Hibridni genetskom algoritam grupa u rešavanju jednodimenzionog problema minimalnog pakovanja

Andrija Urošević

Univerzitet u Beogradu, Matematicki fakultet
andrija.urosevic@protonmail.com

Septembar 2022.

Sažetak

Genetski algoritmi imaju široku primenu u oblasti optimizacije. Specijalno koriste se za probleme rapsoređivanja i planiranja. Problem minimalnog pakovanje je jedan takav problem koji je NP-težak. Ovaj rad istražuje moguće implementacije genetskog algoritma, i potencijalne probleme koji nastaju pri implementaciji. Tačnije, kako najbolje predstaviti jedinke i kako definisati operatore ukrštanja i mutacije, zajedno sa odgovarajućom funkcijom prilagođenosti. Operatori ukrštanja i mutacije sadrže dodatne heuristike pa se zbog toga genetski algoritam nazivamo hibridnim. Takođe, postavlja se pitanje kako definisati jedinke, te se uvodi koncept grupa. Zbog toga ovaj algoritam ima naziv hibridni genetski algoritam grupa.

1 Uvod

Problem minimalnog pakovanja u prostoru (engl. Minimal Bin Packing Problem) podrazumeva pokovanje objekata različitih zapremina u konačno mnogo kontejnera sa ciljem da se minimizuje broj potrebnih kontejnera za skladištenje datih objekata. Formalno, neka je dat konačan skup elemenata I, neka svaki element $i \in I$ ima veličinu $s(i) \in \mathbb{Z}^{+^n}$, i neka je dat kapacitet kontejnera B. Treba naći particiju $\{I_1, I_2, \ldots, I_k\}$ skupa elemenata I, tako da je $\sum_{i \in I_k} s(i) \leq B$ i broj kontejnera k najmanji moguć. Ovaj problem je NP-težak[1]. Njegova varijanta problema odlučivanja: Da li za broj kontejnera k postoji particija $\{I_1, I_2, \ldots, I_k\}$ skupa elemenata I, tako da $\sum_{i \in I_k} s(i) \leq B$?

Jednodimenzioni problem minimalnog pakovanje je problem minimalnog pakovanja u prostoru gde su veličine elemenata $s(i) \in \mathbb{Z}^+$ i gde je kapacitet kontejnera $B \in \mathbb{Z}^+$. Optimalan vrednost broj kontejnera k za dati skup elemenata I, obeležavamo i sa $OPT(I)$. U literaturi se koristi i normalizovani oblik problema. Naime, $B = 1$ i $s(i) \in [0, 1]$ za svaki element $i \in I$ (normalizujemo instance tako što podelimo veličinu $s(i)$ sa B, a B postavimo na 1).

Problem jednodimenzionog minimalnog pakovanja napadamo koristeći genetski algoritam sa specijalizovanom reprezentacijom i specijalizovanim operatorima genetskog algoritma. Trenutno najbolja polinomijalna aproksimativna rešenja upadaju u $\frac{3}{2}OPT(I)$[10] i $\frac{71}{60}OPT(I)$[5, 11]. Ovaj rad, eksperimentalno, pokušava da upadne u ove granice. Postoje i bolje, kompleksnije polinomijalne aproksimacije čije rešenje nije veće od $OPT(I) + O(\log^2(OPT(I))$[7]. Dalje imamo poboljšanja ove ideje gde rešenje nije veće od $OPT(I) + O(\log(OPT(I)) \log\log(OPT(I)))$[8] i na

kraju $OPT(I) + O(\log(OPT(I)))$[4].

2 Genetski algoritam za minimalno pokovanje

Problem jednodimenzionog minimalnog pakovanja pokušavamo da rešimo genetskim algoritmom. Postoje mnoga pitanja i problemi koji nastaju pri implementiranju genetskog algoritma za ovaj problem. Glavni problemi koji nastaju je genetski algoritam koji se ponaša kao nasumična pretraga[3]. Naime, izbor reprezentacija jedinki, a kasnije i odgovarajućih specijalizovanih operatora ukrštanja i mutacije je od ključnog značaja za rešavanje ovog problema. U daljem tekstu su opisani načini na koje treba implementirati genetski algoritam za efikasno rešavanja problema jednodimenzionog pakovanja.

2.1 Reprezentacija jedinki

Razmatramo tri načina za predstavljanje hromozoma: Naivni pristup, permutacije i grupe. Svaki sledeći, respektivno, smanjuje nivo redudandnosti koja se javljaju u reprezentaciji hromozoma.

