Εργασία στο μάθημα “Αλγόριθμοι και Πολυπλοκότητα” Χρονοπρογραμματισμός εξετάσεων Πανεπιστημίου V1.0

Γκόγκος Χρήστος
Μεταπτυχιακό Πληροφορικής και Δικτύων (MSc)
Τμήμα Πληροφορικής και Τηλεπικοινωνιών, Πανεπιστημίου Ιωαννίνων

Άρτα, Οκτώβριος 2020

1 Εισαγωγή

Η αποδοτική δημιουργία προγραμμάτων εξετάσεων είναι ένα σημαντικό και επαναλαμβανόμενο πρόβλημα το οποίο καλούνται να αντιμετωπίσουν τα εκπαιδευτικά ιδρύματα σε όλο τον κόσμο. Μια απλοποιημένη μορφή του προβλήματος έχει προταθεί από τους Carter κ.ά. [CLL96] οι οποίοι διέθεσαν δημόσια 13 στιγμιότυπα προβλημάτων που εν συνεχεία χρησιμοποιήθηκαν σε πληθώρα επιστημονικών εργασιών χρονοπρογραμματισμού. Στα πλαίσια της παρούσας εργασίας ζητείται να κατασκευάσετε μια εφαρμογή που θα είναι σε θέση να παράγει λύσεις για τα προβλήματα αυτά.

2 Περιγραφή προβλήματος

Το πρόβλημα αφορά εξετάσεις, σπουδαστές και συνεχόμενες περιόδους σε κάθε μια από τις οποίες μπορούν να διεξαχθούν μια ή περισσότερες εξετάσεις. Κάθε εξέταση διαθέτει μια λίστα από σπουδαστές και κάθε σπουδαστής μπορεί να είναι εγγεγραμμένος σε μια ή περισσότερες εξετάσεις. Η λύση του προβλήματος συνίσταται στην ανάθεση εξετάσεων σε περιόδους έτσι ώστε να μην υπάρχουν **συγκρούσεις**, δηλαδή να μην υπάρχουν σπουδαστές που θα έπρεπε να συμμετάσχουν σε εξετάσεις σε περισσότερα του ενός μαθήματα στην ίδια περίοδο. Καθώς είναι ενδεχόμενο να υπάρχουν πολλά εναλλακτικά προγράμματα που ικανοποιούν τον ανωτέρω περιορισμό, προτιμότερο θεωρείται εκείνο το πρόγραμμα που διαθέτει επαρκή διαστήματα προετοιμασίας ανάμεσα σε διαδοχικές εξετάσεις για όλους τους φοιτητές συνολικά. Ειδικότερα, ορίζονται τιμές ποινής που είναι 16, 8, 4, 2 ή 1 σε κάθε περίπτωση που ένας φοιτητής συμμετέχει σε δύο εξετάσεις που απέχουν 1, 2, 3, 4 ή 5 περιόδους αντίστοιχα. Η συνολική ποινή για όλους τους φοιτητές, διαιρεμένη με το πλήθος των φοιτητών αποτελεί το κόστος της λύσης.

3 Δεδομένα προβλήματος

Τα δεδομένα του προβλήματος που θα χρησιμοποιηθούν στα πειράματα μπορείτε να τα κατεβάσετε από τη σελίδα https://github.com/chgogos/datasets/blob/main/UETT/toronto.zip. Συγκεντρωτικά στοιχεία των προβλημάτων παρουσιάζονται στον Πίνακα 1 όπου εμφανίζεται και το πλήθος των διαθέσιμων περιόδων για κάθε πρόβλημα.

Τα αρχεία δεδομένων (κατάληξη .stu) διαθέτουν για κάθε σπουδαστή μια γραμμή που περιέχει τους αριθμούς των μαθημάτων στα οποία είναι εγγεγραμμένος χωρισμένους μεταξύ τους με κενά. Η πρώτη γραμμή του αρχείου αντιστοιχεί στον πρώτο σπουδαστή, η δεύτερη γραμμή στο δεύτερο σπουδαστή κ.ο.κ. Για παράδειγμα το αρχείο car-f-92.stu περιέχει 18419 σειρές δεδομένων και ξεκινά με τις ακόλουθες σειρές:

0170
0156

0281
0006
0154 0156
0383
0534 0535 0536
...

που σημαίνουν ότι ο φοιτητής 1 έχει εγγραφεί στο μάθημα 0170, ο φοιτητής 2 έχει εγγραφεί στο μάθημα 0156, ο φοιτητής 3 έχει εγγραφεί στο μάθημα 0281, ο φοιτητής 4 έχει εγγραφεί στο μάθημα 0006, ο φοιτητής 5 στα μαθήματα 0154 0156 κ.ο.κ.

