Pakovanje konveksnih poligona u pravougaonik minimalne površine

Seminarski rad u okviru kursa
Geometrijski algoritmi
Matematički fakultet

Nikola Belaković 1023/2023

28. januar 2024.

Sažetak

Ovaj rad istražuje problem pakovanja konveksnih poligona u pravougaonik minimalne površine uz dozvoljenu translaciju, ali ne i rotaciju. Cilj je pronaći optimalno pozicioniranje i dimenzije pravougaonika koji će minimalno obuhvatiti dati konveksni poligon. Ovaj problem ima široku primenu u oblastima kao što su računarski vid, obrada slika, robotika i optimizacija rasporeda. Rad istražuje jedan aproksimativni algoritam koji se koristi za rešavanje problema pakovanja, fokusirajući se na specifične zahteve ovog problema. Rezultati istraživanja pokazuju efikasnost odabranog pristupa u rešavanju problema pakovanja konveksnih poligona u pravougaonik minimalne površine, nudeći perspektivu za dalja poboljšanja i primene u praksi.

Sadržaj

1 Uvod

U radu "Improved Approximations for Translational Packing of Convex Polygons" autora Adam Kurpisz i Silvan Suter predstavljeni su revolucionarni algoritmi za dvodimenzionalno pakovanje konveksnih poligona bez rotacije. Oni su proučavali različite zahteve, u zavisnosti od tipa kontejnera. Jedan od problema koji su rešavali je i problem pakovanja konveksnih poligona u pravougaonik minimalne površine.

Mnoge situacije iz realnog života zahtevaju donošenje odluka o optimalnom pakovanju kolekcije objekata u određeni prostor. Jedna posebna kategorija tih problema je dvodimenzionalno pakovanje, koje se susreće u svakodnevnim scenarijama poput raspoređivanja predmeta na polici i u industrijskim primenama poput rezanja kolačića iz razvijenog testa ili proizvodnje setova pločica iz panela standardnih veličina od drva, stakla ili metala. Još jedan intrigantan primer uključuje rezanje komada tkanine za proizvodnju odeće. Široka primenjivost problema dvodimenzionalnog pakovanja dovela je do porasta interesa za dizajniranje učinkovitih aproksimativnih algoritama za njihovo rešavanje [2].

Implementacija aproksimativnog algoritma koji je predstavljen u radu "Improved Approximations for Translational Packing of Convex Polygons", kao i analiza njegove efikasnosti, predstavljaju važan korak ka razumevanju novog pristupa u rešavanju problem pakovanja i oni će biti prikazani u daljem tekstu.

2 Opis problema

U ovom delu, definišemo problem pakovanja konveksnih poligona u pravougaonik minimalne površine. Prvo ćemo precizirati šta podrazumevamo pod konveksnim poligonima, translacijom i ostalim relevantnim pojmovima.

Konveksni poligoni: Konveksni poligon je geometrijski oblik za koji važi da za svaku njegovu stranicu sve njegove tačke su sa iste njene strane. Drugim rečima, poligon je konveksan ako su mu svi unutrašnji uglovi manji od opruženog [1].

Translacija: Translacija je osnovna operacija u geometriji koja podrazumeva pomeranje geometrijskog oblika (u ovom slučaju konveksnog poligona) duž pravca, bez rotacije ili promene oblika. Translacija se vrši tako da se svaka tačka poligona pomeri za određenu udaljenost i u određenom smeru.

Dati su skupovi konveksnih poligona, svaki određen nizom koordinata svojih tačaka u rasponu od 0 do 1 duž obe ose. Cilj je pronaći najmanji mogući pravougaonik u koji se mogu smestiti svi poligoni, uz ograničenje da se poligoni ne smeju rotirati već samo translirati. Ovo pitanje ima široku primenu u različitim kontekstima, uključujući organizaciju objekata na polici, rezanje materijala ili proizvodnju setova pločica.

Ovaj problem ima značajne izazove i kompleksnosti zbog specifičnih zahteva za pakovanjem konveksnih poligona bez rotacije, što zahteva razvoj efikasnih algoritama za postizanje optimalnih ili približnih rešenja.

