Τεχνική αναφορά: επίλυση προβλημάτων knapsack 0-1 με χρήση 2 επιλυτών και σύγκριση χρόνων εκτέλεσης

Στραβοράβδης Σπύρος

Ιανουάριος 2020

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Περίληψη

Οι ακόλουθες σημειώσεις είναι περιγραφή της εργασίας του μαθήματος *Αλγόριθμοι και Πολυπλοκότητα*. Περιγράφουν την επίλυση προβλημάτων knapsack και γίνεται σύγκριση των χρόνων μεταξύ διαφορετικών επιλυτών.

Δημιουργήθηκε και δοκιμάστηκε σε Debian 10 Buster (CPython 3.7, Pypy 7.0).

Το σύστημα που χρησιμοποιήθηκε για χρονομέτρηση είναι ένα laptop με επεξεργαστή Intel Core i3 4030U, 8GB μνήμης RAM και SSD 1TB.

1 Γενικά στοιχεία

1.1 Το πρόβλημα του knapsack

Για δεδομένο πλήθος αντικειμένων μεγέθους V και αξίας W 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. [1]

Πιο απλά, αν έχουμε ένα σακίδιο με συγκεκριμένη χωρητικότητα, θα πρέπει να αποφασίσουμε τους βέλτιστους συνδυασμούς αντικειμένων που θα βάλουμε σε αυτό ώστε η αξία τους να είναι μέγιστη.

1.2 Τεχνικές λεπτομέρειες

Ο κώδικας επιλύει 320 προβλήματα (παραχθέντα με αυτόματη γεννήτρια [2]), με διαφορετικό αριθμό στοιχείων, βάρη, αξίες και μέγεθος "σακιδίου", τα οποία βρίσκονται στον φάκελο problems.

Εξετάζονται οι ακόλουθες περιπτώσεις επιλυτών:

- Σε CPython (η συνηθισμένη υλοποίηση της Python) ένας απλός επιλυτής με χρήση δυναμικού προγραμματισμού και ο επιλυτής που παρέχεται από το OR-Tools [3]

- Σε Pypy (εναλλακτική, πιο γρήγορη υλοποίηση της Python λόγω χρήσης JIT) ο ίδιος απλός επιλυτής

2 Υλοποίηση με CPython

Ο κώδικας έχει ελεγχτεί μόνο με Python3 (3.7), δεν είναι γνωστό αν τρέχει σε Python2.

2.1 Main.py

Η συνάρτηση `knapSack` περιέχει την λογική του απλού επιλυτή, ο κώδικας της οποίας προέρχεται από την ιστοσελίδα GeeksForGeeks [4], σε περιγραφή του σχετικού προβλήματος.

Η συνάρτηση `mainSolver` είναι η κύρια συνάρτηση η οποία εκτελεί τους δύο επιλυτές, εξάγει τα δεδομένα με τις λύσεις του κάθε προβλήματος στο αρχείο ResultsCPython.csv μαζί με τους χρόνους εκτέλεσης τους και κατόπιν δημιουργεί γράφημα μέσω της βιβλιοθήκης Matplotlib στο οποίο συγκρίνει τους χρόνους εκτέλεσης του κάθε προβλήματος και για τους δύο τρόπους επίλυσης.

> Όπως αναφέρεται παρακάτω, στο αρχείο `plot.py` υπάρχει διάγραμμα της Matplotlib που περιλαμβάνει και αποτελέσματα της Pypy.

2.2 Test.py (unit tests)

Περιέχει την κλάση TestSolvers η οποία έλεγχει την σωστή λειτουργία του κυρίου προγράμματος (main.py) μέσω 3 μεθόδων οι οποίες εφαρμόζονται σε 3 προβλήματα (μικρού-μεσαίου-μεγάλου μεγέθους).

2.3 ResultsCPython.csv

Περιέχει τα αποτελέσματα απο την εκτέλεση των αλγορίθμων με CPython (όνομα αρχείου, χρόνος εκτέλεσης με OR-Tools, χρόνος εκτέλεσης απλού επιλυτή, συνολική αξία, συνολικό βάρος).

3 Υλοποίηση με Pypy

Η Pypy είναι μία εναλλακτική υλοποίηση της Python που παρέχει πολύ πιο γρήγορο χρόνο εκτέλεσης σε σύγκριση με την προεπιλεγμένη υλοποίηση της Python (CPython). Το κυριότερο μειονέκτημα ωστόσο είναι ότι δεν υποστηρίζονται απαραίτητα όλες οι βιβλιοθήκες που είναι διαθέσιμες για Python. [5, 6]

Επειδή οι βιβλιοθήκες OR Tools και Matplotlib δεν υποστηρίζονται σε Pypy, δοκιμάσαμε να τρέξουμε μόνο την δεύτερη υλοποίηση και να αποθηκεύσουμε τους χρόνους σε αρχείο .csv, ώστε να δούμε την διαφορά με την CPython. Οι υλοποιήσεις βρίσκονται στα αρχεία `mainPypy.py` και `testPypy.py`.

