Διερεύνηση προσεγγίσεων επίλυσης του προβλήματος 0-1 σακιδίου

Εργασία στο μάθημα “Αλγόριθμοι και Προχωρημένες Δομές Δεδομένων”
Πανεπιστήμιο Ιωαννίνων-ΠΜΣ Πληροφορικής & Τηλεπικοινωνιών
Χειμερινό εξάμηνο ακαδημαϊκού έτους 2019-2010
Γκόγκος Χρήστος

14 Νοεμβρίου 2019

Περίληψη

Το πρόβλημα του 0-1 σακιδίου αποτελεί ένα από το βασικά προβλήματα συνδυαστικής βελτιστοποίησης. Στην παρούσα εργασία ζητείται ο προγραμματισμός μιας εφαρμογής που να επιλύει διάφορα στιγμιότυπα του προβλήματος με έξι διαφορετικούς τρόπους (άπληστα, εξαντλητικά, με διακλάδωση και φραγή, με δυναμικό προγραμματισμό, με ακέραιο προγραμματισμό και με έναν εξειδικευμένο επιλυτή ανοικτού κώδικα). Επιπλέον, ζητείται η συγγραφή τεχνικής αναφοράς που να περιγράφει εν συντομία τις προσεγγίσεις επίλυσης και τα αποτελέσματα. Ο κώδικας της εφαρμογής να αναρτηθεί σε αποθετήριο και να κατασκευαστεί μια ιστοσελίδα που να εμφανίζει τα κύρια αποτελέσματα της εργασίας.

1 Περιγραφή του προβλήματος

Το πρόβλημα 0-1 σακιδίου (0-1 knapsack) αφορά ένα σύνολο από αντικείμενα για τα οποία γνωρίζουμε το βάρος και την αξία κάθε αντικειμένου. Ζητείται η επιλογή ενός υποσυνόλου των αντικειμένων έτσι ώστε το συνολικό βάρος από τα επιλεχθέντα αντικείμενα να μην ξεπερνά μια συγκεκριμένη τιμή βάρους και ταυτόχρονα να επιτυγχάνεται η μεγαλύτερη δυνατή αξία. Το πρόθεμα 0-1 στο όνομα του προβλήματος υποδηλώνει ότι κάθε αντικείμενο μπορεί είτε να επιλεχθεί είτε να μην επιλεχθεί στο σύνολό του και όχι τμηματικά.

2 Δημιουργία στιγμιοτύπων του προβλήματος

Δημιουργήστε στιγμιότυπα προβλημάτων 0-1 σακιδίου χρησιμοποιώντας τον generator του άρθρου [Pis99] και τον κώδικα που βρίσκεται στη διεύθυνση (http://hjemmesider.diku.dk/~pisinger/generator.c). Δημιουργήστε από 5 στιγμιότυπα για κάθε συνδυασμό των ακόλουθων παραμέτρων: n={10,50,100,500}, r={50,100,500,1000} και type={1,2,3,4}, δηλαδή σύνολο $5 \times 4 \times 4 \times 4 = 320$ στιγμιότυπα. Δώστε κατάλληλο όνομα στο αρχείο που παράγεται έτσι ώστε να διακρίνονται τα αρχεία μεταξύ τους και να προκύπτει από το όνομά τους με ποιες αρχικές τιμές δημιουργήθηκαν καθώς και σε ποιο σετ προβλημάτων ανήκουν. Για παράδειγμα ονομάζοντας ένα αρχείο “problem_10_50_1_1_5.txt”, αυτό σημαίνει ότι πρόκειται για αρχείο που δημιουργήθηκε με n=10, r=50, t=1 και είναι το πρώτο από τα συνολικά πέντε αρχεία που δημιουργήθηκαν με τις ίδιες παραμέτρους. Τα δεδομένα που μπορεί να περιέχει το αρχείο αυτό παρουσιάζονται στη συνέχεια.

```
10
     1      42      31
     2      42      42
     3      18       6
     4      18      25
     5      27      14
     6       7      13
     7      17      25
     8      19      41
     9      44      19
    10      34      12
51
```

