SVEUČILIŠTE J. J. STROSSMAYERA U OSIJEKU
ELEKTROTEHNIČKI FAKULTET

DOMAGOJ ŠEVERDIJA

UNAPRIJEĐENI APROKSIMACIJSKI ALGORITMI ZA POKRIVANJE 1.5D TERENA

Doktorski rad

Osijek, 2013

Doktorski rad je izrađen na Elektrotehničkom fakultetu Osijek, Sveučilišta J. J. Strossmayera u Osijeku.

MENTOR: prof.dr.sc. Goran Martinović

Doktorski rad ima _____ stranica.

Doktorski rad br.: _____

POVJERENSTVO ZA OCJENU DOKTORSKOG RADA:

1. Dr.sc. Željko Hocenski, redoviti profesor, predsjednik, ETFOS
2. Dr.sc. Goran Martinović, redoviti profesor, mentor, ETFOS
3. Dr.sc. Domagoj Matijević, izvanredni profesor, član, Odjel za matematiku Sveučilišta u Osijeku
4. Dr.sc. Vlado Sruk, izvanredni profesor, član, Fakultet elektrotehnike i računarstva Sveučilišta u Zagrebu
5. Dr.sc. Alfonzo Baumgartner, docent, član, ETFOS

POVJERENSTVO ZA OBRANU DOKTORSKOG RADA:

1. Dr.sc. Željko Hocenski, redoviti profesor, predsjednik, ETFOS
2. Dr.sc. Goran Martinović, redoviti profesor, mentor, ETFOS
3. Dr.sc. Domagoj Matijević, izvanredni profesor, član, Odjel za matematiku Sveučilišta u Osijeku
4. Dr.sc. Vlado Sruk, izvanredni profesor, član, Fakultet elektrotehnike i računarstva Sveučilišta u Zagrebu
5. Dr.sc. Alfonzo Baumgartner, docent, član, ETFOS

datum obrane: 13. prosinca, 2013.

Science is what we understand
well enough to explain to a computer.
Art is everything else we do.

— Donald E. Knuth

ZAHVALA

Ovaj doktorski rad ne bi mogao postojati da nije bilo pomoći i podrške ljudi oko mene.

U prvom redu, želim se zahvaliti svojoj ženi Ivani na njenoj strpljivosti i brizi, te svojim roditeljima i setri koji su me bodrili da se bavim stvarima koje volim od malih nogu.

Ovaj doktorski rad ne bi bio moguć da nije bilo budne brige i skrbi mog kolege doc.dr.sc. Matijevića koji je uvijek bio na raspologanju u mom znanstvenom sazrijevanju. Posebno se zahvaljujem mentoru prof.dr.sc. G. Martinoviću na smjernice i profesionalni angažman prilikom nastajanja ovog doktorskog rada.

Želim se zahvaliti svim svojim kolegama te svim nastavnicima i profesorima koje su direktnim i indirektnim utjecajem omogućili stvaranje ovog doktorskog rada.

SADRŽAJ

POPIS SLIKA

POPIS TABLICA

POPIS ALGORITAMA

1

UVOD

Doktorski rad je organiziran u 2 bitna dijela: u prvom dijelu istražuju se problemi čuvanja na terenima kao kombinatorni optimizacijski problemi vidljivosti, te u drugom dijelu, implementacijski model problema čuvanja za 1.5D terene na temelju brzih aproksimativnih rješavatelja linearnih programa.

U nastavku, predstavlja se motivacija za probleme vidljivosti u izračunljivoj geometriji te se definiraju problemi čuvanja koji se razmatraju u doktorskom radu. Osnovni pregled pojmova iz teorije složenosti i rezultati iz linearne optimizacije predstavljeni su kao temelj za analizu algoritama koje će bit predstavljeni u sljedećim poglavljima.

Sadržaj

1.1 PROBLEMI VIDLJIVOSTI

1.1.1 *Motivacija*

Problemi vidljivosti predstavljaju posebno područje proučavanja u izračunljivoj geometriji (engl. *computational geometry*). Pod pojmom vidljivosti u geometriji podrazumijeva se odnos između dvije točke u prostoru tako da se kaže da dvije točke vide jedna drugu ukoliko ne postoji prepreka između njih. Pojam vidljivosti može se shvatiti u općenitom smislu kao relacija između 2 točke koja je uvjetovana geometrijskim svojstvima prostora u kojem se točke nalaze (npr. širenje signala između 2 točke u nekom mediju se u ovom smislu može interpretirati kao vidljivost). Mnogi problemi u robotici i planiranju putanje usko su vezani s pojmom vidljivosti [64], postavljanje komunikacijskih antena [24] i sustava za nadziranje [35] itd. Motivacija će započeti s klasičnim problemom *čuvanja galerije* kao jednim od prvih formuliranih problema vidljivosti.

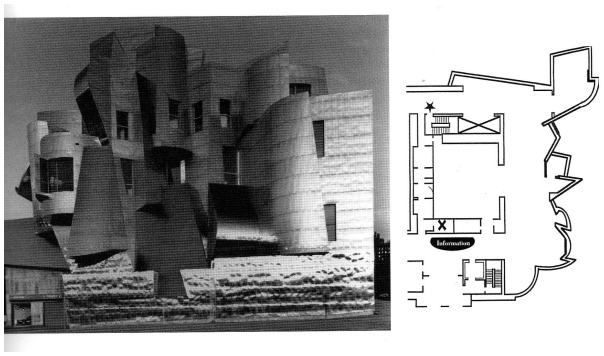
U pregledu [71] predstavljen je problem čuvanja galerije s umjetninama zajedno s još nekim varijacijama.

1.1.2 *Problemi vidljivosti u poligonu*

Problem čuvanja galerije (engl. *Art Gallery problem*) jedan je od klasičnih geometrijskih problema vidljivosti i dodirna točka robotike, izračunljive geometrije i optimizacije. Motiviran je problemom smještanja minimalnog broja čuvara u galeriju s umjetninama tako da svi čuvari zajedno promatraju čitavu galeriju. *Slika 1* [3] pokazuje jednu takvu galeriju s pripadnim tlocrtom. Ova motivacija našla je daljnje primjene u stvarnom svijetu kao što je postavljanje senzora i kamera za pokrivanje interijera neke zgrade [35], pokrivanje telekomunikacijskim antenama [24] nekog planinskog područja, postavljanje izvore svjetla za osvjetljavanje neke pozornice [82] itd. U novije vrijeme zabilježena je primjena u industriji [62] za mjerenje interijera zgrada koristeći stacionarne laserske čitače.

Prema [12], Viktor Klee je 1976. godine postavio sljedeće pitanje na jednoj konferenciji:

> Koliko je čuvara nužno, a koliko dovoljno da čuvaju umjetnine u galeriji s n zidova?



Slika 1: Galerija s umjetninama i njen tlocrt

Upravo ovo, naočigled, naivno pitanje, otvorilo je područje istraživanja u kombinatornoj geometriji što je rezultiralo velikim brojem radova na tu temu i na varijacijama koje su proizašle iz tog problema. Neke od problema bit će i ovdje predstavljeni.

Galerija se kao geometrijski lik predstavlja kao jednostavni poligon kojeg je potrebno čuvati čuvarima postavljenima u točkama unutar poligona.

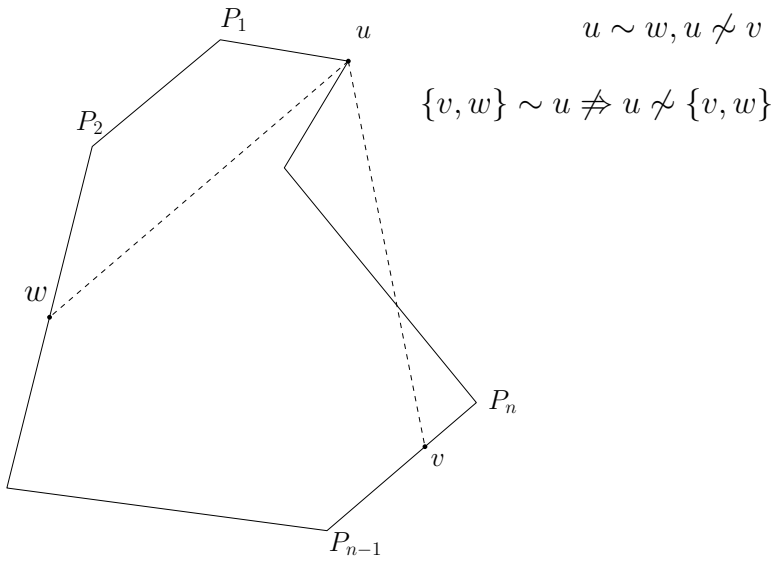
DEFINICIJA 1: Poligon P čini uređeni niz točaka $V = (P_1, P_2, \ldots, P_n)$ za $n \geq 3$ Euklidske ravnine $\mathbb{R}^2$ koji se zovu *vrhovi* poligona P zajedno sa skupom dužina $\delta P = \{\overline{P_1P_2}, \overline{P_2P_3}, \ldots, \overline{P_{n-1}P_n}\}$, koje se zovu *bridovi*. Poligon P je *jednostavan* ako se nikoja dva nesusjedna ruba ne sijeku.

Jednostavan poligon, prema [81, str. 37], je Jordanov luk te stoga dijeli ravninu u 3 podskupa: sam poligon s bridovima $P \cup \delta P$, omeđeno područje unutrašnjosti poligona (interijer) u oznaci $IntP$ i vanjsko neomeđeno područje (ekterijer) u oznaci $ExtP$.

U daljnjem kontekstu, pod pojmom jednostavnog poligona P, smatrat će se skup $V \cup \delta P \cup IntP$ koji je zatvoren i ograničen skup u ravnini.

DEFINICIJA 2: Za točke $v \in P$ se kaže da je *vidljiva* iz točke $u \in P$ u odnosu na P i piše se $u \sim_P v$ ukoliko je dužina $\overline{uv}$ u potpunosti sadržana u poligonu P, odnosno $\overline{uv} \in P$. Za skup $Q \subseteq P$ kaže se da *vidi* skup točaka $R \subseteq P$ u odnosu na P ukoliko za svaki $r \in R$ postoji točka $q \in Q$ tako da je $q \sim_P r$ i piše se $Q \sim_P P$.

Relacija vidljivosti je simetrična s obzirom na pojedinačne točke. Za relaciju vidljivosti među skupovima točaka to općenito ne mora vrijediti, stoga će se točke iz skupa Q zvati *čuvari*, a točke iz R *klijenti*.



Slika 2: Vidljivost u poligonu

Ukoliko $Q \sim R$ kaže se da Q *čuva* ili *pokriva* R. *Slika 2* daje primjer koja pokazuje kako relacija vidljivosti nije općenito simetrična.

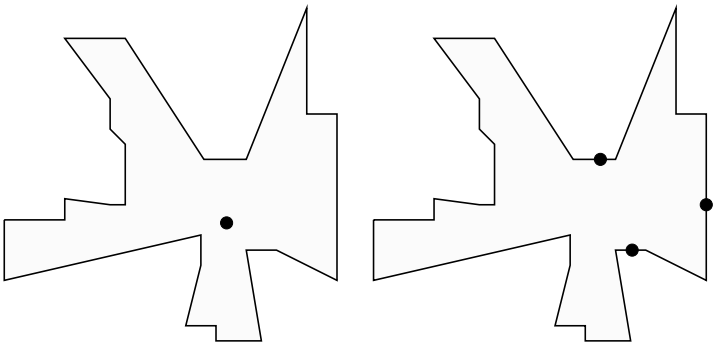
Radi jednostavnosti oznaka, ispušta se '*u odnosu na*' u definiciji relacije vidljivosti, jer će se iz konteksta podrazumijevati o kojem je poligonu riječ.

PRIMJEDBA 1: Pojam *čuvanje* koje se koristi u nazivu "problemi čuvanja" predstavlja posebnu vrstu *pokrivanja*. Naime, u pojmu *pokrivanje* podrazumijeva se da su dani skupovi koji imaju pokriti neke elemente, formulacija koja može biti vrlo općenita. Pojam *čuvanje* podrazumijeva pokrivanje u kojem su skupovi definirani na temelju područja vidljivosti čuvara, a elementi su sami klijenti. Relacija incidencije je onda relacija vidljivosti između čuvara i klijenata.

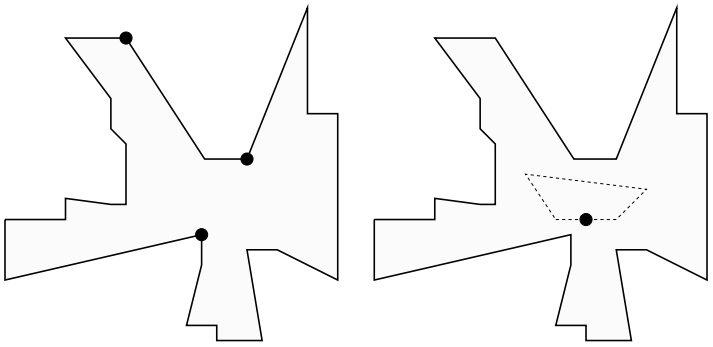
Shermer ističe u svom pregledu problema vidljivosti [77] korištenje različitih tipova poligona te pripadne rezultate vezane uz njih od kojih će biti istaknuti poligon s rupama, ilustriran na *Slici 4(a)*.

DEFINICIJA 3: Jednostavni poligon P s rupama sastoji se od konačnog k broja jednostavnih poligona $P_1, P_2, \ldots, P_k$ sa svojstvom $P_i \subset P$ i $P_i \cap P_j = \emptyset$, $i \neq j$ gdje su $i, j \in \{1, 2, \ldots, k\}$.

Osim varijacije u tipu poligona, Shermer [77], Urrutia [82] i O'Rourke [74] ističu određene restrikcije na čuvare, pa primjerice čuvari mogu biti smješteni isključivo u vrhovima odnosno bridovima poligona ili mogu biti smješteni u interijeru poligona kao stacionarni čuvari. Neki



(a) čuvari u interijeru $IntP$ (b) čuvari u bridovima δP



(c) čuvari u vrhovima $V(P)$ (d) određivanje puta $\delta \in IntP$ za koje $\delta \sim P$

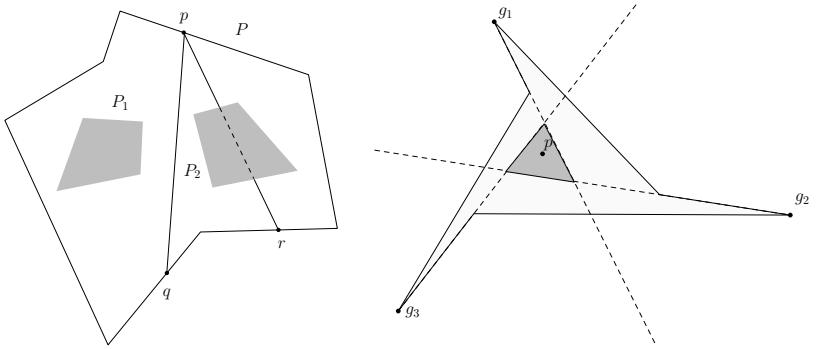
Slika 3: Različiti pristupi postavljanja čuvara u galeriju

autori povezuju probleme čuvanja poligona s optimizacijom rute gibanja robota kao mobilnog čuvara [75]. Čuvanje poligona u tom kontekstu je definirano kao pronalaženje puta u kojem je svaka točka poligona vidljiva na nekom dijelu puta. Ilustracija je prikazana na *Slici 3(d)*.

U doktorskom radu je od koristi sljedeći spomenuti problem: za neki jednostavni poligon P i podskup $G \subseteq P$ pronaći minimalni podskup $S \subseteq G$ koji čuva poligon P. Valja naglasiti kako čuvanje poligona podrazumjeva i pokrivanje njegovog interijera, što ne mora biti slučaj ukoliko se želi pokriti samo bridove poligona.

LEMA 1.1. *Ukoliko neki skup točaka Q poligona P čuva $IntP$, onda skup Q čuva i bridove poligona δP. Obrat općenito ne vrijedi.*

Dokaz. Na *Slici 4(b)* vidljivo je da obrat općenito ne vrijedi, dakle, točku p ne vidi niti jedan čuvar iz $Q = \{g_1, g_2, g_3\}$, a $Q \sim \delta P$.



(a) poligon P s "rupama" P_1 i P_2.

(b) pokrivenost bridova poligona ne implicira pokrivenost interijera

Slika 4: Poligon s "rupama" i nepokriveni interijer poligona

Pretpostavimo da postoji neki proizvoljni skup točaka $Q \subset P$ za koje $Q \sim IntP$ i neka točka $p \in \delta P$ za koje $p \notin Q$ i $Q \not\sim p$. Neka je $q \in Q$. Kako $q \not\sim p$, onda po definiciji vidljivosti, $\overline{qp} \cap ExtP \neq \emptyset$. Jednostavni poligon P tvori Jordanov luk, što za posljedicu ima da je $ExtP$ povezan, implicira da postoji neka točka $r \in IntP$ koja leži na spojnici $\overline{qp}$ za koje je $\overline{qr} \cap ExtP \neq \emptyset$ i $Q \setminus \{q\} \not\sim r$. Drugim riječima, $Q \not\sim r$ što je kontradikcija s pretpostavkom da $Q \sim IntP$. □

Prema *slici 3* može se definirati sljedeće varijante čuvanja poligona:

DEFINICIJA 4: Neka je dan jednostavni poligon P s n vrhova. Problem VERTEX-POLYGON-GUARD je problem pronalaženja minimalnog skupa čuvara S iz skupa vrhova poligona P tako da S čuva P.

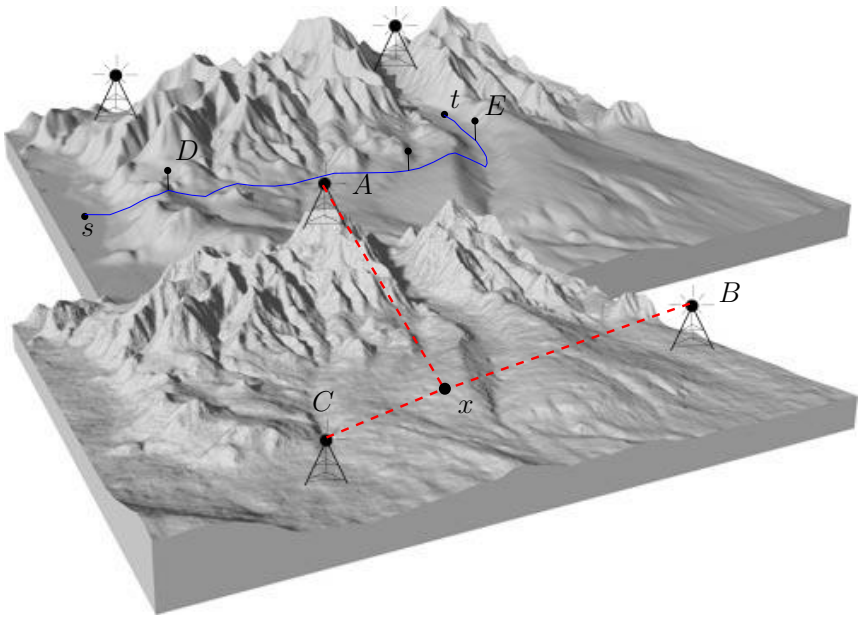
DEFINICIJA 5: Neka je P jednostavni poligon s n vrhova. Problem EDGE-POLYGON-GUARD je problem pronalaženja minimalnog skupa čuvara S iz bridova poligona od P tako da S čuva P.

DEFINICIJA 6: Neka je P jednostavni poligon s n vrhova. Problem POINT-POLYGON-GUARD je problem pronalaženja minimalnog skupa čuvara S iz interijera poligona od P tako da S čuva P.

1.1.3 *Problemi vidljivosti na terenima*

Problemi čuvanja terena nalaze motivaciju u postavljanju bežične komunikacijske mreže na nekom reljefnom području kao što je prikazano na *slici 5*.

Motivacija 1: Tvrtka investitor želi postaviti komunikacijske antene na takav način da je svaka točka tog reljefnog područja pokrivena, ilus-



Slika 5: Reljefno područje pokriveno komunikacijskim antenama

tracija prikazana na *slici 5*. Kako reljef utječe na širenje signala, firma želi osigurati dovoljan broj antena te signalom pokriti interesno područje. U obzir se može uzeti cijena postavljanja tih antena i robusnije pokrivanje područja: neku točku područja trebaju pokrivati signali iz više izvora (npr. točku x pokrivaju 3 antene A, B, C na *slici 5*).

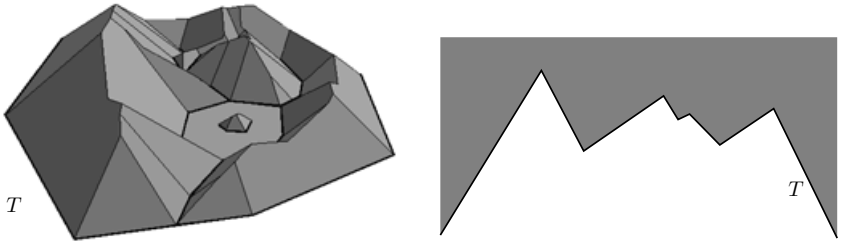
Antene se mogu isto tako intepretirati kao postavljanje promatračnica na tom području i cilj je da svaka točka tog područja bude viđena.

Motivacija 2: Za neku cestu na reljefnom području treba postaviti nadzorne kamere tako da svaka dionica ceste bude pokrivena nadzornim kamerama. Na *slici 5* je prikazan put st koji predstavlja cestu od nekog mjesta s do mjesta t. Nadzorne kamere D i E pokrivaju tu cestu.

Geometrijski gledajući, reljefno područje se može opisati kao poliedarska ploha, a točke koje treba čuvati kao točke te plohe. Pokriva li neka točka plohe neku drugu točku na toj plohi može se modelirati kao dužina koja spaja te 2 točke. Ukoliko ta dužina ne siječe nigdje teren kaže se da točke vide jednu drugu. Cesta se može modelirati kao poligonalna linija koja leži na poliedarskoj plohi.

U daljnjim razmatranjima, promatrat će se 2 tipa formulacije terena, naime, 2.5D teren i 1.5D teren od kojeg će ovaj zadnji biti predmet interesa ovog doktorskog rada. Oba terena su ilustrirani na *slici 6*.

DEFINICIJA 7: Neka je n broj proizvoljnih točaka u $\mathbb{R}^3$ koje zovemo vrhovi. Ploha $T := \{(x,y,z) : z = f(x,y)\} \subset \mathbb{R}^3$ gdje je $f : \mathbb{R}^2 \to \mathbb{R}$ funkcija linearne interpolacije između vrhova zove se *2.5D teren složenosti n*.



