Optimizacija problema sečenja

Veljko Prodan Maja Milenković

1 Uvod

Ovaj rad se bavi optimizacijom problema sečenja uz pomoć simuliranog kaljenja i genetskog algoritma. Za testiranje algoritama korišćen je 21 test primer, svaki različite dužine. Test primeri do određene dužine su testirani CPLEX rešavačem radi poređenja sa genetskim algoritmom i simuliranim kaljenjem, kao i kombinacijom ova dva algoritma.

2 Problem sečenja

Problem sečenja (Cutting stock problem - CSP) je NP težak problem u kojem je cilj naći najbolji način da se materijal određenih dimenzija iseče na manje komade zadatih dimenzija, tako da se minimizuje otpad i poveća iskorišćenost materijala. Ovaj problem se često javlja u industriji papira, čelika, drveta, tekstila i sl. Rešavanje ovog problema je složeno zbog velikog broja mogućih kombinacija isecanja narudžbina.

U ovom radu, za dužinu jednog komada materijala je korišćena fiksna vrednost od 100 cm, a rešenje problema je predstavljeno kombinacijom zadatih celih brojeva između 1 i 100. Vrednost jednog rešenja se računa tako što se lista otpadaka jedne narudžbine sortira u opadajućem poretku pa se zatim dužina svakog otpatka pomnoži sa svojom pozicijom u sortiranoj listi. Zbir dobijenih vrednosti predstavlja vrednost datog rešenja.

Item list	1	2	3	4	5	Stock no.
	12	12	12	12	12	Initial stock length
5	7					Sum of trim loss
5	2					
5	2	7				
5	2	2				
3	2	2	9			
3	2	2	6			
3	2	2	3			
3	2	2	0			
2	0	2	0			
2	0	0	0			
2	0	0	0	10		
2	0	0	0	8		
2	0	0	0	6		
2	0	0	0	4		
2	0	0	0	2		
2	0	0	0	0		
2	0	0	0	0	10	
2	0	0	0	0	8	
2	0	0	0	0	6	
Trim loss	**0**	**0**	**0**	**0**	**6m**	**6m**

Stock#	1	2	3	4	5
VC	1	2	3	4	5
w_j	2	2	2	0	0

$TVC = 2*1+2*2+2*3+0*4+0*5= 12$

Stock#	1	2	3	4	5
VC	1	2	3	4	5
w_j	6	0	0	0	0

$TVC = 6*1+0*2+0*3+0*4+0*5= 6$

Računanje vrednosti rešenja

3 Simulirano kaljenje

Simulirano kaljenje je metaheuristički algoritam za rešavanje optimizacionih problema, inspirisan procesom kaljenja metala u metalurgiji. Cilj algoritma je pronaći globalni optimum funkcije, izbegavajući zaglavljivanje u lokalnim optimumima.

Algoritam započinje od početnog rešenja i iterativno ga poboljšava. U svakoj iteraciji, trenutno rešenje se modifikuje i evaluira se ciljna funkcija. Ako je novo rešenje bolje, ono se prihvata. Međutim, ako je novo rešenje lošije, ono se prihvata sa određenom verovatnoćom, koja opada tokom izvršavanja algoritma. Ova verovatnoća prihvatanja lošijih rešenja omogućava algoritmu da izađe iz lokalnih optimuma i istraži širi prostor rešenja.

3.1 Parametri i funkcije

U ovom radu, za implementaciju simuliranog kaljenja korišćeni su sledeći parametri:

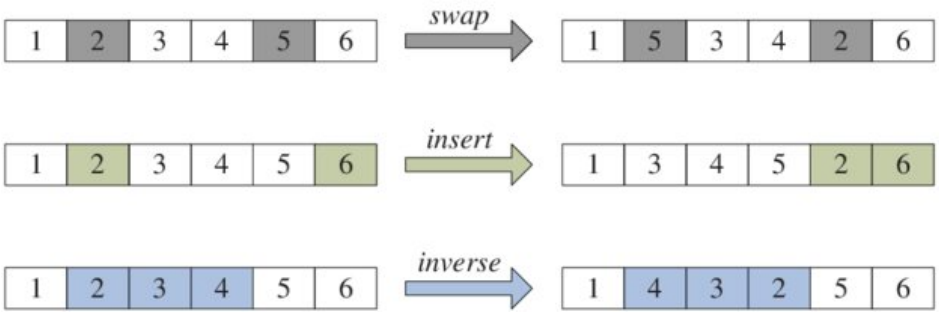
1. Početna temperatura
2. Minimalna temperatura
3. Alfa: koristi se za izračunavanje sledeće vrednosti temperature. Ima vrednost između 0 i 1
4. Maksimalan broj iteracija
5. Maksimalan broj prihvaćenih rešenja tokom trenutne temperature

