Rješavanje problema pakiranja kutija korištenjem metaheuristika

Opis problema

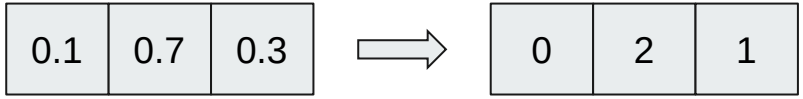
- Više varijanti
  - Rad se bavi 1D varijantom
- Pakirati elemente u kutije
- Cilj: minimizirati neistkorišten prostor kutija
- Pravila
  - Niti jedan element nije veći od same kutije
  - Neograničen broj kutija na raspolaganju
  - Sve kutije iste su veličine
- Zadaci algoritama
  - Heuristički algoritam (Best-Fit) - pakira elemente u kutije
  - Metaheuristički algoritmi - uče poredak elemenata koji se predaju heurističkom algoritmu

Klasične heurističke metode

- Next-Fit
  - Ne vraća se na stare kutije.
- First-Fit
  - Stavi element u najstariju kutiju u koju stane.
- Best-Fit
  - Stavi element u kutiju koja ima najmanje prostora.

Kodiranje jedinki

- Jedinka
  - Vektor s brojem dimenzija jednakim broju elemenata za pakiranje.
  - Predstavlja redoslijed predavanja elemenata heurističkom algoritmu.
- Redoslijed se iz jedinke dobiva rangiranjem vrijednosti unutar vektora.



Algoritam krijesnice

- Pravila
  - Krijesnice su unisex.
  - Sjajnija privlači slabiju.
  - Svjetlost je određena funkcijom dobrote.
- Svjetlost vs intenzitet
- Parametri
  - Koeficijent apsorpcije γ
  - Zadani intenzitet β_0
  - Koeficijent nasumičnog pomaka α

$$\beta = \beta_0 * e^{-\gamma r^2}$$

$$s_i^{t+1} = s_i^t + \beta_0 e^{-\gamma r_{ij}^2}(s_j^t - s_i^t) + \alpha \epsilon_i^t$$

function FIREFLY($\gamma, \alpha, \beta_0, \pi, degradacijaMutacije$)

Inicijaliziraj populaciju od N jedinki

Izračunaj intenzitet krijesnica pomoću funkcije dobrote

while (t < MaxGeneracija) **do**

for i = 0 .. N **do**

for j = 0 .. i **do**

if $I_j > I_i$ **then**

Pomakni i-tu krijesnicu prema j-oj prema formuli **(3.3)**

Mutiraj i-tu jedinku prema vjerojatnosti π

Izračunaj novi intenzitet jedinke

end if

end for

end for

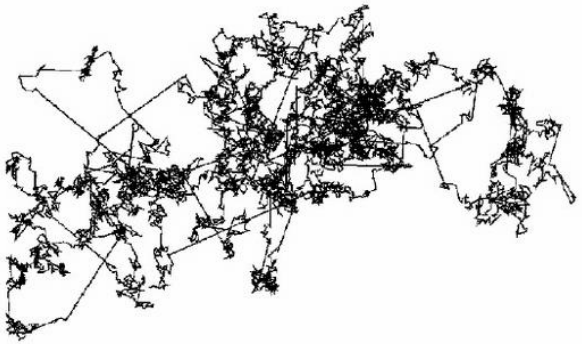
$\pi = \pi * degradacijaMutacije$

Nađi najbolju krijesnicu

end while

end function

Kukavičji algoritam



- Lévyjev let
  - Mantegnov algoritam

$$L(\lambda, \sigma^2) = \frac{u}{|v|^{\frac{1}{\lambda}}} \qquad u \sim N(0, \sigma^2), v \sim N(0, 1)$$

- Lokalno pretraživanje

$$x_i^{t+1} = x_i^t + \alpha s * H(p_a - \epsilon) * (x_j^t - x_k^t)$$

- Globalno pretraživanje

$$x_i^{t+1} = x_i^t + \alpha s L(\lambda, \sigma^2)$$

function CUCKOOSEARCH($\alpha, s, p_a, \lambda, \sigma^2$)

Inicijaliziraj N gnijezda

while (t < MaxGeneracija) **do**

Nasumično odaberi gnijezdo x_i i x_j koristeći k-turnir

Generiraj x_i' formulom **(4.2)**

if $dobrota(x_i') > dobrota(x_j)$ **then**

Izbaci x_j i zamijeni ga novim x_i'

end if

p_a najlošijih gnijezda zamijeni njihovim lokalnim nasumičnim hodom danim formulom **(4.1)**