Πίνακας 1: Δεδομένα προβλημάτων

Πρόβλημα	Αρχείο Δεδομένων	Εξετάσεις	Φοιτητές	Εγγραφές	Περίοδοι	Πυκνότητα
car-f-92	car-f-92.stu	543	18419	55522	32	0.14
car-s-91	car-s-91.stu	682	16925	56877	35	0.13
ear-f-83	ear-f-83.stu	190	1125	8109	24	0.27
hec-s-92	hec-s-92.stu	81	2823	10632	18	0.42
kfu-s-93	kfu-s-93.stu	461	5349	25113	20	0.06
lse-f-91	lse-f-91.stu	381	2726	10918	18	0.06
pur-s-93	pur-s-93.stu	2419	30029	120681	42	0.03
rye-s-93	rye-s-93.stu	486	11483	45051	23	0.07
sta-f-83	sta-f-83.stu	139	611	5751	13	0.14
tre-s-92	tre-s-92.stu	261	4360	14901	23	0.18
uta-s-92	uta-s-92.stu	622	21266	58979	35	0.13
ute-s-92	ute-s-92.stu	184	2749	11793	10	0.08
yor-f-83	yor-f-83.stu	181	941	6034	21	0.29

3.1 Ενδεικτικές λύσεις προβλημάτων

Στον πίνακα 2 παρουσιάζονται αρχεία με λύσεις των προβλημάτων (κατάληξη .sol) και το κόστος της κάθε λύσης. Τα αρχεία αυτά υπάρχουν διαθέσιμα προς μεταφόρτωση στη διεύθυνση https://github.com/chgogos/datasets/blob/main/UETT/good_solutions1.zip και μπορούν να χρησιμοποιηθούν για επαλήθευση του ορθού υπολογισμού του κόστους της λύσης. Σχετικά με τα περιεχόμενα των αρχείων λύσης παρουσιάζεται ως παράδειγμα το αρχείο car-f-92(3.71).sol που περιέχει 543 εγγραφές (όσες οι εξετάσεις) και ξεκινά με τις ακόλουθες σειρές:

291 0
135 19
382 11
...

που σημαίνουν ότι η εξέταση 291 έχει τοποθετηθεί στην περίοδο 0, η εξέταση 135 έχει τοποθετηθεί στην περίοδο 19, η εξέταση 382 έχει τοποθετηθεί στην περίοδο 11 κ.ο.κ.

4 Ζητούμενα

Να κατασκευάσετε ένα πρόγραμμα που θα ελέγχεται από ένα μενού (ή ένα GUI=Graphical User Interface) με τις ακόλουθες επιλογές:

1. Φόρτωση προβλήματος
2. Φόρτωση λύσης

Πίνακας 2: Λύσεις

Αρχείο λύσης	Κόστος λύσης
car-f-92(3.71).sol	3.71
car-s-91(4.39).sol	4.39
ear-f-83(32.63).sol	32.63
hec-s-92(10.05).sol	10.04
kfu-s-93(12.90).sol	12.90
lse-f-91(9.82).sol	9.82
pur-s-93(4.49).sol	4.49
rye-s-93(7.93).sol	7.93
sta-f-83(157.03).sol	157.03
tre-s-92(7.72).sol	7.72
uta-s-92(3.04).sol	3.04
ute-s-92(24.77).sol	24.77
yor-f-83(34.71).sol	34.71

3. Επίλυση προβλήματος

4. Μαζική επίλυση προβλημάτων

Η λειτουργικότητα που ζητείται για κάθε μια επιλογή περιγράφεται στις ακόλουθες παραγράφους.

4.1 Φόρτωση προβλήματος

Μέσω της συγκεκριμένης επιλογής το πρόγραμμα θα επιτρέπει να φορτωθούν στη μνήμη του υπολογιστή τα δεδομένα ενός προβλήματος που θα επιλέγει ο χρήστης. Στη συνέχεια θα υπολογίζει τα βασικά χαρακτηριστικά του προβλήματος όπως τον αριθμό εξετάσεων, τον αριθμό σπουδαστών, τον αριθμό εγγραφών σπουδαστών και την πυκνότητα συγκρούσεων. Για τον υπολογισμό της πυκνότητας θα πρέπει να κατασκευαστεί ο πίνακας συγκρούσεων. Ο πίνακας συγκρούσεων c είναι ένας δισδιάστατος πίνακας στον οποίο κάθε στοιχείο $c_{ij} = 1$ αν η εξέταση i βρίσκεται σε σύγκρουση με την εξέταση j ενώ ισχύει ότι $c_{ij} = 0$ σε άλλη περίπτωση. Η πυκνότητα συγκρούσεων υπολογίζεται διαιρώντας τον αριθμό των στοιχείων του πίνακα συγκρούσεων που έχουν την τιμή 1 με το συνολικό πλήθος των στοιχείων του πίνακα. Τα αποτελέσματα θα πρέπει να συμπίπτουν με τον πίνακα 1.