(a) 2.5D teren (b) 1.5D teren

Slika 6: Dva osnovna tipa terena

Intuitivno, svaki vertikalni pravac siječe 2.5D teren u najviše jednoj točki.

Nadalje, neka je $\mathbb{R}^2$ euklidska ravnina. Točke ravnine predstavljaju uređeni parovi u oznaci $p = (x_p, y_p)$. Za dvije točke $p, q \in \mathbb{R}^2, p \neq q$ definira se skup $\{p + t(p - q) : t \in [0,1]\}$ koji se zove *segment* između p i q te označava s $\overline{pq}$. Ukoliko u definiciji segmenta $\overline{pq}$ se dozvoli $t \in \mathbb{R}$ onda se govori o *spojnici* točaka p i q i označava s pq.

U ovom doktorskom radu promatrat će se uređaj na 1.5D terenu:

DEFINICIJA 8: za točku p kaže se da je *lijevo* od q ukoliko $x_p \leq x_q$ za bilo koje točke $p, q \in \mathbb{R}^2$. Relacija "bit strogo lijevo" se definira na sličan način i piše $p < q$ ako i samo ako $x_p < x_q, \forall p, q \in \mathbb{R}^2$.

Relacije "biti desno" i "biti strogo desno" se definiraju simetrično.

DEFINICIJA 9: Neka su dane točke $P_1, P_2, \ldots, P_n \in \mathbb{R}^2, 1 < n < \infty$. Za skup segmenata $T = \{\overline{P_1P_2}, \overline{P_2P_3}, \ldots, \overline{P_{n-1}P_n}\}$ kaže se da je *x-monotona poligonalna linija* ukoliko vrijedi $P_1 < P_2 < \ldots < P_n$. Poligonalna linija T zove se 1.5*D teren složenosti n*.

Odgovarajući problemi vidljivosti na terenima su definirani na sljedeći način:

DEFINICIJA 10: Neka je dan 2.5D teren T dimenzije n. Problem ZVERTEX-2.5D-TERRAIN-GUARD jest problem pronalaženja minimalnog skupa čuvara iz vrhova od T koji će čuvati T.

DEFINICIJA 11: Neka je dan 2.5D teren T dimenzije n. Problem POINT-2.5D-TERRAIN-GUARD jest problem pronalaženja minimalnog skupa čuvara s terena T koji će čuvati T.

U većini rezultata za čuvanje 1.5D terena ([51], [7], [24]), promatraju se 2 osnovna problema:

DEFINICIJA 12: Neka je dan 1.5D teren T dimenzije n. Problem VERTEX-1.5D-TERRAIN-GUARD jest problem pronalaženja minimalnog skupa čuvara X iz vrhova od T koji će čuvati T.

DEFINICIJA 13: Neka je dan 1.5D teren T dimenzije n. Problem 1.5D-TERRAIN-GUARD jest problem pronalaženja minimalnog skupa čuvara X s terena T koji će čuvati T.

U ovom doktorskom radu promatraju se problemi čuvanja 1.5D terena koji, osim čisto geometrijskog gledišta, imaju i kombinatorni aspekt. Naime, promatraju se proizvoljni podskupovi terena G odnosno N, iz kojih se odabiru čuvari odnosno klijenti. Dopušta se da čuvari imaju cijenu i klijenti mogu imati zahtjev na broj čuvara koji ih trebaju čuvati.

DEFINICIJA 14: Neka je dan teren T sa skupom čuvara $G \subseteq T$ i skupom točaka koje treba čuvati $N \subseteq T$ te neka težinska funkcija $w\colon G \to \mathbb{Q}_+$ definirana nad G. *Težinsko čuvanje 1.5D terena* u oznaci WEIGHTED-1.5D-TERRAIN-GUARD jest problem u kojem treba odrediti skup minimalne težine $X \subseteq G$ takav da čuva N, odnosno

$$X = \arg\min\{w(S)\colon S \subseteq G, S \sim N\}$$

gdje $w(S) = \sum_{g \in S} w(g)$.

Robusnija inačica čuvanja klijenata:

DEFINICIJA 15: Neka je dan 1.5D teren T sa skupom čuvara $G \subset T$ i skupom klijenata $N \subset T$ te funkcija zahtjeva $d\colon N \to \mathbb{Z}_+$ definirana na klijentima N. *Čuvanje 1.5D terena s višestrukim pokrivanjem* u oznaci 1.5D-TERRAIN-MULTI-GUARD jest problem određivanja najmanjeg skupa čuvara $X \subseteq G$ takav da čuva N s obzirom na zahtjeve, odnosno, za svaku točku $p \in N$ postoji barem $d(p)$ čuvara u X koji čuvaju p.

Analiza koja bude predstavljena uključuje i inačicu parcijalnog čuvanja 1.5D terena:

DEFINICIJA 16: U *parcijalnoj inačici problema čuvanja 1.5D terena* u oznaci PARTIAL-1.5D-TERRAIN-GUARD dan je profit $p\colon N \to \mathbb{Q}_+$ za skup

klijenata $N \subseteq T$ i ograničenje na nepokrivenost terena (engl. *budget*) b. Cilj je pronaći skup čuvara minimalne težine X iz T tako da ukupan profit od nečuvanih klijenata najviše b.

Ukoliko u problemu Weighted-1.5D-Terrain-Guard skup čuvara G i skup klijenata N je konačan, govori se o *diskretnoj varijanti* problema. Problem ima oznaku Discrete-1.5D-Terrain-Guard ukoliko su težine na čuvarima jedinične, u protivnom problem se označava Discrete-Weighted-1.5D-Terrain-Guard.

U *jednostranoj varijanti* problema čuvari vide samo u jednom smjeru: ili lijevo ili desno. Točnije, za dana 3 skupa točaka terena N, G_L, G_R želi se pronaći skupovi čuvara $X_L \subseteq G_L$ i $X_R \subseteq G_R$ tako da za svaki $p \in N$ postoji $g \in X_L$ tako da je $g < p$ i $g \sim p$ ili $g \in X_R$ tako da je $g > p$ i $g \sim p$. Dakle, skupovi G_L i G_R, odnosno X_L i X_R ne moraju biti disjunktni.

Posebni slučaj jednostrane varijante predstavlja *lijevo* odnosno *desno čuvanje 1.5D terena* gdje čuvari u G mogu vidjeti samo na lijevo, odnosno samo na desno (drugim riječima, postavljajući $G_L = G$ i $G_R = \emptyset$ dobiva se problem lijevog čuvanja terena odnosno za $G_R = G$ i $G_L = \emptyset$ dobiva se problem desnog čuvanja terena).

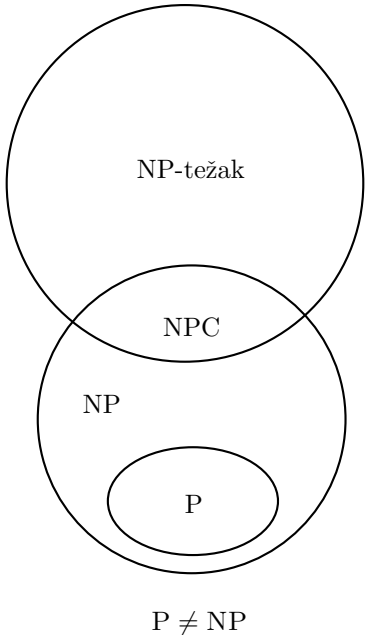
1.2 OPTIMIZACIJSKI PROBLEMI I RAZREDI APROKSIMABILNOSTI

U ovom uvodnom dijelu uvest će se osnovni pojmovi iz teorije složenosti koje su potrebne u ovom radu. Svi ovi pojmovi su uvelike poznati i mogu se naći u npr. [84], te [44] i [78].

1.2.1 *Razredi složenosti*

Standardna podjela optimizacijskih problema na temelju vremenske složenosti mogu se podijeliti u 3 osnovne skupine: P, NP i NPC. Neformalno, kaže se da razred složenosti P sadrži sve one optimizacijske probleme koji su rješivi u polinomnom vremenu, odnosno, za problem u P postoji rješenje koje se može izračunati u $O(n^k)$ za neku konstantu $k \geq 0$ gdje je n veličina ulaza problema. Za problem se kaže da pripada razredu NP ukoliko postoji algoritam koji u polinomnom vremenu može provjeriti korektnost rješenja za taj problem. Za neki optimizacijski problem kaže se da pripada klasi NPC odnosno kaže se da je NP-kompletan ukoliko je u NP i ujedno je NP-težak. Problem je NP-težak ukoliko se svaki problem iz NP može *reducirati* u

polinomnom vremenu na taj problem. Pod pojmom *redukcije u polinomnom vremenu* podrazumijeva se da se bilo koja instanca α nekog problema Q može transformirati u polinomnom vremenu u instancu β nekog drugog problema Q' čije rješenje daje rješenje za α. Odnos razreda složenosti vidljivo je na *slici 7*.



Slika 7: Razredi složenosti

Iz ovih neformalnih definicija jasno je da je svaki problem iz P ujedno u NP, dakle, P $\subseteq$ NP. Vrijedi li stroga inkluzija jest otvoren problem. Ono što je zanimljivo, ukoliko se pokaže da se neki problem koji je u NPC može riješiti u polinomnom vremenu, onda se svaki problem u NP može riješiti u polinomnom vremenu [78]. Upravo ova činjenica daje uvjerenje da je mala vjerovatnost za P = NP. Uz pretpostavku P $\neq$ NP, problemi koji pripadaju NPC klasi zbog svoje važnosti rješavaju se ili raznim heuristikama ili algoritmima u polinomnom vremenu koji nalaze aproksimativno rješenje unutar neke teorijske granice u odnosu na optimalno rješenje.

Proširivajući teoriju složenosti s pristupom od [39] za paralelno računanje, razred NC predstavlja sve one probleme koji se mogu riješiti u polilogaritamskom vremenu na paralelnom računalu s polinomno mnogo procesora. Kako se paralelna računala mogu simulirati na sekvencijalnom računalu, vrijedi inkluzija NC $\subseteq$ P. Vrijedi li stroga inkluzija jest otvoreno pitanje. Štoviše, PC razred ili P-kompletan razred služi za proučavanje problema koji su teški za efikasnu paralelizaciju.

Za neki problem kaže se da je P-kompletan ukoliko je u P i svaki problem u P se može transformirati u polilogaritamskom vremenu na paralelnom računalu s polinomno mnogo procesora na taj problem. Ukoliko vrijedi NC $\neq$ P, onda svi problemi koji se nalaze u PC se ne nalaze u NC. S druge strane, ukoliko bi se za neki PC problem pokazalo da se može pronaći rješenje na efikasan paralelan način onda bi to značilo da je P = NC.

1.2.2 *Aproksimacijski algoritmi*

U ovom doktorskom radu promatrat će se *optimizacijski problemi*, dakle, problemi čije se optimalno rješenje odabire iz skupa svih dopustivih rješenja. Ukoliko skup rješenja je konačan govori se o *kombinatornim optimizacijskim problemima*.

NP optimizacijski problem Π je predstavljen instancom I. Skup svih valjanih instanci problema Π neka je označen s D_Π. Veličinu instance I mjeri se količinom bitova za prezentiranje $|I|$ uz pretpostavku da su brojevi u I pisani binarno. Neka je $S_\Pi(I)$ *skup svih dopustivih rješenja* za instancu I problema Π i neka je dana funkcija cilja $f_\Pi(I,s)$ izračunljiva u polinomnom vremenu koja svakom paru (I,s) pridružuje racionalni broj. Svako dopustivo rješenje $s \in S_\Pi(I)$ je polinomne veličine u ovisnosti o $|I|$ i postoji algoritam u polinomnom vremenu koji za dani par (I,s) utvrđuje je li $s \in S_\Pi(I)$. Vrijednost optimalnog rješenja za instancu I označava se sa $OPT_\Pi(I)$. Dopustivno rješenje s^* za koje je $f_\Pi(I,s^*) = OPT_\Pi(I)$ zove se *optimalno rješenje* instance I za problem Π.

Problem Π se može promatrati kao *minimizacijski problem* odnosno *maksimizacijski problem*, drugim riječima, $f_\Pi(I,s) \geq OPT_\Pi(I)$ za svaki $s \in S_\Pi(I)$ odnosno $f_\Pi(I,s) \leq OPT_\Pi(I)$ za svaki $s \in S_\Pi(I)$.

PRIMJEDBA 2: U formalnoj definiciji razreda NP i NPC koriste se problemi odlučivanja čije rješenje je ili DA ili NE koji se interpretiraju kao formalni jezici, odnosno podskup od $\{0,1\}^*$ (vidjeti npr. [84],[78]). Jezik se sastoji od svih stringova koji kodiraju DA instance problema odlučivanja. Turingov stroj služi kao *verifikator* koji određuje pripada li neki string tom jeziku ili ne. Svi NP optimizacijski mogu se svesti na probleme odlučivanja, naime, problem odlučivanja nekog Π optimizacijskog problema sastoji se od parova (I,B) gdje je I instanca od Π, a B neki racionalni broj. Ukoliko je Π minimizacijski problem tada je odgovor za pripadnu inačicu problema odlučivanja DA ako i samo ako

postoji $I \in D_\Pi$ cijene $\leq B$. Analogno se formulira za maksimizacijski problem. U tom slučaju, kaže se da je (I, B) DA instanca, inače je NE instanca. Težina nekog NP optimizacijskog problem je pokazana tako da se pokaže da je pripadni problem odlučivanja NP-težak. Stoga, kad kažemo da neki optimizacijski problem pripada nekom razredu složenosti, u principu, podrazumijevamo da se formalno može pokazati da njegova inačica kao problem odlučivanja pripada tom razredu složenosti.

Nekoliko kombinatornih optimizacijskih problema bit će od koristi u ovom doktorskom radu:

DEFINICIJA 17: Neka je dan skup od n elemenata $U = \{e_1, e_2, \ldots, e_m\}$ i familija podskupova od U, $\mathcal{S} = \{S_1, S_2, \ldots, S_n\}$ tako da je $\bigcup_{S \in \mathcal{S}} S = U$. Instancu sljedećih problema predstavlja sustav skupova $(U, \mathcal{S})$.

> SET-COVER problem predstavlja pronalaženje najmanjeg pokrivača $\mathcal{C} \subseteq \mathcal{S}$ tako da je $\bigcup_{S \in \mathcal{C}} S = U$.
>
> SET-MULTI-COVER problem predstavlja pronalaženje najmanjeg pokrivača $\mathcal{C} \subseteq \mathcal{S}$ tako da za svaki element $e \in U$ postoji d_e različitih skupova u $\mathcal{C}$ za koje $e \in S$.
>
> HITTING-SET problem predstavlja pronalaženje najmanjeg podskupa elemenata $E \subseteq U$ takvog da za svaki $S \in \mathcal{S}$ postoji $e \in E$ za koje $e \in S$.
>
> PRIZE-COLLECT-SET-COVER problem predstavlja pronalazak pokrivača $\mathcal{C} \subseteq \mathcal{S}$ i podskup elemenata $E \subseteq U$ koji minimiziraju broj skupova u pokrivaču $\mathcal{C}$ zajedno s brojem nepokrivenih elemenata $U \setminus E$.

U prethodnoj definiciji, svi definirani problemi imaju težinske inačice, odnosno umjesto nalaženja pokrivača s najmanjim kardinalitetom traži se pokrivač najmanje težine. U problemu HITTING-SET težinska je funkcija definirana na elementima U, a u problemu PRIZE-COLLECT-SET-COVER osim težine na skupovima, elementi mogu imati pridružen profit te je onda cilj minimizirati težinu pokrivača zajedno s profitom koji se plaća za nepokrivene elemente.

Ilustracije radi, SET-COVER problem je NP optimizacijski problem Π, instancu problema I čini par $(U, \mathcal{S})$. Skup svih dopustivih rješenja $S_\Pi(I)$ jest skup svih podfamilija $\mathcal{F}$ od $\mathcal{S}$ koji pokrivaju U. Ukoliko

je dana težinska funkcija na skupovima $c\colon \mathcal{S} \to \mathbb{Q}_+$ onda ona čini težinsku funkciju optimizacijskog problema, dakle, za $s \in \mathcal{F}$, $f_\Pi(I, s) = \sum_{x \in s} c(x)$. Vrijednost optimalnog rješenja SET-COVER je $OPT_\Pi(I) = \min_{s \in \mathcal{F}} \sum_{x \in s} c(x)$.

Uz uvriježenu hipotezu P $\neq$ NP, mnoge NP-teške optimizacijske probleme se zbog svoje neprivlačnosti, pokušava aproksimirati algoritmom u polinomnom vremenu uz teorijsku garanciju kvalitete aproksimacije. Iscrpan pregled metoda za dizajn aproksimacijskih algoritama može se naći u [84] i [41].

DEFINICIJA 18: Neka je Π optimizacijski problem i neka $\alpha\colon \mathbb{Z}_+ \to \mathbb{Q}_+$, $\alpha(n) \geq 1, \forall n \in \mathbb{Z}_+$. Algoritam $\mathcal{A}$ je *α-aproksimacijski algoritam za* Π ukoliko za svaku instancu I problema Π algoritam $\mathcal{A}$ vraća dopustivo rješenje s za I tako da je

$$f_\Pi(I, s) \leq \alpha(|I|) \cdot OPT_\Pi(I)$$

u slučaju minimizacije, odnosno

$$f_\Pi(I, s) \geq \alpha(|I|) \cdot OPT_\Pi(I)$$

u slučaju maksimizacije.
Vrijeme izvršenja od $\mathcal{A}$ je ograničeno fiksnim polinomom u ovisnosti o $|I|$.

Sljedeći pojmovi su definirani za minimizacijski optimizacijski problem, definicije su analogne za maksimizacijski optimizacijski problem.

Neka je $\mathcal{A}$ aproksimacijski algoritam za minimizacijski problem Π i neka je $\mathcal{A}(I)$ dopustivo rješenje koje vraća $\mathcal{A}$ za neku instancu I problema Π.

Prema definiciji, za aproksimacijski algoritam $\mathcal{A}$ problema Π kažemo da ostvaruje aproksimacijski omjer $R_\mathcal{A}$ ukoliko

$$R_\mathcal{A} \geq \max_{I \in D_\Pi} \frac{OPT_\Pi(I)}{f_\Pi(I, \mathcal{A}(I))}$$

Iz definicije od $R_\mathcal{A}$ može se primjetiti da $R_\mathcal{A} \geq 1$ u slučaju minimizacije.

Kaže se da optimizacijski problem Π se ne može aproksimirati unutar aproksimacijskog omjera R ukoliko za svaki aproksimacijski algoritam $\mathcal{A}$ za Π postoji instanca I od Π tako da je

$$R < \frac{OPT_\Pi(I)}{f_\Pi(I, \mathcal{A}(I))}.$$

U tom slučaju govorimo o *neaproksimabilnosti* problema unutar R faktora pod pretpostavkom P $\neq$ NP.

Neki aproksimacijski algoritmi dopuštaju proizvoljnu aproksimabilnost:

DEFINICIJA 19: Neka je Π optimizacijski minimizacijski problem s funkcijom cilja f_Π. Za $\mathcal{A}$ kaže se da je *aproksimacijska shema za* Π ukoliko za ulaz (I, ϵ) gdje je I instanca od Π i $\epsilon > 0$ parametar pogreške, vraća rješenje s za koje je

$$f_\Pi(I, s) \leq (1 + \epsilon) \cdot OPT_\Pi(I)$$

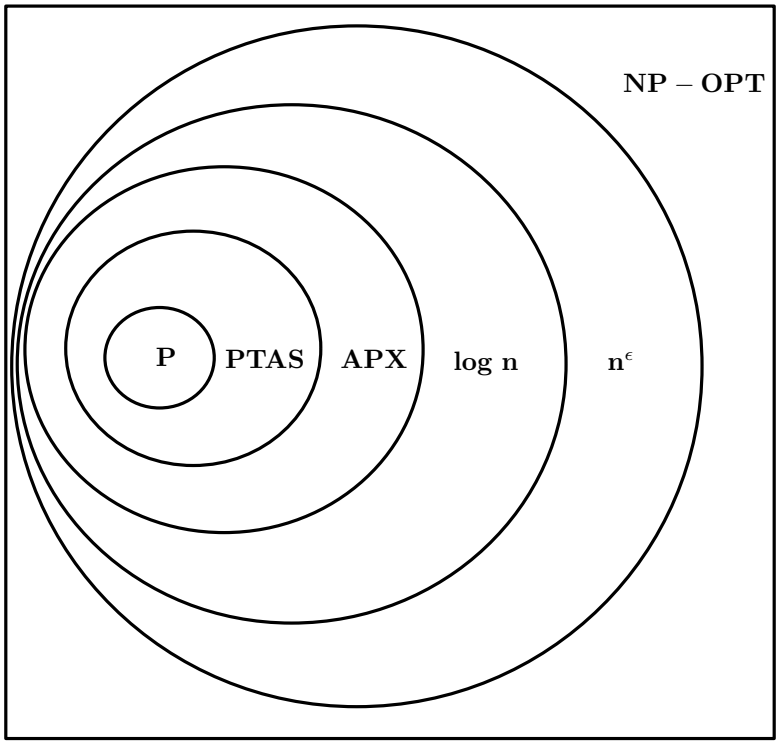
Za $\mathcal{A}$ kaže se da je *aproksimacijska shema u polinomnom vremenu*, u oznaci PTAS, ukoliko za svaki fiksni $\epsilon > 0$ vrijeme izvršenja od $\mathcal{A}$ je ograničeno polinomom u ovisnosti o veličini instance I.

Ukoliko se u prethodnoj definiciji zahtjeva da vrijeme izvršenja algoritma bude ograničeno polinomno u ovisnosti o veličini instance I i parametru $1/\epsilon$ tada se $\mathcal{A}$ zove *aproksimacijska shema u potpunom polinomnom vremenu* i skraćeno piše FPTAS.

1.2.3 *Razredi aproksimabilnosti*

Promatrajući karakter aproksimabilnosti NP optimizacijskih problema mogu se podijeliti u nekoliko istaknutih razreda, kako je predloženo u [23]:

- **NP − OPT** razred – razred koja predstavlja skup svih kombinatornih optimizacijskih problema u NP.
- $\mathbf{n}^\epsilon$ razred – optimizacijski problem se nalazi u razredu $\mathbf{n}^\epsilon$ ukoliko postoji $\epsilon > 0$ takav da se problem može aproksimirati unutar aproksimacijskog omjera $O(n^\epsilon)$ gdje je n veličina instance problema. Za problem se kaže da je $\mathbf{n}^\epsilon$-težak ukoliko postoji $\epsilon > 0$ takav da ne postoji aproksimacijski algoritam koji postiže $\Theta(n^\epsilon)$ aproksimacijski omjer.
- **log n** razred – problem se nalazi u klasi **log n** ukoliko se može aproksimirati unutar aproksimacijskog omjera $O(\log n)$ gdje je n veličina instance problema. Za problem kažemo da je **log n**-težak ukoliko ne postoji aproksimacijski algoritam za problem koji ostvaruje aproksimacijski omjer R za $R \in \Theta(\log n)$.



