

Minimum Array Partition Optimization

Daniil Grbić, Ivan Gogić

September 29, 2023

Opis problema

Data je matrica dimenzija $n \times n$ nenegativnih celih brojeva, i broj $p \in \mathbb{N}$.

Potrebno je izabrati $p - 1$ vertikalnih i $p - 1$ horizontalnih pregrada koje partitionišu matricu na p^2 blokova tako minimizujemo sledeći izraz:

$$\max_{\substack{1 \leq i \leq p \\ 1 \leq j \leq p}} \sum_{\substack{v_{i-1} \leq x \leq v_i \\ h_{j-1} \leq y \leq h_j}} A[x, y]$$

Odnosno treba da minimizujemo najveću među sumama elemenata tih blokova.

Brute Force (BF)

Algoritam grube sile nam je potreban kako bi utvrdili ispravnost optimizacionih metoda na malim primerima.

Gledamo sve moguće particije (ima ih $\binom{n}{p}^2$), a sume blokova efikasno računamo koristeći matricu parcijalnih suma (dinamičko programiranje).

Ubrzanje 3 do 5 puta sečenjem loših rešenja posmatrajući sume proverenih blokova po Dirihelevom principu.

Genetski Algoritam

Genetski algoritam obećava dobre rezultate, ako možemo da smislimo kako da vršimo ukrštanje i mutaciju.

Van tih funkcija koristimo "školski" genetski algoritam.

Mutacija je jednostavna: za svaku pregradu u rešenju postoji verovatnoća da se u toku generacije pomeri levo ili desno ukoliko je to mesto prazno.

Genetski Algoritam - Ukrstanje

Koristimo aritmetički pristup:

roditelj A : 1 3 9

roditelj B : 2 5 7

dete 1 : 1 4 8

dete 2 : 2 4 8

roditelj A : 1 2 3

roditelj B : 2 3 4

dete 1 : 1 2 3

dete 2 : 2 2 4

...

...

Nastaje problem kada se pregrade preklapaju - to rešavamo tako što problematične prepreke premestimo na slučajno izabrana slobodna mesta. To garantuje korektnost i pored toga poboljšava rešenje.

Simulirano kaljenje i VNS

Klasični algoritmi kaljenja i VNS.

Kod algoritma kaljenja, male promene postižemo slučajnim pomeranjem pregrada, čime i istražujemo prostor rešenja.

Kod VNS, promene postižemo slučajnim pomeranjem više pregrada odjednom.

Rezultati - Generisanje Testova

Testove generišemo po različitim pravilima, naime:

Random popunjava matricu brojevima iz $(1, n/2)$

Linear gde je $M_{i,j} = i + j + 1$

Squared gde je $M_{i,j} = i^2 + j^2 + 1$

Za svaki tip testa generišemo po 4 konkretna test primera, sledećih dimenzija:

Tiny $n = 50, p = 3$

Small $n = 80, p = 6$

Medium $n = 100, p = 4$

Large $n = 500, p = 10$

Svaki test primer propuštamo kroz svaki algoritam po 5 puta, i beležimo najbolje, najgore, i srednje rezultate i vremena izvršavanja.

Rezultati - Random testovi

Testcase: tests/random_n100_p4.in

GENETIC ALGORITHM (GA) [pop_size=100, num_iters=600, elitism_size=20]

Results : BEST=17061	WORST=17061	AVG=17061.00
Time (s) : BEST=5.06	WORST=5.22	AVG=5.11

SIMULATED ANNEALING (SA)

Results : BEST=17061	WORST=17061	AVG=17061.00
Time (s) : BEST=0.68	WORST=0.68	AVG=0.68

VARIABLE NEIGHBOURHOOD SEARCH (VNS) [num_iters=10000]

Results : BEST=17061	WORST=17061	AVG=17061.00
Time (s) : BEST=4.75	WORST=4.85	AVG=4.80

Testcase: tests/random_n500_p10.in

GENETIC ALGORITHM (GA) [pop_size=100, num_iters=600, elitism_size=20]