2.1.1 Naivni pristup

Hromozom kod naivnog pristupa podrazumeva listu dužine $\|I\|$, gde je i-ti element u listi kontejner u koji pakujemo element $i \in I$. Na primer, hromozom

$$ABCCAC$$

predstavlja pakovanje gde se prvi element pakuje u kontejner A, drugi element se pakuje u kontejner B, dok se treći i četvrti element pakuju u kontejner C.

Ovaj pristup sa sobom nosi puno redudandnosti. Naime, sledeća dva hromozoma enkodiraju istu informaciju:

$$ABCCAC;$$
$$CBAACA.$$

2.1.2 Permutacije

Radi smanjenja redudandnosti veoma često se koriste permutacije. Neka su elementi I obeleženi rednim brojevima. Hromozom predstavlja permutaciju elemenata tako da se pakovenje dobija dekodiranjem te permutacije. Mehanizam dekodiranja za problem minimalnog pakovanje predstavlja dodavanje elemenata u kontejner sve dok je to moguće, ukoliko nije otvara se novi kontejner.

Na primer, neka je data permutacije 0123456789. Neka je tada jedno validno pakovanje

$$0123|45678|9.$$

Jasno je da je i permutacija 3210456789 enkodira isto pakovanje, tj. odgovarajuće pakovanje je oblika

$$3210|45678|9.$$

2.1.3 Grupe

Reprezentacija hromozoma predstavlja listu grupa elemenata, tj. listu kontejnera nekog dopustivog pakovanja. Na primer, razmotrimo primer pakovanja kod naivnog pristupa $ABCCAC$. Njegovo ekvivalentno pakovanje predstavljeno hromozomom grupa se enkodira kao:

$$A = \{0, 4\},\ B = \{1\},\ C = \{2, 3, 5\}.$$

Ovo možemo zapisati i kao:

$$\{0, 4\},\ \{1\},\ \{2, 3, 5\}.$$

Ovim pristupom smanjujemo stepen redudandnosti koji nastaje i kod naivnog pristupa i kod permutacija. Te će genetski algoritam implementirati grupe kao jedinke.

2.2 Operator ukrštanja

Operator ukrštanja PMX za hromozome predstavljenje pomoću permutacije ne daje smislene naslednike[3]. Zbog toga je najbolje predstaviti hrozomozome preko grupa. Uobičajeni operatori ukrštanja nije moguće

primeniti na grupe, kako pri razmenjivanju grupa dobijamo nedopustivo rešenje. Zbog toga treba osmisliti specijalizovani operator ukrštanja.

Algoritam 1 Operator ukrštanja

Require: X, Y ▷ Hromozomi roditelja
1: $(Start_X, End_X) \leftarrow \mathcal{U}^2(0, \|X\|)$
2: $(Start_Y, End_Y) \leftarrow \mathcal{U}^2(0, \|Y\|)$
3: $X \leftarrow \text{INSERT}(X, Y[Start_Y, End_Y])$
4: $Y \leftarrow \text{INSERT}(Y, X[Start_X, End_X])$
5: $X, B_X \leftarrow \text{REMOVE DUPLCATES}(X)$
6: $Y, B_Y \leftarrow \text{REMOVE DUPLCATES}(Y)$
7: $X \leftarrow \text{PACK}(X, B_X)$
8: $Y \leftarrow \text{PACK}(Y, B_Y)$

Operaciju ukrštanja izvodimo po algoritmu 1. Algoritam započinje tako što bira granice segmenata koji će se ukrštati $(Start_X, End_X)$ i $(Start_Y, End_Y)$ za hromozom X i Y, respektivno. Nakon toga, vrši se umetanje segmetna $Y[Start_Y, End_Y]$ u X, nakon pozicije End_X, dok segment $X[Start_X, End_X]$ umetamo u Y ispred poziciji $Start_Y$. Sada hromozomi u sebi sadrže duplikate, tj. elemente koji se nalaze u umetnutim segmentima zajedno sa elementima drugih kontejnera. Sledeći korak podrazumeva uklanjanje svih kontejnera koji u sebi sadrže elemente iz umetnutog segmenta. Pri uklanjanju ovih kontejnera potencijalno smo izbacili i one koji nisu bili duplikati. Njih smeštamo u skup B_X i B_Y, respektivno u odnosu na hromozome X i Y. Sada je potrebno odrediti novo pakovanje dodavanjem izbačenih elemenata u grupe iz hromozoma. Ako dodavanje nije moguće ni za jednu grupu otvaramo novu grupu i dodajemo je u hromozom.

