Različiti približni algoritmi za problem pakovanja

Seminarski rad iz Konstrukcije i Analize Algoritama 2

Popović Ognjen

27.8.2024

1. Uvod

Problemi pakovanja pripadaju klasi optimizacionih problema. U ovakvim problemima zadat je skup objekata, potencijalno različitih veličina i jedna ili više kutija fiksnog kapaciteta. Cilj je najčešće spakovati sve objekte u kutije koristeći što je moguće manje kutija ili, u slučaju problema kada je data jedna kutija, što je kompaktnije moguće napakovati istu.

Postoje razne varijacije ovog problema, po broju dimenzija koje se posmatraju, po obliku kutija i objekata koji se smeštaju, po cilju optimizacije, pa i varijacija gde se posmatra pakovanje u beskonačni, Euklidski prostor.

2. Definicija problema

U ovom radu fokus će biti na problemu koji je formalno definisan na sledeći način:

- Neka je I skup objekata, gde za $\forall i \in I$ važi da ima veličinu $s(i)$, gde $s(i) \in (0,1]$.
- Neka je B vrednost koja predstavlja kapacitet kutija koji ne sme biti prevaziđen, i neka je $B = 1$.
- Neka je $K \in \mathbb{Z} \wedge K > 0$ vrednost koja predstavlja iskorišćen broj kutija.
- Neka je K vrednost koju je potrebno minimizovati uz uslove $(\forall i) \sum_{j=1}^{n} s(x_{ij}) \leq B = 1$ i $I' = I$, tj. uz uslove da suma svih veličina objekata u jednoj kutiji ne prevazilazi B, odnosno 1 i da je skup spakovanih objekata I' jednak skupu svih objekata.
- Rešenje problema je optimalno ako je K minimalno.
- Vrednost K za optimalno rešenje je definisano kao $K = OPT(I)$.

Neformalno objašnjenje problema glasi:

Objekte različite veličine treba rasporediti u kutije sa predefinisanim, fiksnim kapacitetom jednakim 1 tako da se minimizuje broj kutija koje se koriste, gde se ne smeju preklapati objekti smešteni u istu kutiju i ne smeju se premašiti kapaciteti kutija.

3. Algoritmi

Ovaj problem spada u klasu NP kompletnih problema i samim time ne postoji efikasan algoritam za njegovo rešavanje [1].

Postoje optimalni algoritmi za ovaj problem: **MPT**, **Bin Completion**, **Improved Bin Completion** i **BCP.** Sa druge strane, približni algoritmi ne garantuju optimalnost rešenja, ali se izvršavaju

mnogo brže nego egzaktni algoritmi. Za problem pakovanja donja granica faktora aproksimacije je $\frac{3}{2}$ [2].

3.1. Online algoritmi

Kod online algoritama objekti nisu poznati unapred. Na početku izvršavanja poznat je samo prvi objekat i tek po njegovom pakovanju otkriva se sledeći objekat. Operacija pakovanja je nepovratljiva. Online algoritmi se mogu koristiti i za rešavanje offline verzije ovog problema.

- **Next Fit**

Ako sledeći objekat ne može da stane u trenutno otvorenu kutiju, kutija se zatvara i nova, prazna, kutija se otvara. Ovaj postupak se ponavlja dok se ne smeste svi objekti u kutije.

- **Next-k-Fit**

Varijanta Next Fit algoritma gde se poslednjih k kutija drže otvorenim i trenutni objekat se pakuje u prvu u koju može da stane.

- **First Fit**

Sve kutije se drže otvorenim i trenutni objekat se pakuje u prvu kutiju u koju može da stane.

- **Best Fit**

Sve kutije se drže otvorenim i trenutni objekat se pakuje u kutiju koja je najviše napunjena, a u koju i dalje može da stane.

- **Worst Fit**

Sve kutije se drže otvorenim i trenutni objekat se pakuje u kutiju koja je najmanje napunjena, a u koju i dalje može da stane.

