Εικόνα που περιέχει πίνακας, εσωτερικό

Περιγραφή που δημιουργήθηκε αυτόματα

**Διερεύνηση προσεγγίσεων επίλυσης του προβλήματος σακιδίου 0-1 (Knapsack 0-1)**

Ιωάννης Παναγιώτης Μποτίλιας

ΑΜ:62

**Περιγραφή του προβλήματος Knapsack 0-1**

Το πρόβλημα 0-1 σακιδίου (Knapsack 0-1) 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 σύνολο του και όχι τμηματικά

**Δημιουργία στιγμιότυπων για το πρόβλημα Knapsack 0-1**

Για τη δημιουργία προβλημάτων χρησιμοποιήθηκε ο κώδικας της γεννήτριας του άρθρου του David Pisinger:

<http://hjemmesider.diku.dk/~pisinger/generator.c>.

Δημιουργήθηκαν 320 διαφορετικά στιγμιότυπα για το πρόβλημα με διαφορετικές τιμές για τις παραμέτρους που παίρνει ως είσοδο ο κώδικας generator.c.

Οι παράμετροι αυτοί διαφοροποιούν ως εξής τα στιγμιότυπα:

* n: ο αριθμός αντικειμένων,
* r: εύρος τιμών βάρους / αξίας
* t: τύπος συσχέτισης βάρους – αξίας (1=άσχετα, 2=χαμηλή σχέση, 3=υψηλή σχέση, 4=υποσύνολο)
* i: αριθμός στιγμιότυπου
* S: σύνολο στιγμιότυπων

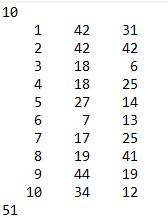
Δημιουργήθηκαν 5 στιγμιότυπα για κάθε ένα πρόβλημα με τις εξής διαφορετικές τιμές:

* n={10,50,100,500}
* r={50,100,500,1000}
* t={1,2,3,4}

Χρησιμοποιήθηκαν τα στιγμιότυπα από το αποθετήριο του κ. Γκόγκου στο GitHub:

<https://github.com/chgogos/uoi_algorithms_and_complexity/tree/master/knapsack_prj>

Τα στιγμιότυπα τα οποία δημιουργούνται είναι της εξής μορφής:



Στην πρώτη γραμμή αναγράφεται ο αριθμός των αντικειμένων(n=10). Κάθε επόμενη γραμμή περιέχει το id, το profit και το weight ενός αντικειμένου(πχ Για την πρώτη γραμμή id = 1, profit = 42 και weight = 31) Τέλος, ο αριθμός της τελευταίας γραμμής είναι η συνολική χωρητικότητα του σακιδίου(W = 51)

**Επίλυση προβλημάτων και Πειράματα**

Για την επίλυση ακολουθήθηκαν έξι διαφορετικές προσεγγίσεις:

* Άπληστη μέθοδος ( Greedy )
* Εξαντλητική απαρίθμηση συνδυασμών ( Brute Force )
* Διακλάδωση και φραγή ( Branch & Bound )
* Δυναμικός προγραμματισμός ( Dynamic Programming )
* Δυναμικός προγραμματισμός με χρήση εξειδικευμένου επιλυτή(OR tools)
* Ακέραιος προγραμματισμός με χρήση εξειδικευμένου επιλυτή(OR tools)

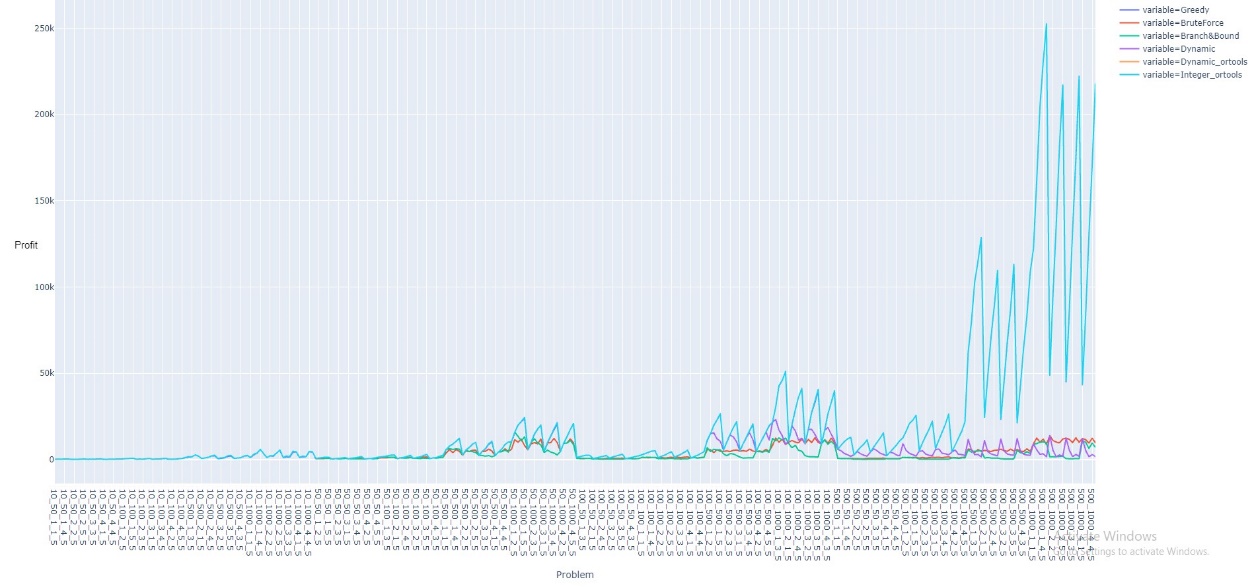
Όλα τα στιγμιότυπα του προβλήματος λύθηκαν και με τις έξι προσεγγίσεις και τα αποτελέσματα (συνολικό κέρδος, συνολικό βάρος, επιλεγμένα αντικείμενα και χρόνος επίλυσης) καταγράφηκαν σε αρχεία csv. Τέλος, τέθηκε χρονικός περιορισμός σε κάποιες από τις προσεγγίσεις για την εύρεση λύσης τα δέκα (10) δευτερόλεπτα. Όταν ξεπερνάει το όριο αυτό ο αλγόριθμος σταματάει να τρέχει και δίνει την μερική λύση που έχει βρει μέχρι στιγμής.