Slika 8: Inkluzija aproksimacijskih klasa

- **APX** razred – optimizacijski problem se nalazi u **APX** klasi ukoliko se može aproksimirati unutar $1+\epsilon$ aproksimacijskog omjera za neki $\epsilon > 0$. Za problem kažemo da je **APX** težak ukoliko postoji $\epsilon > 0$ za kojeg se problem ne može aproksimirati unutar $1+\epsilon$ aproksimacijskog omjera.
- **PTAS** razred – optimizacijski problem se nalazi u **PTAS** ukoliko se može aproksimirati unutar $1+\epsilon$ aproksimacijskog omjera za bilo koji $\epsilon > 0$.
- **P** razred – optimizacijski problem se nalazi u razredu **P** ukoliko se može optimalno riješiti u polinomnom vremenu.

Za razrede aproksimabilnosti vrijedi inkluzija:

$$\mathbf{P} \subset \mathbf{PTAS} \subset \mathbf{APX} \subset \mathbf{log\ n} \subset \mathbf{n^{\epsilon}} \subset \mathbf{NP - OPT}.$$

Primjerice, problem Set-Cover pripada **log n** razredu i on je **log n**-težak [27]. Discrete-1.5D-Terrain-Guard problema pripada **PTAS** razredu [33]. U ovom doktorskom radu cilj je pokazati da problemi

WEIGHTED-1.5D-TERRAIN-GUARD i 1.5D-TERRAIN-MULTI-GUARD pripadaju barem **APX** klasi. Uz pretpostavku P $\neq$ NP, FPTAS iz klase **PTAS** je algoritamski najpoželjnije rješenje koje se može ostvariti za NP-težak problem jer se garantira proizvoljna aproksimabilnost problema i poznata je eksplicitna zavisnost vremenske složenosti o ϵ parametru.

Mjera za složenost sustava skupova SET-COVER problema $(U, \mathcal{S})$ se pokazala u novije vrijeme bitna komponenta u aproksimabilnosti geometrijskih SET-COVER problema [11], [16]. Jedna od takvih mjera dali su Vapnik i Chervonenkis u [83]:

DEFINICIJA 20: Neka je dan sustav skupova $(U, \mathcal{S})$. Za sustav skupova $(U, \mathcal{S})$ kažemo da ima VC-dimenziju k ukoliko je k kardinalitet najvećeg podskupa X od U takav da za svaki podskup $Y \subseteq X$ postoji $S \in \mathcal{S}$ za koje $Y \subseteq S$ i $S \cap (X \setminus Y) = \emptyset$.

1.3 LINEARNA OPTIMIZACIJA

Linearno programiranje ima značajnu ulogu u dizajnu kombinatornih optimizacijskih algoritama. Dobar uvod u područje linearne optimizacije može se naći u [9] ili u kontekstu konveksne optimizacije [10]. Tehnike temeljene na teoriji linearnog programiranja za dizajn aproksimacijskih algoritama se može naći u [84].

1.3.1 *Linearno programiranje*

Linearno programiranje predstavlja problem uvjetne optimizacije linearnog funkcionala (minimizacija ili maksimizacija) gdje su uvjeti formulirani kao linearne nejednakosti.

Minimizacijski linearni problem (LP) postavljen je u jednostavnom obliku na sljedeći način:

$$\min \sum_{j=1}^{n} c_j x_j \tag{LP1}$$

uz uvjete

$$\sum_{j=1}^{n} a_{i,j} x_j \geq b_i \qquad i = 1, 2, \ldots, m \tag{1}$$

$$x_j \geq 0 \qquad j = 1, 2, \ldots, n \tag{2}$$

gdje je $x = (x_1, x_2, \ldots, x_n), x_i \in \mathbb{R}$ vektor varijabli po kojem se vrši optimizacija, varijable uvjeta $a_{i,j} \in \mathbb{R}$ zajedno s vrijednostima $b_i \in \mathbb{R}$ određuju linearne uvjete koje mora zadovoljiti neko rješenje x te vektor $c = (c_1, c_2, \ldots, c_n), c_i \in \mathbb{R}$ opisuje funkciju f koju treba optimizirati, dakle, $f(x) = \sum_{j=1}^{n} c_j x_j$ koja se zove *funkcija cilja*.

Linearni program LP1 se može kraće zapisati kao:

$$\min_{x \in \mathbb{R}^n} \{c^T x \colon Ax \geq b, x \geq 0\} \tag{3}$$

gdje je A *matrica uvjeta* i b, c vektori:

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & \ldots & a_{1,n} \\ a_{2,1} & a_{2,2} & \ldots & a_{2,n} \\ \vdots & \vdots & \ldots & \vdots \\ a_{m,1} & a_{m,2} & \ldots & a_{m,n} \end{bmatrix}, c = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix}, b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix},$$

a operator nejednakosti $\geq$ u matričnom smislu podrazumijeva nejednakost po komponentama vektora.

Za bilo koje rješenje $x = (x_1, x_2, \ldots, x_n)$ za koje vrijede linearni uvjeti (1) i (2) kaže se da je *dopustivo rješenje* za LP1, inače, riječ je o *nedopustivom rješenju* za LP1.

DEFINICIJA 21: Neka je $\mathcal{P} = \{x \colon Ax \geq b, x \geq 0\}$ skup dopustivih rješenja za LP1, odnosno skup svih točaka u $\mathbb{R}^n$ koje se nalaze u presjeku poluprostora $\bigcap_{i=1}^{m} \{a_i^T x \geq b_i\}$ s poluprostorom $\bigcap_{j}^{n} \{x_j \geq 0\}$ kojeg zovemo *poliedar*.

Za točku $x \in \mathcal{P}$ kažemo da je *vrh poliedra* $\mathcal{P}$ ako postoji $a \in \mathbb{R}^n$ takav da je $a^T x < a^T y$ za svaki $y \in \mathcal{P}, y \neq x$.

DEFINICIJA 22: *Problem linearnog programiranja* predstavlja problem pronalaženja $x^* \in \mathcal{P}$ tako da je $f(x^*) \leq f(x)$ za sve $x \in \mathcal{P}$ gdje je $\mathcal{P}$ pripadajući poliedar, a f pripadajuća funkcija cilja.

Nužan i dovoljan uvjet za postojanje i nalaženje rješenja problema linearnog programiranja okarakterizirana je sljedećim teoremom:

TEOREM 1.1 ([10]). *Problem linearnog programiranja LP1 ima rješenje ako i samo ako je poliedar $\mathcal{P}$ neprazan i ako je funkcija cilja f omeđena odozdo na tom poliedru. Ako rješenje postoji, postiže se na barem jednom vrhu poliedra $\mathcal{P}$.*

Koristeći Lagrangeovu metodu multiplikatora za uvjetnu optimizaciju formulira se sljedeći linearni program:

$$\max \sum_{i=1}^{m} b_i y_i \tag{LP2}$$

uz uvjete

$$\sum_{i=1}^{m} a_{i,j} y_i \leq c_j \qquad j = 1, 2, \ldots, n \tag{4}$$

$$y_i \geq 0 \qquad i = 1, 2, \ldots, m$$

Problem LP1 zove se *primalni*, a problem LP2 *dualni linearni program*. U skladu s tim nazivima, varijable x se zovu *primalne varijable*, a linearni uvjeti (1) *uvjeti primala* odnosno varijable y zovu se *dualne varijable*, a linearni uvjeti (4) *uvjeti duala*.

Jedan od centralnih rezultata linearnog programiranja jest *teorem LP dualnosti*:

Teorem 1.2 (Teorem LP dualnosti). *Primalni program ima konačan optimum ako i samo ako njegov dual ima konačan optimum. Štoviše, ako su* $x^* = (x_1^*, x_2^*, \ldots, x_n^*)$ *i* $y^* = (y_1^*, y_2^*, \ldots, y_n^*)$ *optimalna rješenja za primalni odnosno dualni program tada vrijedi*

$$\sum_{j=1}^{n} c_j x_j^* = \sum_{i=1}^{m} b_i y_i^*.$$

i njegova slabija varijanta:

Teorem 1.3 (Slabi teorem LP dualnosti). *Ako su* $x = (x_1, x_2, \ldots, x_n)$ *i* $y = (y_1, y_2, \ldots, y_m)$ *dopustiva rješenja primalnog odnosno dualnog programa, tada vrijedi sljedeća nejednakost:*

$$\sum_{j=1}^{n} c_j x_j \geq \sum_{i=1}^{m} b_i y_i$$

Jednakost vrijedi jedino u slučaju kad su x i y optimalna rješenja, štoviše, optimalna rješenja imaju odgovarajuća komplementarna svojstva:

Teorem 1.4 (Primal-dual komplementarnost). *Neka su* x *i* y *primalna odnosno dualna dopustiva rješenja. Rješenja* x *i* y *su optimalna ako i samo ako su zadovoljena oba sljedeća svojstva:*

Uvjeti primalne komplementarnosti
Za svaki $1 \leq j \leq n$ *ili* $x_j = 0$ *ili* $\sum_{i=1}^{m} a_{i,j} y_i = c_j$.

UVJETI DUALNE KOMPLEMENTARNOSTI
Za svaki $1 \leq i \leq m$ *ili* $y_i = 0$ *ili* $\sum_{j=1}^{n} a_{i,j} x_i = b_i$.

1.3.2 *Cjelobrojno linearno programiranje*

Nemali broj NP optimizacijskih problema se mogu formulirati kao linearni program u kojima varijable moraju biti cjelobrojne.

DEFINICIJA 23: Neka je dana racionalna matrica A i racionalni vektori b i c. *Problem cjelobrojnog linearnog programiranja* (ILP) jest određivanje minimizatora $x \in \mathbb{Z}_+^n$ za minimizacijski problem

$$\min_{x \in \mathbb{Z}_+^n} \{c^T x \colon Ax \geq b\}. \tag{5}$$

Relaksacija cjelobrojnog linearnog programa predstavlja linearni program definiran kao (5) u kojem uvjet $x \in \mathbb{Z}_+^n$ supstituiran s $x \in \mathbb{R}_+^n$.

U kontekstu dualne relacije, za cjelobrojno linearno programiranje općenito vrijedi nejednakost u *Teoremu 1.2* čak i u slučaju optimalnih rješenja:

$$\min_{x \in \mathbb{Z}_+^n} \{c^T x \colon Ax \geq b\} \geq \max_{y \in \mathbb{Z}_+^m} \{b^T y \colon A^T y \leq c\} \tag{6}$$

PRIMJEDBA 3: Ukoliko se uzme $A = [2], b = [1], c = [1]$ cjelobrojno optimalno rješenje primala jest 1, dok za dualni problem cjelobrojno optimalno rješenje jest 0. Dakle, optimalna rješenja se razlikuju. Optimalna vrijednost LP relaksacije jest u slučaja primala i duala $1/2$.

PRIMJEDBA 4: Zahtjev da ulazi cjelobrojnom linearnom programu budu racionalni ima svoju opravdanost u tome što postoje primjeri cjelobrojnih formulacija linearnog programa sa iracionalnim vrijednostima čija optimalna vrijednost jest ograničena, ali se ne može ostvariti u nijednoj cjelobrojnoj vrijednosti. Ilustracija takvog primjera može se naći npr. u [76].

Općenito rješavanje cjelobrojnih linearnih problema je težak problem:

TEOREM 1.5 ([76]). *Problem cjelobrojnog linearnog programiranja je NP-kompletan problem.*

Dakle, prethodni teorem kaže da problem rješavanja ILP-a spada među najteže probleme klase NP i kao takav općenito je nerješiv u polinomnom vremenu pod pretpostavkom P $\neq$ NP. U posebnim slučajevima, ILP problemi se mogu rješiti u polinomnom vremenu. Dobar pregled takvih slučaja i metoda daje [76].

Od predmeta interesa za ovaj doktorski rad bit će klasa cjelobrojnih linearnih programa s posebnim tipom matrice uvjeta.

1.3.3 *Neke specijalne klase cjelobrojnih linearnih programa*

Neki specijalne klase ILP-a mogu se rješiti optimalno u polinomnom vremenu uz određene uvjete. Primjerice, ukoliko ILP ima matricu uvjeta koje ima svojstvo da je svaka poddeterminanta jednaka $0, -1$ ili 1 onda se ILP može cjelobrojno riješiti u polinomnom vremenu [76]. U ovom radu promatraju se razredi ILP čije matrice uvjeta imaju svojstvo uravnoteženosti.

DEFINICIJA 24: Za binarnu matricu A kažemo da je *uravnotežena*[1] ukoliko ne sadrži kvadratnu podmatricu neparnog reda s dvije 1 po retku i po stupcu.

Uravnotežene matrice je promatrao Berge [8] kao poopćenja matrice incidencije vrhova-bridova u bipartitivnim grafovima i odredio karakter pripadnog poliedra.

TEOREM 1.6 ([8]). *Neka je dan LP oblika* $\min_{x\in\mathbb{R}^n}\{c^T x\colon Ax \geq 1, x \geq 0\}$ *gdje je A uravnotežena matrica. Pripadni poliedar* $P = \{x\colon Ax \geq 1, x \geq 0\}$ *sadrži cjelobrojne vrhove.*

Drugim riječima, ILP problem s matricom uvjeta koja je uravnotežena može se rješiti direktno rješavajući LP relaksaciju.

Hoffman i ostali u [42] proučavajući lokacijske probleme na mrežama koje imaju strukturu stabla proširuju pojam uravnoteženih matrica na *totalno uravnotežene matrice*[2]:

DEFINICIJA 25 ([42]): Za uravnoteženu matricu A kažemo da je *totalno uravnotežena* ukoliko ne sadrži kvadratnu podmatricu čija suma po recima odnosno po stupcima je jednaka 2 i nema identičnih stupaca.

U pregledu [58] Kolen i Tamir pokazuju korespondenciju totalno uravnoteženih matrica s matricom incidencije klijenata i postrojenja

1 engl. *balanced*
2 engl. *totally balanced matrices*

koji ih moraju posluživati u tzv. lokacijskim problemima na mrežama i daju optimalne algoritme u polinomnom vremenu koji rješavaju ILP formulaciju tih problema. Algoritmi su temeljeni na primal-dual tehnikama i pretpostavljaju da je matrica incidencije stavljena u standardni pohlepni oblik.

TEOREM 1.7 ([58]). *Matrica A je totalno uravnotežena ako i samo ako se može permutirati u* standardni pohlepni oblik, *odnosno, ako se permutacijom redaka i stupaca ne može dobiti podmatrica*

$$\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}.$$

Upravo prethodni teorem će biti od ključnog značaja za nalaženje cjelobrojnih rješenja problema koje su predstavljeni u ovom doktorskom radu.

1.4 DOPRINOS DOKTORSKOG RADA

Koristeći jedinstveni matematički aparat predstavlja se 4-aproksimacijski algoritam za skoro sve varijante problema težinskog čuvanja 1.5D terena. Za razliku od dosadašnjih geometrijski temeljenih pristupa, koriste se kombinatorne tehnike temeljene na linearnom programiranju i tehnike zaokruživanja koje iskorištavaju strukturalno svojstvo 1.5D terena koje ima korespodenciju s totalnom uravnoteženošću matrice uvjeta pripadne LP formulacije. Pristup je stupnjevit: svaka varijanta problema reducira se na jednostavniji problem za kojeg se nalazi cjelobrojno optimalno rješenje u polinomnom vremenu. Aproksimacijski omjer za svaku varijantu ovisi o načinu podjele problema. Rezultati za težinsko i parcijalno čuvanje 1.5D terena su objavljeni u [25].

Problem čuvanja 1.5D terena s višestrukim pokrivanjem slijedi sličan okvir kao i prethodno opisana inačica te se može aproksimirati s aproksimacijskim faktorom 5. Ključna ideja postupka je podijeliti zahtjeve klijenata koje za posljedicu ima definiranje jednostavnijih varijanti koji se mogu optimalno cjelobrojno riješiti. Konačan rezultat, a i time aproksimacijski omjer, definiran je kombiniranjem rješenja tih varijanti. Rezultati za čuvanje 1.5D terena s višestrukim pokrivanjem su objavljeni u [26].

Prednost ovog matematičkog aparata i algoritama iz njega izvedenih su vrlo jednostavne primjene linearnog programiranja. Sami algo-

ritmi su esencijalno jednostavni za analiziranje za razliku od prijašnjih pristupa s relativno kompliciranim slučajevima [7], [51] čiji opis i analiza uključuje relativno podužu listu slučajeva. Kao dodatak, ovaj pristup omogućuje rješavanje općenitijih inačica problema nego onih koji su predstavljeni u tim radovima: čuvari mogu imati težine te se želi minimizirati težina odabranih čuvara, varijanta u kojoj potrebno pokrivati čitav teren već samo predefinirani dio ili kad se želi robusnije čuvati 1.5D teren - svaka točka koja treba biti čuvana ima zahtjev na broj čuvara koji ju trebaju čuvati. Čini se da su ovakve inačice teže za analizirati ukoliko se koriste samo geometrijske tehnike korištene u [7], [51] za osnovni problem. Valja naglasiti da za mnoge geometrijske probleme skupovnog pokrivača (npr. pokrivanje točaka s proizvoljnim diskovnim radijusima [11]) za koje postoje aproksimacijski algoritmi konstantnog faktora je vrlo nejasno kako te rezultate proširiti na inačice s težinama ili zahtjevima što u pristupu koji će biti predstavljen nije slučaj.

Kako algoritmi prezentirani u ovom radu koriste zaokruživanje frakcionalnog rješenja odgovarajuće formulacije linearnog programa predstavlja se model implementacije rješavatelja linearnog programa. Jedan od alata za rješavanje općenitih linearnih programa jest simpleks metoda koja u praksi radi dobro, premda ima lošu teorijsku granicu za najgori slučaj izvršavanja. Ovaj doktorski rad predstavlja model implementacije aproksimativnog rješavatelja linearnih programa koji opisuju probleme pakiranja i pokrivanja izvorno danog u [31] temeljen na CUDA platformi. Model je testiran na jednoprocesorskom sustavu i CUDA omogućenoj platformi. U ovisnosti o parametru aproksimacije, primjećuje se ubrzanje CUDA implementacije u uspredbi s implementacijom na jednoprocesorskom sustavu te uspoređuje sa simpleks rješavateljem iz GLPK [68] biblioteke. Rezultati implementacije su dani u [69]. Dodatno, sekvencijalna i CUDA implementacija aproksimacijske sheme zajedno s GLPK implementacijom simpleks metode su uspoređeni na instancama 1.5D terena u kojem se i dalje primjećeno ubrzanje CUDA implementacije.

1.5 ORGANIZACIJA DOKTORSKOG RADA

U *poglavlju 1* predstavljeni su problemi vidljivosti u poligonima i terenima. Osnovni pregled teorije složenosti i linearnog programiranja daju pripremu za alate koje će se koristiti u kasnijim poglavljima.

U *poglavlju 2* iznose se neka geometrijska svojstva terena te se daje kratak pregled dosadašnjih pristupa u dizajnu aproksimacijskih algoritama za probleme vidljivosti u poligonima i terenima.

Kombinatorni aspekt čuvanja 1.5D terena proučava se u *poglavljima 3, 4 i 5*. Inačica problema u kojem čuvari mogu imati težine riješen je u *poglavlju 3* u kojem je, kao krajni rezultat, dan aproksimacijski algoritam konstantne aproksimacije za nekoliko varijanti tog problema.

U *poglavlju 4* pristup se prilagođava za inačicu kada klijenti na terenu zahtjevaju određeni broj čuvara da ih pokrivaju te kao rezultat dan je aproksimacijski algoritam konstantne aproksimacije.

U *poglavlju 5* proučava se parcijalna inačica problema u kojem se pokazuje kako se problem može svesti na problem parcijalnog pokrivanja u kojem matrica uvjeta ima svojstvo totalne uravnoteženosti. Aproksimacijski algoritam temeljen je na rezultatu iz [72] za općenite probleme parcijalnog pokrivanja.

Implementacija brzih aproksimativnih rješavatelja linearnih programa problema pakiranja i pokrivanja istražena je u *poglavlju 6*. Model implementacije se temelji na aproksimacijskoj shemi danoj u [31] te je dana paralelna implementacija za CUDA platformu. Ekperimentalnom analizom uspoređena je sekvencijalna i CUDA implementacija aproksimacijske sheme u usporedbi sa simpleks metodom iz GLPK paketa. Ulazi za ekperimente su kvadratne matrice različitih gustoća koje sadrže racionalne odnosno binarne brojeve. Na takvim uzorcima eksperimenata primjećeno je dva reda veličine ubrzanje CUDA implementacije aproksimacijske sheme naspram sekvencijalne inačice aproksimativnog rješavatelja i simpleks metode.

U *poglavlju 7* predložen je implementacijski model algoritama za težinski problem čuvanja 1.5D terena i problem čuvanja 1.5D s višestrukim pokrivanjem koristeći optimalni rješavatelj temeljen na simpleks metodi i aproksimativni rješavatelj temeljen na paralelnoj CUDA implementaciju za rješavanje linearnih programa problema pakiranja i pokrivanja.

Dio I

PROBLEMI ČUVANJA TERENA

2

ČUVANJE TERENA

Problem čuvanja terena se može interpretirati kao posebna vrsta problema čuvanja poligona te se stoga daje pregled osnovnih rezultata za probleme vidljivosti u poligonima koji su bitni za složenost i aproksimabilnost problema čuvanja na 1.5D i 2.5D terenima. Za problem čuvanja 1.5D terena predstavit će se razvoj dizajna aproksimacijskih algoritama te će se istaknuti neka geometrijska svojstva koje će biti od bitne koristi za algoritme koje će biti predstavljeni. Neki od problema vidljivosti za poligone i terene su predstavljeni u pregledu [71].

Sadržaj

2.1 ČUVANJE POLIGONA

Povijesno gledajući, prve formulacije problema vidljivosti definirani su na jednostavnim poligonima i razvojem tehnika za rješavanje postupno su se uvodile različite varijacije poligona.