- **Almost Worst Fit**

Sve kutije se drže otvorenim i trenutni objekat se pakuje u drugu najmanje napunjenu kutiju ukoliko u nju može da stane. Ukoliko ne, prelazi se na najmanje napunjenu kutiju.

- **Refined First Fit** [3]

Objekti i kutije se kategorišu u 4 kategorije po veličini: $\left(0, \frac{1}{3}\right]$, $\left(\frac{1}{3}, \frac{2}{5}\right]$, $\left(\frac{2}{5}, \frac{1}{2}\right]$, $(\frac{1}{2}, 1]$. Svaki novi objekat se prvo kategoriše, pa se onda pakuje u kutiju u njegovoj klasi po principu First Fit.

- **Harmonic-k** [4]

Sličan algoritam kao i Refined First Fit, osim što se kategorije određuje prema Harmonijskom redu.

- **Refined-harmonic** [4]

Kombinuje ideje iz Refined First Fit i Harmonic-k algoritama. Za objekte veće od $\frac{1}{3}$ koristi se Refined First Fit princip, dok se za ostale objekte koristi Harmonic-k princip.

3.2. Offline algoritmi

U offline verziji ovog problema, svi objekti i njihove veličine su poznati unapred, što omogućava bolje faktore aproksimacije. Ovi algoritmi se ne mogu koristiti u online verziji ovog problema.

Većina offline algoritama se svodi na sortiranje liste objekata opadajuće po veličini i onda na korišćenje nekog online algoritma: **First-Fit-Decreasing (FFD)**, **Next-Fit-Decreasing**, **Modified-First-Fit-Decreasing (MFFD)**, itd.

Postoje i napredniji offline algoritmi koji imaju bolji faktor aproksimacije, ali su dosta manje efikasni, sa visokim stepenima polinomijalnog izvršavanja:

- **Karmarkar-Karp**

Pohlepni algoritam čija je najpoznatija primena u particionisanju brojeva. Može biti adaptiran da se koristi u problemima poput problema pakovanja. Pruža dobru heuristiku za balansiranje popunjenosti kutija [5].

- **Rothvoss**

Algoritam zasnovan na tehnikama linearnog programiranja kao što su probabilističko zaokruživanje i LP relaksacija. Poboljšava faktor aproksimacije rešavanjem relaksirane verzije problema pakovanja [6].

- **Hoberg and Rothvoss**

Zasniva se na Rothvoss algoritmu i koristi unapređene tehnike probabilističkog zaokruživanja u odnosu na isti [7].

3.3. Karakteristike algoritama

- Neka je L lista objekata
- $A(L)$ predstavlja broj kutija iskorišćen kada se na listu L primeni algoritam A
- $OPT(L)$ predstavlja optimalni broj kutija za listu L

Karakteristike nekih od online algoritama do sada predstavljenih u radu prikazani su u tabeli Table 1 [3] [8] [9] [10]:

Table 1

Naziv algoritma	Faktor aproksimacije	Najgori slučaj	Vremenska složenost
Next Fit	$NF(L) \leq 2 \cdot OPT(L) - 1$	$NF(L) \leq 2 \cdot OPT(L) - 2$	$0(\|L\|)$
First Fit	$FF(L) \leq \lfloor 1.7 \cdot OPT(L) \rfloor$	$FF(L) \leq \lfloor 1.7 \cdot OPT(L) \rfloor$	$O(\|L\| \log(\|L\|))$
Best Fit	$BF(L) \leq \lfloor 1.7 \cdot OPT(L) \rfloor$	$BF(L) \leq \lfloor 1.7 \cdot OPT(L) \rfloor$	$O(\|L\| \log(\|L\|))$
Worst Fit	$WF(L) \leq 2 \cdot OPT(L) - 1$	$WF(L) \leq 2 \cdot OPT(L) - 1$	$O(\|L\| \log(\|L\|))$
Refined First Fit	$RFF(L) \leq \frac{5}{3} \cdot OPT(L) + 5$	$RFF(L) \leq \frac{5}{3} \cdot OPT(L) + \frac{1}{3}$	$O(\|L\| \log(\|L\|))$