Evaluiraj gnijezda i nađi najbolje

end while

end function

Genetski algoritam

- Selekcija
- Križanje

$$Roditelj\ 1: \ [0.1, 0.7, 0.3, 0.1]$$

$$Roditelj\ 2: \ [0.8, 0.3, 0.6, 0.2]$$

$$Dijete\ dobiveno\ križanjem\ roditelja\ 1\ i\ 2: \ [0.1, 0.7, 0.6, 0.2]$$

- Mutacija

$$[0.1, 0.7, 0.6, 0.2] \Rightarrow [0.6, 0.7, 0.1, 0.2]$$

function GENETICALGORITHM(π)

Inicijaliziraj N kromosoma

while (t < MaxGeneracija) **do**

Nasumično odaberi tri kromosoma koristeći k-turnir

Napravi dijete od bolja dva koristeći operator križanja

Operatorom mutacije mutiraj dobiveno dijete

Zamijeni najlošiji kromosom dobiven iz k-turnira sa novo kreiranim djetetom

Evaluiraj kromosome i nađi najbolje rješenje

end while

end function

Rezultati - postavke

Parametar	Vrijednost
γ	0.3
α	0.01
β_0	0.01
π	0.05
$degradacijaMutacije$	0.8
$veličinaPopulacije$	20
$MaxGeneracija$	100

Parametri Firefly algoritma

Parametar	Vrijednost
αs	2
p_a	0.1
λ	0.1
σ^2	2
$veličinaPopulacije$	20
$MaxGeneracija$	100

Parametri Cuckoo algoritma

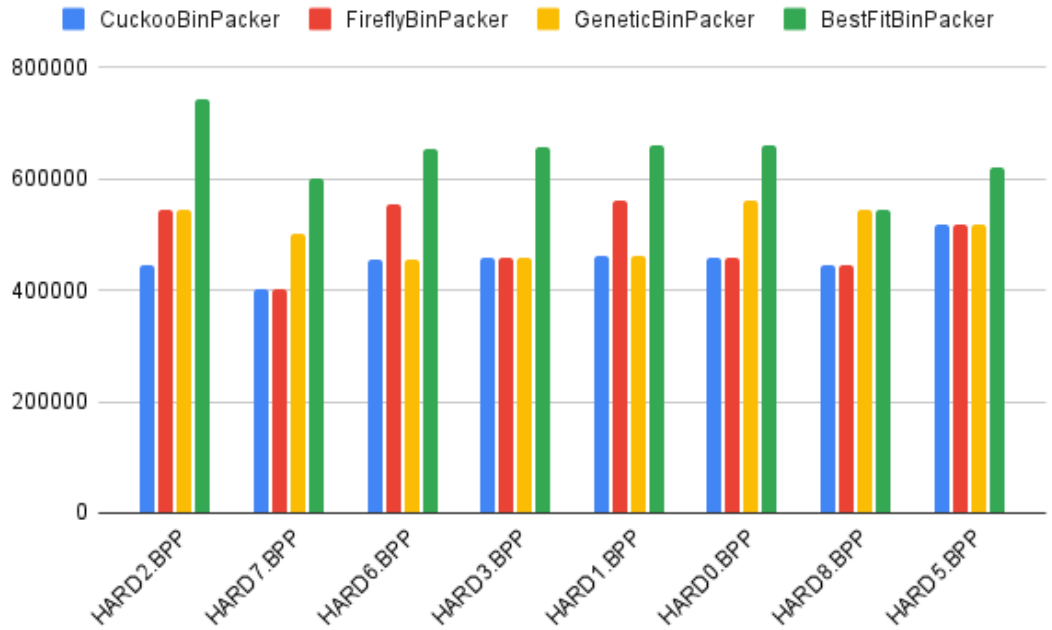
Parametar	Vrijednost
π	0.05
$veličinaPopulacije$	50
$MaxGeneracija$	200