3 Opis algoritma i analiza složenosti

U ovom odeljku opisujemo algoritam za minimizaciju površine pravougaonika, koji pruža efikasnu aproksimaciju sa faktorom 9.45.

3.1 Opis algoritma

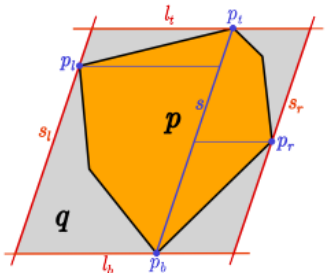
Pre nego što detaljno opišemo algoritam, definišemo nekoliko ključnih pojmova i lema koje će biti korisne:

- **Konveksni poligon** p: Skup tačaka $p \subseteq [0,1]^2$ u ravni.
- **Kičma** s **poligona** p: Prava linija koja povezuje tačku sa najmanjom y-koordinatom i tačku sa najvećom y-koordinatom u poligonu p. Ugao između x-ose u pravcu povećanja i s nazivamo ugao od s . Kažemo da je ugao od s nagnut udesno ako je taj ugao u opsegu $(0, \frac{\pi}{2}]$ i nagnut ulevo ako je u opsegu $[\frac{\pi}{2}, \pi]$.
- **X-paralelogram** q: Paralelogram u ravni sa dve stranice paralelne sa x-osom. Od te dve stranice, onu sa nižom y-koordinatom nazivamo bazom q i pišemo $base(q)$. Širinu jedne od stranica q koja nije paralelna sa x-osom označavamo sa $wside(q)$. Nagnutost x-paralelograma se definiše kao i nagnutost kičme poligona.

Lema: Za svaki konveksni poligon p postoji x-paralelogram q koji ga sadrži i zadovoljava sledeće uslove:

1. $height(q) = height(p)$
2. $base(q) \leq width(p)$
3. $wside(q) \leq width(p)$
4. $area(q) \leq 2 \cdot area(p)$

Dokaz: Prvo, konstruišemo omeđujući x-paralelogram kao onaj koji se može videti na slici 1. Neka su l_b i l_t linije paralelne sa x-osom i tangente na p, dodirujući dno p i vrh p redom. Izaberimo $p_b \in p \cap l_b$ i $p_t \in p \cap l_t$ i definišemo s kao liniju koja povezuje p_b i p_t. Napomenimo da je s kičma p. Neka su s_l i s_r tangente na p i paralelne s, one leže na levoj i desnoj strani p redom. Neka su $p_l \in p \cap s_l$ i $p_r \in p \cap s_r$. Sada definišemo q kao skup omeđen sa l_b, l_t, s_l i s_r.



Slika 1: Konstrukcija paralelograma

Sa slike sa jasno vidi da važe (1), (3) i (4). Međutim, q ne mora nužno da zadovoljava i (2), ali ako zadovoljava, onda sigurno zadovoljava i ostale pretpostavke iz leme. Posmatrajmo slučaj kada je $base(q) > width(p)$. Razmotrimo pravougaonik koji omeđuje p. To jest, r sadrži p i svaka njegova strana ima neprazan presek sa p. Sigurno, r zadovoljava (1), (2) i (3). Da bismo videli da zadovoljava i (4), primetimo da je

$$\begin{aligned} area(r) &= width(r) \cdot height(r) \\ &= width(p) \cdot height(p) \\ &< base(q) \cdot height(q) \\ &= area(q) \\ &\leq 2 \cdot area(p) \end{aligned}$$

Pokazali smo da r zadovoljava sve zahteve (1) - (4) leme, a pošto je i r x-paralelogram, zaključujemo da lema važi.

Pomoću leme, sada možemo predstaviti glavni rezultat ove glave.