Karakteristike nekih od offline aloritama do sada predstavljenih u radu prikazani su u tabeli Table 2 [5] [7] [11] [12]:

Table 2

Naziv algoritma	Faktor aproksimacije
FFD	$FFD(L) \leq \frac{11}{9} \cdot OPT(L) + \frac{6}{9}$
MFFD	$MFFD(L) \leq \frac{71}{60} \cdot OPT(L) + 1$
Karmarkar-Karp	$KK(L) \leq OPT(L) + O(\log^2(OPT(L))$
Hoberg-Rothvoss	$HB(L) \leq OPT(L) + O(\log(OPT(L))$

4. Implementacija

Naredni algoritmi implementirani su u Python-u: **Next Fit**, **Next k Fit** (sa vrednostima 2, 10, 100 za k), **First Fit**, **Best Fit**, **Worst Fit**, **Almost Worst Fit**, **Refined First Fit**, **Harmonic k** (sa vrednošću 20 za k) i **Refined Harmonic** (sa vrednošću 20 za k).

Algoritam NF ima vremensku složenost $O(n)$, algoritmi NkF, HK i RH imaju $O(n * k)$, dok FF, BF, WF, AWF i RFF imaju $O(n * \log(n))$. Posebno treba naglasiti da su se algoritmi FF i RFF u praksi pokazali kao značajno sporiji zbog konstrukcije binarnog stabla pre izvršavanja samog algoritma. Implementirane su i offline verzije ovih algoritama tako što je niz objekata prvo sortiran opadajuće, pa su onda primenjene online verzije. Treba napomenuti da online verzije FF i MFF algoritama unapred znaju broj objekata u nizu kako bi bila moguća konstrukcija stabla

neophodna za efikasnu implementaciju algoritama, kao i da offline verzija RFF algoritma efikasnije konstruiše stabla nego online verzija.

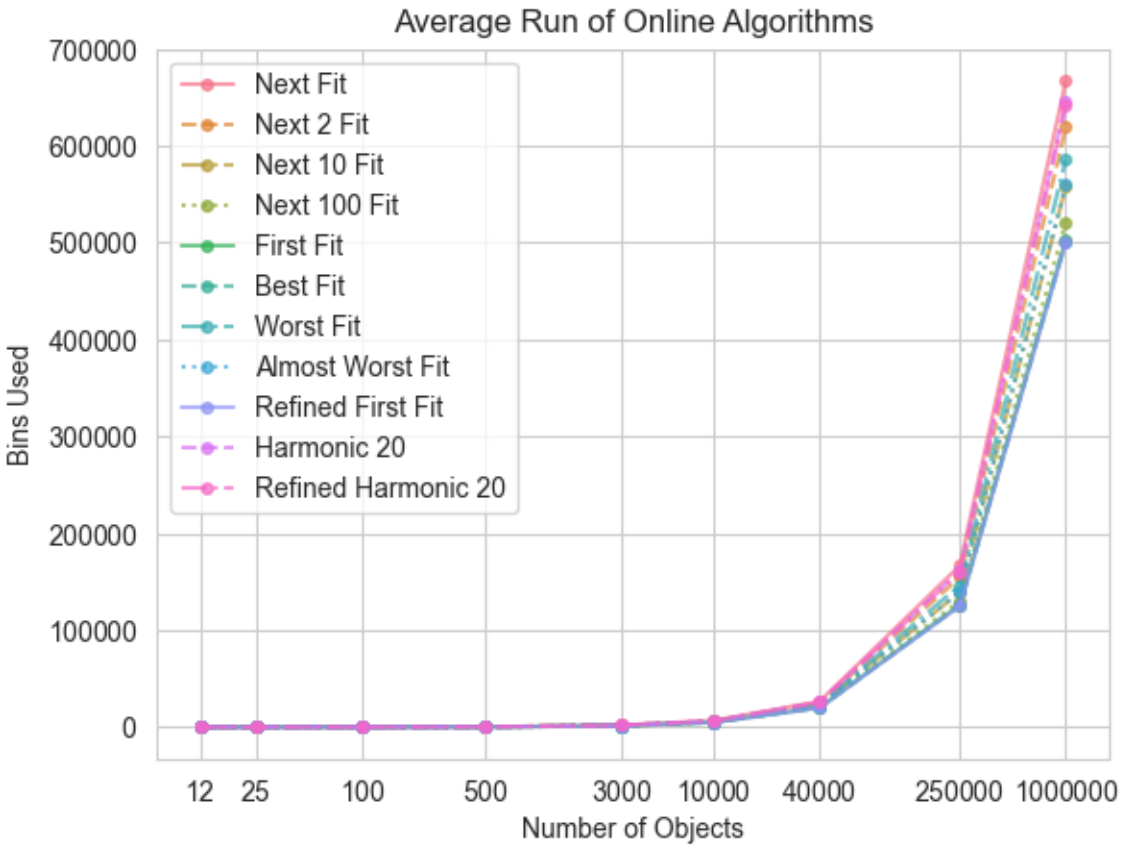
Navedene vremenske složenosti se odnose na online verzije algoritama, dok će za offline verzije dominirati sortiranje sa $O(n * \log(n))$ složenošću. U praksi, offline verzije algoritama će se pokazati kao brže jer se sami algoritmi brže izvršavaju na sortiranom nizu, a funkcija sortiranja je implementirana kroz C kod, koji se značajno brže izvršava nego Python kod.