4.2 Φόρτωση λύσης

Με δεδομένο ότι με το προηγούμενο ερώτημα έχει φορτωθεί στη μνήμη του προγράμματος ένα πρόβλημα, με την επιλογή φόρτωση λύσης θα φορτώνεται στη μνήμη και ένα αρχείο λύσης, θα εξετάζεται η εγκυρότητα της λύσης που περιέχει και θα εμφανίζεται το κόστος της λύσης εφόσον είναι έγκυρη. Μια λύση είναι έγκυρη (εφικτή) αν δεν περιέχει συγκρούσεις εξετάσεων και έχει τοποθετημένες όλες τις εξετάσεις σε περιόδους.

4.3 Επίλυση προβλήματος

Ζητείται για το πρόβλημα που κάθε φορά θα έχει φορτωθεί (από την επιλογή 1), η δημιουργία λύσεων που θα είναι έγκυρες και θα έχουν το κατά το δυνατόν ελάχιστο κόστος. Ένας τρόπος για να πραγματοποιηθεί η επίλυση του προβλήματος είναι αρχικά η κατασκευή μιας έγκυρης λύσης [LTMG12] και στη συνέχεια η εφαρμογή κινήσεων που θα βελτιώνουν τη λύση. Τέτοιες κινήσεις μπορεί να είναι η μεταφορά μιας εξέτασης σε άλλη περίοδο, η ανταλλαγή περιόδων ανάμεσα σε δύο εξετάσεις, ή ακόμα και η χρήση κινήσεων βασισμένων σε kempe chains [LMR19]. Η επίλυση θα επιχειρείται για χρονικό διάστημα που θα εισάγει ο χρήστης σε δευτερόλεπτα

και μπορεί να γίνει με οποιοδήποτε τρόπο κριθεί σκόπιμος. Όταν λήξει ο χρόνος επίλυσης θα εμφανίζεται το κόστος της λύσης που προέκυψε και θα εξάγεται η λύση σε αρχείο (.sol).

4.4 Μαζική επίλυση προβλημάτων

Ο χρήστης θα δίνει το χρονικό διάστημα επίλυσης προβλήματος σε δευτερόλεπτα και το πρόγραμμα θα επιλύει όλα τα στιγμιότυπα προβλημάτων, ένα προς ένα, εμφανίζοντας όταν λήξει ο χρόνος για κάθε πρόβλημα, το κόστος της λύσης και εξάγοντας κάθε λύση σε αρχείο (.sol).

5 Παραδοτέα εργασίας

Τα παραδοτέα της εργασίας θα είναι τα ακόλουθα:

1. Τεχνική αναφορά για την εργασία, στα πρότυπα σύντομου επιστημονικού άρθρου (5 σελίδες).
2. Github project που θα εμφανίζεται ως Github page και το οποίο θα περιέχει:
   (α') Σύντομη περιγραφή της εφαρμογής
   (β') Τον πλήρη κώδικα της εφαρμογής
   (γ') Οδηγίες εγκατάστασης
   (δ') Οδηγίες εκτέλεσης

6 Παρατηρήσεις

- Η υλοποίηση του κώδικα θα πρέπει να γίνει κατά προτίμηση στη γλώσσα προγραμματισμού C++. Εναλλακτικά, μπορεί να χρησιμοποιηθεί η Java ή η Python ή κάποια άλλη γλώσσα προγραμματισμού.
- Μερικοί αλγόριθμοι που μπορούν να χρησιμοποιηθούν για την επίλυση του προβλήματος περιγράφονται στο https://github.com/clever-algorithms/CleverAlgorithms καθώς και στο βιβλίο [Luk13].
- Επιπλέον πληροφορίες για το πρόβλημα του χρονοπρογραμματισμού εξετάσεων μπορούν να ληφθούν από τις ακόλουθες αναφορές [BP02], [QBM+09].

6.1 Σχετικές ιστοσελίδες

- `http://www.cs.nott.ac.uk/~rxq/data.htm`
- `http://nemertes.lis.upatras.gr/jspui/handle/10889/6721`

Αναφορές

[BP02] Edmund Kieran Burke and Sanja Petrovic. Recent research directions in automated timetabling. *European Journal of Operational Research*, 140(2):266–280, 2002.

[CLL96] Michael W Carter, Gilbert Laporte, and Sau Yan Lee. Examination timetabling: Algorithmic strategies and applications. *Journal of the Operational Research Society*, pages 373–383, 1996.

[LMR19] Nuno Leite, Fernando Melício, and Agostinho C Rosa. A fast simulated annealing algorithm for the examination timetabling problem. *Expert Systems with Applications*, 122:137–151, 2019.

[LTMG12] Rhyd Lewis, J Thompson, C Mumford, and J Gillard. A wide-ranging computational comparison of high-performance graph colouring algorithms. *Computers & Operations Research*, 39(9):1933–1950, 2012.

[Luk13] Sean Luke. *Essentials of Metaheuristics*. Lulu, second edition, 2013. Available for free at http://cs.gmu.edu/∼sean/book/metaheuristics/.

[QBM+09] Rong Qu, Edmund K Burke, Barry McCollum, Liam TG Merlot, and Sau Y Lee. A survey of search methodologies and automated system development for examination timetabling. *Journal of scheduling*, 12(1):55–89, 2009.