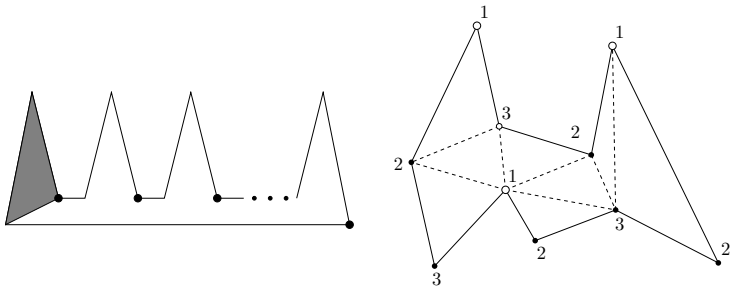
2.1.1 *Nužni i dovoljni broj čuvara za čuvanje poligona*

Jedan od prvih rezultata po pitanju dovoljnog broja čuvara je dao Chvátal u [18] kao odgovor na pitanje V. Klee za varijantu čuvanja jednostavnog poligona iz njegovih vrhova.

TEOREM 2.1 (Chvátalov teorem). *Neka je $V(P)$ skup vrhova nekog jednostavnog poligona P s n vrhova. Za čuvanje P dovoljno je, a ponekad i nužno $\lfloor n/3 \rfloor$ čuvara iz $V(P)$.*

Elegantniji dokaz Chvátalovog teorema ponudio je Fisk koristeći svojstvo *kromatskog broja* grafa induciranog trianguliranjem poligona [28]. Dokaz je konstruktivan u smislu što direktno opisuje algoritam postavljanja čuvara. Tek do novijeg vremena, većina algoritama je temeljena na postavljanju gornje granice na broj čuvara potrebnih za čuvanje poligona. Avis i Toussaint u [4] su pokazali kako, oponašajući Fiskov dokaz, mogu riješiti problem u $O(n \log n)$ vremenu.

Nužnost $\lfloor n/3 \rfloor$ čuvara se ogleda u primjeru "češalj" poligona COMB_n na *slici 9(a)* gdje svaki džep poligona (trokuti na gornjem dijelu poligona) čuva najmanje jedan čuvar iz vrha.



(a) COMB_n poligon sa sivo označenim džepom

(b) trianguliranje poligona

Slika 9: Nužnost $n/3$ čuvara i Fiskov dokaz

FISKOV DOKAZ I ALGORITAM ZA POSTAVLJANJE ČUVARA *Trianguliranje* poligona P je podjela poligona P u skup trokuta koji su u parovima disjunktni i vrhovi tih trokuta su vrhovi poligona P, a stranice trokuta su rubovi poligona P ili dijagonale koje spajaju 2 nesusjedna vrha u poligonu P.

Za dano trianguliranje T poligona P definira se graf $GT(P)$ na način da su mu vrhovi upravo vrhovi poligona P, a bridovi stranice trokuta iz trianguliranja.

Kromatski broj grafa G je najmanji prirodni broj k različitih boja za kojeg je bojanje tog grafa moguće u smislu da nikoja 2 susjedna vrha nisu obojana istom bojom. U tom slučaju, kaže se da G ima kromatski broj k.

Fiskov dokaz temeljen je na sljedećim činjenicama:

LEMA 2.1 ([82]). *Svaki se jednostavni poligon može triangulirati. Štoviše, svaka triangularizacija jednostavnog poligona od n vrhova sadrži $n-2$ trokuta.*

LEMA 2.2 ([82]). *Neka je P jednostavni poligon i T triangularizacija poligona P. Tada $GT(P)$ ima kromatski broj 3.*

FISKOV DOKAZ Neka je P jednostavni poligon s n vrhova. Vrhove induciranog grafa trianguliranja $GT(P)$ po *Lemi 2.1* i *Lemi 2.2* može se klasificirati u 3 disjunktna skupa C_1, C_2, C_3. Neka su $\{1,2,3\}$ oznake tih klasa. Očito, jedna od particija, npr. C_1 sadrži $\lfloor \frac{n}{3} \rfloor$ točaka. Kako svaki trokut u trianguliranju sadrži sve 3 oznake, dovoljno je staviti čuvara na vrhove iz C_1. Tvrdnja teorema dalje slijedi iz činjenice da se svaki trokut može čuvati čuvarom iz jednog vrha i da vrhovi poligona tvore vrhove grafa $GT(P)$.

Direktno iz ovog rezultata može se dati algoritam postavljanja čuvara u vrhove jednostavnog poligona:

ALGORITAM 1 Algoritam pokrivanja poligona od n vrhova s $\lfloor n/3 \rfloor$ čuvara

```
1: procedure Polygon-Vertex-Cover((P))
2:     G ← Triangulate(P)
3:     {C1, C2, C3} ← 3-Color-Graph(G)
4:     return C1
5: end procedure
```

TEOREM 2.2. Algoritam 1 *se može implementirati u* $O(n \log n)$ *vremenu gdje je* n *broj vrhova jednostavnog poligona* P.

Dokaz. Učinkovita implementacija u linearnom vremenu procedure TRIANGUALATE za trianguliranje jednostavnog poligona s n vrhova dano je u [15]. Procedura 3-COLOR-GRAPH(G) definira 3 particije vrhova $G := GT(P)$ na temelju 3-obojivosti tog grafa, koja se, prema [4], može odrediti *podijeli-pa-vladaj* strategijom u $O(n \log n)$ vremenu. Algoritam vraća jednu od particija vrhova grafa na koje se mogu postaviti čuvari u pripadnom poligonu. □

2.1.2 *Nalaženje optimalnog broja čuvara za čuvanje poligona*

Optimizacijski problemi za čuvanje poligona, dakle, MIN-VERTEX-POLYGON-GUARD, MIN-EDGE-POLYGON-GUARD i MIN-POINT-POLYGON-GUARD gdje je pripadni poligon jednostavni ili sadrži rupe pripada razredu NP-teških problema pokazano je u [65] i [2]. Dokaz je temeljen na redukciji problema čuvanja poligona u polinomnom vremenu na 3-SAT problem. Eidenbenz, Stamm i Widmayer su u [24] pokazali da su varijante problema MIN-VERTEX-POLYGON-GUARD, MIN-EDGE-POLYGON-GUARD i MIN-POINT-POLYGON-GUARD u kojima poligon ne sadrži rupe **APX**-teški odnosno **log n**-teški ukoliko poligon sadrži rupe.

Jedan od prvih aproksimacijskih rezultata dao je Ghosh u [32] koji koristi tehniku podjele danog poligona (bez rupa ili s rupama) u polinomno mnogo komponenti sa svojstvom da svaka točka unutar neke komponente jest viđena istim skupom čuvarima iz vrhova. Ovaj način diskretizacije svodi problem na instancu SET-COVER problema koje standardnim tehnikama rješavanja za takvu vrstu problema nalazi aproksimativno rješenje unutar $O(\log n)$ aproksimacijskog omjera gdje je n broj vrhova poligona. Algoritam radi u $O(n^4)$ vremenu. Nedavno su, neovisno, Kwon *et al* u [47] i King [53] ubrzali vrijeme izvršavanja na $O(n^3)$. King je aproksimacijski faktor za jednostavni poligon s n vrhova poboljšao na $O(\log \log \mathrm{OPT})$ za čuvanje iz vrhova gdje je OPT broj optimalnih čuvara koji pokrivaju poligon.

Premda aproksimacijski omjer $O(\log n)$ dolazi iz općenite formulacije SET-COVER problema zbog razloga što pokrivač skup može biti bilo koji podskup elemenata, Ghosh u [32] smatra da se taj aproksimacijski omjer ne može niti ostvariti zbog geometrijskih restrikcija samog poligona. Prvi egzaktni algoritam temeljen na realnoj algebarskoj geometriji dali su Efrat i Har-Peled [22] koji radi u $O((n\mathrm{OPT})^{3(2\mathrm{OPT}+1)})$

vremenu gdje OPT predstavlja optimalni broj čuvara proizvoljno postavljanjih unutar poligona. Dodatno, daju aproksimacijski algoritam temeljen na redukciji problema s proizvoljnim lokacijama čuvara u poligonu na inačicu u kojoj su čuvari smješteni na gustoj mreži unutar poligona te dobivaju $O(\log \mathrm{OPT})$ aproksimaciju u očekivanom vremenu $O(n\mathrm{OPT}^2 \log n \log(n\mathrm{OPT}) \log^2 \Delta)$ u ovisnosti o omjeru dijametra poligona i veličini mreže Δ.

Kröller *et al* u [62] su dali kombinatorni algoritam temeljen na linearnom programiranju u kojem čuvari u poligonu s rupama mogu biti postavljeni na proizvoljnim lokacijama unutar poligona i potrebno je čuvati čitav poligon. Njihov pristup reducira linearni program s neprebrojivo mnogo uvjeta na diskretni linearni program te rješavaju iterativno separacijske probleme tako da utvrđuju točke koje narušavaju uvjete primala i duala. Te točke definiraju nove separacijske probleme za primal i dual. Postupak se zaustavlja kada nema točaka koji narušavaju uvjete. Vraćeno rješenje primala i duala predstavlja donju i gornju među na optimalno rješenje čuvanja poligona.

2.2 ČUVANJE 2.5D TERENA

Eidenbenz i ostali [24] razriješili su problem neaproksimabilnosti čuvanja 2.5D terena. Pokazuju kako se čuvanje 2.5D terena ne može aproksimirati bolje od logaritamskog faktora. Pristup baziraju na redukciji problema RESTRICTED-SET-COVER, odnosno SET-COVER u kojem sustav skupova $(U, \mathcal{S})$ ima svojstvo $|\mathcal{S}| \leq |U|$, na problem POINT-POLYGON-GUARD u kojem poligon ima rupe. Poligon trianguliraju i oblikuju tako da dobiju 2.5D teren.

Cjelovit rezultat je izražen teoremom:

TEOREM 2.3. *[24] Problem* POINT-2.5D-TERRAIN-GUARD *i problem* VERTEX-2.5D-TERRAIN-GUARD *se ne može aproksimirati algoritmom u polinomnom vremenu unutar* $((1-\epsilon)/12) \ln n$ *za bilo koji* $\epsilon > 0$*, gdje je* n *broj vrhova 2.5D terena.*

2.3 ČUVANJE 1.5D TERENA

Kako je predstavljeno u prethodnom odjeljku, rješavanje problema čuvanja 2.5D terena je jednako teško kao i rješavanje općenitog SET-COVER problema. Ukoliko se problemi čuvanja na terenu promatraju

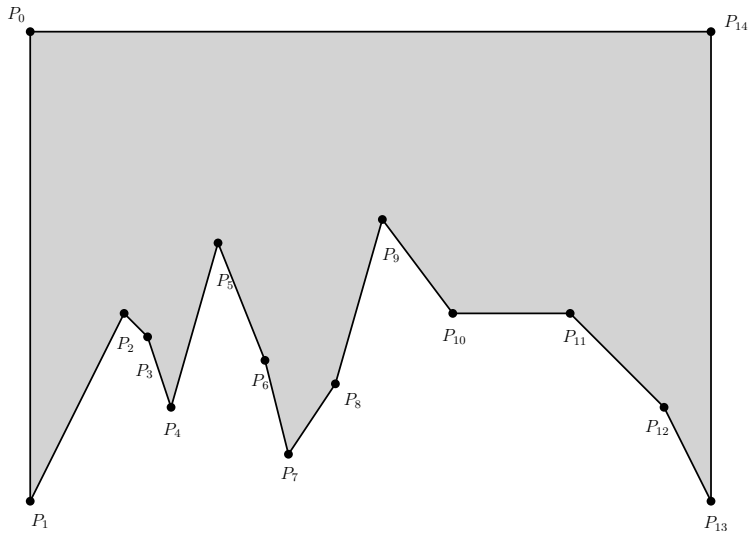
na 1.5D terenima pokazuje se da se ti problemi mogu dobro aproksimirati.

2.3.1 *Neka svojstva 1.5D terena*

Neka je dan 1.5D teren T predstavljen poligonalnom linijom $P_1P_2 \dots P_n$ s n vrhova. Uvođenjem dodatne 2 točke na T može se formirati jednostavni poligon. Dakle, u ravnini se odaberu 2 točke "vertikalno dovoljno daleko" od P_1 i P_n, odnosno $P_0 = (x_{P_1}, y_\infty)$ te $P_{n+1} = (x_{P_n}, y_\infty)$ za $y_\infty > \max\{y_i \colon P_i = (x_i, y_i), i = 1, 2, \dots, n\}$. Definiran je jednostavni poligon P_T s $n+2$ vrha na sljedeći način: vrhovi poligona čini uređeni skup $P_0, P_1, P_2, \dots P_n, \dots P_{n+1}$ s bridovima

$$\delta P_T = \{\overline{P_0P_1}, \overline{P_1P_2}, \overline{P_2P_3}, \ \dots, \overline{P_{n-1,n}P_{n,n+1}}, \overline{P_{n+1}P_0}\}.$$

$IntP_T$ je zatvoren i omeđen povezan skup.



Slika 10: x-monotona poligonalna linija T s 13 vrhova kao dio jednostavnog poligona P_T

Gornja ograda na broj potrebnih čuvara iz vrhova 1.5D terena može se naći u $O(n \log n)$ vremenu gdje je n broj vrhova terena. Zaista, neka je dan teren T dimenzije n i pridruženi mu poligon P_T. Po *Teoremu 2.1* poligon P_T uvijek mogu čuvati $\lfloor (n+2)/3 \rfloor = \lceil n/3 \rceil$ čuvara iz vrhova. Odnosno, postoji 3 disjunktne particije C_1, C_2, C_3 vrhova od P_T koja svaka ima najviše $\lceil n/3 \rceil$ elemenata. Ukoliko se uzmu čuvari na vrhovima isključivo iz jedne particije, recimo C_1, onda će poligon P_T biti pokriven. Ukoliko neki vrh iz C_1 nije iz T može se zamijeniti

sa susjednim vrhom iz druge particije koji se nalazi u T. Takav vrh postoji zbog *Leme 2.1* i konstrukcije poligona P_T. Ti čuvari mogu se naći u $O(n \log n)$ vremenu koristeći *Algoritam 1*.

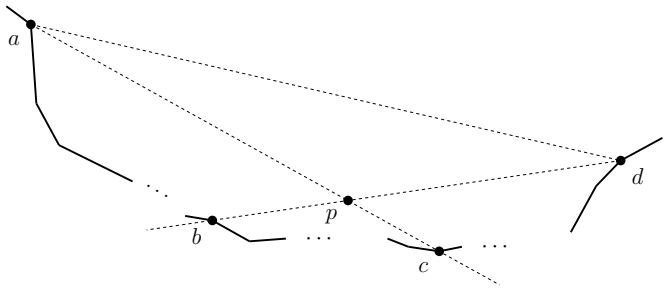
PROPOZICIJA 2.1. *Postoji algoritam u $O(n \log n)$ vremenu koji nalazi dopustiv skup čuvara veličine $\lceil n/3 \rceil$ za problem* VERTEX-1.5D-TERRAIN-GUARD *za 1.5D teren složenosti n.*

Neka je dan 1.5D teren T veličine n i neka je P_T poligon koji opisuje teren T. Uvjet x-monotonosti daje nužan i dovoljan uvjet čuvanja unutrašnjosti poligona P_T:

LEMA 2.3. *Skup točaka $S \subseteq T$ čuva čitav P_T ako i samo ako S čuva čitav T.*

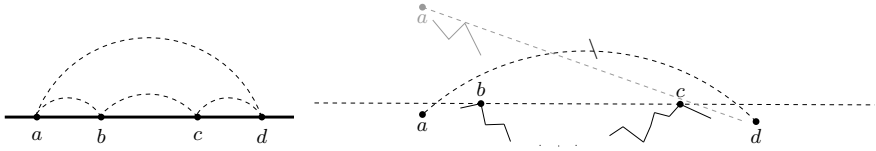
Svojstvo x-monotonosti 1.5D terena pokazalo se od presudne važnosti za geometrijske algoritme opisane u [51] i [7] te u isto tako u ovom doktorskom radu za definiranje kombinatornih algoritama u idućim poglavljima.

LEMA 2.4 (*Tvrdnja uređaja*). *Neka su $a \leq b < c \leq d$ četiri točke koje pripadaju 1.5D terenu T. Ako $a \sim c$ i $b \sim d$ tada $a \sim d$.*



Slika 11: Segmenti $\overline{ac}$ i $\overline{bd}$ se sijeku u točki p unutar jednostavnog poligona $abcd$

Dokaz. Neka su $a, b, c, d \in T$ točke tako da je $a \leq b < c \leq d$. Slučaj $a = b$ je trivijalan jer $a \sim d$ jer $b \sim d$. Simetričan argument i za $c = d$. Neka je $a < b$ i $c < d$. Točka b se mora nalaziti ispod ili na segmentu $\overline{ac}$ (inače, b bi smetao vidljivosti između a i c). Slično, točka c mora ležati ispod $\overline{bd}$. Stoga, segmenti $\overline{ac}$ i $\overline{bd}$ se moraju sijeći u barem jednoj točki, recimo, p (vidjeti *sliku 11*). Primjećuje se sada kako nijedna točka terena ne može biti iznad segmenata $\overline{ap}$ i $\overline{pd}$ što implicira da a vidjeti d. □



(a) četvorka točaka (b) nedopustiv 1.5D teren

Slika 12: Četvorka točaka ravnine koja ne kontradiktira *Tvrdnji uređaja*, a nije instanca 1.5D terena

Valja naglasiti kako svaka instanca problema čuvanja 1.5D terena za koje čuvari i klijenti u relaciji vidljivosti ne kontradiktiraju *Tvrdnji uređaja* ne mora nužno biti instanca 1.5D terena. Upravo iz *slike 12(a)* mogu se vidjeti točke a, b, c, d za koje $a < b < c < d$ i $a \sim b$, $b \sim c$, $c \sim d$, $a \sim d$. Na *slici 12(b)* može se vidjeti nemogućnost implementacije u 1.5D teren: točke a, d moraju biti ispod spojnice bc kako bi vrijedilo $a \not\sim c, b \not\sim d$ što je u kontradikciji da $a \sim d$ jer bi barem jedan od a ili d morao biti iznad spojnice bc.

Za svaku točku q 1.5D terena T definira se *područje vidljivosti*

$$\mathcal{V}(q) = \{p \in P_T \colon q \sim p\},$$

odnosno, ukoliko točka vidi samo lijevo

$$\mathcal{V}_L(q) = \{p \in P_T \colon q \sim p, p < q\}$$

i ukoliko vidi samo desno

$$\mathcal{V}_R(q) = \{p \in P_T \colon q \sim p, p > q\}.$$

Neka je N skup točka 1.5D terena T koje trebaju biti pokriveni (klijenti) i skup G točaka terena na koje se želi postaviti čuvare. Čuvanje klijenata iz N sa G je ekvivalentno Set-Cover problemu gdje je sustav skupova dan kao $(N, \{\mathcal{V}(g) \colon g \in G\})$.

King je u [52] pokazao kako VC dimenzija sustava skupova inducirani problemom čuvanja 1.5D terena ima ograničenu VC dimenziju:

Lema 2.5 ([52]). *VC-dimenzija sustava skupova $(N, \{\mathcal{V}(g) \colon g \in G\})$ induciranim skupom čuvara G i skupom klijenata N u problemu čuvanja 1.5D terena je najviše 4.*

Štoviše, King je u [52] koristeći redukciju poligona s rupama na 2.5D teren pokazao kako sustav skupova induciran 2.5D terenom ima neograničenu VC-dimenziju.

2.3.2 *Aproksimacijski rezultati za čuvanje 1.5D terena*

Od 1995. god kad je predložen prvi dokaz za NP kompletnost problema, premda nedovršen, ali naslućivan, stvorilo je niz aproksimacijskih algoritama za problem. Ovdje će se dati kratak pregled algoritamskih tehnika koje su korištene, kao svojevrsni pregled prethodnog rada na problemu čuvanja 1.5D terena.

2.3.2.1 $O(\log \mathrm{OPT})$ *aproksimacija*

Koristeći rezultat od Brönemanna i Goodricha [11] zajedno s *Lemom 2.5* čuvanje 1.5D terena kao SET-COVER problem može se deterministički aproksimirati unutar $O(d \log(d\mathrm{OPT}))$ faktora, gdje je d VC-dimenzija pripadnog sustava skupova. Dakle, pristup je baziran na definiranju HITTING-SET problema za sustav skupova $(N, \{\mathcal{V}(g) : g \in G\})$ za koje se onda može primjeniti aproksimacijska tehnika ϵ-mreže za HITTING-SET probleme ograničene VC-dimenzije d što u konačnici implicira $O(\log \mathrm{OPT})$ aproksimaciju za problem čuvanja 1.5D terena.

Za geometrijske SET-MULTI-COVER probleme rezultat od Chekuri i ostalih [16] pokazao je da postoji $O(\log z^*)$ aproksimacija u kojima je sustav skupova ograničene VC-dimenzije, a z^* predstavlja optimalnu vrijednost LP relaksacije odgovarajuće ILP formulacije SET-MULTI-COVER problema. Zbog *Leme 2.5* slijedi da postoji $O(\log z^*)$ aproksimacija i za problem čuvanja 1.5D terena s višestrukim pokrivanjem.

2.3.2.2 *Aproksimacije konstantnog faktora*

Prvi rezultat konstantnog faktora dali su Ben-Moshe i ostali [7] koristeći *podijeli-pa-vladaj* strategiju za rješavanje DOMINATING-SET problema u grafu induciranim vidljivošću vrhova na 1.5D terenu. Aproksimacijski omjer nisu iskazali eksplicitno, premda se u nekim radovima tvrdilo da je 6 [51]. King je u [51] aproksimacijski faktor za čuvanje 1.5D terena spustio na 5. Clarkson i ostali u [19] pokazali su kako se tehnika ϵ-mreža predstavljena za geometrijske SET-COVER probleme može primjeniti na čuvanje 1.5D terena i dobiti konstantu aproksimaciju za čuvanje 1.5D terena. Valja naglasiti kako je nejasno kako ovi pristupi mogu rješavati težinsku inačicu problema čuvanja 1.5D terena.