Takođe su testirane dve različite funkcije za izračunavanje naredne vrednosti temperature:

1. *calc_temperature_1*: $[T_{next} = \frac{T_{current}}{1+\alpha \cdot T_{current}}]$, $[\alpha = \frac{T_0 - T_{min}}{N \cdot T_0 \cdot T_{min}}]$
2. *calc_temperature_2*: $[T_{next} = \alpha \cdot T_{current}]$, α je odabrana fiksna vrednost

Testirane su i tri funkcije za kreiranje novog rešenja:

1. *Swap*
2. *Insert*
3. *Inverse*



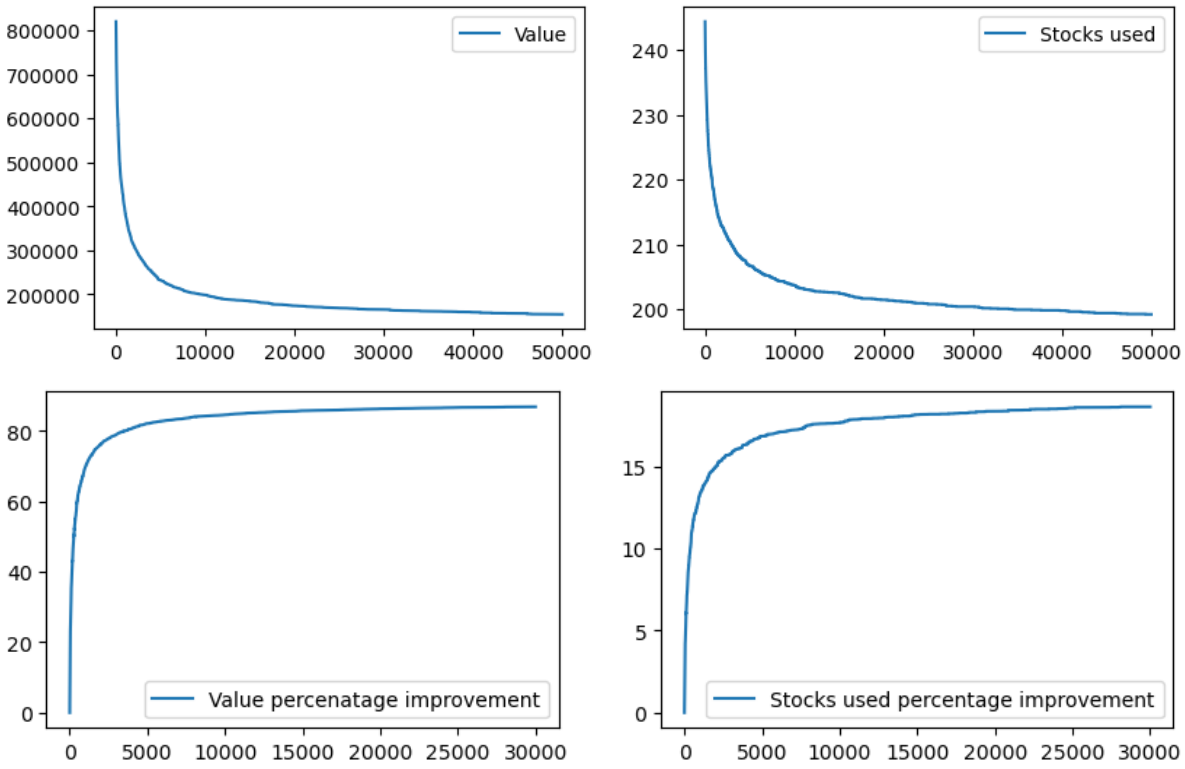
3.2 Testiranje i odabir parametara i funkcija

Za biranje početne temperature, minimalne temperature, i maksimalan broj prihvaćenih rešenja po temperaturi pokrenut je *grid search* nad svim test primerima. Izabrana početna temperatura je 100, minimalna temperatura 1, a maksimalan broj prihvaćenih rešenja po temperaturi 2.