Što se tiče testiranja i poređenja algoritama korišćeni su liste objekata veličine: 12, 25, 100, 500, 3000, 10000, 40000, 250000 i 1000000. Objekti su slučajno generisani tako da imaju vrednosti između 0 i 1. Svaki algoritam je pokrenut 100 puta za svaku veličinu ulaza i izračunat je prosek iskorišćenih kutija, kao i vreme koje je utrošeno za izvršavanje. Za offline algoritme računat je prosek vremena koje je bilo potrebno za sortiranje listi svih veličina i taj prosek je dodavan na prosek izvršavanja algoritama.

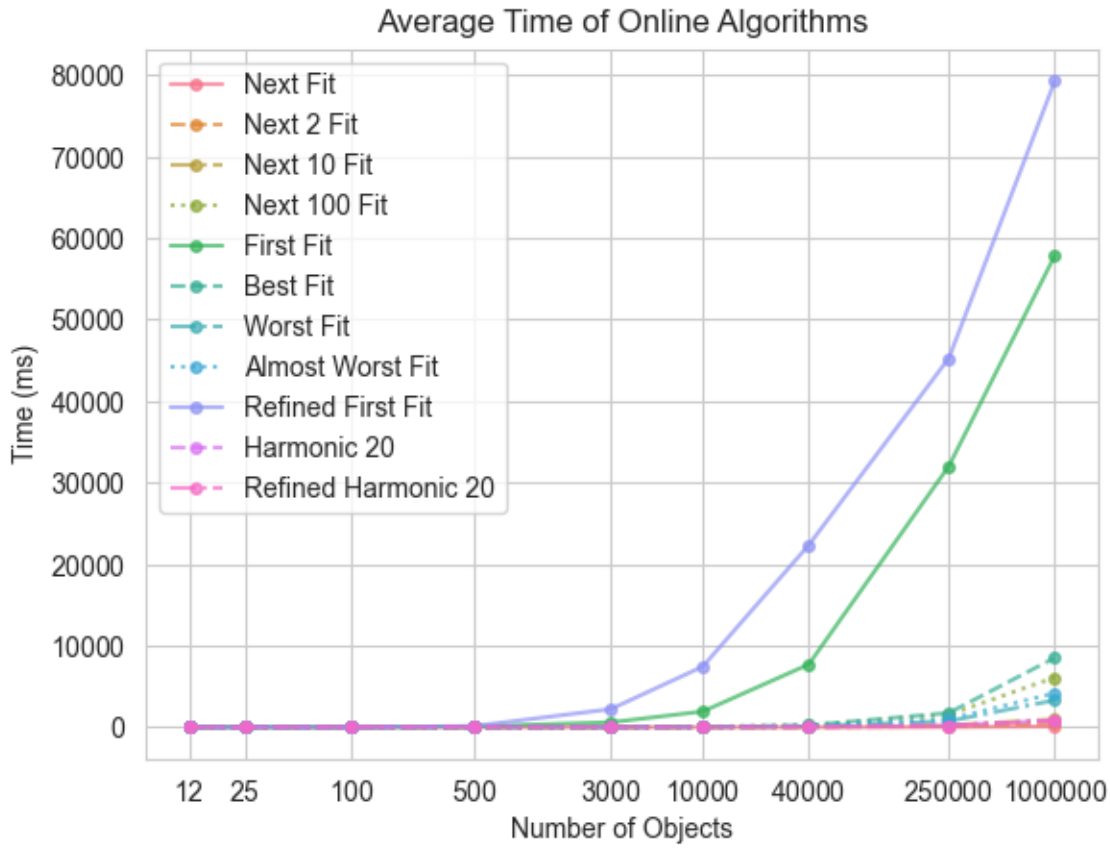
Za izvršavanje celokupnog testiranja i poređenja algoritama bilo je potrebno 14 sati. Celokupan kod, implementacija i grafici mogu se videti na sledećem linku.