Teorema: Postoji algoritam aproksimacije polinomijalnog vremena 9.4 za minimizaciju površine poligona. Osim toga, postoji takav algoritam sa vremenom izvršavanja $O(n^2 + N)$, gde je n broj poligona, a N ukupan broj temena u datom ulazu $I \in \mathcal{I}$, pod pretpostavkom da je svaki poligon $p \in I$ dat kao lista svojih temena.

Dokaz: Neka je $I \in \mathcal{I}$. Konstruišemo pakovanje P.

Pakujemo svaki poligon $p \in I$ u x-paralelogram q kao u Lemi. Nazovimo instancu svih tako dobijenih x-paralelograma IQ. Ideja našeg algoritma je da koristimo FFDH (pakovanje prvog odgovarajućeg po opadajućoj visini) da bismo spakovali ispravljene, osnoparalelne verzije x-paralelograma IQ i zatim koristimo ovo pakovanje da bismo dobili jedno za IQ i time i I koje nije mnogo veće.

Definišimo još jednu instancu IR koja, za svaki $q \in IQ$, sadrži pravougaonik $r = (\text{base}(q), \text{height}(q))$. Sa FFDH, sada pakujemo IR u traku širine $cw_{\max}(I)$, gde je $c \geq 1$ koji će biti kasnije određen. Nazovimo tako dobijeno pakovanje PR. Sledi

$$\text{height}(P_R)(cw_{\max}(I)) \leq \left(1 + \frac{1}{m}\right) \cdot \text{area}(I_R) + c \cdot h_{\max}(IR) \cdot w_{\max}(I),$$

gde je $m = \lfloor c \rfloor$, jer je $w_{\max}(IR) = \max_{q \in IQ} \text{base}(q) \leq w_{\max}(I)$ po lemi.

Neka je $S \subseteq IQ$ paralelogrami koji odgovaraju pravougaonicima u određenoj polici u pakovanju PR. Možemo spakovati S u novu policu širine $(c+2)w_{\max}(I)$ tako što ćemo prvo urediti paralelograme S po opadajućem uglu. Zaista, ako ih, nakon ovog poređanja, postavimo jedan pored drugog u polici, primećujemo da su sada sve osnove paralelograma povezane jedna sa drugom i da se preklapanje sa svake strane najviše postiže sa $\max_{q \in S} \text{wside}(q) \leq w_{\max}(I)$ po lemi. Stavite sve takve police jednu preko druge i nazovite tako dobijeno pakovanje PQ. Napomenimo da PQ ima istu visinu kao i PR, ali $\frac{c+2}{c}$ puta njegovu širinu. Na kraju, spakujemo svaki poligon u odgovarajući paralelogram u pakovanju PQ. Nazovite ovo pakovanje P. Tada

$$
\begin{aligned}
\text{area}(P) &\leq \text{area}(P_Q) \\
&\leq \frac{c+2}{c} \cdot \text{height}(P_R) \cdot (cw_{\max}(I)) \\
&\leq \frac{c+2}{c}\left(1+\frac{1}{m}\right)\text{area}(I_R) + (c+2)h_{\max}(I_R)w_{\max}(I) \\
&= \frac{c+2}{c}\frac{m+1}{m}\text{area}(I_Q) + (c+2)h_{\max}(I_Q)w_{\max}(I) \\
&\leq 2\frac{c+2}{c}\frac{m+1}{m}\text{area}(I) + (c+2)h_{\max}(I)w_{\max}(I) \\
&\leq \left(2\frac{c+2}{c}\frac{m+1}{m} + (c+2)\right)\text{opt}(I).
\end{aligned}
$$

Možemo proveriti da se minimum postiže kod $c = 3$, u kom slučaju dobijamo aproksimaciju 9.4. Iz dokaza algoritma se jasno vide koraci algoritma za pakovanje konveksnih poligona u pravougaonik minimalne površine.

3.2 Složenost izvršavanja

Vreme izvršavanja algoritma prati sledeće korake:

1. Konstrukcija IQ iz I: $O(N)$ vremena.
2. Konstrukcija IR iz IQ: $O(n)$ vremena.
3. Konstrukcija PR iz IR: $O(n^2)$ vremena.
4. Konstrukcija PQ iz PR sortiranjem: $O(n\log(n))$ vremena.
5. Konstrukcija P iz PQ: $O(N)$ vremena.