Parametri genetskog algoritma

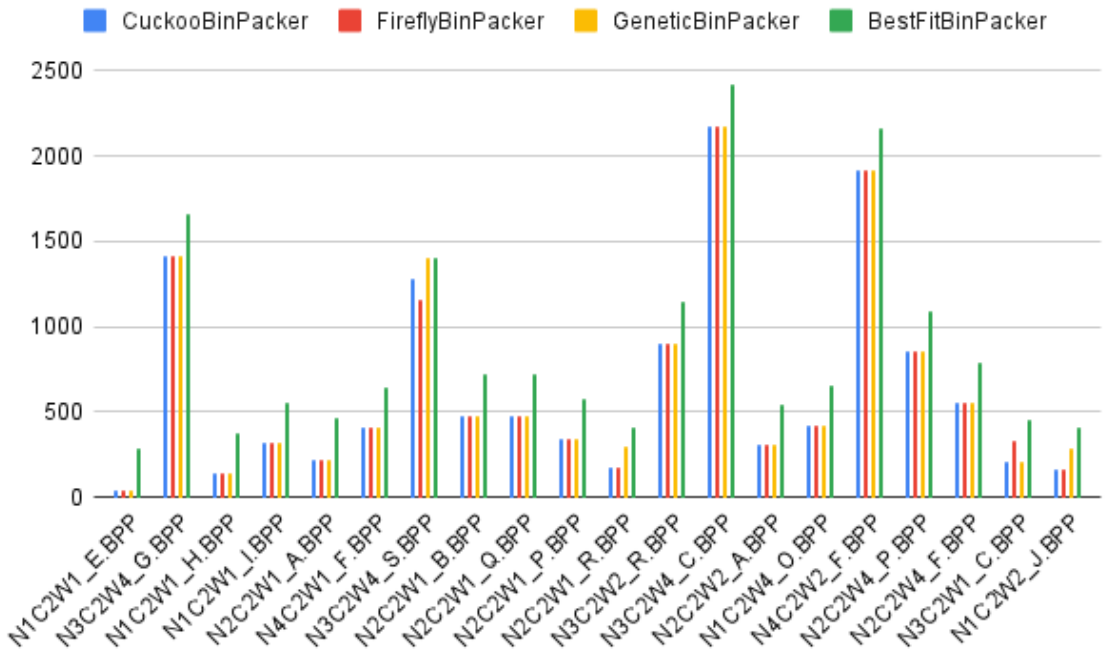
Rezultati - skup podataka

- Ulazi
  - Veličina kutije
  - Niz veličina elemenata koje treba pakirati
- Klase problema
  - Jednostavni
    - Mnogo elemenata po kutiji
    - Mala raspršenost veličina elemenata
    - Rezultati algoritama ujednačeni
  - Srednji
  - Teški
    - Malo elemenata po kutiji (3 do 5)
    - Velika raspršenost veličina elemenata