5. Rezultati

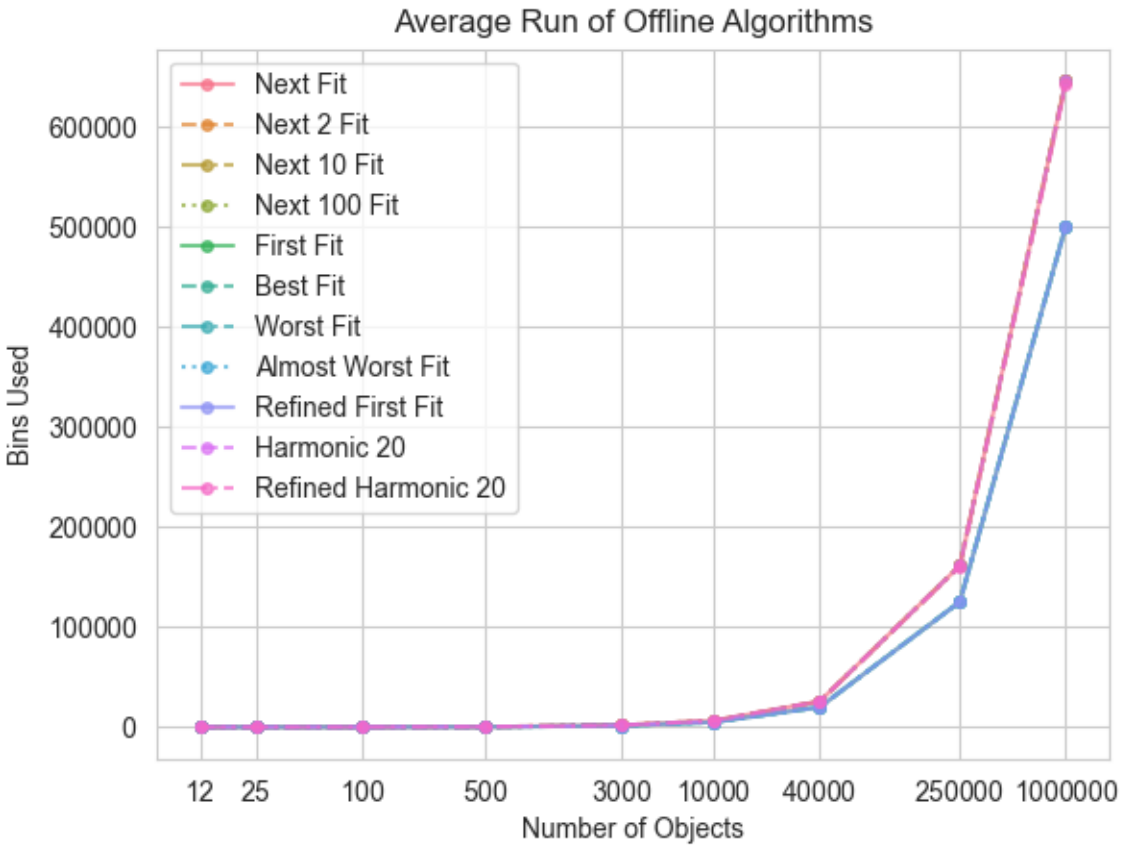
Na sledećim graficima prikazani su rezultati testiranja i poređenja algoritama. Grafici prikazuju iskorišćenost kutija i potrebnog vremena za izvršavanje algoritma u odnosu na povećanje veličine ulaza.