Results : BEST=326181	WORST=338514	AVG=330243.80
Time (s) : BEST=24.44	WORST=25.95	AVG=24.95

SIMULATED ANNEALING (SA)

Results : BEST=360022	WORST=552450	AVG=424930.40
Time (s) : BEST=9.21	WORST=10.39	AVG=9.89

VARIABLE NEIGHBOURHOOD SEARCH (VNS) [num_iters=10000]

Results : BEST=323192	WORST=417608	AVG=362644.00
Time (s) : BEST=24.83	WORST=28.67	AVG=26.29

Rezultati - Linear testovi

```
Testcase: tests/linear_n100_p4.in
  GENETIC ALGORITHM (GA) [pop_size=100, num_iters=600, elitism_size=20]
    Results : BEST=67760      WORST=68355      AVG=67879.00
    Time (s) : BEST=5.16      WORST=5.34      AVG=5.23
  SIMULATED ANNEALING (SA)
    Results : BEST=67760      WORST=68355      AVG=67879.00
    Time (s) : BEST=0.71      WORST=0.71      AVG=0.71
  VARIABLE NEIGHBOURHOOD SEARCH (VNS) [num_iters=10000]
    Results : BEST=67760      WORST=72500      AVG=69065.00
    Time (s) : BEST=4.03      WORST=4.87      AVG=4.38

Testcase: tests/linear_n500_p10.in
  GENETIC ALGORITHM (GA) [pop_size=100, num_iters=600, elitism_size=20]
    Results : BEST=1371552     WORST=1621573     AVG=1454183.80
    Time (s) : BEST=23.79      WORST=26.55      AVG=24.81
  SIMULATED ANNEALING (SA)
    Results : BEST=1383025     WORST=2073344     AVG=1565182.40
    Time (s) : BEST=9.15       WORST=11.14       AVG=9.76
  VARIABLE NEIGHBOURHOOD SEARCH (VNS) [num_iters=10000]
    Results : BEST=1378620     WORST=2168270     AVG=1566251.40
    Time (s) : BEST=21.42       WORST=29.76       AVG=25.80
```

Rezultati - Squared testovi

Testcase: tests/squared_n100_p4.in

GENETIC ALGORITHM (GA) [pop_size=100, num_iters=600, elitism_size=20]

Results : BEST=4808345	WORST=4808345	AVG=4808345.00
Time (s) : BEST=5.10	WORST=5.16	AVG=5.14

SIMULATED ANNEALING (SA)

Results : BEST=4807152	WORST=4807152	AVG=4807152.00
Time (s) : BEST=0.65	WORST=0.70	AVG=0.67

VARIABLE NEIGHBOURHOOD SEARCH (VNS) [num_iters=10000]

Results : BEST=4808345	WORST=5033479	AVG=4853371.80
Time (s) : BEST=4.16	WORST=4.76	AVG=4.61

Testcase: tests/squared_n500_p10.in

GENETIC ALGORITHM (GA) [pop_size=100, num_iters=600, elitism_size=20]

Results : BEST=522518392	WORST=619537980	AVG=571390189.00
Time (s) : BEST=23.39	WORST=26.13	AVG=25.28

SIMULATED ANNEALING (SA)

Results : BEST=533205270	WORST=1038957839	AVG=680165623.40
Time (s) : BEST=9.39	WORST=11.87	AVG=10.50

VARIABLE NEIGHBOURHOOD SEARCH (VNS) [num_iters=10000]

Results : BEST=551863221	WORST=851774417	AVG=660668487.00
Time (s) : BEST=19.00	WORST=25.25	AVG=22.11

Zakljucak

Pokazali smo da je moguće uspešno koristiti optimizacione algoritme za rešavanje Minimum Array Partition problema.

Genetiski algoritam daje najkonzistentnije rezultate, VNS je tu odmah iza njega.

Simulirano kaljenje je dobra alternativa kada možemo da prihvatimo neoptimalno rešenje, a sprečava nas vreme.