Ovaj algoritam pruža efikasno rešenje za minimizaciju površine sa zadovoljavajućim vremenom izvršavanja i aproksimacijom.

4 Implementacija algoritma

U kodu su implementirana dva algoritma: aproksimativni algoritam pakovanja opisan u prethodnoj glavi i naivni algoritam, čije rešenje nije jednako opt(I) iz dokaza prethodne teoreme, ali je dobra polazna osnova za procenu kvaliteta aproksimativnog algoritma. Oba algoritma se sastoje od funkcije za izvršavanje algoritma sa dodatnim pomoćnim funkcijama i funkcije za vizualizaciju poligona i procesa pakovanja. Vizualizacija uključuje crtanje poligona i njihovih okvirnih pravougaonika sa dimenzijama njegovih stranica.

4.1 Naivni algoritam

U ovom algoritmu se iterira kroz sve permutacije poligona kako bi pronasao optimalni raspored. Za svaku permutaciju, vrši se pakovanje i porede se rezultati, zadržavajući raspored koji minimizuje površinu pravougaonika.

Poligoni se pakuju u kontejner koristeći metod koji prolazi kroz niz poligona i translira ih dok god god se oni presecaju. Presek poligona se proverava proverom preseka svih njegovih stranica.

4.2 Aproksimativni algoritam

Neke od najbitnijih funkcija za implementaciju aproksimativnog algoritma su:

Pravljenje X-paralelograma: Implementirana je funkcija `napraviXParalelogram` koja za dati poligon kreira x-paralelogram koji ga obuhvata, prateći korake iz opisa algoritma.

FFDH: Implementirana je funkcija `FFDHPakovanje` koja implementira algoritam First-Fit Decreasing Height (prvi odgovarajući opadajuće visine) za pakovanje pravougaonika.

Određivanje kičme poligona: Funkcija `kicmaPoligona` pronalazi kičmu poligona, pravu koja spaja najnižu i najvišu tačku.

I ostali delovi algoritma su implementirani prateći opis algoritma iz prethodne glave. Jedina razlika u kodu je što poligoni nisu zadati u koordinatama od 0 do 1, zbog iscrtavanja poligona. Pored toga, prilikom FFDH pakovanja širina trake je postavljena da odgovara širini scene na kojoj se iscratavaju poligoni.

5 Rezultati

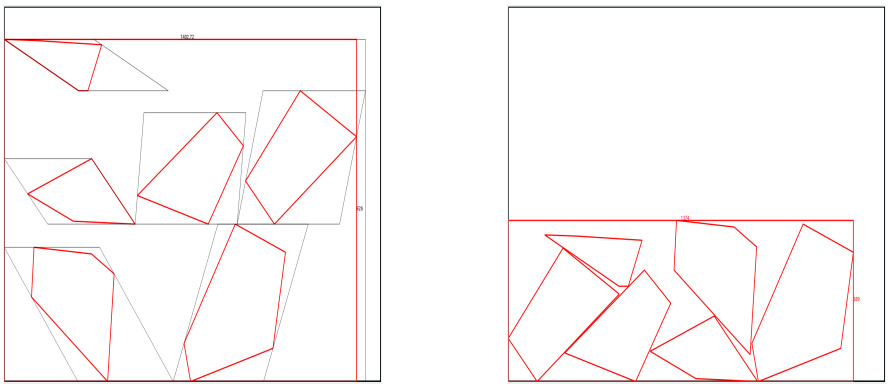
U nastavku će biti prikazani rezultati rada algoritma na 4 različita skupa od po 6 poligona kao i poređenje vremena rada oba algoritma na nasumično generisanim poligonima.

5.1 Skup 1

Podaci o poligonima se nalaze u datoteci 'input.txt' u direktorijumu 'ga_Poligoni'.