Zatim su testirane dve funkcije za smanjivanje temperature. Za prvu varijantu se alfa računa po formuli, a za drugu varijantu su testirane sledeće vrednosti: 0.95, 0.98, 0.99, 0.999. Za svaku kombinaciju odabrane funkcije i vrednosti alfa je algoritam pokrenut po 5 puta nad svim test primerima. Bolje rezultate je ostvarila prva funkcija.

Na isti način su testirane i funkcije za kreiranje novog rešenja: za svaku funkciju su po 5 puta pokrenuti svi test primeri. Funkcija *swap* je imala najbolje rezultate.

Radi odabira broja iteracija je algoritam pušten nad svim test primerima, i izračunata je prosečna vrednost rešenja po iteracijama, kao i prosečno procentualno poboljšanje rešenja po iteracijama u odnosu na početno rešenje.



4 Genetski algoritam

Genetski algoritam je optimizaciona tehnika inspirisana principima prirodne selekcije i genetike. Pripada široj klasi evolutivnih algoritama, koji koriste mehanizme poput mutacije, ukrštanja i selekcije kako bi iterativno poboljšali rešenja problema.

Osnovni koncept genetskih algoritama je da se potencijalna rešenja problema kodiraju kao hromozomi ili genomi, a zatim se iterativno primenjuju genetski operatori kako bi se stvorile nove generacije rešenja. Svaka generacija se evaluira na osnovu funkcije prilagođenosti (*fitness* funkcije), koja meri kvalitet svakog rešenja u odnosu na definisani problem.

Proces započinje inicijalizacijom početne populacije slučajno generisanih rešenja. Zatim se kroz više iteracija primenjuju funkcije selekcije, ukštanja i mutacije, postepeno evoluirajući populaciju ka boljim rešenjima.

4.1 Parametri i funkcije

Za implementaciju genetskog algoritma korišćeni su sledeći parametri:

1. Veličina populacije
2. Elitizam
3. Mutacija i ukrštanje
    - DHM/ILC (Decreasing High Mutation Rate (DHM) and an Increasing Low Crossover Rate (ILC))
    - ILM/DHC (Increasing Low Mutation Rate (ILM) and a Decreasing High Crossover Rate (DHC))
    - Fiksna verovatnoća mutacije i ukrštanja
4. Broj iteracija
5. Veličina turnira (turnirska selekcija)
6. Selekcioni pritisak (rangovska selekcija)

Testirane su tri funkcije selekcije:

1. Turnirska selekcija
2. Ruletska selekcija
3. Rangovska selekcija

Testirane su tri funkcije ukrštanja, ali zbog toga što nisu direktno primenjive na originalno rešenje, bilo je potrebno prvo modifikovati ih.

solution	1	2	2	3	1	2	3	1	3
index of occurrence	1	1	2	1	2	3	2	3	3

Tri funkcije ukrštanja:

1. *Partially mapped crossover*

parent 1	3	2	2	2	3	1	1	1	3
parent 2	~~1~~	1	3	2	~~2~~	1	~~2~~	~~3~~	3
GPMX offspring	1	3	2	2	3	1	2	1	3

2. *Generalized order crossover*

parent 1	3	2	2	2	3	1	1	1	3
parent 2	~~1~~	1	3	2	~~2~~	1	~~2~~	~~3~~	3
GOX offspring	1	3	2	2	2	3	1	1	3

3. *Precedence preservative crossover*

parent 1	3	2	2	2	3	1	1	1	3
parent 2	1	1	3	2	2	1	2	3	3
gene of parent	1	1	2	2	2	2	1	1	1
PPX offspring	3	2	1	1	2	1	2	3	3

Četiri funkcije mutacije:

1. *Swap*
2. *Insert*
3. *Inverse*
4. *Shuffle*

4.2 Testiranje i odabir parametara i funkcija

Zbog dugačkog vremena izvršavanja, odabrana je fiksna veličina populacije od 100 jedinki.