2.3.2.3 *PTAS korištenjem metode lokalne pretrage*

Prvi PTAS za čuvanje 1.5D terena na kojem je dan skup čuvara G i skup klijenata N dali su Gibson i ostali [33] koristeći metodu lokalne pretrage. Algoritam započinje s dopustivim skupom čuvara koji iterativno, ukoliko je moguće, popravlja skup čuvara tako da zamjeni najviše b čuvara s nekim manjim brojem čuvara. Vrijeme izvršenja je u ovisnosti o b i asimptotski je $n^{O(b)}$ gdje je n broj vrhova 1.5D terena. Algoritam je baziran na ideji od Chana i Har-Peleda [13] te od Mustafe i Raya [73]. Analiza je temeljena na sljedećoj pretpostavci: za povoljno odabrani $b = O(1/\epsilon^2)$ lokalna pretraga s najviše b zamjena po iteraciji vraća $(1+\epsilon)$-aproksimaciju. Dokaz ove tvrdnje temeljen je na pokazivanju egzistencije planarnog grafa koji povezuje skup čuvara X s optimalnim skupom Y. Neka je $X' = X \setminus Y$ i $Y' = Y \setminus X$. Promatrajući bipartitivni graf s vrhovima $X' \cup Y'$ u kojem svaki vrh $x \in X'$ čini brid s nekim $y \in Y'$ ako i samo ako x, y vide neku točku koja nije čuvana od $X \cap Y$. Pokazujući da je taj graf planaran, koristeći analizu iz Mustafa i Ray [73], slijedi da X' nije proizvoljno veći od Y' što implicira da X nije puno veći od Y. Proizvoljnu $1+\epsilon$ aproksimaciju dobivaju primjenom separatorskih teorema za planarne grafove iz Fredrickson [30]. Za sada je nepoznato može li se pristup proširiti i na težinsku inačicu problema.

2.3.3 *NP težina problema čuvanja 1.5D terena*

Prvi pokušaji dokazivanja NP težine problema čuvanja 1.5D terena započeli su Chen i ostali [17] tvrdeći kako se rezultat može postići koristeći ideju dokaza od Lee i Lin [65] za pokazivanje NP-težine čuvanja poligona reducirajući problem čuvanja 1.5D terena na 3-SAT problem. U radu su osim ovog rezultata dali optimalni algoritam za lijevo odnosno desno čuvanje. Kompletan dokaz nije nikad publiciran, i svaki pokušaj redukcije nailazio je na prepreku *Tvrdnje uređaja*. Nedavno su King i Krohn [54] uspjeli pokazati NP težinu problema koristeći redukciju PLANAR-3-SAT problema na problema čuvanja 1.5D terena. Njihova analiza uključuje diskretnu i neprekidnu inačicu problema. Štoviše, NP-težina dopušta postojanje PTAS-a (pokazano u [33]), ali ne i FPTAS ukoliko P $\neq$ NP.

TEOREM 2.4. *[54] Problem čuvanja 1.5D terena je NP-težak problem.*

2.4 OSVRT NA PROBLEME VIDLJIVOSTI

Doprinos

Ovo poglavlje predstavlja svojevrstan pregled dosadašnjeg rada na problemima čuvanja na 1.5D terenima. U [71] je publiciran pregled problema vidljivosti na različitim tipovima poligona.

Daljni rad

Kako je problem čuvanja 1.5D terena praktički rješeno s dokazom NP-težine i danim PTAS-om za problem, postavlja se pitanje može li se što reći o aproksimabilnosti *maksimalnog čuvanja 1.5D terena*? Dakle, za dani 1.5D teren T i parametar $k \in \mathbb{Z}_+$ treba odrediti skup čuvara G veličine najviše k tako da je duljina pokrivenog terena maksimalna. Kao instanca MAX-k-COVERAGE problema postoji već pohlepni algoritam koji vraća $1 - 1/e$ aproksimaciju. Može li se faktor aproksimacije popraviti ostaje otvoreno pitanje.

3

TEŽINSKI PROBLEM ČUVANJA 1.5D TERENA

U ovom poglavlju predstavlja se općenito prvi 4-aproksimacijski algoritam za problem težinskog čuvanja 1.5D terena. Metodologija rada se temelji na formulaciji linearnog programa problema težinskog čuvanja 1.5D terena te koristeći rješenje tog linearnog programa problem se razdvaja na lijevi i desni potproblem koji se mogu riješiti optimalno zahvaljujući svojstvu potpune uravnoteženosti. Nekoliko varijanti izvornog problema se uzima u obzir i daju aproksimacije. Diskretizacijski postupak pokazuje kako čuvanje čitavog 1.5D terena se može reducirati na konačan skup točaka koje treba pokriti. Rezultat predstavlja unaprijeđivanje rezultata iz [51] u kojem je promatrana klasična geometrijska inačica problema. Napominjemo da za aproksimacijsku shemu predstavljenu u [33] nije još jasno može li se poopćiti i za težinsku inačicu problema.

Sadržaj

3.1 FORMULACIJA LINEARNOG PROGRAMA

Neka je zadana instanca 1.5D terena T, skup čuvara $G \subseteq T$ i skup točaka koje treba čuvati $N \subseteq T$ (klijenti). Neka je dana težinska funkcija $w\colon G \to \mathbb{Q}_+$ na čuvarima s vrijednostima $w(g) \mapsto w_g$ i funkcija zahtjeva $d\colon N \to \mathbb{Z}_+$, $d(p) \mapsto d_p$ na klijentima. Problem utvrđivanja skupa X čuvara iz G minimalne težine u kojem svaki klijent p iz N ima barem d_p čuvara iz X koji ga pokrivaju naziva se WEIGHTED-1.5D-TERRAIN-MULTI-GUARD problem.

Odgovarajuća ILP formulacija za problem WEIGHTED-1.5D-TERRAIN-MULTI-GUARD dana je s

$$\min \sum_{g \in G} w_g x_g \tag{LP3}$$

uz uvjete

$$\sum_{g \in \mathcal{V}(p)} x_g \geq d_p \qquad \forall p \in N \tag{7}$$

$$x_g \in \{0,1\} \qquad \forall g \in G$$

Odabir čuvara $g \in G$ u optimalno rješenje je predstavljeno indikator varijablom $x_g \in \{0,1\}$. Uvjet (7) zahtjeva da nekog klijenta $p \in N$ čuva barem d_p čuvara. Težina skupa odabranih čuvara modelirano je funkcijom cilja. Dakle, optimalno cjelobrojno rješenje LP3 činio bi skup optimalnih čuvara $X \subseteq G$ čija je ukupna vrijednost OPT .

Instanca WEIGHTED-1.5D-TERRAIN-MULTI-GUARD problema jest instanca SET-MULTI-COVER problema sa sustavom skupova $(N, \mathcal{V})$, težinskom funkcijom $w\colon \mathcal{V} \to \mathbb{Q}_+$ i funkcijom zahtjeva $d\colon N \to \mathbb{Z}_+$ gdje $\mathcal{V} = \{\mathcal{V}(g)\colon g \in G\}$. Prema poznatom rezultatu iz aproksimacije SET-MULTI-COVER problema:

THEOREM 3.1. *[84] Problem* WEIGHTED-1.5D-TERRAIN-MULTI-GUARD *kao* SET-MULTI-COVER *instanca se može u polinomnom vremenu aproksimirati unutar* $O(\log n)$ *faktora.*

Za sada, pripada li problem WEIGHTED-1.5D-TERRAIN-MULTI-GUARD barem **APX** razredu ostaje i dalje otvoreno pitanje. Zanimljivo, ako se promatraju restrikcije problema ili na jedinične težine ili na jedinične zahtjeve, problem pripada razredu APX. Štoviše, ukoliko su težine i zahtjevi jedinični, problem pripada **PTAS** klasi prema [33].

Problem WEIGHTED-1.5D-TERRAIN-GUARD predstavljen kao problem cjelobrojnog linearnog programiranja može se formulirati kao LP3 uz jedinični zahtjev, tj. u (7) $d_p = 1, \forall p \in N$. Cilj je riješiti problem WEIGHTED-1.5D-TERRAIN-GUARD kad su $G = N = T$. Valja primjetiti da ILP formulacija LP3 daje jednu vrlo općenitu formulaciju: u prvom redu, može imati neprebrojivo mnogo uvjeta u (7), a u drugom slučaju, rješavanje ILP-a je općenito NP-težak problem. Ispostavlja se da se ILP može riješiti s konačno mnogo uvjeta i proširiti na općeniti slučaj kad $G = N = T$. Glavni izazov je definirati strategiju zaokruživanja LP relaksacije pripadnog ILP-a. Ispostavlja se da ukoliko se problem podijeli u lijevi i desni problem čuvanja, onda se pripadne ILP formulacije mogu riješiti cjelobrojno direktno rješavajući njihove relaksacije što je posljedica potpune uravnoteženosti njihovih matrica uvjeta. Rješenje lijevog i desnog problema čuvanja 1.5D terena definiraju dopustivo rješenje za polazni problem za koje će se pokazati da nije proizvoljno daleko od optimalnog rješenja.

U nastavku postupno će se krenuti od najjednostavnijih varijanti problema i postupno graditi rješenje za problem kad $G = N = T$. Algoritmi koji budu bili predstavljeni bit će definirani za lijevo čuvanje 1.5D terena gdje algoritmi za desno čuvanje slijede simetrično.

3.2 LIJEVO ČUVANJE 1.5D TERENA

Promotrimo prvo problem lijevog čuvanja 1.5D terena u oznaci WEIGHTED-1.5D-TERRAIN-LEFT-GUARD problem, u kojem je skup čuvara i skup klijenata konačan (u slučaju desnog čuvanja, problem se označava s WEIGHTED-1.5D-TERRAIN-RIGHT-GUARD). Napomenimo da za slučaj s jediničnim težinama postoji optimalni algoritam opisan u [17]. Ovdje se daje prvi optimalni algoritam za slučaj s težinama.

Neka je dan teren T sa skupom potencijalnih čuvara $G \subset T$ i skupom klijenata $N \subset T$ te svaki klijent mora biti čuvan s barem jednim čuvarom iz G koji se nalazi strogo lijevo od nj.

Pripadni frakcionalni linearni program za problem lijevog čuvanja glasi:

$$\min \sum_{g \in G} w_g x_g \tag{LP4}$$

uz uvjete

$$\sum_{g \in \mathcal{V}_L(p)} x_g \geq 1 \qquad \forall p \in N \tag{8}$$
$$x_g \geq 0 \qquad \forall g \in G$$

gdje varijabla x_g označava koliko je čuvar g frakcionalno izabran za lijevog čuvara. Vrijednost z^* označava frakcionalnu optimalnu vrijednost od LP4.

Neka je $A \in \{0,1\}^{|N|\times|G|}$ binarna matrica. Matrica A se zove matrica *lijeve vidljivosti* ukoliko odgovara matričnom opisu relacije vidljivosti između čuvara i klijenata za neku instancu lijevog problema čuvanja, tj.

$$A(i,j) := \begin{cases} 1, & \text{ako } g_j < p_i,\ g_j \sim p_i \\ 0, & \text{inače} \end{cases}, \quad 1 \leq i \leq m, 1 \leq j \leq n$$

stoga matrični zapis od LP4 glasi:

$$\min_{x \in \mathbb{R}^n} \{w^T x \colon Ax \geq 1, x \geq 0\}$$

Ključno svojstvo koje se želi pokazati u ovom poglavlju jest da matrica vidljivosti A ima svojstvo potpune uravnoteženosti.

LEMA 3.1. *Neka je A matrica lijeve vidljivosti. Tada je A potpuno uravnotežena.*

Dokaz. Neka je dana matrica lijeve vidljivosti A veličine $m \times n$. Pokazat će se kako se ona može staviti u standarnu pohlepnu formu što je ekvivalentno da je matrica potpuno uravnotežena prema *Teoremu 1.7*.

Retci i stupci matrice A se permutiraju tako da retci poredani odozgo prema dolje odgovaraju klijentima presloženima slijeva na desno odnosno stupci poredani slijeva na desno odgovaraju čuvarima presloženima s desna na lijevo.

Dakle, retci i stupci od A su permutirani tako da za proizvoljne retke $i, j \in \{1, 2, \ldots, |N|\}$, i proizvoljne stupce $k, l \in \{1, 2, \ldots, |G|\}$ vrijedi:

$$i < j \Rightarrow p_i < p_j \quad \& \quad k < l \Rightarrow g_l < g_k.$$

Pretpostavimo da postoji inducirana podmatrica 2×2 oblika

$$H = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$$

čiji retci i stupci su indeksirani s $p_1, p_2 \in N$ odnosno s $g_1, g_2 \in G$. Stoga, poredak $g_2 < g_1 < p_1 < p_2$ označava elemente inducirane podmatrice

$$\begin{array}{c} \begin{array}{cccc} & g_2 & & g_1 & \\ \end{array} \\ \left[\begin{array}{cccc} & \vdots & & \vdots & \\ \dots & 1 & \dots & 1 & \dots \\ & \vdots & & \vdots & \\ \dots & 1 & \dots & 0 & \dots \\ & \vdots & & \vdots & \end{array}\right] \begin{array}{c} \\ p_1 \\ \\ p_2 \\ \\ \end{array} \end{array}$$

Primjenjujući *Tvrdnju uređaja* za $a = g_2, b = g_1, c = p_1$ i $d = p_2$ zaključuje se $g_2 \sim p_2$ što je u kontradikciji s pretpostavkom da A sadrži H. Dakle, A je potpuno uravnotežena.

□

Prema [58], poliedar $\mathcal{P} = \{x \colon Ax \geq 1, x \geq 0\}$ gdje je A potpuno uravnotežena ima cjelobrojne vrhove.

Hoffman i ostali u [42] daju efikasni kombinatorni algoritam u $O(mn)$ vremenu za probleme pokrivanja na mrežama u kojima se matrica uvjeta može permutirati u standardni pohlepni oblik. Dodatno, pokazuju kako svaka matrica koja se može staviti u standarni pohlepni oblik ujedno potpuno uravnotežena i obrnuto. Primal-dual algoritam iz [42] zahtjeva da matrica A bude permutirana u standardni pohlepni oblik, stoga daju algoritam koji prepoznaje potpunu uravnoteženost matrice i permutira istu u standardni pohlepni oblik u vremenu $O(mn + m \log m)$.

Nadalje, pokazat će se da problem lijevog čuvanja se prevede u vrlo jednostavnu proceduru kad je $w_g = 1, \forall g \in G$ u kojem nije potrebno permutiranje u standardni pohlepni oblik.

3.2.1 *Uniformno lijevo čuvanje*

Za svaku točku $p \in N$ neka $L(p)$ označava najljevijeg čuvara koji vidi p. Uniformno (netežinsko) lijevo čuvanje je opisano *algoritmom 2*. Ključna ideja algoritma jest dodavanje najljevijeg čuvara $L(p)$ u skup čuvara X ukoliko klijent p još nije pokriven s X.

LEMA 3.2. *Postoji algoritam koji u $O((|G| + |N|) \log(|G| + |N|))$ vremenu vraća optimalno rješenje za problem uniformnog lijevog čuvanja 1.5D terena.*

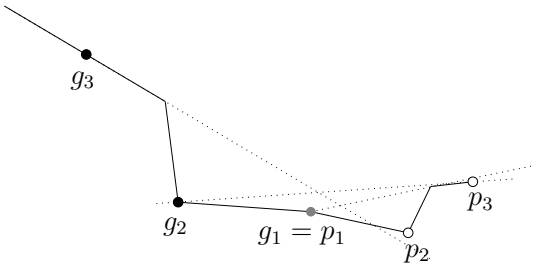
ALGORITAM 2 Nalaženje lijevih čuvara za 1.5D teren

```
1: procedure LEFT-GUARDING(T, N, G)
2:     X ← ∅
3:     for p ∈ N obrađen s lijeva na desno do
4:         if p nije viđen od X then
5:             X ← X ∪ {L(p)}
6:         end if
7:     end for
8:     return X
9: end procedure
```

Dokaz.
Kako bi se uvjerili da algoritam vraća optimalno rješenje, označimo s $N' \subseteq N$ one klijente koje čine algoritam da doda nove čuvare. Pretpostavimo, radi kontradikcije, kako postoje 2 klijenta p' i p'' u N' koji su viđeni slijeva istim čuvarom $g \in G$, drugim riječima, $g < p' < p''$ i $g \sim \{p', p''\}$. Neka je $g' = L(p')$ i mora biti $g' \leq g$. Ako je $g' = g$ tada $g' \sim p''$, i stoga p'' ne bi bio nečuvan kad je obrađen. Dakle, $g < g'$, ali *Tvrdnja uređaja* kaže da onda $g' \sim p''$ što vodi kontradikciji. Zaključujemo, svaki čuvar u G može vidjeti najviše jednog kijenta u N' što znači da je $|N'|$ donja ograda na optimalno rješenje. Kako kardinalnost of X je jednaka onoj od N' slijedi da je X optimalan.

Koristeći Graham [37] ideju iz algoritma za računanje konveksne ljuske od n točaka u ravnini u $O(n \log n)$ vremenu, *algoritam 2* se može implementirati u $O((|G| + |N|) \log(|G| + |N|))$ vremenu. Točke $G \cup N$ se obrađuju slijeva na desno. Koristeći strukturu stoga prati se u svakoj iteraciji kandidate za $L(p)$ prilikom obrađivanja klijenta p. Vidljivost između točaka se može čitati iz matrice lijeve vidljivosti koja se može izračunati u koraku prije same obradbe. □

PRIMJEDBA 5: U definiciji lijevog i desnog čuvanja 1.5D terena čuvar ne može vidjeti točku na kojoj leži. Upravo primjer u *Slici 13* pokazuje da bez tog kriterija poliedar $\{x \colon Ax \geq 1, x \geq 0\}$ ne mora imati sve vrhove cjelobrojne. Čuvari $G = \{g_1, g_2, g_3\}$ i klijenti $N = \{p_1, p_2, p_3\}$ su prikazani na *Slici 13*. Matrica vidljivosti A prikazana je desno od primjera. Čuvar g_1 čuva točku na kojoj leži, tj. g_1 čuva p_1. Vrhovi poliedra $\{x \in \mathbb{R}^n : \ Ax \geq 1, x \geq 0\}$ su $(0,1,1)$, $(1,1,0)$, $(1,0,1)$, $(1/2,1/2,1/2)$.



$$A = \begin{array}{c} \\ p_1 \\ p_2 \\ p_3 \end{array} \begin{array}{c} \begin{array}{ccc} g_1 & g_2 & g_3 \end{array} \\ \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \end{array}$$

Slika 13: Čuvar g_1 i klijent p_1 se nalaze na istoj točki terena pa po definiciji vidljivosti $g_1 \sim p_1$

3.3 JEDNOSTRANO ČUVANJE 1.5D TERENA

Analizu se nastavlja na poopćenju lijevog čuvanja, konkretno, dan nam je konačan skup klijenata N te 2 konačna skupa čuvara G_L i G_R gdje svaki čuvar u G_L (odnosno, G_R) može vidjeti klijenta iz N strogo s lijeva (odnosno, strogo s desna). Intuicija iza algoritma jest koristiti LP rješenje koje će pomoći u određivanju onih klijenata koji će biti čuvani s lijeva, odnosno s desna. Bez smanjenja općenitosti pretpostavlja se da svaki klijent u N može biti viđen slijeva ili sdesna. Inače, mora biti čuvan samim sobom što daje nedopustivu instancu problema, situacija koje se može otkriti u predobradnom koraku.

THEOREM 3.2. *Postoji 2-aproksimacijski algoritam za diskretnu varijantu jednostranog čuvanja 1.5D terena.*

Promotrimo sljedeći LP za nalaženje optimalnog skupa lijevih i desnih čuvara:

$$\min \sum_{g \in G_L} w_g \, x_{g,L} + \sum_{g \in G_R} w_g \, x_{g,R} \tag{LP5}$$

uz uvjete

$$\sum_{g \in \mathcal{V}_L(p) \cap G_L} x_{g,L} + \sum_{g \in \mathcal{V}_R(p) \cap G_R} x_{g,R} \geq 1 \qquad \forall p \in N \tag{9}$$

$$x_{g,L} \geq 0 \qquad \forall g \in G_L \tag{10}$$

$$x_{g,R} \geq 0 \qquad \forall g \in G_R$$

U uvjetu (10) varijabla $x_{g,L}$ ukazuje koliko je frakcionalno g izabran u skup lijevih čuvara X_L, a $x_{g,R}$ ukazuje koliko je g frakcionalno izabran u skup desnih čuvara X_R. Uvjet (9) traži da je svaka točka viđena nekim čuvarom, bilo slijeva ili sdesna.

Algoritam prvo pronađe optimalno frakcionalno rješenje x^* za LP5. Koristeći x^*, klijenti se podijele u 2 skupa

$$\begin{aligned} N_L &= \left\{ p \in N \mid \textstyle\sum_{g \in \mathcal{V}_L(p) \cap G_L} x^*_{g,L} \geq \frac{1}{2} \right\} \\ N_R &= \left\{ p \in N \mid \textstyle\sum_{g \in \mathcal{V}_R(p) \cap G_R} x^*_{g,R} \geq \frac{1}{2} \right\}. \end{aligned} \tag{11}$$

Intuitivno, želi se utvrditi klijenti čiji je frakcionalni doprinos lijevih čuvara značajniji nego desnih i ti klijenti trebaju biti čuvani s lijeva, a one čiji je frakcionalni doprinos čuvara veći s desna bit će čuvani s desna. Skupovi N_L i N_R ne moraju biti disjunktni.

Optimalno rješenje X^*_L za lijevo čuvanje daje WEIGHTED-1.5D-TERRAIN-LEFT-GUARD problem za parove (N_L, G_L) i optimalno desno rješenje X^*_R daje WEIGHTED-1.5D-TERRAIN-RIGHT-GUARD za par (N_R, G_R). Potrebno je pokazati kako par (X^*_L, X^*_R) čini dopustivo rješenje za LP5. Koristeći frakcionalno rješenje od LP5 određuje se gornja međa na cijene od X^*_L, X^*_R:

LEMA 3.3. *Neka su X^*_L, X^*_R optimalna rješenja parova (N_L, G_L) odnosno (N_R, G_R). Tada je $w(X^*_L) \leq 2\sum_{g \in G_L} w_g\, x^*_g$ i $w(X^*_R) \leq 2\sum_{g \in G_R} w_g\, x^*_g$.*

Dokaz. Pokazuje se prva nejednakost. Uzevši $x_{g,L} = 2x^*_g$ konstruira se dopustivo rješenje x_L od LP4 za $G = G_L$ i $N = N_L$. Rješenje x_L je dopustivo po definiciji od N_L i njegova cijena je $2\sum_{g \in G_L} w_g x^*_g$. Optimalno frakcionalno rješenje može biti samo manje. *Lema 3.1* kaže da cijena optimalnog frakcionalnog rješenja z^*_L od LP4 je jednaka cijeni optimalnog cjelobrojnog rješenja, naime, $w(X^*_L)$. Dakle,

$$w(X^*_L) = z^*_L \leq 2 \sum_{g \in G_L} w_g x^*_g$$

.

