχρι αυτή τη στιγμή, οπότε πάλι δεν είναι εγγυημένο ότι βρέθηκε η βέλτιστη λύση. Αν δοθεί για `Timeout` το `0`, τότε δεν υπάρχει περιορισμός χρόνου. Έστω ότι τα δεδομένα του προβλήματος είναι αυτά που δίνονται στη συνέχεια. Η σημασία των κατηγορημάτων `job/2`, `task/4`, και `machine/2` είναι ακριβώς η ίδια όπως και στην άσκηση 2. 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, ούτε κατηγόρημα `staff/1`, αφού το διαθέσιμο προσωπικό δίνεται ως όρισμα στο κατηγόρημα που θα υλοποιηθεί.

```
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).
```

Κάποια παραδείγματα εκτέλεσης για τα παραπάνω δεδομένα είναι τα εξής:

```
?- jobshop_opt([j1,j2,j3,j4,j5], 10, Schedule, Cost, 1.0, 0).
Found a solution with cost 18
Found a solution with cost 16
Found a solution with cost 14
Found a solution with cost 13
Found a solution with cost 12

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

?- jobshop_opt([j1,j2,j3,j4,j5], 7, Schedule, Cost, 1.0, 0).
Found a solution with cost 19
Found a solution with cost 18
Found a solution with cost 17
Found a solution with cost 16
Found a solution with cost 14
Found a solution with cost 13
Found no solution with cost 12.0 .. 12.0

Schedule = [execs(m1,[t(t11,0,3),t(t41,3,4),t(t13,6,8),
                      t(t42,8,11),t(t43,11,13)]),
            execs(m1,[t(t21,0,2),t(t31,2,5),t(t23,6,9),
                      t(t24,9,13)]),
            execs(m2,[t(t51,0,3),t(t12,3,5),t(t32,5,6),
```

```
                    t(t33,6,10),t(t14,10,13)]),
            execs(m2,[t(t22,3,6)])]
Cost = 13

?- jobshop_opt([j1,j2,j3,j4], 6, Schedule, Cost, 1.0, 0).
Found a solution with cost 18
Found a solution with cost 16
Found a solution with cost 15
Found a solution with cost 14
Found no solution with cost 12.0 .. 13.0

Schedule = [execs(m1,[t(t11,0,3),t(t41,3,4),t(t23,5,8),
                    t(t13,8,10),t(t24,10,14)]),
            execs(m1,[t(t21,0,2),t(t31,2,5),t(t42,5,8),
                    t(t43,12,14)]),
            execs(m2,[t(t22,2,5),t(t12,5,7),t(t32,7,8),
                    t(t14,10,13)]),
            execs(m2,[t(t33,8,12)])]
Cost = 14

?- jobshop_opt([j1,j2,j3,j4], 5, Schedule, Cost, 1.0, 20).
Found a solution with cost 19
Found a solution with cost 18
Found a solution with cost 17
Found a solution with cost 16
Branch-and-bound timeout while searching for solution
better than 16

Schedule = [execs(m1,[t(t11,0,3),t(t41,3,4),t(t13,5,7),
                    t(t23,9,12),t(t24,12,16)]),
            execs(m1,[t(t21,0,2),t(t31,2,5),t(t42,7,10),
                    t(t43,10,12)]),
            execs(m2,[t(t12,3,5),t(t22,5,8),t(t32,8,9),
                    t(t14,9,12),t(t33,12,16)]),
            execs(m2, [])]
Cost = 16

?- jobshop_opt([j1,j2], 3, Schedule, Cost, 1.0, 0).
Found a solution with cost 20
Found a solution with cost 19
Found no solution with cost 12.0 .. 18.0

Schedule = [execs(m1,[t(t11,0,3),t(t13,5,7),t(t21,7,9),
                    t(t23,12,15),t(t24,15,19)]),
            execs(m1,[]),
            execs(m2,[t(t12,3,5),t(t22,9,12),t(t14,12,15)]),
            execs(m2,[])]
Cost = 19

?- jobshop_opt([j1,j2], 2, Schedule, Cost, 1.0, 0).
Found no solution with cost 12.0 .. 22.0

No

?- jobshop_opt([j1,j4,j5], 3, Schedule, Cost, 1.0, 0).
Found a solution with cost 16
```

```
Found a solution with cost 15
Found no solution with cost 10.0 .. 14.0

Schedule = [execs(m1,[t(t11,0,3),t(t41,3,4),t(t13,8,10),
                      t(t42,10,13),t(t43,13,15)]),
            execs(m1,[]),
            execs(m2,[t(t51,3,6),t(t12,6,8),t(t14,10,13)]),
            execs(m2,[])]
Cost = 15

?- jobshop_opt([j1,j4,j5], 3, Schedule, Cost, 3.0, 0).
Found a solution with cost 16
Found no solution with cost 10.0 .. 13.0

Schedule = [execs(m1,[t(t11,0,3),t(t41,3,4),t(t13,9,11),
                      t(t42,11,14),t(t43,14,16)]),
            execs(m1,[]),
            execs(m2,[t(t12,4,6),t(t51,6,9),t(t14,11,14)]),
            execs(m2,[])]
Cost = 16

?- jobshop_opt([j1,j2,j3,j4,j5], 6, Schedule, Cost, 2.0, 0).
Found a solution with cost 19
Found a solution with cost 17
Found a solution with cost 15
Found no solution with cost 12.0 .. 13.0

Schedule = [execs(m1,[t(t11,0,3),t(t41,3,4),t(t23,5,8),
                      t(t13,8,10),t(t24,11,15)]),
            execs(m1,[t(t21,0,2),t(t31,2,5),t(t42,5,8),
                      t(t43,10,12)]),
            execs(m2,[t(t12,4,6),t(t32,6,7),t(t33,7,11),
                      t(t14,11,14)]),
            execs(m2,[t(t22,2,5),t(t51,12,15)])]
Cost = 15
```

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]

Παραδοτέο για την άσκηση είναι **ένα πηγαίο αρχείο Prolog** με όνομα **`jobshop_opt.pl`**, μέσα στο οποίο **δεν** θα πρέπει να περιέχονται γεγονότα που ορίζουν τα δεδομένα του προβλήματος.

[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.