3.1 Εγκατάσταση

Για να εγκατασταθεί η Pypy σε Debian 10 Buster χρειάστηκαν τα ακόλουθα βήματα:

- Εγκατάσταση του συστήματος διαχείρισης πακέτων και επεκτάσεων Conda (χρησιμοποιήθηκε το Miniconda)
- Ενεργοποίηση του Conda για το κέλυφος Bash που χρησιμοποιείται: `conda init bash`
- Δημιουργία και ενεργοποίηση ενός εικονικού περιβάλλοντος για το Pypy:

  ```
  conda create --name pypy_env

  conda activate pypy_env
  ```

- Εγκατάσταση του Pypy3 σε αυτό:

  ```
  conda install pypy3
  ```

- Εγκατάσταση του pip (διαχειριστής πακέτων και προαπαιτούμενων στην Python):

  ```
  pypy3 -m ensurepip
  ```

- Εγκατάσταση βιβλιοθηκών απαραίτητες για να δουλέψει το Pandas:

  ```
  pypy3 -mpip install cython
  ```

```
pypy3 -mpip install numpy
```

- Τέλος, εγκατάσταση του Pandas:

```
pypy3 -mpip install pandas
```

3.2 ResultsPypy.csv

Περιέχει τα αποτελέσματα απο την εκτέλεση του απλού επιλυτή με Pypy (όνομα αρχείου, χρόνος εκτέλεσης του απλού επιλυτή).

4 Αποτελέσματα

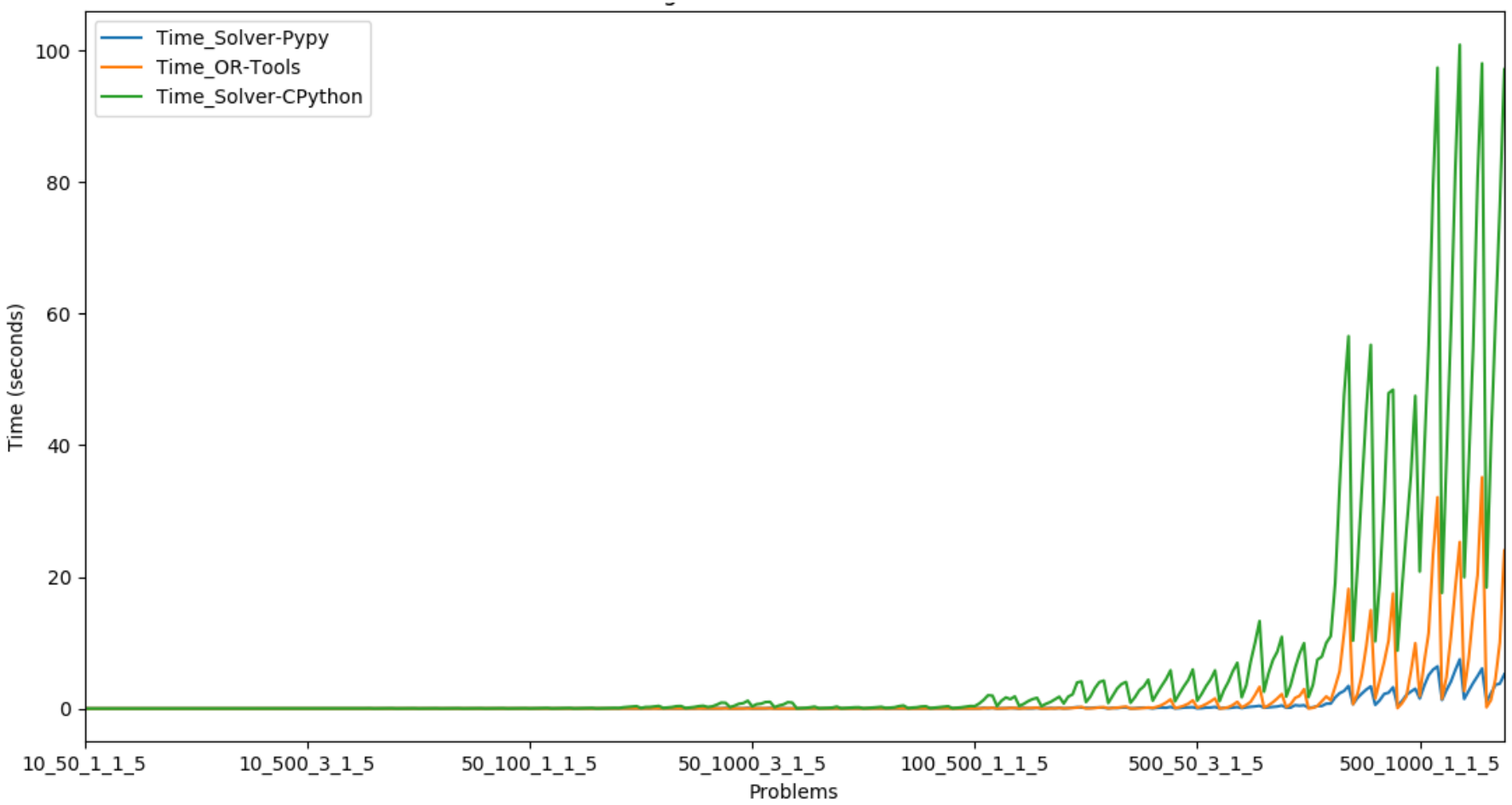
Στο αρχείο `plot.py` γίνεται δημιουργία διαγράμματος με τη βιβλιοθήκη Matplotlib, διαβάζοντας τα δεδομένα που είχαμε παράξει στα αρχεία `resultsCPython.csv` και `resultsPypy.csv`. Το αρχείο πρέπει να εκτελεστεί με CPython, γιατί η βιβλιοθήκη Matplotlib δεν δουλεύει σε Pypy. Τα αποτελέσματα της εκτέλεσης για κάθε πρόβλημα φαίνονται στο *Σχήμα 1* και στα παραγόμενα αρχεία .csv, ενώ ο συνολικός χρόνος των επιλυτών υπάρχει στον *Πίνακα 1*.