Druga nejednakost ide simetrično. Konstruira se dopustivo rješenje x_R gdje $x_{g,R} = 2x^*_g$ za LP relaksaciju ILP formulacije desnog čuvanja za $G = G_R$ i $N = N_R$. Rješenje x_R je i gornja međa za optimalno rješenje ILP-a desnog branjenja. Dakle,

$$w(X^*_R) = z^*_R \leq 2 \sum_{g \in G_R} w_g x^*_g.$$

□

Kako je $\sum_{g \in G_L} w_g \, x^*_{g,L} + \sum_{g \in G_R} w_g \, x^*_{g,R}$ donja granica na cijenu optimalnog čuvanja klijenata iz N, slijedi da je cijena rješenja (X^*_L, X^*_R) najviše 2 puta veća od optimalnog OPT :

$$w(X^*_L) + w(X^*_R) \leq 2\Big(\sum_{g \in G_L} w_g \, x^*_{g,L} + \sum_{g \in G_R} w_g \, x^*_{g,R}\Big) \leq 2\text{OPT}$$

Kako bi se uvjerili da je par (X^*_L, X^*_R) dopustivo rješenje za LP5, promotrimo nekog klijenta $p \in N$. Zbog (9) i pretpostavke da je svaki klijent viđen od barem jednog čuvara s lijeva ili desna, mora biti slučaj da je $p \in N_L$ ili $p \in N_R$. Stoga, p mora biti čuvan s lijeva s X^*_L ili s desna s X^*_R.

PRIMJEDBA 6: Računati X^*_L i X^*_R se može tako da se nađe frakcionalno optimalno rješenje za LP5 koristeći LP rješavatelj te onda koristiti primal-dual algoritme dane u [58] za potpuno uravnotežene matrice za parove (G_L, N_L) i (G_R, N_R).

S ovim je dokaz *Teorema 3.2* završen.

Radi kompletnosti, predstavit će se kombinatorni algoritam u $O(mn)$ vremenu za nalaženje optimalnih čuvara X^*_L kao alternativa *Primjedbi 6*.

LEMA 3.4. *Algoritam* WEIGHTED-LEFT-GUARDING *nalazi u* $O(mn)$ *vremenu optimalni skup lijevih čuvara* X^*_L.

Dokaz. Neka X označava skup čuvara koji će vratiti algoritam, s Y skup klijenata koje će dodavati čuvare u X. S g_p je označen čuvar koji je pridružen klijentu p. *Algoritam 3* nalazi optimalno rješenje za problem WEIGHTED-1.5D-TERRAIN-LEFT-GUARD za skup čuvara G i skup klijenata N.

Optimalnost rješenja X pokazat će se korištenjem *Teorema 1.4*.

Dualni program od LP4 je dan s LP6.

$$\max \sum_{p \in N} y_p \tag{LP6}$$

uz uvjete

$$\sum_{p | g \in \mathcal{V}_L(p)} y_p \leq w_g \qquad \forall g \in G$$

$$y_p \geq 0 \qquad \forall p \in N$$

Radi analize, promotrimo primalne i dualne vrijednosti koje su definirane na temelju algoritma:

ALGORITAM 3 Nalaženje lijevih čuvara za 1.5D s težinama

procedure WEIGHTED-LEFT-GUARDING(T, G, N, w)
    OBRADA S LIJEVA:
    $X \leftarrow \emptyset$, $Y \leftarrow \emptyset$
    $w'(g) \leftarrow w(g), \quad \forall g \in G$
    for $p \in N$ obrađen s lijeva na desno **do**
        if $\mathcal{V}_L(p) \cap X = \emptyset$ **then**
            $g_p \leftarrow \arg\min\{w'(g) : g \in \mathcal{V}_L(p)\}$
            $w'(g) \leftarrow w'(g) - w'(g_p), \forall g \in \mathcal{V}_L(p) \setminus \{g_p\}$
            $X \leftarrow X \cup \{g_p\}, \quad Y \leftarrow Y \cup \{p\}$
        end if
    end for
    KORAK ELIMINIRANJA:
    for $p \in Y$ obrađen s desna na lijevo **do**
        if $(X \setminus \{g_p\}) \cap \mathcal{V}_L(p) \neq \emptyset$ **then**
            $X \leftarrow X \setminus \{g_p\}$
        end if
    end for
    return X
end procedure

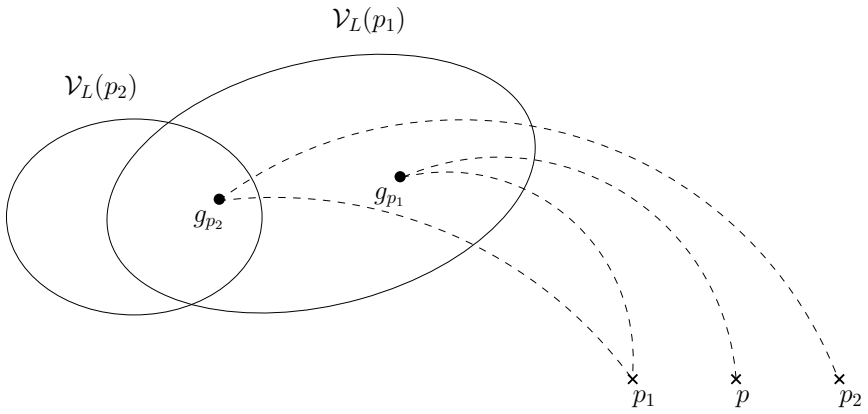
$$x_g = \begin{cases} 1, & g \in X \\ 0, & \text{inače} \end{cases}, g \in G$$

odnosno

$$y_p = \begin{cases} w'_{g_p}, & p \in Y \\ 0, & \text{inače} \end{cases}, p \in N.$$

Inicijalno, dualne varijable su $y_p = 0, \forall p \in N$. Treba argumentirati da par (x, y) na kraju *Algoritma 3* čini optimalno rješenje za primal LP4 i dual LP6.

primalna dopustivost: Treba pokazati da je x dopustivo rješenje primala. U tu svrhu, promotrimo klijenta p_1 i radi kontradikcije, pretpostavimo da će klijent p ostati nečuvan nakon koraka eliminacije. Neka je g_{p_1} čuvar kojeg je aktivirao klijent p_1 i $g_{p_1} \sim p$, ilustracija prikazana na *Slici 14*. Prilikom koraka eliminacije, postoji neki klijent $p_2 \in Y$ za kojeg je aktiviran čuvar $g_{p_2} \in X$ koji zamjenjuje g_{p_1} prema *koraku 14* algoritma. Primjećuje se da $g_{p_2} < g_{p_1}$. Kako $g_{p_2} \sim p_1$, $g_{p_1} \sim p$ slijedi da $g_{p_2} \sim p$ što je kontradikcija s polaznom pretpostavkom.



Slika 14: Argument primalne dopustivosti

dualna dopustivost: Za dopustivost dualnog rješenja y, promotrimo k klijenata iz Y, $p_1 < p_2 < \ldots < p_k$ i neki čuvar $g < p_i, g \sim p_i$ za $i = 1, 2, \ldots, k$. Kako su $y_{p_i} \in Y$ postoje čuvari $g_1, g_2, \ldots, g_k$ tako da je $g_i := g_{p_i}$. Zadnji klijent p_k prema algoritmu može odabrati g_k ili g.

Promotrimo oba slučaja:

$g_{p_k} \neq g$. U i-toj iteraciji u *koraku 7* algoritam je odabrao povoljnijeg čuvara prema rezidualnoj težini:

$$w'_{g_i} \leq w'_g - \sum_{j=1}^{i-1} w'_{g_j}. \tag{12}$$

Posebno, zapisujući za $i = k$:

$$w_g \geq w'_g \geq \sum_{j=1}^{k} w'_{g_j} = \sum_{i=1}^{k} y_{p_i}$$

$g_{p_k} = g$. Ukoliko p_k aktivira g, onda u tom slučaju korak eliminacije isključuje g_{p_i} iz X za $i < k$. Dakle,

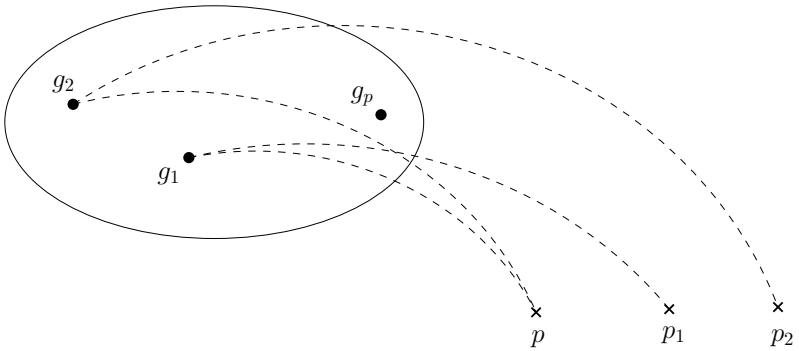
$$\begin{aligned} \sum_{i=1}^{k} y_{p_i} &= \sum_{i=1}^{k-1} w'_{g_i} + \underbrace{(w'_g - \sum_{i=1}^{k-1} w'_{g_i})}_{y_{p_k}} \\ &= w'_g \leq w_g. \end{aligned}$$

uvjeti primalne komplementarnosti: Neka je $x_g \neq 0$ i argumentirajmo da je onda $\sum_{p:\, g \in \mathcal{V}_L(p)} y_p = w_g$. Dovoljno je argumentirati za $g \in X$ jer inače $x_g = 0$. Koristeći sličan argument iz dualne dopustivosti, neka su $p_1, p_2, \ldots, p_k \in Y$ k klijenata koje vidi g. Kako je g uzet za čuvara nakon koraka eliminacije, mora biti da ga je odabrao klijent p_k, inače bi $y_{p_k} \notin Y$. Stoga,

$$\sum_{i=1}^{k} y_{p_i} = \sum_{i=1}^{k-1} w'_{g_i} + (w_g - \sum_{i=1}^{k-1} w'_{g_i}) = w_g.$$

uvjeti dualne komplementarnosti: Neka je $p \in Y$ i želimo argumentirati da vrijedi $\sum_{g \in \mathcal{V}_L(p)} x_g = 1$. Radi kontradikcije, pretpostavimo da postoje 2 aktivirana čuvara g_1, g_2 koji vide p. Prilikom obrade s lijeva algoritam odabire čuvara g_p i poveća dualnu vrijednost varijable y_p.

Ukoliko je $g_p = g_1$ tada u koraku eliminacije, čuvar g_1 će biti zamjenjen s g_2. Analogno bi se dogodilo ukoliko $g_p = g_2$. Promotrimo situaciju kad $g_p \neq g_1$. Po pretpostavci, čuvari g_1, g_2 su aktivirani prilikom obrade klijenata p_1, p_2, dakle, $g_1 = g_{p_1}, g_2 = g_{p_2}$. Po relaciji $g_2 < g_1 < g_p$ i $p < p_1 < p_2$ (vidjeti *sliku 15*) vrijedi $g_2 \sim p_1$ što implicira, da u koraku eliminacije, prilikom obrade p_1, g_1 bi bio zamjenjen s g_2. Posebno, prilikom precesuiranja p, g_p bi bio zamjenjen s g_2 i dolazimo do kontradikcije da p vide barem 2 čuvara iz $X \cap \mathcal{V}_L(p)$.



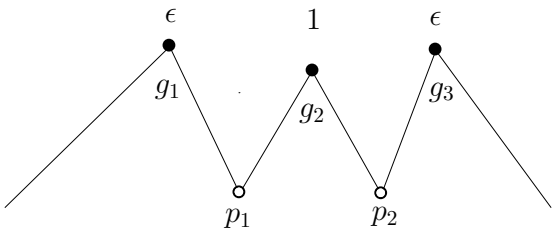
Slika 15: Argument dualne optimalnosti

Vrijeme izvršenja algoritma. Obrada klijenata s lijeva na desno i obrnuto može se implementirati u $O(|N|)$ vremenu kao obrada redaka matrice vidljivosti na temelju informacije o položaju klijenata i čuvara koja može biti kodirana u nekom pomoćnom polju ili povezanoj listi. Provjera $\mathcal{V}_L(p) \cap X = \emptyset$ se implementira jednostavnim obrađivanjem retka u matrici vidljivosti za p što je $O(|G|)$ vrijeme. Za nalaženje minimuma i ažuriranje težina potrebno je $O(|G|)$ vremena. U koraku eliminacije, *korak 14* predstavlja obradu čuvara i može se implementirati u $O(|G|)$ vremenu skenirajući također redak matrice koje odgovara skupu vidljivosti $\mathcal{V}_L(p)$. Dakle, ukupna vremenska složenost je asimptotski $O(mn)$ za $|G| = n, |N| = m$.

□

PRIMJEDBA 7: Iz *Slike 16* lako je vidjeti nužnost podjele točaka na one koje će biti čuvane s lijeva odnosno s desna. Ako se rješava odvojeno lijevi i desni problem čuvanja 1.5D terena te uzimanjem minimalnog rješenja od njih može biti proizvoljno daleko od optimalnog rješenja. Zaista, WEIGHTED-1.5D-TERRAIN-LEFT-GUARD za par (G, N) vraća rješenje s cijenom $w(\{g_1, g_2\}) = 1 + \epsilon$. S istom cijenom vraća i WEIGHTED-1.5D-TERRAIN-RIGHT-GUARD, $w(\{g_2, g_3\}) = 1 + \epsilon$. Kako u oba slučaja mora se odabrati čuvar težine ϵ zaključuje se da minimalno rješenje (koje je $1 + \epsilon$) može biti proizvoljno daleko od OPT za $\epsilon > 0$ gdje je OPT $= 1$.

S druge strane, ukoliko se podjele klijenti u skupove N_L i N_R prema (11) dobiva se $N_L = \{p_2\}$ i $N_R = \{p_1\}$. Nadalje, među lijevim i desnim čuvarima $G_L = \{g_1, g_2\}$ i $G_R = \{g_2, g_3\}$ pronalazi se optimalni čuvari, $X_L^* = X_R^* = \{g_2\}$ za koje $w(X_L^*) + w(X_R^*) \leq 2$, dakle, unutar faktora 2 aproksimacije.



Slika 16: Instanca 1.5D terena u kojem samo lijevo ili samo desno rješenje može biti proizvoljno daleko od optimalnog rješenja

3.4 OBOSTRANO ČUVANJE 1.5D TERENA

Problem DISCRETE-WEIGHTED-1.5D-TERRAIN-GUARD je problem čuvanja 1.5D terena T u kojoj je dan konačan skup čuvara $G \subset T$ te konačan skup klijenata $N \subset T$. Čuvari mogu vidjeti u oba smjera.

TEOREM 3.3. *Za problem* DISCRETE-WEIGHTED-1.5D-TERRAIN-GUARD *postoji* 4*-aproksimacijski algoritam kad* $G \cap N = \emptyset$*. Inače, dobiva se aproksimacija s faktorom* 5*.*

Dokaz. Slučaj $G \cap N = \emptyset$ je jednostavno riješen tako da svakog čuvara koji može vidjeti u oba smjera zamijeni se s lijevim i desnim čuvarom. Dakle, uzima se $G_L = G_R = G$ i rješava se problem jednostranog čuvanja 1.5D terena. Rješenje (X_L^*, X_R^*) jest najviše 4OPT gdje OPT predstavlja optimalno rješenje za problem DISCRETE-WEIGHTED-1.5D-TERRAIN-GUARD:

$$\begin{aligned} w(X_L^*) + w(X_R^*) &\leq \textstyle\sum_{g \in G} w_g x_{g,L} + \sum_{g \in G} w_g x_{g,R} \\ &\leq 2 \textstyle\sum_{g \in G} w_g x_g^* + 2 \sum_{g \in G} w_g x_g^* \\ &\leq 4\text{OPT} \end{aligned}$$

Intuitivno, u najgorem slučaju svaki čuvar može biti 2 puta odabran, stoga, faktor 2 je induciran redukcijom problema na jednostrani problem.

Isti pristup za slučaj $G \cap N \neq \emptyset$ implicira da redukcija mora inducirati dodatni faktor 3 jer klijent koji je sam sebi čuvar mora se zamijeniti s jednim ili lijevim ili desnim čuvarom što bi u konačnici rezultiralo s faktorom aproksimacije 6 za diskretan slučaj. Pažljivim odvajanjem klijenata iz N može se dobiti faktor aproksimacije 5.

U tu svrhu promatra se dodatni linearni program:

$$\min \sum_{g \in G} w_g \, x_g \tag{LP7}$$

uz uvjete

$$\begin{aligned} \sum_{g \in \mathcal{V}(p)} x_g &\geq 1 && \forall p \in N \\ x_g &\geq 0 && \forall g \in G \end{aligned}$$

Neka je x^* optimalno frakcionalno rješenje od LP7. Kao i u jednostranoj varijanti, koristit će se rješenje x^* da diktira koji klijenti će čuvati sami sebe (eventualno još neke druge klijente), a koji će biti čuvani od ostalih čuvara.

Skup čuvar-klijenata definiran je kao

$$X_0^* = \left\{ g \in N \cap G \mid x_g^* \geq \tfrac{1}{5} \right\}. \tag{13}$$

čija je težina najviše

$$w(X_0^*) \leq 5 \sum_{g \in X_0^*} w_g x_g^*. \tag{14}$$

Neka je N' skup točaka u N koji nisu viđeni od X_0^*, dakle, $N' = N \setminus \{p \colon X_0^* \sim p\}$ i neka je $G' = G \setminus X_0^*$. Konstruirat će se frakcionalno rješenje x za jednostrano čuvanje klijenata u N' za skupove lijevih odnosno desnih čuvara $G_L = G_R = G'$. Za svaki $g \in G'$ neka je $x_{g,L} = x_{g,R} = 5/4 x_g^*$. Frakcionalno rješenje x je dopustivo za LP5 jer za svaki $p \in N'$

$$\begin{aligned} \sum_{g \in \mathcal{V}_L(p) \cap G_L} x_{g,L} + \sum_{g \in \mathcal{V}_R(p) \cap G_R} x_{g,R} &= \frac{5}{4} \left(\sum_{g \in \mathcal{V}(p) \setminus \{p\}} x_g^* + x_p^* \right) \\ &\geq \frac{5}{4} \left(1 - \frac{1}{5} \right) \\ &= 1. \end{aligned} \tag{15}$$

Problem je sveden na jednostrano čuvanje uz skup čuvara G' i skup klijenata N' za koje $G' \cap N' = \emptyset$. Neka su (X_L^*, X_R^*) rješenja vraćena problemom Weighted-1.5D-Terrain-Left-Guard odnosno Weighted-1.5D-Terrain-Right-Guard. Cijena tih rješenja je garantirana da je 2 puta veća od cijene za x, odnosno $5/2 \sum_{g \in G'} w_g x_g^*$.

Skupovi čuvara X_0^*, X_L^*, X_R^* imaju cijenu

$$\begin{aligned} w(X_0^*) + w(X_L^*) + w(X_R^*) &\leq 5\sum_{g\in X_0^*} w_g x_g^* + 5\sum_{g\in G'} w_g x_g^* \\ &\leq 5\text{OPT}. \end{aligned} \tag{16}$$

gdje je OPT optimalna cijena čuvanja za problem WEIGHTED-1.5D-TERRAIN-GUARD kad $G \cap N \neq \emptyset$.

□

VREMENSKA SLOŽENOST ALGORITMA Aproksimacijski algoritam za problem DISCRETE-WEIGHTED-1.5D-TERRAIN-GUARD kad $G \cap N \neq \emptyset$ se sastoji od nekoliko rutina. U prvom redu, rješavatelj LP relaksacije originalnog problema LP7 nalazi frakcionalno rješenje na temelju kojeg se definira podjela na potprobleme: nalaženje čuvara-klijenata te problemi čuvanja slijeva odnosno sdesna. Vrijeme potrebno za nalaženja klijent-čuvara i redukciju problema na slučaj $G' \cap N' = \emptyset$ može se izvesti u $O(|G||N|)$ vremenu. Algoritam dalje rješava problem lijevog i desnog čuvanja koristeći optimalni algoritmom WEIGHTED-LEFT-GUARDING za input (T, G', N'_L, w) odnosno WEIGHTED-RIGHT-GUARD za input (T, G', N'_R, w) što je asimptotski $O(|G||N|)$. Ukupno vrijeme algoritma je $O(|G||N|) \cdot T_{\text{LP}}(|N|, |G|)$ gdje je $T_{\text{LP}}(|N|, |G|)$ asimptotsko vrijeme potrebno za rješavanje LP problema pakiranja opisanog s LP7.

3.5 ČUVANJE ČITAVOG 1.5D TERENA

U općenitom problemu 1.5D-TERRAIN-GUARD, pretpostavlja se da se čuvari uzimaju proizvoljno s terena, i skup klijenta pretpostavlja čitav teren. Pokazat će se kako iskoristiti diskretizacijski postupak predložen u [7] za svođenje problema na diskretnu varijantu.

3.5.1 *Diskretizacijski postupak*

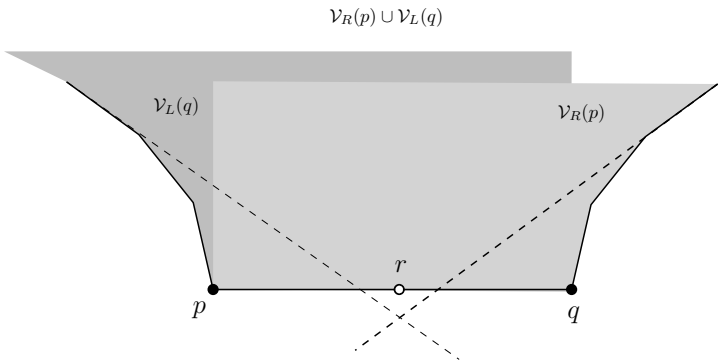
Dan je 1.5D teren T složenosti n s vrhovima $V(T) = \{P_1, P_2, \ldots, P_n\}$ i pripadnim nizom segmenata $\delta T = \{\overline{P_1P_2}, \overline{P_2P_3}, \ldots, \overline{P_{n-1}P_n}\}$. Neka je $r \in T$ proizvoljna točka s terena. Točka r može biti ili $r \in V(T)$ ili $r \in \overline{pq} \setminus \{p, q\}, \overline{pq} \in \delta T$. Iz geometrijskih svojstava 1.5D terena lako se zaključuje sljedeće svojstvo vidljivosti iz proizvoljne točke terena:

LEMA 3.5. *Neka je dan 1.5D teren T. Vrijedi jedna od sljedećih tvrdnji:*

a) ukoliko $r \notin V(T)$ *i* $r \in \overline{pq}$ *tada* $\mathcal{V}(r) \subseteq \mathcal{V}_R(p) \cup \mathcal{V}_L(q)$.