**Παρουσίαση αποτελεσμάτων**

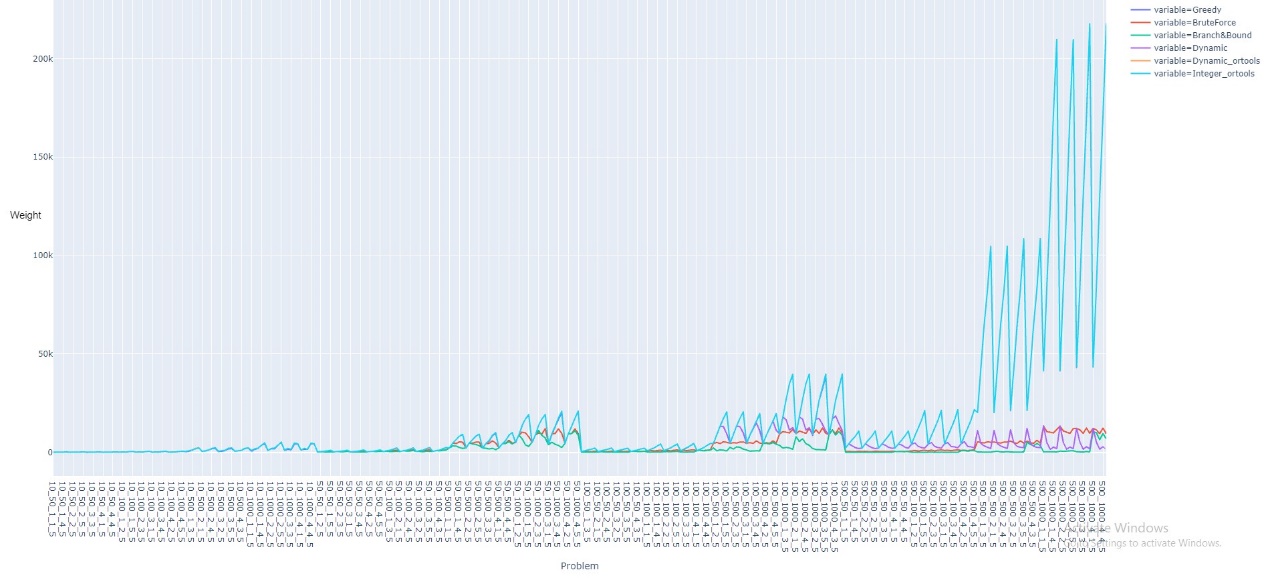
Παρακάτω δίνονται τα αποτελέσματα για κάθε επιλυτή για όλα τα στιγμιότυπα του προβλήματος σακιδίου 0-1 αφού πρώτα έγινε μια ομαδοποίηση σύμφωνα με:

* Πρόβλημα – Συνολικό κέρδος κάθε αλγορίθμου
* Πρόβλημα – Συνολικό βάρος κάθε αλγορίθμου
* Πρόβλημα – Χρόνος επίλυσης κάθε αλγορίθμου

Γράφημα συνολικού κέρδους



Γράφημα συνολικού βάρους



Γράφημα χρόνου επίλυσης



\*\* Για καλύτερη απεικόνιση των γραφημάτων (live) μπορείτε να τρέξετε το αρχείο plot.py που βρίσκεται στο αποθετήριο μου στο GitHub στο παρακάτω link

<https://github.com/johnboti/0_1_knapsack/tree/master/plots>

**Συμπεράσματα**

Από τα αποτελέσματα φαίνονται οι διαφορές της κάθε προσέγγισης για το πρόβλημα.

Greedy:

* Θετικό: Μικρός χρόνος επίλυσης
* Αρνητικό: Μικρό κέρδος και βάρος

Brute Force:

* Θετικό: Βέλτιστη λύση
* Αρνητικό: Μεγάλος χρόνος επίλυσης(Για μεγάλα προβλήματα)

Branch & Bound:

* Θετικό: Βέλτιστη λύση
* Αρνητικό: Μεγάλος χρόνος επίλυσης(Για μεγάλα προβλήματα)

Dynamic Programming

* Θετικό: Βέλτιστη λύση
* Αρνητικό: χρόνος επίλυσης(10 sec από τη μέση των προβλημάτων και μετά)

Dynamic OR-Tools:

* Θετικό: Βέλτιστη λύση
* Αρνητικό: χρόνος επίλυσης( Για τα πολύ μεγάλα προβλήματα)

Integer OR-Tools:

* Θετικό:
* βέλτιστος επιλυτής για μικρό αλλά και μεγάλο αριθμό αντικειμένων.
* μικρός χρόνος επίλυσης
* Αρνητικό: χρόνος επίλυσης(Για τα μικρά προβλήματα της τάξης των ms)

Σύμφωνα με τα παραπάνω καταλήγουμε πως ο ακέραιος προγραμματισμός με χρήση εξειδικευμένου επιλυτή (OR tools) είναι η βέλτιστη προσέγγιση για την επίλυση του προβλήματος σακιδίου 0-1.

.