Analiza rezultata pokazuje da aproksimativni algoritam koji radi mnogo brže od naivnog algoritma daje lošije rešenje, što se može videti na slici 2, ali je aproksimacija u skladu sa pretpostavkom.

1. Rezultat aproksimativnog algoritma: 1402.72 x 826
2. Rezultat naivnog algoritma: 1374 x 389



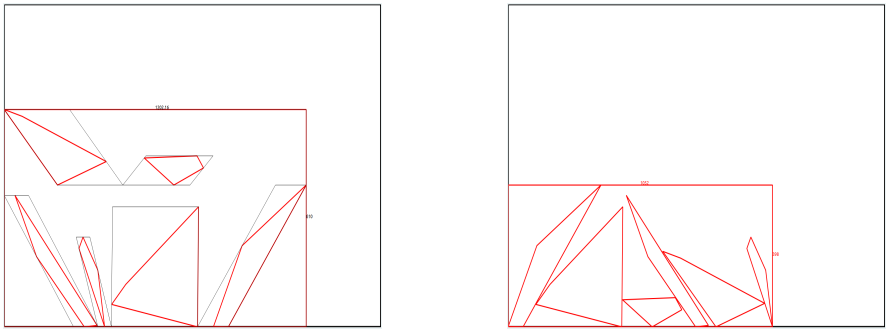
Slika 2: Rezultati rada aproksimativnog i naivnog algoritma

5.2 Skup 2

Podaci o poligonima se nalaze u datoteci 'input2.txt' u direktorijumu 'ga_Poligoni'.

Analiza rezultata pokazuje da aproksimativni algoritam koji radi mnogo brže od naivnog algoritma daje lošije rešenje, što se može videti na slici 3, ali je aproksimacija u skladu sa pretpostavkom.

1. Rezultat aproksimativnog algoritma: 1202.16 x 610
2. Rezultat naivnog algoritma: 1052 x 398



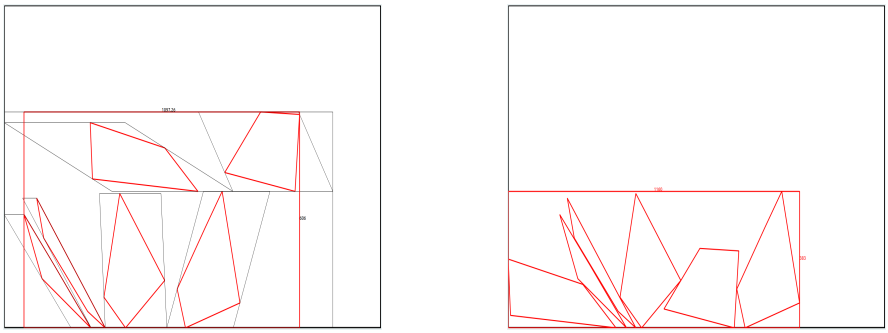
Slika 3: Rezultati rada aproksimativnog i naivnog algoritma

5.3 Skup 3

Podaci o poligonima se nalaze u datoteci 'input3.txt' u direktorijumu 'ga_Poligoni'.

Analiza rezultata pokazuje da aproksimativni algoritam koji radi mnogo brže od naivnog algoritma daje lošije rešenje, što se može videti na slici 4, ali je aproksimacija u skladu sa pretpostavkom.

1. Rezultat aproksimativnog algoritma: 1097.26 x 606
2. Rezultat naivnog algoritma: 1160 x 383



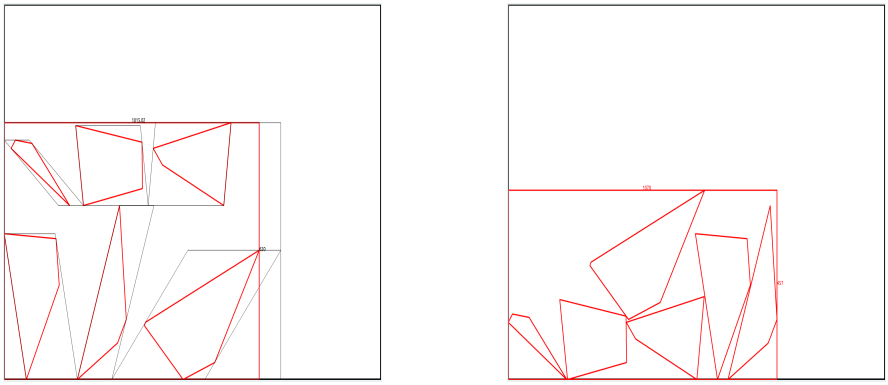
Slika 4: Rezultati rada aproksimativnog i naivnog algoritma