Prvo je pokrenut *random search* nad svim test primerima, sa 150 kombinacija parametara koji se tiču elitizma, ukrštanja, i mutacije. Pristup sa fiksnom verovatnoćom mutacije i ukrštanja se pokazao bolje od druga dva pristupa, dok za elitizam nije mogao da se izvede definitivan zaključak.

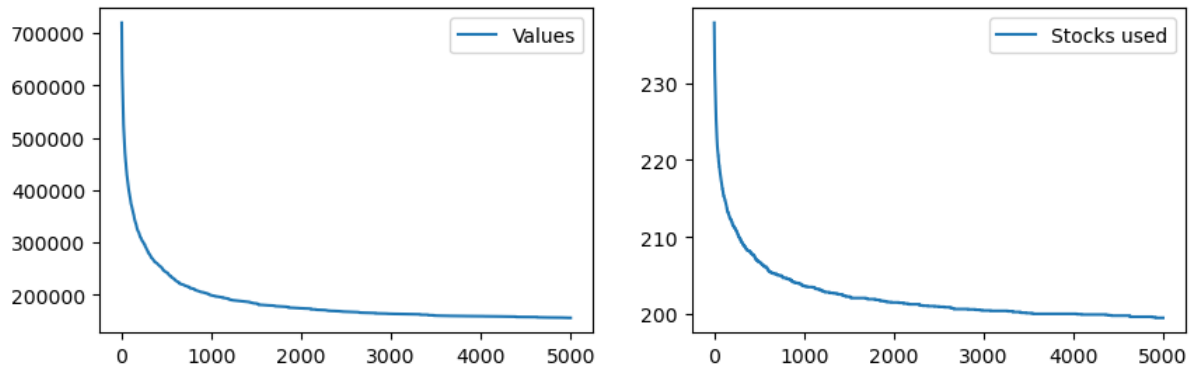
Za verovatnoću mutacije su testirane vrednosti od 0.03 do 0.08. Za svaku verovatnoću su pokrenuti svi test primeri po 3 puta. Najbolji rezultati su ostvareni sa verovatnoćom 0.07.

Za verovatnoću ukrštanja su testirane vrednosti od 0.6 do 1, i za svaku verovatnoću su takođe pokrenuti svi test primeri po 3 puta. Vrednost 0.85 je ostvarila najbolje rezultate.

Radi biranja najboljih funkcija selekcije, ukrštanja i mutacije pokrenut je *grid search*, gde je ispitana svaka kombinacija ovih funkcija nad svim test primerima. Sa rezultatima su se istakle kombinacije sa turnirskom selekcijom i *swap* funkcijom mutacije, dok za funkciju ukrštanja nije moglo da se dođe do definitivnog zaključka: *precedence preservative crossover* je imao najbrže vreme izvršavanja, ali najgore rezultate, dok između ostale dve funkcije nije mogla da se primeti značajna razlika u rezultatima. Zato je za obe funkcije algoritam pokrenut nad svim test primerima po 3 puta. Ovog puta je *generalized order crossover* ostvario najbolje rezultate.

Za elitizam su testirane veličine od 0 do 10% od ukupnog broja populacije, gde je veličina od 4% ostvarila najbolje rezultate. Takođe su testirane različite veličine turnira za selekciju. Odabrana je veličina od 6 jedinki po turniru.