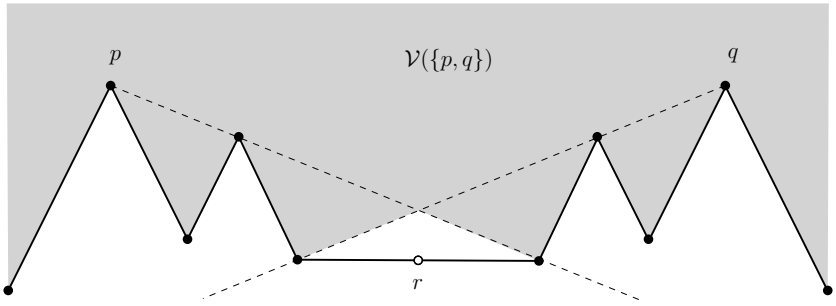
b) ukoliko $r \in V(T)$ *tada* $\mathcal{V}_L(r) \cup \mathcal{V}_R(r) = \mathcal{V}(r) \setminus \{r\}$.

Dakle, svaki se čuvar na terenu može zamjeniti s lijevim i desnim čuvarom u odgovarajućim vrhovima terena koji zajedno vide barem koliko vidi taj čuvar (*Slika 17*).



Slika 17: Lijevi čuvar u p i desni čuvar u q vide zajedno barem što vidi r

PRIMJEDBA 8: Zahtjev u formulaciji problema da se pokriju svih neprebrojivo mnogo točaka terena predstavlja pravi izazov, prvenstveno što bi odgovarajuća formulacija problema kao linearni program imalo neprebrojivo mnogo uvjeta na koje ne može primjeniti standardne metode za rješavanje linearnih programa. S druge strane, ne postoji trivijalni podskup točaka terena koje, kad pokrijemo, implicira pokrivenost čitavog 1.5D terena. Upravo na *slici 18* je prikazana instanca terena u kojoj su čuvari i klijenti ograničeni na vrhove terena $G = N = V(T)$. Rješenje problema je $X = \{p, q\}$ što je nedopustivo za čitav T odnosno, $X \not\sim T$.



Slika 18: Čuvanje isključivo vrhova terena ne implicira nužno pokrivenost čitavog terena

Sljedećom tvrdnjom pokazat će se da postoji postupak koji u polinomnom vremenu vraća konačan skup $M \subset T$ čija pokrivenost garantira i pokrivenost od T.

THEOREM 3.4. *Neka je dan 1.5D teren T složenosti n. Postoji konačan skup točaka $M \subset T$ takav da za bilo koji $G \subset T$ za koje G čuva M onda G čuva čitav T. Štoviše, $|M| = O(n^2)$ i M se može izračunati u $O(n^2)$ vremenu.*

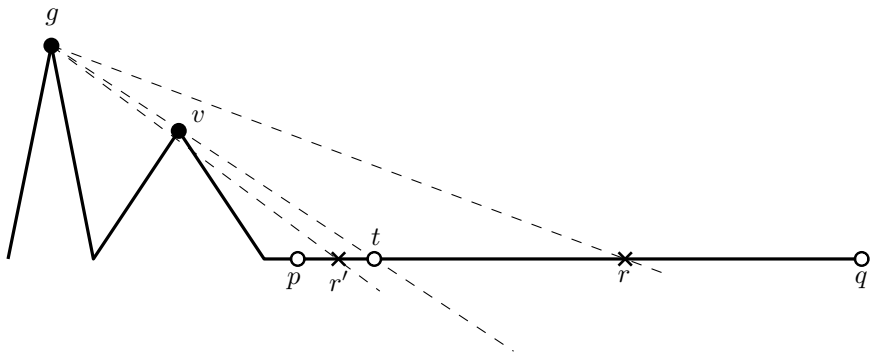
Dokaz. Konstruktivnim dokazom pokažimo da takav skup postoji. Promotrimo proizvoljna 2 vrha $v_1, v_2 \in V(T)$ i uvedimo sve točke $t \in T$ za koje vrijedi $t \in \mathcal{V}(v_1) \cap \mathcal{V}(v_2) \cap v_1v_2 \cap T$. Takvih točaka ima najviše dvije za fiksni par v_1, v_2. Označimo sa N' skup svih takvih točaka. Obrađivajući teren s lijeva na desno, točke iz $V(T) \cup N'$ označimo redom $t_1, t_2, \ldots, t_k$ na temelju x-monotonog uređaja. Za segment $\overline{t_{i-1}t_i}$, $i = 2, 3, \ldots, k$ kažemo da je *esencijalni segment*.

Ključno svojstvo koje proizlazi iz tvrdnje uređaja, jest ukoliko postoji neki čuvar $g \in T$ koji vidjeti neku točku r iz esencijalnog segmenta, recimo $\overline{pq}$, tada vidjeti sve točke iz $\overline{pq}$.

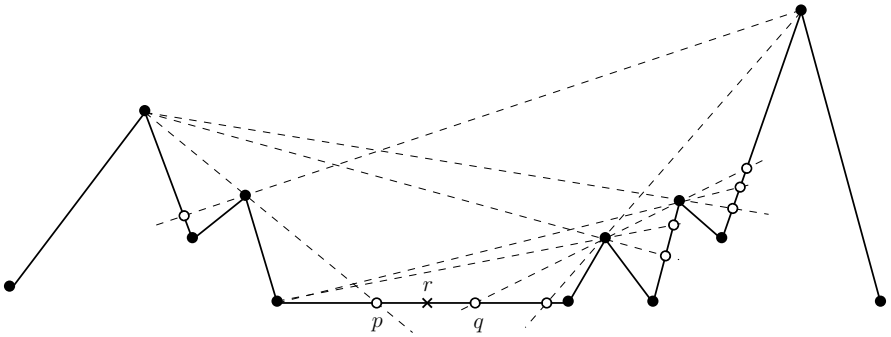
Dovoljno je pokazati za vidljivost slijeva, gdje se za vidljivost sdesna pokazuje simetrično. Neka je $g \in T$ neki čuvar s terena T za koje $g < p$ za neki esencijalni segment $\overline{pq}$ i neka $g \sim r$ za neki r u interijeru segmenta $\overline{pq}$. Radi kontradikcije, pretpostavimo da postoji točka r' iz segmenta $\overline{pq}$ koju ne vidi g. Ukoliko $r < r'$ tada iz definicije esencijalnog segmenta i tvrdnje uređaja za $a = g, b = p, c = r, d = r'$ imamo $g \sim r'$ što je kontradikcija s polaznom pretpostavkom. Neka je $r' < r$. Kako $g \not\sim r'$ mora postojati neki vrh terena $v \in T$, $g < v < p$ koji se nalazi iznad spojnice gr', a ispod spojnice gr. To znači da spojnica gv presjeca esencijalni segment $\overline{pq}$ u točki t za koje $r' < t < r$ što je kontradikcija s konstrukcijom od $\overline{pq}$ (*slika 19*).

U najgorem slučaju mora se uzeti $\frac{1}{2}\binom{n}{2}$ sjecišta što implicira $O(n^2)$ esencijalnih segmenata. Svaki esencijalni segment može se predstaviti proizvoljnom točkom iz svog interijera, stoga se odabire polovište kao predstavnik (točka r na *slici 20*). Sjecište para pravaca se može izračunati u $O(1)$ vremenu koristeći tehnike iz analitičke geometrije. Sjecišta svih pravaca se skeniraju s lijeva na desno i odabiru se odgovarajuća sjecišta koja leže na terenu, korak koji je izvediv u $O(n^2)$ vremenu.

□



Slika 19: Ukoliko neki čuvar vidi točku unutar esencijalnog segmenta, onda vidi čitav segment.



Slika 20: Diskretizacija 1.5D terena, točka r je predstavnik esencijalnog segmenta pq.

3.5.2 *Aproksimacijski algoritam za čuvanje čitavog 1.5D terena*

Neka je dan problem čuvanja čitavog 1.5D terena i radi jednostavnosti analize, promatra se problem s jediničnim težinama. Koristeći diskretizacijski postupak opisan *Teoremom 3.4* pokazuje se kako se problem može svesti na diskretnu varijantu problema.

Teorem 3.5. *Za* 1.5D-Terrain-Guard *problem postoji* 4*-aproksimacijski algoritam.*

Dokaz. Neka je X^* optimalni skup čuvara za čuvanje 1.5D terena T u kojem se mora pokriti čitav T.

Promotrimo nekog čuvara $g \in X^*$. Ukoliko g nije vrh od T tada mora ležati na nekom segmentu pq od T, po definiciji. Pretpostavimo bez smanjenja općenitosti da $p < q$, tada, prema *Lemi 3.5*, lijevi čuvar u točki p i desni čuvar u q vide barem koliko vidi g. Ukoliko je g vrh od T tada lijevi i desni čuvar u g zajedno vide koliko i g, ali ne vide

g. Stoga, postoji rješenje X' koji koristi isključivo lijeve i desne čuvare na vrhovima od T koji pokriva $T \setminus V(T)$ tako da je $|X'| = 2|X^*|$.

Kako bi se nosili s činjenicom da svaka točka terena mora biti pokrivena, iskoristimo skup točaka M koje vraća *Teorem 3.4*. Stoga, dopustivo rješenje za jednostrani problem čuvanja s $G_L = G_R = V$ i $N = M$ također čini dopustivo rješenje za problem čuvanja $T \setminus V(T)$. Neka je X'' optimalno rješenje za ovaj jednostrani problem i neka je X''' rješenje vraćeno *Teoremom 3.2*. Kako je X' dopustivo rješenje za jednostrani problem, vrijedi $|X'''| \leq 2|X''| \leq 2|X'| = 4|X^*|$ što daje ukupni aproksimacijski omjer 4. Skup čuvara X''' pokriva M što po njegovo konstrukciji, omogućava da vidi i čitave segmente na kojima se nalaze elementi od M, što uključuje i vrhove $V(T)$. □

PRIMJEDBA 9: U slučaju kad težine nisu jedinične koriste se standardne diskretizacijske tehnike na uštrb dodatnog faktora $1 + \epsilon$ u aproksimaciji. Promotrimo neki segment $\overline{pq}$ i pretpostavimo da težinska funkcija $w(x)$ za $x \in \overline{pq}$ ima r lokalnih minimuma u tom intervalu. Tada za svaki segment $\overline{st} \subseteq \overline{pq}$ na kojem w monotono raste ili pada, recimo od t_1 do t_2 (odnosno, od t_2 do t_1 ukoliko monotono pada) može se podijeliti na $\log_{1+\epsilon}(t_2/t_1)$ podsegmenata, gdje je $\epsilon > 0$ proizvoljno mala konstanta. Ukoliko se na nekom od tih segmenata nalazi optimalni čuvar g, onda, zamjenjujući g s lijevim čuvarom g_L i desnim čuvarom g_R na krajevima podsegmenata osigurava se da ta zamjena nije proizvoljno velika, odnosno $w(g_L) + w(g_R) \leq (2 + \epsilon)w(g)$ (u slučaju jediničnih težina, faktor je 2). Za fiksni segment $\overline{pq}$ uvodi se dodatno najviše $2r \log_{1+\epsilon}(t_2/t_1)$ čuvara, s dodatnim faktorom u općoj aproksimaciji $1 + \epsilon$.

Kako bi se uvjerili u to svojstvo promotrimo slučaj kad $t_1 \leq t_2$, gdje obrnuti slučaj ide analogno.

Definirajmo niz točaka intervala $[t_1, t_2]$ kao

$$a_1 := t_1, a_2 := a_1(1+\epsilon), \ldots, a_{h+1} := a_1(1+\epsilon)^h \leq t_2, \quad h \in \mathbb{N} \tag{17}$$

Neka je $w(g)$ cijena optimalnog čuvara koji se nalazi na nekom podintervalu $[a_k, a_{k+1}] \subseteq [t_1, t_2]$ definiranom kao u (17). Neka je čuvaru g_L pridružena težina a_k, a čuvaru g_R težina a_{k+1}. Primjetimo da $g_L < g_R$ iz konstrukcije. Zbrajanje težina lijevog i desnog čuvara ne može biti proizvoljno veće od $w(g)$. Zaista,

$$\frac{w(g_L) + w(g_R)}{w(g)} = \frac{a_k + a_{k+1}}{w(g)} \leq \frac{a_k}{w(g)}(2+\epsilon) \leq 2 + \epsilon.$$

Ukoliko se h računa kao $h = \log_{1+\epsilon}(t_2/t_1)$ onda se definira najviše $h+1$ pointervala $[t_1, a_2], [a_2, a_3], \ldots, [a_h, a_{h+1}], [a_{h+1}, t_2]$ čija je relativna veličina unutar ϵ.

Iskažimo konačan rezultat,

THEOREM 3.6. *Za* WEIGHTED-1.5D-TERRAIN-GUARD *postoji* $(4+\epsilon)$*-aproksimacijski algoritam za proizvoljni* $\epsilon > 0$.

3.6 OSVRT NA PROBLEM TEŽINSKOG ČUVANJA 1.5D TERENA

Doprinos

U ovom poglavlju predstavljen je općenito prvi aproksimacijski algoritam konstantne aproksimacije za inačicu problema čuvanja 1.5D terena u kojem čuvari imaju težine/cijenu. Koristeći linearno programiranje dano je nekoliko aproksimacija varijanti osnovnog problema. Za lijeve i desne probleme čuvanja 1.5D terena dani su optimalni algoritmi kao alternativa općenitim primal-dual algoritmima za probleme pokrivanja s potpuno uravnoteženim matricama uvjeta, danih u [42]. Rezultati su kompletno objavljeni u [25].

Daljni rad

Gibson i ostali u [33] su pokazali da se problem čuvanja 1.5D terena bez težina može proizvoljno aproksimirati generičkom rutinom lokalne pretrage. Može li se ista metodologija primjeniti i na proizvoljnu aproksimaciju za inačicu problema s težinama ostaje otvoren problem.

4

PROBLEM ČUVANJA 1.5D TERENA S VIŠESTRUKIM POKRIVANJEM

U ovom poglavlju predstavlja se problem višestrukog pokrivanja klijenata na 1.5D terenu i daje $5/2(1+1/d_{\min})$-aproksimacijski algoritam gdje je $d_{\min}$ minimalni zahtjev. Glavna ideja pristupa je na temelju izračunatog optimalnog rješenja pripadne LP relaksacije odlučiti za svakog klijenta količinu zahtjeva koji se mora ispuniti s lijeve strane i količinu zahtjeva s desne strane. Na takav način formuliraju se 2 potproblema, naime, lijevi i desni problem višestrukog pokrivanja čije je rješenje cjelobrojno i njihovim kombiniranjem dobiva se rješenje koje je unutar konstantne aproksimacije. Predstavljeni rezultat je općenito prvi aproksimacijski algoritam konstantne aproksimacije za problem koji poboljšava rezultat za geometrijske Set-Multi-Cover probleme [16] s ograničenom VC dimenzijom. Rezultat je kompletno objavljen u [26].

Sadržaj

4.1 VIŠESTRUKO POKRIVANJE 1.5D TERENA

U *problemu čuvanja 1.5D terena s višestrukim pokrivanjem* u oznaci 1.5D-TERRAIN-MULTI-GUARD dan je 1.5D teren T s konačnim skupom čuvara $G \subset T$ jediničnih težina i konačnim skup klijenata $N \subset T$ zajedno s definiranim zahtjevima na klijentima $d_p \colon N \to \mathbb{Z}_+, d(p) \mapsto d_p$. ILP formulacija problema čuvanja 1.5D terena s višestrukim pokrivanjem je oblika LP3 gdje $w_g = 1, \forall g \in G$ i opći oblik uvjeta (7). Kao i dosada, označava se s x^* optimalno rješenje ILP relaksacije od LP3 koja glasi

$$\min \sum_{g \in G} x_g \tag{LP8}$$

uz uvjete

$$\sum_{g \in \mathcal{V}(p)} x_g \geq d_p \qquad \forall p \in N \tag{18}$$

$$0 \leq x_g \leq 1 \qquad \forall g \in G \tag{19}$$

Uvjet (19) stavlja ograničenje na broj kopija čuvara g koje mogu sudjelovati u rješenju problema.

Linearni program LP8 predstavlja posebni tip problema miješanog pakiranja i pokrivanja koje će se u ovom kontektu zvati *problem ograničenog višestrukog pokrivanja*.

Koristeći rješenje x^* definirat će se 3 skupa klijenata koje trebaju biti pokriveni.

4.1.1 *Klijenti koji su ujedno i čuvari*

Fiksira se parametar $\alpha \in (0, 1/2)$ koji će biti definiran kasnije. Neka je X_0^* skup čuvara-klijenata za koje je frakcionalni doprinos x_g^* dovoljno velik, odnosno $X_0^* = \{g \in G \colon x_g^* \geq \alpha\}$ koje se uzima u konačno rješenje. Problem se sada reducira na sljedeću instancu: redefinirani zahtjevi $d'_p = d_p - |\mathcal{V}(p) \cap X_0^*|$ za sve $p \in N$, i skup N' predstavlja klijente koji nisu pokriveni s X_0^*, dakle, $N' = \{p \in N \colon d'_p \geq 1\}$ te $d_{\min} = \min\{d'_p \colon p \in N'\}$. U obzir se uzimaju čuvari koji imaju malu frakcionalnu vrijednost, dakle, $G' = G \setminus X_0^*, \mathcal{V}'(p) = \mathcal{V}(p) \cap G'$, te slično, $\mathcal{V}'_L(p) = \mathcal{V}'_L(p) \cap G'$ i $\mathcal{V}'_R(p) = \mathcal{V}_R(p) \cap G'$.

4.1.2 *Lijevi i desni problem čuvanja s višestrukim pokrivanjem*

Formulacija linearnog programa lijevog i desnog problema čuvanja 1.5D terena s višestrukim pokrivanjem dano je kao:

$$\min \sum_{g \in G'} x_g \qquad \text{(LP9)}$$

uz uvjete

$$\sum_{g \in \mathcal{V}'_L(p)} x_g \geq d_{p,L} \quad \forall p \in N'$$
$$0 \leq x_g \leq 1 \quad \forall g \in G'$$

$$\min \sum_{g \in G'} x_g \qquad \text{(LP10)}$$

uz uvjete

$$\sum_{g \in \mathcal{V}'_R(p)} x_g \geq d_{p,R} \quad \forall p \in N'$$
$$0 \leq x_g \leq 1 \quad \forall g \in G'$$

gdje su $d_{p,L}$ i $d_{p,R}$ nenegativne cijele vrijednosti

$$d_{p,L} = \left\lfloor \left(1 + \frac{1}{d_{\min}}\right) \left(\sum_{g \in \mathcal{V}'_L(p)} x^*_g + \frac{1}{2} x^*_p \right) \right\rfloor$$

odnosno,

$$d_{p,R} = \left\lfloor \left(1 + \frac{1}{d_{\min}}\right) \left(\sum_{g \in \mathcal{V}'_R(p)} x^*_g + \frac{1}{2} x^*_p \right) \right\rfloor$$

definirane kao količine zahtjeva d'_p za klijenta $p \in N'$ koje se mora osigurati slijeva odnosno sdesna. Za sve ostale $p \notin G'$ postavlja se $x^*_p = 0$.

Neka su z^*, z^*_L, z^*_R optimalne vrijednosti za LP8, LP9 odnosno LP10. Matrice uvjeta za LP9 i LP10 su potpuno uravnotežene prema *Lemi 3.1*. Cjelobrojnost rješenja je garantirana za poliedar $\{x\colon Ax \geq 1, x \geq 0\}$ no rezultat ne vrijedi općenito za poliedar $\{x\colon Ax \geq d, x \geq 0\}$. Kolen i Tamir [58] su pokazali da se problem višestrukog pokrivanja s jediničnim težinama može ipak svesti na ekvivalentan problem s težinama i da LP9 i LP10 imaju optimalna cjelobrojna rješenja.

U usporedbi s njihovim pristupom i radi kompletnosti analize, predstavlja se jednostavna procedura koja radi u $O(mn)$ vremenu i ne prolazi takvu redukciju. Algoritam vraća optimalni skup čuvara za lijevi problem čuvanja (simetrično se formulira za desni problem). Analiza algoritma se bazira na argumentima iskazanima u *Teoremu 1.4*.

LEMA 4.1. *Neka su X^*_L i X^*_R skupovi optimalnih čuvara za problem lijevog odnosno desnog čuvanja 1.5D terena s višestrukim pokrivanjem. Tada*

$|X_L^*| = z_L^*$ *i* $|X_R^*| = z_R^*$. *Štoviše, skupovi* X_L^* *i* X_R^* *se mogu izračunati u* $O(mn)$ *vremenu.*

Dokaz. Formulira se dualni linearni problem od LP9 za lijevo čuvanje 1.5D terena s višestrukim pokrivanjem:

$$\min \sum_{p \in N'} d_{p,L} y_p - \sum_{g \in G'} z_g \tag{LP11}$$

uz uvjete

$$\begin{aligned} \sum_{p:\, g \in \mathcal{V}_L'(p)} y_p - z_g \ &\leq 1 && \forall g \in G' \\ y_p &\geq 0 && \forall p \in N' \\ z_g &\geq 0 && \forall g \in G' \end{aligned}$$

Nadalje, radi jednostavnosti notacije, pretpostavlja se da $X_0^* = \emptyset$ i stoga $G' = G, N' = N$. Opisujemo prvo kombinatorni algoritam u $O(mn)$ vremenu za problem čuvanja klijenata iz N s lijeva.

ALGORITAM 4 Algoritam višestrukog pokrivanja 1.5D terena s lijeva

```
1: procedure LEFT-MULTI-GUARDING(T, G, N, d_L)
2:     X ← ∅
3:     for p ∈ N obrađen s lijeva na desno do
4:         while |X ∩ V_L(p)| ≤ d_{p,L} do
5:             X ← X ∪ L(p)
6:         end while
7:     end for
8:     return X
9: end procedure
```

U algoritmu, s $L(p)$ označava se najljeviji čuvar u skupu $\mathcal{V}_L(p) \setminus X$.

Radi svrhe analize, definira se algoritam dualnih ažuriranja DUAL-UPDATES u kojem se sa $X(p)$ označava skup aktiviranih čuvara u X koji su odabrani u rješenje prilikom obradbe p u *koraku 5* algoritma LEFT-MULTI-GUARDING.

Inicijalno, svi su čuvari neoznačeni. Treba pokazati da primalno x odnosno dualno rješenje (y, z) algoritma je dopustivo.

Primalna dopustivost: Slijedi iz *koraka 5* LEFT-MULTI-GUARDING algoritma.