5.4 Skup 4

Podaci o poligonima se nalaze u datoteci 'input4.txt' u direktorijumu 'ga_Poligoni'.

Analiza rezultata pokazuje da aproksimativni algoritam koji radi mnogo brže od naivnog algoritma daje lošije rešenje, što se može videti na slici 5, ali je aproksimacija u skladu sa pretpostavkom.

1. Rezultat aproksimativnog algoritma: 1015.02 x 620
2. Rezultat naivnog algoritma: 1070 x 457



Slika 5: Rezultati rada aproksimativnog i naivnog algoritma

5.5 Poređenje rezultata

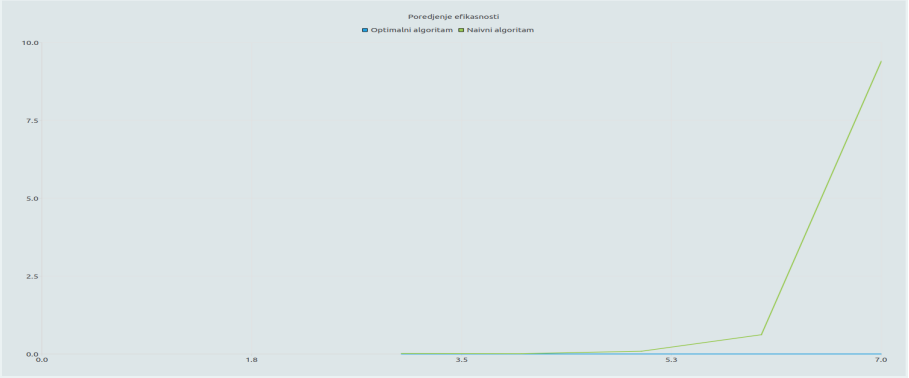
U tabeli 1 možemo videti da aproksimativni algoritma, za ove skupove podataka, vraća kao rezultat pravougaonik koji je uglavnom oko 2 puta veći od pravougaonika koji vrati naivni algoritam, koji je mnogo sporiji od aproksimativnog, a dokaz za to ćemo videti u nastavku.

Skup poligona	Aproksimativni algoritam			Naivni algoritam			Odnos površina
	Širina	Visina	Površina	Širina	Visina	Površina	
1	1402.72	826	1158646.72	1374	389	534486	2.1678
2	1202.16	610	733317.6	1052	398	418696	1.7514
3	1097.26	606	664939.56	1160	383	444280	1.4967
4	1015.02	620	629312.4	1070	457	488990	1.2870

Tabela 1: Prikaz rezultata oba algoritma i odnos površina dobijenih pravougaonika

5.6 Poređenje vremena izvršavanja

Sa slike 6 se jasno vidi da je vreme izvršavanja naivnog algoritma mnogo veće od vremena za koje aproksimativni algoritam završi svoj rad. Primetno je da naivni algoritam za skup od 7 poligona radi skoro 10 sekundi. Ovaj algoritam je praktično neupotrebljiv za neke veće skupove poligona, jer je njegova složenost eksponencijalna po broju poligona, dok aproksimativni algoritam radi brzo i za velike skupove poligona.



Slika 6: Vreme izvršavanja algoritama

Literatura

[1] Predrag Janičić. Računarska geometrija, 2016.

[2] Vijay V. Vazirani Ketan Mulmuley, Umesh V. Vazirani. Matching is as easy as matrix inversion. *Combinatorica 7*, 1987.