Σύγκριση επιλυτών στο πρόβλημα του Knapsack.

Νασιώτης Κωνσταντίνος

Δεκέμβριος 2019

Περίληψη

Το παρόν έγγραφο αποτελεί αναφορά στην εργασία που διεκπεραιώθηκε στα πλαίσια του μαθήματος **Αλγόριθμοι και Πολυπλοκότητα** του τμήματος Πληροφορικής & Τηλεπικοινωνιών του Πανεπιστημίου Ιωαννίνων. Στόχος της είναι η περιγραφή του προβλήματος του **Knapsack** καθώς και οι μέθοδοι δυναμικού προγραμματισμού που εφαρμόστηκαν για την επίλυσή του. Δοκιμάστηκαν και συγκρίθηκαν οι επιδόσεις αλγορίθμων της Google μέσω της βιβλιοθήκης που διαθέτει για προβλήματα βελτιστοποίησης, **Google OR Tools** αλλά και μία προσωπική υλοποίηση.

Περιεχόμενα

1 Εκφώνηση εργασίας

Το πρόβλημα 0-1 knapsack (0-1 σακιδίου) 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 έχει επιτευχθεί 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.

1. Δημιουργήστε στιγμιότυπα προβλημάτων 0-1 knapsack χρησιμοποιώντας τον generator του άρθρου [1] 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 $5 * 4 * 4 * 4 = 320$ στιγμιότυπα.

2. Χρησιμοποιώντας δυναμικό προγραμματισμό αναπτύξτε (κατά προτίμηση σε C++) έναν επιλυτή για το πρόβλημα. Καταγράψτε το αποτέλεσμα και το χρόνο που χρειάστηκε για να επιλυθεί το κάθε πρόβλημα με τον επιλυτή σας.

3. Χρησιμοποιώντας το λογισμικό Google OR-Tools και τον εξειδικευμένο επιλυτή που διαθέτει για προβλήματα knapsack επιλύστε τα στιγμιότυπα προβλημάτων που δημιουργήσατε. Καταγράψτε το αποτέλεσμα και το χρόνο που χρειάστηκε για να επιλυθεί το κάθε πρόβλημα.

Ο κώδικας με οδηγίες εκτέλεσης, μοναδιαίους ελέγχους (unit tests), παραδείγματα εκτέλεσης και μια σύντομη τεχνική αναφορά θα πρέπει να ανέβει στο GitHub ή σε κάποιο άλλο δημόσια διαθέσιμο αποθετήριο.

2 Θεωρητικό υπόβαθρο

2.1 Το πρόβλημα του Knapsack

Για δεδομένο πλήθος αντικειμένων μεγέθους V και αξίας W, το πρόβλημα του Knapsack ή Σακιδίου είναι να προσδιοριστεί ένα σύνολο από τα παραπάνω αντικείμενα που το συνολικό του μέγεθος να είναι μικρότερο από ένα προδιαγεγραμμένο όριο ενώ ταυτόχρονα η συνολική του αξία να γίνει όσο το δυνατόν

μεγαλύτερη[2]. Με λίγα λόγια ο στόχος είναι να μεγιστοποιήσουμε το κέρδος τοποθέτησης όπως στο 1 και να να ικανοποιείται ο περιορισμός 2.

$$max \sum_{i=1}^{n} w_i x_i \tag{1}$$

$$\sum_{i=1}^{n} v_i x_i \leq V, x_i \in \{0, 1\} \tag{2}$$

2.2 Δυναμικός προγραμματισμός

Σύμφωνα με τον ορισμό, **δυναμικός προγραμματισμός** είναι η μαθηματική θεωρία των πολυσταδιακών αποφάσεων, που αναφέρεται στη βελτιστοποίηση της λειτουργίας τους, βάσει κάποιου επιλεγμένου κριτηρίου, γνωστού ως συνάρτηση κόστους ή αντικειμενικής συνάρτησης.[3].

Με λίγα λόγια, η βασική ιδέα του Δυναμικού Προγραμματισμού είναι ότι μπορούμε να χωρίσουμε κατάλληλα το πρόβλημα που αντιμετωπίζουμε σε τόσα υποπροβλήματα όσες είναι και οι άγνωστες μεταβλητές του και να προσδιορίζουμε κάθε φορά την τιμήμιας μόνο μεταβλητής. Έτσι αντί να έχουμε ένα πρόβλημα με τρείς, παραδείγματος χάρη μεταβλητές, σχηματίζουμε τρία αλληλοσυνδεόμενα υπο-προβλήματα με μία μεταβλητή στο καθένα[3].