Επιλυτής	Συνολικός χρόνος εκτέλεσης
Απλός επιλυτής (CPython) + OR-Tools	2533 seconds
Απλός επιλυτής (CPython)	1974 seconds
Απλός επιλυτής (Pypy)	96 seconds

Πίνακας 1: Χρόνος εκτέλεσης για κάθε επιλυτή

Τέλος, στον *Πίνακα 2* βλέπουμε πόσος χρόνος απαιτείται για να εκτελεστούν τα tests για τον έλεγχο του απλού επιλυτή και με τις δύο υλοποιήσεις Python.

Αξίζει να σημειωθεί ότι για να τρέξουν τα tests πρέπει πρώτα να εκτελεστεί ο κώδικας της υλοποίησης, κάνοντας τη διαφορά στον χρόνο ακόμα πιο σημαντική (33 + 2 λεπτά για την CPython, έναντι 2 λεπτών για την Pypy).



Σχήμα 1: Συγκριτικό διάγραμμα χρόνων εκτέλεσης με χρήση της Matplotlib

5 Συμπεράσματα

Είναι φανερό ότι η Pypy προσφέρει αρκετά πλεονεκτήματα σε απόδοση σε σύγκριση με την κλασσική Python. Δεν αποτελεί ασφαλώς drop-in αντικαταστάτρια υλοποίηση γιατί δεν υποστηρίζεται από όλες τις βιβλιοθήκες του οικοσυστήματος της Python και η εγκατάσταση της απαιτεί κάποια χρονοβόρα και πιθανόν μη εμφανή βήματα. Ωστόσο μετά από έλεγχο των απαιτήσεων μας μπορεί να φανεί απρόσμενα χρήσιμη.

Όσον αφορά τη σύγκριση των 2 επιλυτών σε CPython είναι φανερό από το *Σχήμα 1* ότι αυτός του OR-Tools είναι πιο αποδοτικός από την απλή υλοποίηση δυναμικού προγραμματισμού.

Εκτέλεση unit tests με	Συνολικός χρόνος εκτέλεσης
CPython	133 seconds
Pypy	6 seconds

Πίνακας 2: Χρόνος εκτέλεσης του `testPypy.py`

Αναφορές

[1] Νικόλας Μανδέλλος (Ανάκτηση 09/01/2020). *Πρόβλημα Σακιδίου - Knapsack Problem (http://users.ntua.gr/nmand/Knapsack.html)*

[2] David Pisinger (1994). *Core problems in Knapsack Algorithms - Operations Research (http://hjemmesider.diku.dk/~pisinger/codes.html)*

[3] Google Developers (2020). *OR-Tools (https://developers.google.com/optimization)*

[4] GeeksForGeeks.com, Bhavya Jain et al (Ανάκτηση 09/01/2020). *0-1 Knapsack Problem | DP-10 (https://www.geeksforgeeks.org/0-1-knapsack-problem-dp-10/)*

[5] Pypy contributors (Ανάκτηση 10/01/2020). *What is PyPy? (https://pypy.org/features.html)*

[6] Benjamin Peterson (2012). *The Architecture of Open Source Applications (Volume 2): PyPy (https://www.aosabook.org/en/pypy.html)*