Ovim postupkom smo opisali opšti oblik specijalizovanog operatora ukrštanja za hromozome koji su predstavljeni listom grupa. Treba još definisati na koji način vršimo ubacivanje izbačenih elemenata i tako dobijamo novo pakovanje, tj. dopustivo rešenje. Postoje mnoge efikasne heuristike koje to rade u vremenskom ograničenju $O(n \log n)$. Na primer, FFD (engl. First-Fit Descending), NFD (engl. Next-Fit Descending), MFFD (engl. Modified First-Fit Descending)[6]. Ovaj radi koristi FFD, koji sortira izbačene elemente po opadajućoj veličini i tako sortirane ubacije jedan po jedan u prvu odgovarajuću grupu u koju može da se upakuje. Ako ne može da se upakuje ni u jednu grupu, otvara se nova grupa u koju se ubacuje.

2.3 Operator mutacija

Slično, kao i kod operatora ukrštanja, treba odrediti specijalizovani operator mutacije. Jedan način definisanja operatora mutacije se sastoji u tome da iz hromozoma sa nekom verovatnoćom izbacujemo grupe. Izbačene elemente onda pakujemo nazad nekom heuristikom isto kao i kod operatora ukrštanja.

2.4 Računanje prilagodjenosti

Kako problem predstavlja minimizaciju broja potrebnih kontejnera za pakovanje, prirodno je za ocenu prilagodjenosti izabrati broj grupa u hromozomu. Problem nastaje kada dve jedinke imaju istu ocenu prilagodjenosti, tj. broj kontejnera je isti za oba pakovanja. Kako odrediti koje pakovanje je bolje od ta dva? Ovo prevazilazimo tako što merimo koliko je svaki od kontejnera dobro upakovan. Naime, dobro upakovan kontejner je onaj koji ima malo neiskorišćenog kapaciteta.

Formalno, prilagodjenost računamo kao

$$\text{fitness} = \frac{\sum_{j=1}^{N} (F_j/B)^k}{N}, \qquad (1)$$

gde je N broj grupa u hromozomu, B je kapacitet kontejnera, F_j je napunjenost kontejnera j, i $k > 1$ je predefinisani koeficijent koji pojačava odnose dobre upakovanosti. Eksperimentalo $k = 2$ daje dobre rezultate[3].

Napunjenost kontejnera j računamo kao

$$F_j = \sum_{i \in I_j} s(i), \qquad (2)$$

gde je je I_j skup elemenata iz grupe j, a $s(i)$ veličina elementa i.

Kako ovako definisana funkcija prilagođenosti predstavlja meru dobre upakovanosti, i kako se problem svodi na određivanje minimalnog broja potrebnih kutija, onda će genetski algoritam vršiti maksimizaciju funkcije prilagođenosti.

2.5 Ostali parametri genetskog algoritma

- verovatnoća mutacije: 0.66
- verovatnoča ukrštanja: 1.00
- broj jedinki ukrštanja: 2
- broj naslednika: 2
- tip zamene: roditelje menjaju naslednici
- veličina populacije: 100
- sortirana populacija: da
- kriterijum sortiranja: maksimizacija funkcije prilagođenosti
- broj selekcija: 2×50
- tip selekcije: turnirksa
- broj turnira: 5
- broj iteracija: 50
- kriterijum zaustavljanja: broj iteracija

3 Rezultati

Algoritam je pokrenut na računaru sa sledećim specifikacijama: CPU — Intel Core i5–2520M @ $4 \times 3.2GHz$, RAM — 7836MiB, Kernel — x86_64 Linux 5.16.15-arch1–1, g++ — (GCC) 11.2.0.

Za ekspreimentalno testiranje koristimo ručno napravljene lake instance iz grupe EZ. Za dalje testiranje koristmo N1, N2 i H instance i njihove optimalne vrednosti koje su dobijene pomoću tačnog algoritma BISON[9].

3.1 EZ instance

Hibridni genetski algoritam grupa

Instanca	Rezultat	Vreme(ms)
EZ0	3	0
EZ1	3	1
EZ2	3	0
EZ3	2	0
EZ4	2	0
EZ5	3	0
EZ6	3	0
EZ7	2	0
EZ8	5	1
EZ9	2	0

Algoritam grube sile

Instanca	Rezultat	Vreme(ms)
EZ0	3	0
EZ1	3	0
EZ2	3	0
EZ3	2	1
EZ4	2	1
EZ5	3	0
EZ6	3	1
EZ7	2	24
EZ8	5	1
EZ9	2	323

Primećujemo da hibridni genetski algoritam grupa daje ista rešenja kao i algoritam grube sile. Takođe vidimo da sa malim povećanjem broja elemenata u instanci EZ9 vreme izvršavanja algoritma grube sile drastično povećava u odnosu na hibridni genetski algoritma grupa koji ima slično vreme izvršavanja. U ostalim instancama, zbog toga, ne pokrećemo algoritam grube sile, kako se neće završiti u realnom vremenu.