3 Η υλοποίηση

3.1 Βιβλιοθήκες και εργαλεία

Οι βιβλιοθήκες που χρησιμοποιήθηκαν είναι η STD της γλώσσας προγραμματισμού C++, η OR Tools της Google και επίσης για την προσωπική μου υλοποίηση δανείστηκα την υλοποίηση δυναμικού προγραμματισμού που βρίσκεται σε μία ιστοσελίδα με tutorials σε διάφορες γλώσσες προγραμματισμού, την Geeks for Geeks. Για τους ελέγχους μονάδων χρησιμοποιήθηκε η ανοιχτού κώδικα βιβλιοθήκη Catch2. Τα προβλήματα προς επίλυση παράχθηκαν με το εργαλείο generator.c [6].

3.2 Περιγραφή μεθόδων

Αρχικά χρειάστηκαν να εγκαταστηθούν οι βιβλιοθήκες. Έπειτα χρειάστηκε να γραφεί μία μέθοδος ανάγνωσης και των 320 στιγμιότυπων. Η υλοποίησή μου

κατασκευάζει ένα διάνυσμα που περιέχει τα ονόματα των 320 *.*txt* αρχείων που παρέχονται από την εκφώνηση. Ο αλγόριθμος για κάθε θέση του διανύσματος ονομάτων, ανοίγει το αντίστοιχο αρχείο, καλεί μία μέθοδο που φορτώνει τα δεδομένα σε διανύσματα ακολουθώντας το προκαθορισμένο format του αρχείου, ξεκινά τη χρονομέτρηση, καλεί τον προσωπικό επιλυτή, αρχικοποιεί το χρονόμετρο και καλεί τον επιλυτή του OR Tools. Τα αποτελέσματα αποθηκεύονται σε ένα αρχείο με όνομα **results.csv**.

3.3 Δοκιμές μονάδων

Σχεδόν κάθε μέθοδος επικυρώθηκε για την ορθή λειτουργία της μέσω της Catch2 βιβλιοθήκης και, πιο συγκεκριμένα, μέσω της συνάρτησης **TEST_CASE**.

4 Πειραματική διαδικασία

Για να συγκρίνουμε τα αποτελέσματα, θα πάρουμε δεδομένα από το .csv αρχείο που δημιουργήθηκε από την υλοποίησή μου.

4.1 Υλικό

Ο αλγόριθμος έτρεξε σε υπολογιστή με τα εξής χαρακτηριστικά:

OS	CPU	MEMORY
WINDOWS 10 64-bit 1909	Intel core i7 5820k @ 4.4GHz	16GB DDR4 RAM

4.2 Αποτελέσματα

Ο μέσος χρόνος που χρειάστηκαν τα παραπάνω προβλήματα για να επιλυθούν φαίνεται στον παρακάτω πίνακα:

	Προσωπική Υλοποίηση	Google OR Tools
Χρόνος	$\simeq 39.2ms$	$\simeq 572.9ms$

5 Συμπέρασμα

Όπως μπορούμε να δούμε στον πίνακα της προηγούμενης ενότητας, τα προβλήματα που επιλύθηκαν με τον προσαρμοσμένο προσωπικό επιλυτή χρειάστηκαν πολύ λιγότερο χρόνο απ'ότι με τον επιλυτή του Google OR Tools. Συνεπώς, είτε θα δοκίμαζα άλλων επιλυτή της συγκεκριμένης εργαλειοθήκης (λ.χ. KNAPSACK_MULTIDIMENSION_BRANCH_AND_BOUND_SOLVER) είτε την προσωπική μου υλοποίηση, ειδικά αν το πρόβλημα είναι απλό (μίας διάστασης).

Αναφορές

[1] *Google OR Tools* - https://developers.google.com/optimization

[2] *Πρόβλημα Σακιδίου* - http://users.ntua.gr/nmand/Knapsack.html

[3] *ΕΦΑΡΜΟΓΕΣ ΔΥΝΑΜΙΚΟΥ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ*, Ουζούνης Παναγιώτης, Τ.Ε.Ι. ΚΑΒΑΛΑΣ, 2008.

[4] *0-1 Knapsack problem, Geeks for Geeks* - https://www.geeksforgeeks.org/0-1-knapsack-problem-dp-10/

[5] *Catch2* - https://github.com/catchorg/Catch2

[6] *David Pisinger GENERATOR.c* - http://hjemmesider.diku.dk/ pisinger/generator.c

[7] Pisinger, David. Core problems in knapsack algorithms. Operations Research 47.4, pp. 570-575, 1999.