Στην πρώτη γραμμή είναι ο αριθμός των αντικειμένων n=10. Κάθε επόμενη από τις 10 γραμμές περιέχει τον αριθμό του αντικειμένου, την αξία του αντικειμένου και το βάρος του αντικειμένου. Συνεπώς, το αντικείμενο 1 έχει κέρδος 42 και βάρος 31, το αντικείμενο 2 έχει κέρδος 42 και βάρος 42 και ομοίως για τη συνέχεια. Στην τελευταία γραμμή είναι η συνολική χωρητικότητα του σακιδίου, δηλαδή στο συγκεκριμένο παράδειγμα 51.

3 Επίλυση του προβλήματος

Η επίλυση του προβλήματος να γίνει με τους ακόλουθους έξι διαφορετικούς τρόπους.

3.1 Άπληστη μέθοδος

Στην άπληστη μέθοδο (greedy approach) θα υπολογίζεται για κάθε αντικείμενο ο λόγος αξία προς βάρος και θα δίνεται προτεραιότητα στην εισαγωγή των αντικειμένων με τις μεγαλύτερες τιμές και μέχρι να μην μπορεί να προστεθεί άλλο αντικείμενο στο σακίδιο.

3.2 Εξαντλητική απαρίθμηση συνδυασμών

Στην εξαντλητική απαρίθμηση (brute force - full enumeration) θα πρέπει να δοκιμαστούν όλοι οι πιθανοί συνδυασμοί τοποθέτησης αντικειμένων στο σακίδιο (ανά ένα αντικείμενο, ανά δύο αντικείμενα, ανά τρία αντικείμενα κ.ο.κ.) και να επιλέγεται η πλέον συμφέρουσα. Η συγκεκριμένη προσέγγιση δεν μπορεί να λειτουργήσει για μεγάλα στιγμιότυπα προβλημάτων οπότε εάν το πρόβλημα είναι επαρκώς μεγάλο θα πρέπει να επιστρέφει το καλύτερο αποτέλεσμα που εντοπίζει με μέγιστο χρονικό περιθώριο εκτέλεσης τα 10 δευτερόλεπτα. Ο περιορισμός του χρόνου εκτέλεσης των 10 δευτερολέπτων να ισχύει και για τις άλλες προσεγγίσεις.

3.3 Διακλάδωση και φραγή

Η μέθοδος διακλάδωσης και φραγής (branch and bound) αποτελεί βελτίωση της μεθόδου πλήρους απαρίθμησης. Αρχικά τα αντικείμενα διατάσσονται σε φθίνουσα σειρά αξίας προς βάρος. Στη συνέχεια δημιουργείται ένα δυαδικό δένδρο μερικών λύσεων. Σε κάθε κόμβο του δένδρου λαμβάνεται η απόφαση επιλογής ενός αντικειμένου η οποία οδηγεί στο αριστερό υποδένδρο ενώ η μη επιλογή του οδηγεί στο δεξιό υποδένδρο. Επιπλέον σε κάθε κόμβο καταγράφεται το συνολικό βάρος και η συνολική αξία των αντικειμένων του. Η βασική ιδέα είναι ότι κάποιοι κλάδοι του δένδρου δεν έχει νόημα να δημιουργηθούν καθώς δεν πρόκειται να οδηγήσουν σε καλύτερη λύση από κάποια που έχει ήδη βρεθεί. Αυτό γίνεται με την καταγραφή της καλύτερης τιμής που έχει εντοπιστεί ανά πάσα στιγμή και τον υπολογισμό ενός φράγματος καλύτερης περίπτωσης (άνω φράγμα) για κόμβους που πρόκειται να αναπτυχθούν προσθέτοντας επιπλέον αντικείμενα στα ήδη υπάρχοντα. Ένας απλός τρόπος υπολογισμού του άνω φράγματος είναι να προστεθεί στην τρέχουσα αξία ενός κόμβου (δηλαδή στη συνολική αξία των αντικειμένων του κόμβου), το γινόμενο της χωρητικότητας του σακιδίου που απομένει ελεύθερο με τον καλύτερο λόγο αξίας προς βάρος των αντικειμένων που απομένουν προς επιλογή. Αν η τιμή του άνω φράγματος είναι μικρότερη από την καλύτερη τιμή που έχουμε καταγράψει τότε δεν έχει νόημα να αναπτύξουμε περαιτέρω τον κόμβο, οπότε γίνεται κλάδεμα του δένδρου. Λεπτομέρειες για την υλοποίηση μπορείτε να βρείτε στο [Tri19a] και στο [Tri19b].