3.2 N1 instance

Neke N1 instanci

Instanca	Rez.	Opt.	$t(ms)$
N1C1W1A	31	25	4
N1C1W1B	35	31	8
N1C1W1C	26	20	5
N1C1W1D	31	28	7
N1C1W1E	31	26	7
N1C1W1F	31	27	5
N1C1W1G	30	25	6
N1C1W1H	36	31	7
N1C1W1I	28	25	6
N1C1W1J	30	26	7
N1C1W1K	30	26	6
N1C1W1L	38	33	7
N1C1W1M	34	30	8
N1C1W1N	30	25	6
N1C1W1O	38	32	8
N1C1W1P	31	26	4
N1C1W1Q	34	28	8
N1C1W1R	31	25	6
N1C1W1S	34	28	5
N1C1W1T	35	28	8

3.3 N2 instance

Neke N2 instanci

Instanca	Rez.	Opt.	$t(ms)$
N1W1B1R0	21	18	3
N1W1B1R1	21	18	3
N1W1B1R2	21	19	3
N1W1B1R3	21	18	3
N1W1B1R4	19	17	3
N1W1B1R5	18	17	3
N1W1B1R6	20	17	3
N1W1B1R7	19	17	5
N1W1B1R8	20	18	4
N1W1B1R9	19	17	4

3.4 H instance

Teške instance

Instanca	Rez.	Opt.	$t(ms)$
HARD0	66	56	13
HARD1	66	57	15
HARD2	66	56	14
HARD3	66	55	14
HARD4	65	57	11
HARD5	66	56	14
HARD6	66	57	11
HARD7	64	55	13
HARD8	66	57	15
HARD9	66	56	12

4 Zaključak

Hibridni genetski algoritam grupa primenjen na jednodimenzioni problem minimalnog pakovanja daje zadovoljavajuće dobra rešenja u kratkom vremenskom periodu, gde broj elemenata nije od ključanog faktor za vremensku efikasnost. Zbog toga se može koristiti kao gornja granica za heuristike koje rešavaju problem minimalnog pakovanja ili neki njegov dualni problem.

Dalji rad podrazumeva eksperimentalno testiranje drugih heuristika pri ubacivanju izbačenih elemenata u grupe kod operatora ukrštanja i operatora mutacije. Takođe, dalji rad može da uključi i podešavanje parametra genetskog algoritma.

Literatura

[1] Ronald V Book. *Michael R. Garey and David S. Johnson, Computers and intractability: A guide to the theory of NP-completeness*. Sv. 3. 2. American Mathematical Society, 1980., str. 898–904.

[2] Pierluigi Crescenzi i dr. „Structure in approximation classes". U: *SIAM Journal on Computing* 28.5 (1999.), str. 1759–1782.

[3] Emanuel Falkenauer. „A hybrid grouping genetic algorithm for bin packing". U: *Journal of heuristics* 2.1 (1996.), str. 5–30.

[4] Rebecca Hoberg i Thomas Rothvoss. „A logarithmic additive integrality gap for bin packing". U: *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM. 2017., str. 2616–2625.

[5] David S Johnson i Michael R Garey. „A 7160 theorem for bin packing". U: *Journal of complexity* 1.1 (1985.), str. 65–106.

[6] E.G. Coffman man Jr, M.R. Garey i D.S. Johnson. „Approximation algorithms for bin packing: A survey". U: *Approximation algorithms for NP-hard problems* (1996.), str. 46–93.

[7] Narendra Karmarkar i Richard M Karp. „An efficient approximation scheme for the one-dimensional bin-packing problem". U: *23rd Annual Symposium on Foundations of Computer Science (sfcs 1982)*. IEEE. 1982., str. 312–320.

[8] Thomas Rothvoß. „Approximating Bin Packing within $O(\log OPT * \log\log OPT)$ Bins". U: *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*. 2013., str. 20–29. DOI: 10.1109/FOCS.2013.11.

[9] Armin Scholl, Robert Klein i Christian Jürgens. „Bison: A fast hybrid procedure for exactly solving the one-dimensional bin packing problem". U: *Computers & Operations Research* 24.7 (1997.), str. 627–645.

[10] David Simchi-Levi. „New worst-case results for the bin-packing problem". U: *Naval Research Logistics (NRL)* 41.4 (1994.), str. 579–585.

[11] Minyi Yue i Lei Zhang. „A simple proof of the inequality $MFFD(L) \leq \frac{71}{60}OPT(L) + 1, \forall L$ for the MFFD bin-packing algorithm". U: *Acta Mathematicae Applicatae Sinica* 11.3 (1995.), str. 318–330.