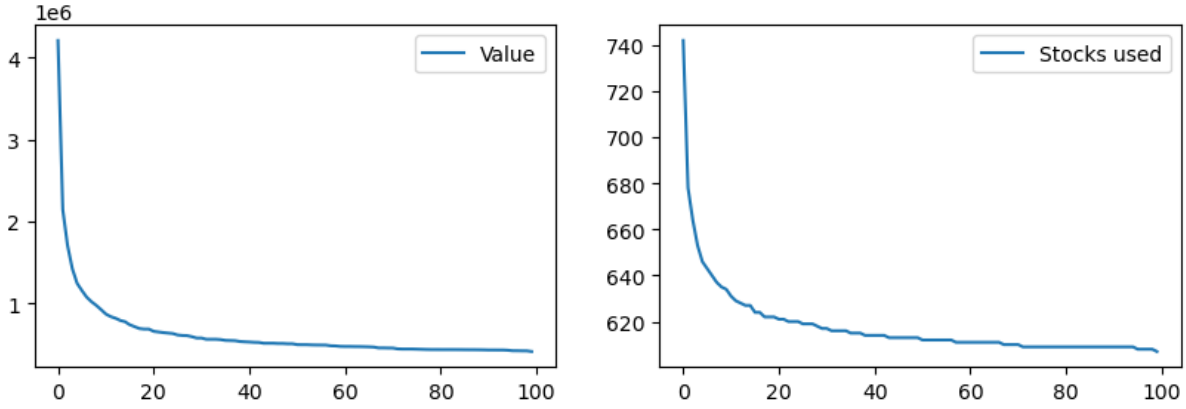
Radi odabira broja iteracija genetski algoritam je pokrenut nad svim test primerima, i izračunata je prosečna vrednost najboljeg rešenja populacije, po svakoj iteraciji. Konačno odabran broj iteracija je 600.



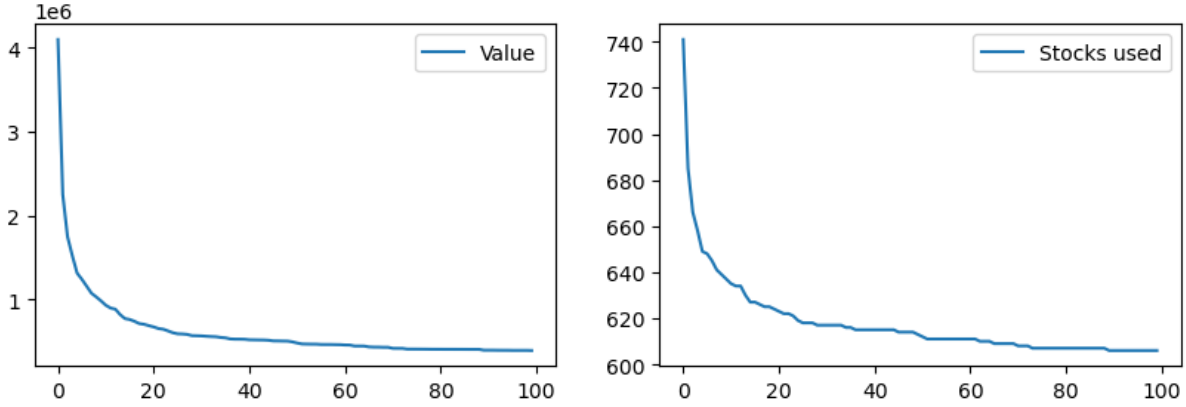
5 Kombinacija genetskog algoritma i simuliranog kaljenja

Testirane su tri varijante kombinacije genetskog algoritma i simuliranog kaljenja. U prvoj varijanti se simulirano kaljenje pokreće nad svakom jedinkom populacije u svakoj