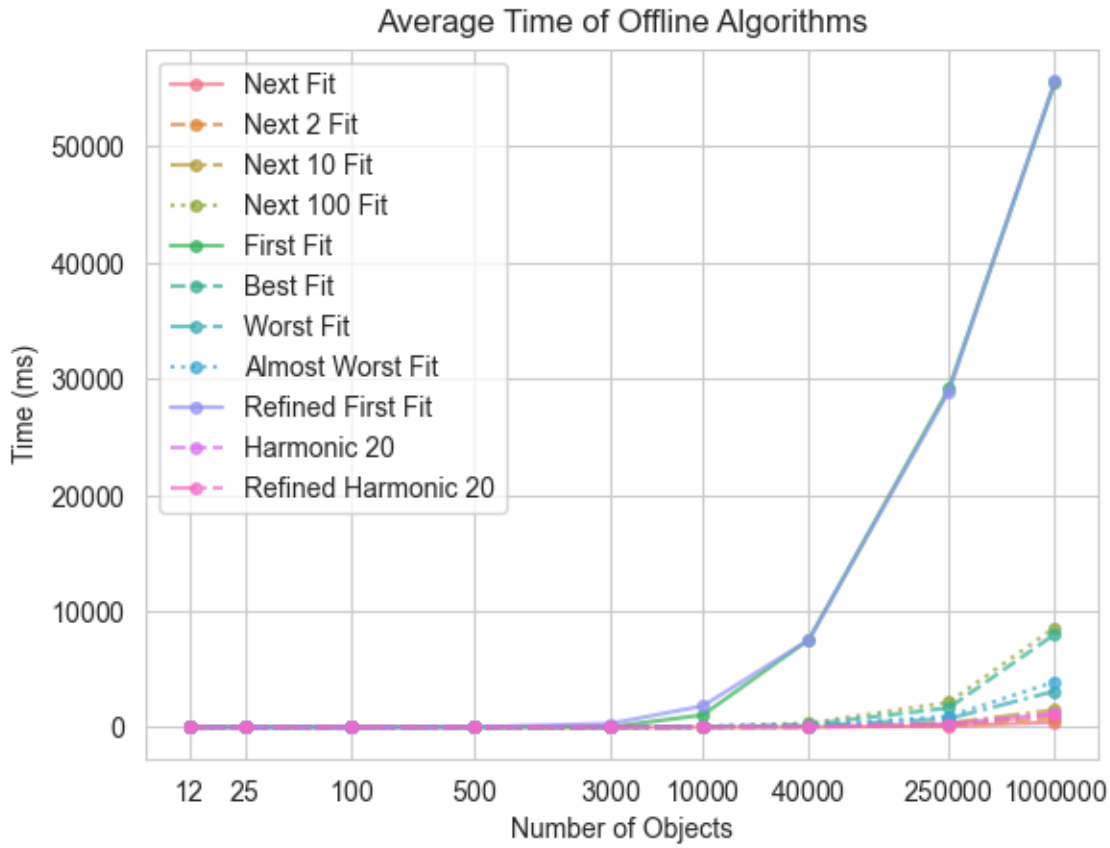
Algoritam NF se pokazao kao najgori sa preko 650000 kutija iskorišćenih pri veličini ulaza od 1000000. Prate ga H20, RH20 i N2F kojima je bilo potrebno preko 600000 kutija. Bolje od njih su se pokazali WF, AWF i N10F sa više od 550000 kutija. Između njih i najboljih algoritama je N100F. Najbolji algoritmi, sa otprilike 500000 iskorišćenih kutija, su FF, BF i RFF.