3.4 Δυναμικός προγραμματισμός

Για την προσέγγιση του δυναμικού προγραμματισμού (dynamic programming) να υλοποιηθεί κώδικας που χρησιμοποιώντας αναδρομή, επίλυση μικρότερων προβλημάτων και καταγραφή ενδιάμεσων αποτελεσμάτων σε έναν πίνακα να οδηγεί σε λύση του προβλήματος. Περισσότερες πληροφορίες για το δυναμικό προγραμματισμό και την εφαρμογή του στο πρόβλημα του 0-1 σακιδίου μπορείτε να βρείτε στο [God04] αλλά και σε πολλές άλλες πηγές στο διαδίκτυο.

3.5 Ακέραιος προγραμματισμός

Ο ακέραιος προγραμματισμός (Integer Programming) αποτελεί έναν γενικό τρόπο επίλυσης προβλημάτων που είναι δυνατόν να μοντελοποιηθούν με συγκεκριμένο τρόπο. Στηρίζεται στον αλγόριθμο επαναληπτικής

βελτίωσης simplex [Wol19] και είναι σε θέση να εντοπίζει λύσεις ακόμα και εάν υπάρχουν περιορισμοί ακεραιότητας για τις τιμές των μεταβλητών απόφασης. Αναλυτικότερα στοιχεία για την εφαρμογή του ακέραιου προγραμματισμού σε προβλήματα όπως το πρόβλημα του 0-1 σακιδίου βρίσκονται στο [CTS95]. Χρησιμοποιήστε τον επιλυτή προβλημάτων ακέραιου προγραμματισμού GLPK [GNU19] μέσω του λογισμικού OR-Tools [PF19b].

3.6 Εξειδικευμένος επιλυτής

Χρησιμοποιήστε τον εξειδικευμένο επιλυτή που διαθέτει το λογισμικό OR-Tools για το πρόβλημα του 0-1 σακιδίου [PF19c].

4 Πειράματα

Και οι έξι προσεγγίσεις επίλυσης να δοκιμαστούν σε όλα τα προβλήματα με μέγιστο επιτρεπτό χρόνο επίλυσης για κάθε στιγμιότυπο προβλήματος τα 10 δευτερόλεπτα. Οι τιμές που θα επιστρέφει ο κάθε επιλυτής για κάθε πρόβλημα (συνολικό κέρδος, συνολικό βάρος, σύνολο επιλεγέντων αντικειμένων) καθώς και ο χρόνος που απαιτήθηκε, να καταγραφούν σε κατάλληλα αρχεία και να παρουσιαστούν σε πίνακες και γραφήματα.

5 Τεχνική αναφορά

Στα πρότυπα επιστημονικής τεχνικής αναφοράς να συνταχθεί έγγραφο (κατά προτίμηση σε LaTeX) που να καταγράφει το πρόβλημα, τις προσεγγίσεις επίλυσης, τα πειράματα που διενεργήθηκαν, τα αποτελέσματα και τα συμπεράσματα.