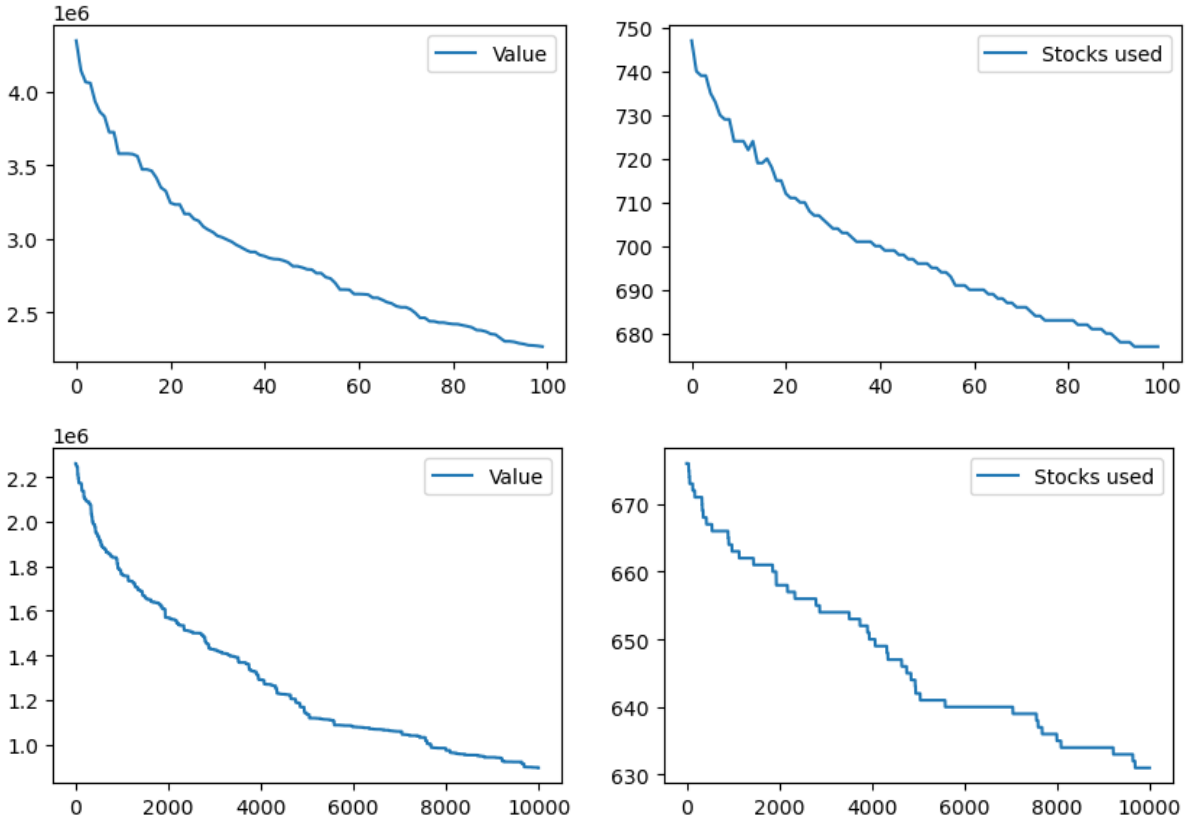
iteraciji tokom genetskog algoritma.



U drugoj varijanti se simulirano kaljenje pokreće samo nad najboljom jedinkom populacije u svakoj iteraciji.



U trećoj varijanti je simulirano kaljenje pokrenuto nad konačnim rešenjem genetskog algoritma.