Ubedljivo najviše vremena za izvršavanje je potrebno algoritmima RFF i FF zbog konstrukcije stabla pred svako izvršavanje. Sledećim algoritmima je bilo potrebno manje od 10s za izvršavanje pri najvećoj veličini ulaza: BF, N100F, AWF, WF, N10F, dok su najbrži algoritmi H20, RH20, N2F i NF.



Offline algoritmi teže dvema tačkama na grafiku, oko 650000 i oko 500000 iskorišćenih kutija. Grupa lošijih algoritama sastoji se od H20, NF, N2F, H10F, RH20 i N100F algoritama, dok boljoj grupi pripadaju algoritmi: WF, AWF, FF, BF i RFF.



Opet je ubedljivo najviše vremena potrebno algoritmima FF i RFF. Za njima slede N100F, BF, a potom i AMF i WF. Najbrži algoritmi su N10F, H20, RH20, N2F i na prvom mestu NF.

6. Zaključak

Po prikazanim rezultatima može se zaključiti da je za izabranu implementaciju bolje koristiti offline algoritme zbog veće efikasnosti i bolje iskorišćenosti kutija kod većeg broja algoritama.

Najkorisniji online algoritam je **Best Fit** zbog jedne od najboljih iskorišćenosti kutija, a istovremeno i vremena izvršavanja koje iznosi manje od 10 sekundi.

Najkorisiji offline algoritmi su **Best Fit**, **Almost Worst Fit** i **Worst Fit** zbog istovremeno dobre iskorišćenosti kutija i vremena izvršavanja koja iznose manje od 10 sekundi.

Napomena je da brzina izvršavanje može jako da varira između različitih implementacija i programskih jezika, tako da su dobijeni i prikazani rezultati validni samo za testiranu implementaciju.

7. Reference

[1] Garey, M. R., Johnson, D. S. Victor Klee (ed.)., Computers and Intractability: A Guide to the Theory of NP-Completeness, 1979.

[2] V. V. Vazirani, Approximation Algorithms, 2013.

[3] A. C.-C. Yao, "New Algorithms for Bin Packing," *Journal of the ACM,* 1980.

[4] C. C. Lee and D. T. Lee, "A simple online bin-packing algorithm," *Journal of the ACM,* 1985.

[5] N. Karmarkar and R. M. Karp, "An efficient approximation scheme for the one-dimensional bin-packing problem," in *23rd Annual Symposium on Foundations of Computer Science (sfcs 1982)*, 1982, pp. 312-320.

[6] T. Rothvoß, "Approximating Bin Packing within O(log OPT · Log Log OPT) Bins," in *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, 2013, pp. 20-29.

[7] R. Hoberg and T. Rothvoss, "A Logarithmic Additive Integrality Gap for Bin Packing," in *Proceedings of the 2017 Annual ACM-SIAM Symposium on Discrete Algorithms*, 2017, pp. 2616-2625.

[8] D. S. Johnson, Near-optimal bin packing algorithms, 1973.

[9] G. Dósa and J. Sgall, "First Fit bin packing: A tight analysis," in *30th International Symposium on Theoretical Aspects of Computer Science*, 2013.

[10] D. György and J. Sgall, "Optimal Analysis of Best Fit Bin Packing," in *Automata, Languages, and Programming*, 2014, pp. 429-441.

[11] G. Dósa, "The Tight Bound of First Fit Decreasing Bin-Packing Algorithm Is FFD(I) ≤ 11/9\mathrm{OPT}(I) + 6/9," in *Combinatorics, Algorithms, Probabilistic and Experimental Methodologies*, 2007.

[12] M. Yue and L. Zhang, "A simple proof of the inequality MFFD(L) ≤ 71/60 OPT(L) + 1,L for the MFFD bin-packing algorithm," *Acta Mathematicae Applicatae Sinica,* vol. 11, pp. 318-330, 1995.