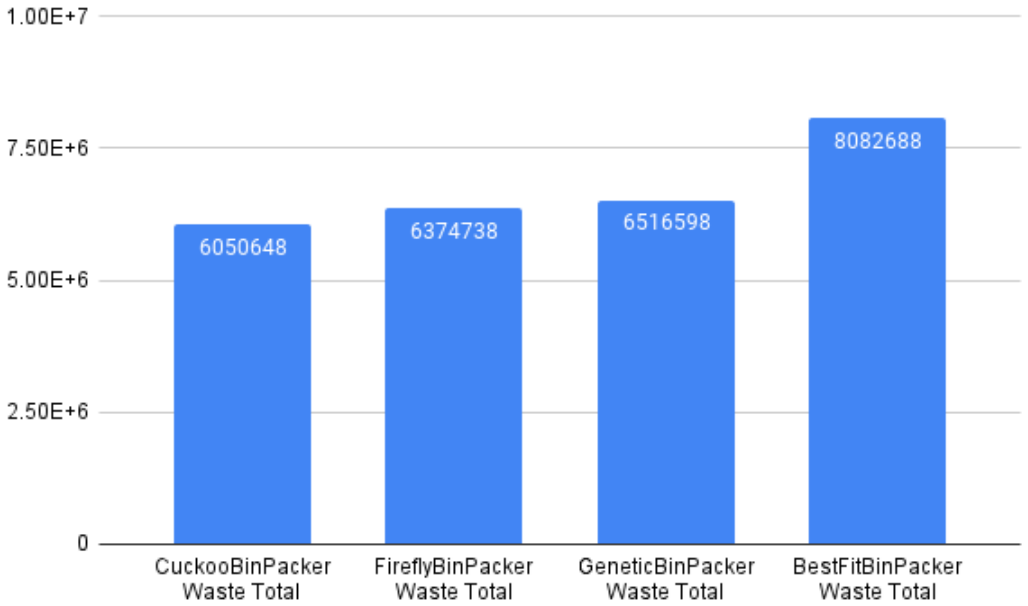
Rezultati - težak skup



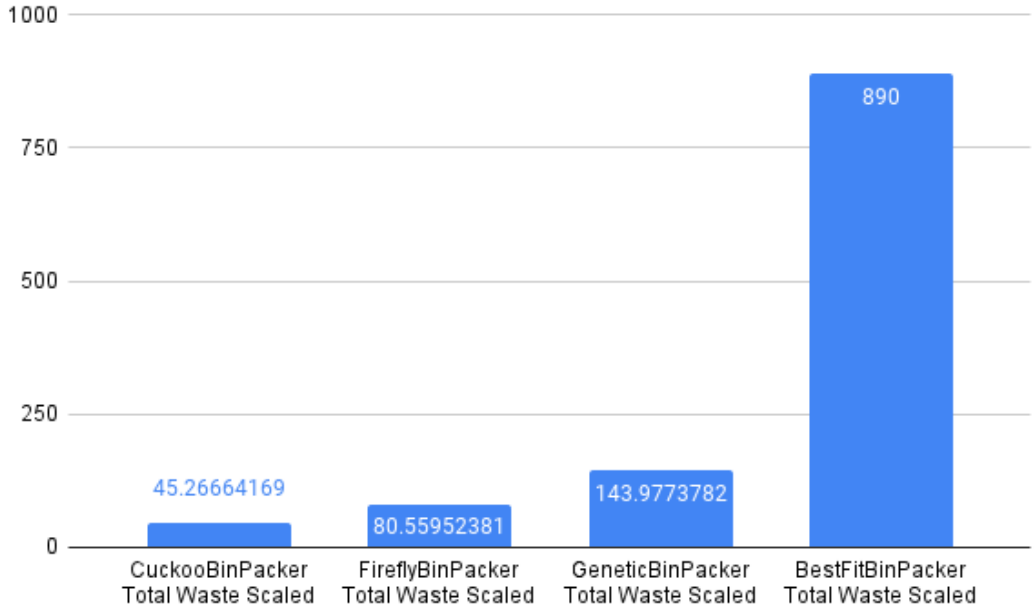
Rezultati - lagan skup



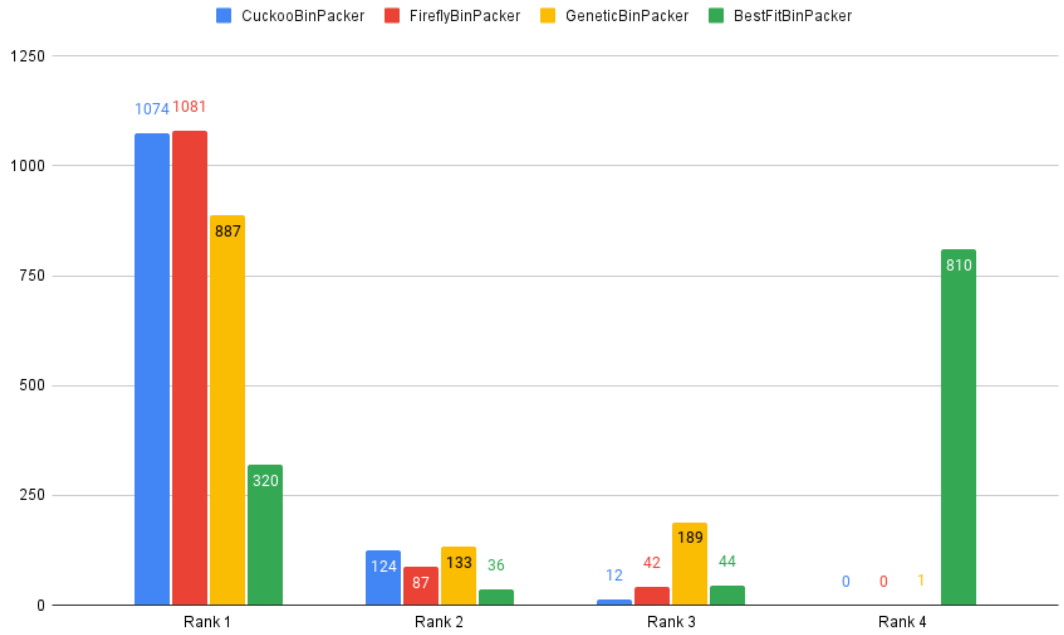
Rezultati - ukupan neiskorišteni prostor



Rezultati - ukupan neiskorišteni prostor (min-max skaliran)



Rezultati - rangovi



Zaključak

- Kukavičji algoritam generalno je najbolji.
  - Levyjev let zaslužan.
- Algoritam krijesnice najbolji je na najviše problema.
- Metaheuristički algoritmi neovisni su od početnom rasporedu elemenata.