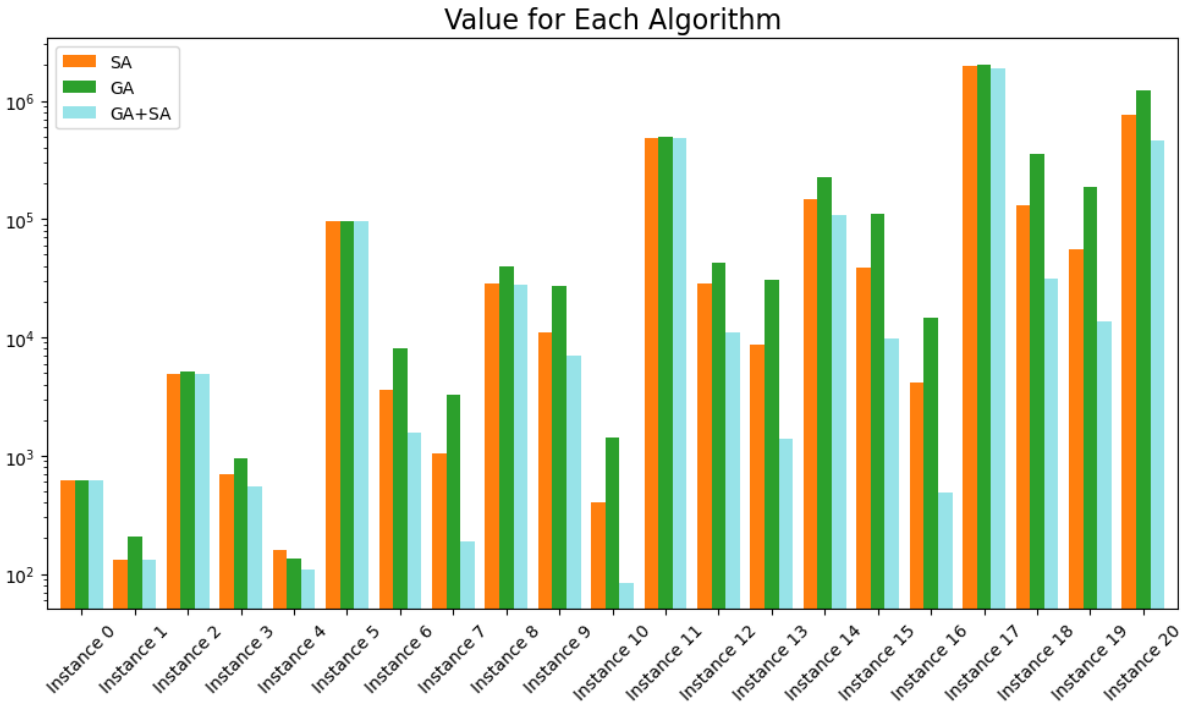


Prva i druga varijanta su ostvarile slične rezultate, značajno bolje od treće. Vreme izvršavanja druge varijante je neuporedivo manja od vremena izvršavanja prve varijante.