6 Ανέβασμα κώδικα και ιστοσελίδα

Ο κώδικας να ανέβει σε αποθετήριο και να είναι προσβάσιμος από τους καθηγητές του μαθήματος (συνίσταται η χρήση του https://github.com/), ενώ θα πρέπει να υπάρχουν οδηγίες εγκατάστασης και εκτέλεσης της εφαρμογής (README.md). Επιπλέον, να δημιουργηθεί μια απλή ιστοσελίδα που επιγραμματικά να αναφέρεται στα αποτελέσματα της εργασίας. Για τη δημιουργία της ιστοσελίδας μπορεί να χρησιμοποιηθεί η δυνατότητα που δίνει το github για ελεύθερη δημιουργία ιστοσελίδων της μορφής username.github.io μέσω του https://pages.github.com/.

7 Σύνοψη υποχρεώσεων εργασίας

Οι υποχρεώσεις της εργασίας παρουσιάζονται στην ακόλουθη λίστα:

1. Δημιουργία 320 στιγμιοτύπων προβλημάτων 0-1 σακιδίου χρησιμοποιώντας τη γεννήτρια προβλημάτων του άρθρου [Pis99].
2. Επίλυση των προβλημάτων με την άπληστη μέθοδο.
3. Επίλυση των προβλημάτων με εξαντλητική εξέταση όλων των πιθανών συνδυασμών αντικειμένων.
4. Επίλυση των προβλημάτων με διακλάδωση και φραγή.
5. Επίλυση των προβλημάτων με δυναμικό προγραμματισμό.
6. Επίλυση των προβλημάτων με ακέραιο προγραμματισμό [GNU19], [PF19a].
7. Επίλυση των προβλημάτων με τον εξειδικευμένο επιλυτή ανοικτού κώδικα του [PF19a].
8. Συγγραφή τεχνικής αναφοράς με παρουσίαση αποτελεσμάτων σε πίνακες και γραφικά.
9. Ανάρτηση του κώδικα στο github.
10. Δημιουργία ιστοσελίδας παρουσίασης αποτελεσμάτων εργασίας.

Η εργασία θα πρέπει να έχει ολοκληρωθεί το αργότερο μέχρι στις **31/12/2019**.

Αναφορές

[CTS95] Gérard Cornuéjol, Michael A. Trick, and Matthew J. Saltzman. A Tutorial on Integer Programming. http://www.math.clemson.edu/~mjs/courses/mthsc.440/integer.pdf, 1995. [Online; accessed 13-11-2019].

[GNU19] GNU. GLPK: (GNU Linear Programming Kit). https://www.gnu.org/software/glpk/, 2019. [Online; accessed 13-11-2019].

[God04] Steve Goddard. Dynamic programming 0-1 Knapsack problem (CSCE 310J). http://cse.unl.edu/~goddard/Courses/CSCE310J/Lectures/Lecture8-DynamicProgramming.pdf, 2004. [Online; accessed 13-11-2019].

[PF19a] Laurent Perron and Vincent Furnon. Google OR-Tools. https://developers.google.com/optimization/, 2019. [Online; accessed 13-11-2019].

[PF19b] Laurent Perron and Vincent Furnon. Google OR-Tools: Mixed-Integer Programming. https://developers.google.com/optimization/mip/integer_opt, 2019. [Online; accessed 13-11-2019].

[PF19c] Laurent Perron and Vincent Furnon. Google OR-Tools: The Knapsack Problem. https://developers.google.com/optimization/bin/knapsack, 2019. [Online; accessed 13-11-2019].

[Pis99] David Pisinger. Core problems in knapsack algorithms. *Operations Research*, 47(4):570–575, 1999.

[Tri19a] Utkarsh Trivedi. 0/1 Knapsack using Branch and Bound. https://www.geeksforgeeks.org/0-1-knapsack-using-branch-and-bound/, 2019. [Online; accessed 13-11-2019].

[Tri19b] Utkarsh Trivedi. Implementation of 0/1 Knapsack using Branch and Bound. https://www.geeksforgeeks.org/implementation-of-0-1-knapsack-using-branch-and-bound/, 2019. [Online; accessed 13-11-2019].

[Wol19] Wolfram Web Resource. Simplex Method. http://mathworld.wolfram.com/SimplexMethod.html, 2019. [Online; accessed 13-11-2019].