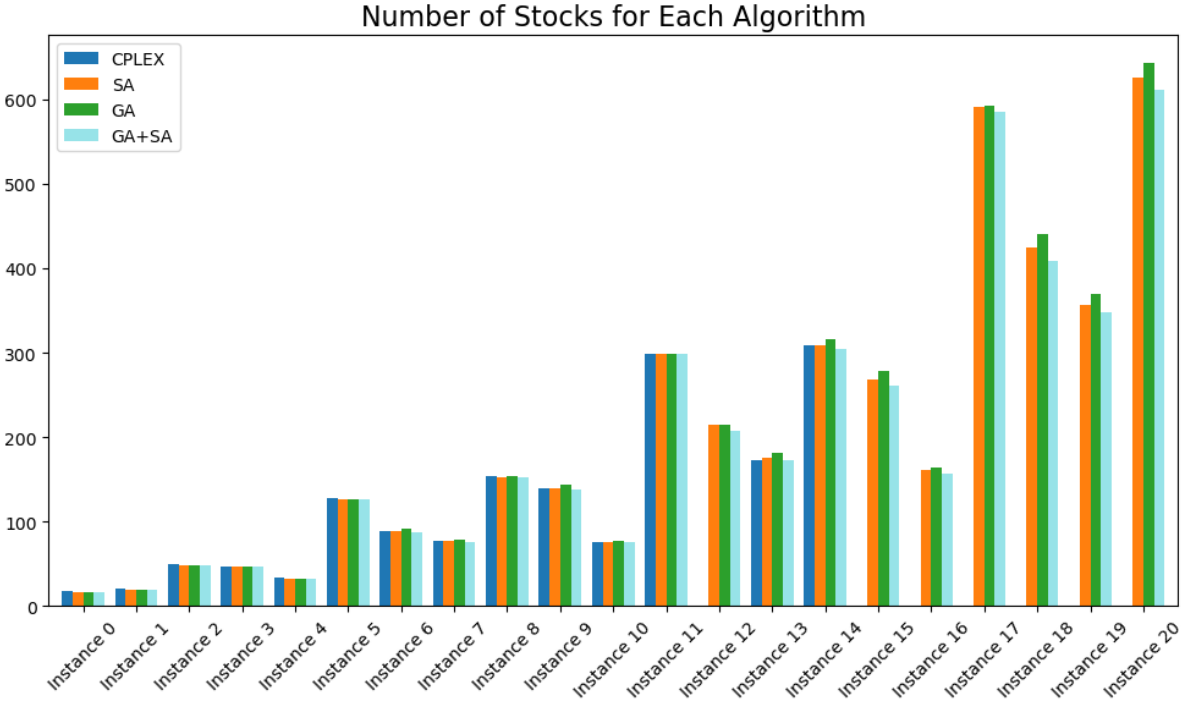
6 Poređenje rezultata

Poređeni su rezultati CPLEX rešavača, simuliranog kaljenja, genetskog algoritma, i druge varijante kombinacije genetskog algoritma i simuliranog kaljenja. Izlaz CPLEX rešavača nije vrednost rešenja kao kod ostalih algoritama, nego broj iskorišćenih komada materijala. Pokrenut je nad prvih 15 test primera, jer je na ostalim previše dugo vreme izvršavanja, a ostali algoritmi su pokrenuti nad svim test primerima.

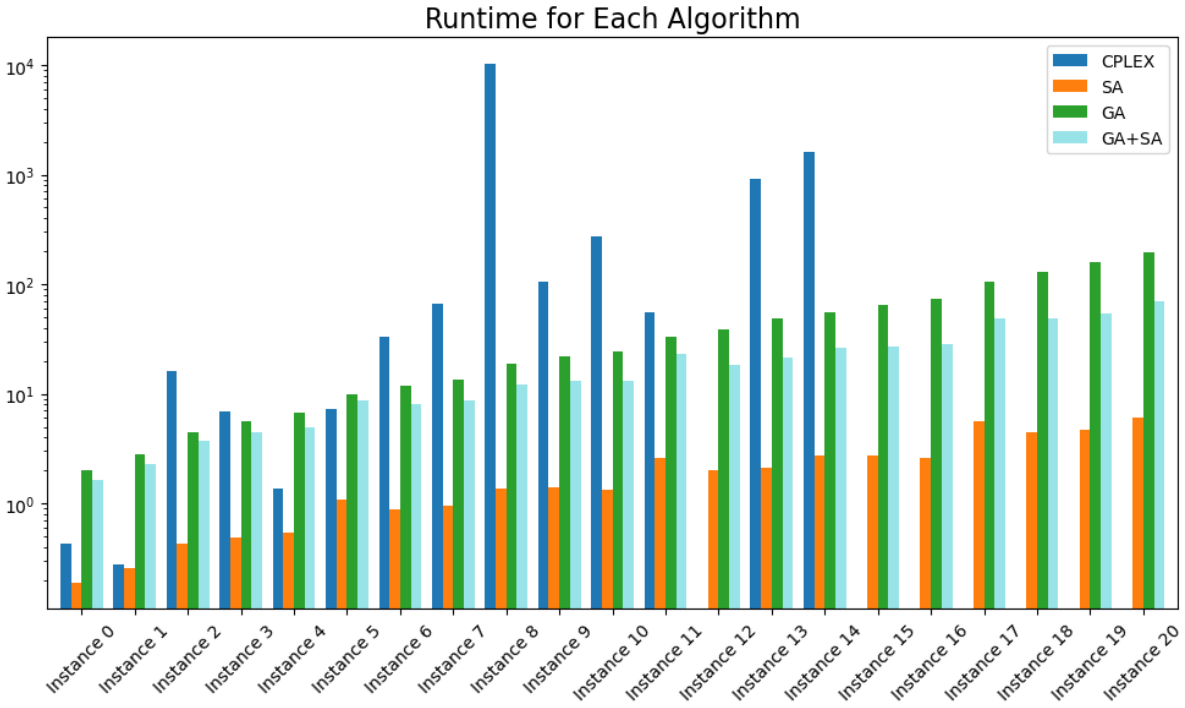
Na svim test primerima je najbolju vrednost rešenja imala kombinacija algoritama, gde je simulirano kaljenje imalo jednak rezultat na dva test primera, i genetski algoritam na jednom.



Za broj iskorišćenih komada materijala je takođe kombinacija algoritama imala najbolje rešenje, gde je CPLEX imao jednak rezultat na 2 test primera, simulirano kaljenje na 8, i genetski algoritam na 6.



Na svim test primerima je najmanje vreme izvršavanja imalo simulirano kaljenje.



	CPLEX	SA	GA	GA+SA
Number of lowest values	0	2	1	21
Number of lowest num stocks	2	8	6	21
Number of lowest runtimes	0	21	0	0