ALGORITAM 5 Kombinatorna procedura za pokazivanje optimalnosti rješenja algoritma LEFT-MULTI-GUARDING

procedure DUAL-UPDATES($T, \{X(p)\}$)
  $y_p \leftarrow 0, \forall p \in N$
  $z_g \leftarrow 0, \forall g \in G$
  $x_g \leftarrow \begin{cases} 1, & g \in X = \cup_{p \in N} X(p) \\ 0, & \text{inače} \end{cases}$
  for $p \in N$ obrađen s desna na lijevo **do**
    if $X(p)$ ima neoznačenog čuvara **then**
      označi čuvare u $\mathcal{V}_L(p)$
      $y_p \leftarrow 1$
    end if
  end for
  $z_g \leftarrow \sum_{p|g \in \mathcal{V}_L(p)} y_p - 1, \forall g \in X$
end procedure

Dualna dopustivost: Dovoljno je pokazati kako $\sum_{p:\, g \in \mathcal{V}_L(p)} y_p \geq 1, \forall g \in X$ i $\sum_{p:\, g \in \mathcal{V}_L(p)} y_p \leq 1, \forall g \in G \setminus X$. Za prvu tvrdnju, pretpostavimo da čuvar $g \in X(p)$ i $y_p = 0$. Tada mora postojati neki $p' > p$ tako da je p' označio g te stoga $y_{p'} = 1$ i $g \in \mathcal{V}_L(p')$ (vidjeti *Sliku 21(a)*).

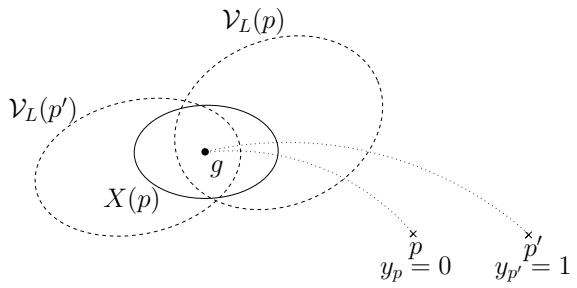
Za drugu tvrdnju (*Slika 21(b)*), promotrimo nekog čuvara $\overline{g} \notin X$ i pretpostavimo da postoje 2 točke $p < p'$ tako da je $\overline{g} \in \mathcal{V}_L(p) \cap \mathcal{V}_L(p')$ i $y_p = y_{p'} = 1$. Kako je $y_p = 1$ postoji čuvar $g < \overline{g}$ tako da je $g \in X(p)$ i g je bio neoznačen kad je *Algoritam 5* obrađivao p. Po *Tvrdnji uređaja* slijedi da g mora vidjeti p' i stoga ne može biti neoznačeno.

Dakle, $x, (y, z)$ su dopustiva cjelobrojna rješenja za primalni odnosno dualni linearni program. Ostaje pokazati da su rješenja ujedno i optimalna. Za analizu se koristi *Teorem 1.4*.

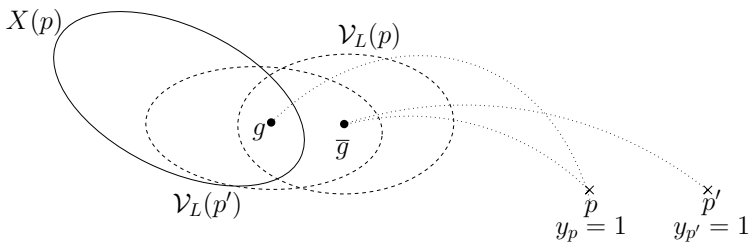
Uvjet primalne komplementarnosti: Treba pokazati da vrijedi

$$\begin{aligned} y_p = 1 \quad &\Rightarrow \quad \sum_{g \in \mathcal{V}_L(p)} x_g = d_{p,L} \\ z_g > 0 \quad &\Rightarrow \quad x_g = 1. \end{aligned}$$

Ukoliko je $z_g > 0$ tada, iz konstrukcije, $g \in X$ i stoga $x_g = 1$. Pretpostavimo da je $y_p = 1$. Želimo pokazati da $\sum_{g \in \mathcal{V}_L(p)} x_g = d_{p,L}$. Kako je $y_p = 1$, postoji čuvar $g \in X(p)$ koji je bio neoznačen kad je p bio obrađen od *Algoritma 5*. Pretpostavimo da $\sum_{g \in \mathcal{V}_L(p)} x_g > d_{p,L}$. Tada postoji čuvar $g' > g$ koji $g' \sim p$ i $g' \in X(p')$ za neki $p' > p$. Pretpostavimo da je čuvar g' označen. Ali tada klijent koji je označio



(a)



(b)

Slika 21: Argumenti dualne dopustivosti

g' mora isto tako označiti g po *Tvrdnji uređaja*. S druge strane, ukoliko g' nije označen, tada točka p' koja je desno od p (dakle, obrađena prije p) bi označila g' jer $g' \in X(p')$ i nije označen zajedno s g.

Uvjet dualne komplementarnosti: Treba pokazati

$$x_g = 1 \Rightarrow \sum_{p|g \in \mathcal{V}_L(p)} y_p - z_g = 1.$$

Implikacija slijedi direktno iz *koraka 11 Algoritma 5* jer $x_g = 1$ ako i samo ako $g \in G$.

Dakle, konstruirano cjelobrojno primalno rješenje x i cjelobrojno dualno rješenje (y, z) su dopustiva i zadovoljavaju uvjete komplementarnosti što po *Teoremu 1.4* implicira da su oni optimalna rješenja za primal LP9 i dual LP11.

□

4.1.3 *Aproksimacijski algoritam za čuvanje 1.5D terena s višestrukim pokrivanjem*

Koristeći čuvare vraćene u zaokruživanju velikih vrijednosti frakcionalnog rješenja x^* i čuvare vraćene u lijevom odnosno desnom pro-

blemu čuvanja s višestrukim pokrivanjem može se formulirati konačno rješenje za izvorni problem.

TEOREM 4.1. *Postoji* $5/2(1+1/d_{\min})$*-aproksimacijski algoritam za problem* 1.5D-TERRAIN-MULTI-GUARD.

Dokaz. Pokazujemo prvo da skup čuvara $X_0^* \cup X_L^* \cup X_R^*$ je dopustivo rješenje za izvorni problem, tj. svaki klijent p je viđen s d_p čuvara iz tog skupa. Zaista, ukoliko je $p \notin N'$ tada je p već čuvano d_p puta od X_0^*. Stoga, pretpostavlja se nadalje da je $p \in N'$ i argumentira da bilo koje cjelobrojno dopustivo rješenje za problem lijevog i desnog čuvanja s višestrukim pokrivanjem, recimo x_L i x_R, vraća barem d'_p čuvara za sve $p \in N'$. Točnije, za svaki $p \in N'$ dobiva se

$$\begin{array}{rcl} \sum_{g\in\mathcal{V}'_L(p)} x_{g,L} + \sum_{g\in\mathcal{V}'_R(p)} x_{g,R} & \geq & d_{p,L} + d_{p,R} \\ & = & \left\lfloor \frac{d_{\min}+1}{d_{\min}} \left(\sum_{g\in\mathcal{V}'_L(p)} x^*_g + \frac{1}{2} x^*_p \right) \right\rfloor \\ & & + \\ & & \left\lfloor \frac{d_{\min}+1}{d_{\min}} \left(\sum_{g\in\mathcal{V}'_R(p)} x^*_g + \frac{1}{2} x^*_p \right) \right\rfloor \end{array}$$

po konstrukciji $d_{p,L}$ i $d_{p,R}$.

Štoviše, zbog $\lfloor a \rfloor + \lfloor b \rfloor \geq \lfloor a+b \rfloor - 1, \forall a, b \in \mathbb{R}$ vrijedi:

$$\begin{array}{rcl} \sum_{g\in\mathcal{V}'_L(p)} x_{g,L} + \sum_{g\in\mathcal{V}'_R(p)} x_{g,R} & \geq & \lfloor \frac{d_{\min}+1}{d_{\min}} (\sum_{g\in\mathcal{V}'_L(p)} x^*_g + \sum_{g\in\mathcal{V}'_R(p)} x^*_g + x^*_p) \rfloor - 1 \\ & \geq & \lfloor d'_p + \frac{d'_p}{d_{\min}} \rfloor - 1 \\ & \geq & d'_p + 1 - 1 \\ & = & d'_p \end{array}$$

što zajedno, $d_{p,L} + d_{p,R} + |\mathcal{V}(p) \cap X_0^*| \geq d_p$ implicira da je $X_0^* \cup X_L^* \cup X_R^*$ dopustiv skup čuvara za izvorni problem.

Treba pokazati da vraćeno rješenje nije proizvoljno daleko od optimalnog rješenja. Iz definicije skupa X_0^*, broj čuvara u X_0^* je omeđen, odnosno $|X_0^*| \leq \frac{1}{\alpha} \sum_{g\in X_0^*} x^*_g$. Pokazat će se u nastavku da je restrikcija od $(1+\beta)x^*$ na G' dopustiva za LP9, za neku konstantu $\beta > 0$. Posljedica toga je nejednakost $z_L^* \leq (1+\beta) \sum_{g\in G'} x^*_g$. Koristeći isti argument za LP10, može se pokazati nejednakost $z_R^* \leq (1+\beta) \sum_{g\in G'} x^*_g$.

Za $\forall p \in N'$ vrijedi:

$$\begin{aligned} d_{p,L} &= \left\lfloor \left(1+\frac{1}{d_{\min}}\right)\left(\sum_{g\in\mathcal{V}'_L(p)} x^*_g + \frac{x^*_p}{2}\right)\right\rfloor \\ &\leq \left(1+\frac{1}{d_{\min}}\right)\left(\sum_{g\in\mathcal{V}'_L(p)} x^*_g + \frac{\alpha}{2}\right) \\ &\leq (1+\beta)\sum_{g\in\mathcal{V}'_L(p)} x^*_g \end{aligned} \tag{20}$$

gdje zadnja nejednakost vrijedi ukoliko

$$\beta \geq \frac{\alpha}{2}\cdot\frac{1}{\sum_{g\in\mathcal{V}'_L(p)} x^*_g}\left(1+\frac{1}{d_{\min}}\right)+\frac{1}{d_{\min}}. \tag{21}$$

Isti argument se može primjeniti i za $d_{p,R}$. U slučaju $d_{p,L}=0$ nejednakost (20) je trivijalno istinita za bilo koju $\beta > 0$, dok inače, za $d_{p,L} \geq 1$ treba vrijediti $\sum_{g\in\mathcal{V}'_L(p)} x^*_g \geq \frac{d_{\min}}{d_{\min}+1} - \frac{\alpha}{2}$ iz definicije od $d_{p,L}$. Stoga, nejednakost (21) vrijedi ukoliko je sljedeći uvjet na β zadovoljen:

$$\begin{aligned} \beta &\geq \frac{\alpha}{2}\cdot\frac{1}{\frac{d_{\min}}{d_{\min}+1}-\frac{\alpha}{2}}\left(1+\frac{1}{d_{\min}}\right)+\frac{1}{d_{\min}} \\ &= \frac{\alpha(d_{\min}+1)^2}{2d^2_{\min}-d_{\min}(d_{\min}+1)\alpha}+\frac{1}{d_{\min}}. \end{aligned} \tag{22}$$

Isto tako, za sve $g \in G'$ treba vrijediti

$$(1+\beta)x^*_g \leq 1$$

koje, zajedno s činjenicom $x^*_g < \alpha$ daje gornju među na β, naime

$$\beta \leq \frac{1}{\alpha} - 1 \tag{23}$$

U konačnici, može se sada ograničiti cijena vraćenog skupa čuvara:

$$\begin{aligned} |X^*_0|+|X^*_L|+|X^*_R| &= |X^*_0| + z^*_L + z^*_R \\ &\leq \frac{1}{\alpha}\sum_{g\in X^*_0} x^*_g + 2\cdot(1+\beta)\sum_{g\in G'} x^*_g \\ &\leq \max\{\frac{1}{\alpha}, 2\cdot(1+\beta)\}z^* \\ &\leq \max\{\frac{1}{\alpha}, 2\cdot(1+\beta)\}\mathrm{OPT} \end{aligned}$$

gdje OPT predstavlja cijenu optimalnog rješenja izvornog problema 1.5D-TERRAIN-MULTI-GUARD.

Uvjet (22) na β ukazuje da je aproksimacijski faktor ograničen s

$$\gamma = \min_{\alpha \in (0,\alpha')} \max \left\{ \frac{1}{\alpha}, \frac{2(d_{\min}+1)^2\alpha}{2d_{\min}^2 - d_{\min}(d_{\min}+1)\alpha} + \frac{2}{d_{\min}} + 2 \right\}.$$

Uspoređujući (22) i (23) dobiva se gornja međa na odabir α, odnosno $\alpha' = \min\{\frac{1}{2}, \frac{2d_{\min}}{3(1+d_{\min})}\}$.

Promatrajući izraze u definiciji γ kao funkcije ovisne o α primjećuje se da je jedna funkcija monotono padajuća, a druga rastuća na intervalu $(0, \alpha')$. Ukoliko grafovi tih dviju funkcija se sijeku na intervalu $(0, \alpha')$ njihovo sjecište može poslužiti za dobivanje vrijednosti za γ. Stoga, izjednačavanjem ta 2 izraza dobiva se $\alpha^* = 2d_{\min}/5(d_{\min}+1)$ što je dopustivo rješenje za γ jer $a^* \in (0, a')$. Konačno, γ se može izraziti eksplicitno kao $\gamma = 5/2(1+d_{\min})/d_{\min}$ čime je dokaz teorema završen.

□

VREMENSKA SLOŽENOST ALGORITMA Aproksimacijski algoritam za problem 1.5D-TERRAIN-MULTI-GUARD prati sličan okvir kao i algoritam za DISCRETE-WEIGHTED-1.5D-TERRAIN-GUARD. U prvom redu, rješavatelj LP relaksacije originalnog problema LP8 nalazi frakcionalno rješenje u vremenu $T_{\text{LP}}(|N|+|G|, |G|)$ na temelju kojeg se može raditi podjela problema. Zaokruživanje velikih vrijednosti frakcionalnog rješenja od LP8 te zatim redukcija na problem s ulazom (T, G', N', d') se može napraviti u $O(|G||N|)$ vremenu. Kombinatorni algoritam LEFT-MULTI-GUARDING uzima ulaz (T, G', N', d'_L) odnosno simetrično definirani RIGHT-MULTI-GUARDING ulaz (T, G', N', d'_R) i vraćaju cjelobrojno optimalno rješenje u $O(|G||N|)$ vremenu. Dakle, ukupna vremenska složenost algoritma jest $O(|G||N|) \cdot T_{\text{LP}}(|N|+|G|, |G|)$.

4.2 ČUVANJE 1.5D TERENA S VIŠESTRUKIM KOPIJAMA

Problem 1.5D-TERRAIN-MULTISET-GUARD jest problem 1.5D-TERRAIN-MULTI-GUARD u kojem se dopušta određeni broj kopija nekog čuvara, te kao takav se može formulirati kao cjelobrojni linearni program:

$$\min \sum_{g \in G} x_g \tag{LP12}$$

uz uvjete

$$\sum_{g \in S(p)} x_g \geq d_p \qquad \forall p \in N$$
$$x_g \in \{0, 1, \ldots, u_g\} \qquad \forall g \in G \qquad (24)$$

gdje (24) dopušta da u rješenju bude odabrano najviše u_g kopija svakog čuvara g iz G. Tretirajući svaku kopiju čuvara g kao novog čuvara, LP12 ima LP relaksaciju oblika LP8 gdje čuvari pripadaju multiskupu $\Gamma = \{g^{u_g} : g \in G\}$. Ova redukcija predstavlja instancu 1.5D-Terrain-Multi-Guard problema s m klijenata i ukupno $\sum_{g \in G} u_g$ čuvara. Ukoliko se u_g ne specificira, dakle, izraz (24) se zamijeni s $x_g \in \mathbb{Z}_+$, može se definirati i dalje problem s konačnim brojem kopija, dakle, za $u_g \geq \max_{p \in N}\{d_p : g \sim p\}$ dobiva se ekvivalentni ILP.

Zaključujemo sa završnim rezultatom:

Teorem 4.2. *Postoji* $5/2(1 + 1/d_{\min})$*-aproksimacijski algoritam za* 1.5D-Terrain-Multiset-Guard *problem.*

4.3 OSVRT NA PROBLEM ČUVANJA 1.5D TERENA S VIŠESTRUKIM POKRIVANJEM

Doprinos

U ovom poglavlju predstavljen je prvi aproksimacijski algoritam konstantne aproksimacije za čuvanje 1.5D terena s višestrukim pokrivanjem. Analiza se temelji na linearnom programiranju i podijeli problema na lijeve i desne potprobleme koji se mogu rješiti optimalno. Za lijeve i desne potprobleme dani su kombinatorni algoritmi koji ih rješavaju optimalno. Rezultat se proširuje na varijantu problema u kojem svaki čuvar može imati nekoliko kopija. Rezultati su objavljeni u [26]. Premda se problem čuvanja 1.5D terena s višestrukim pokrivanjem može svesti na geometrijski Set-Multi-Cover s ograničenom VC dimenzijom i kao takav imati $O(\log z^*)$ aproksimaciju prema [16], ispostavilo se da dodatna struktura problema kao što je *Tvrdnja uređaja* može dovesti do konstantne aproksimacije.

Daljni rad

Ostaje istražiti kako se diskretizacijski postupak dan *Teoremom 3.4* može primjeniti za čuvanje 1.5D terena s višestrukim pokrivanjem.

Problem Weighted-1.5D-Terrain-Multi-Guard se može podijeliti na potprobleme koji imaju potpuno uravnoteženu matricu uvjeta, u trenutku pisanja doktorskog rada, nepoznato je postojanje optimalnog algoritma koji daje cjelobrojno rješenje tih podproblema.

5

PARCIJALNO ČUVANJE 1.5D TERENA

U ovom dijelu predstavit će se aproksimacijski algoritam za parcijalnu inačicu čuvanja 1.5D terena koji se temelji na aproksimacijskom algoritmu od Mestre [72] za probleme parcijalnog pokrivanja koje sadrže svojstvo potpune uravnoteženosti. Jedno od nedostataka standardne formulacije Set-Cover problem jest da postoje određeni elementi koji su skupi za pokrivanje (engl. *outliers*, [14]) gdje onda optimalno rješenje može biti vrlo skupo. Parcijalna inačica Set-Cover problema bi zahtjevala da se ne pokrije čitav skup elemenata već samo k njih. U kontekstu čuvanja 1.5D terena primjenu može naći u nadgledanju autoceste nadzornim kamerama u kojima neke dionice ceste su od veće važnosti (npr. dionice s intenzivnijim prometom ili većom frekventnošću prometnih nesreća). U tom slučaju modelira se njihova vrijednost s većim prihodom, dok dio autoputa koji nije od interesa može imati manju vrijednost prihoda. Ključna ideja analize jest pokazati da parcijalno čuvanje 1.5D terena predstavlja problem parcijalnog pokrivanja u kojem matrica uvjeta odgovarajuće LP formulacije sadrži svojstvo potpune uravnoteženosti. Aproksimacijski algoritmi temeljeni na Lagrangeovoj relaksaciji za parcijalno pokrivanje su uvelike riješeni u [56] i za za probleme pokrivanja koje sadrže svojstvo potpune uravnoteženosti u [72].

Sadržaj

5.1 PROBLEM PARCIJALNOG POKRIVANJA

Neka je $\mathcal{S} = \{S_1, S_2, \ldots, S_n\}$ familija podskupova elemenata od $U = \{e_1, e_2, \ldots, e_m\}$ i neka je dana težinska funkcija $c\colon \mathcal{S} \to \mathbb{Q}_+$ nad skupovima i funkcija profita $\pi\colon U \to \mathbb{Q}_+$ koja svakoj točki $e \in U$ pridružuje profit $\pi(e)$. Problem PARTIAL-SET-COVER za dano ograničenje pokrivanja b predstavlja određivanje pokrivača najmanje težine $\mathcal{C} \subseteq S$ takav da je $\pi(U \setminus \mathcal{C}) \leq b$ gdje $\pi(S) = \sum_{e \in S} \pi(e)$.

Neka $x(S)$ označava indikator varijablu pripada li skup S rješenju $\mathcal{C}$ i neka $r(e)$ označava indikator varijablu je li neki element e ostao nepokriven od $\mathcal{C}$. Formulacija problema kao cjelobrojni linearni program opisan je s LP13.

$$\min \sum_{S \in \mathcal{S}} c(S)\, x(S) \tag{LP13}$$

uz uvjete

$$\begin{aligned} \sum_{e \in U} \pi(e) r(e) &\leq b && && (25)\\ \sum_{e \in S} x(S) + r(e) &\geq 1 && \forall e \in U \\ x(S), r(e) &\in \{0,1\} && \forall S \in \mathcal{S}, \forall e \in U \end{aligned}$$

Primal i dual LP relaksacije od LP13 dano je u matričnom obliku

$$\begin{aligned} &\min\ c^T x && \text{(LP14)}\\ &\text{uz uvjete} \\ \pi^T r &\leq b && (26)\\ Ax + Ir &\geq 1 \\ x_S, r_e &\geq 0 \end{aligned} \qquad \begin{aligned} &\min\ \mathbf{1}^T y - b\lambda && \text{(LP15)}\\ &\text{uz uvjete} \\ A^T y &\leq c && (27)\\ y &\leq \lambda \pi \\ y_S, \lambda &\geq 0 \end{aligned}$$

gdje je relacija incidencije elemenata i skupova opisana matricom $A = [a_{e,S}] \in \{0,1\}^{m \times n}$ za $m = |U|$ i $n = |\mathcal{S}|$ u kojoj vrijednost $a_{e,S} = 1$ ako i samo $e \in S$. Varijabla $x_S = 1$ indicira je li $S \in \mathcal{C}$ te varijabla $r_e = 1$ ukoliko $e \notin \mathcal{C}$.

Ponekad se umjesto ograničenja b definira *ciljani profit* P koji se mora ostvariti pokrivenim elementima, dakle, $P = \pi(U) - b$ ukoliko je zadano ograničenje b. U LP14 uvjet (26) se formulira onda kao $\pi^T r \leq \pi(U) - P$.

Problem parcijalnog pokrivanja opisan s LP13 može se svesti na problem Prize-Collect-Set-Cover s profit funkcijom $\lambda\pi$ tako da se uvede Lagrangeov multiplikator za uvjet (25) i doda funkciji cilja. Problem ima formulaciju cjelobrojnog linearnog programa LP16.

$$\min c^T x + \lambda \pi^T r - \lambda b \tag{LP16}$$

uz uvjete

$$\begin{aligned} Ax + Ir &\geq 1 \\ x_S, r_e &\in \{0,1\} \end{aligned}$$

Neka je OPT cijena optimalnog parcijalnog pokrivanja za problem Partial-Set-Cover i OPT-PC(λ) cijena pokrivanja za Prize-Collect-Set-Cover za dani λ.

Neka je $\mathcal{A}$ α-aproksimacijski algoritam za Prize-Collect-Set-Cover problem. Kaže se da $\mathcal{A}$ čuva svojstvo Lagrangeovog multiplikatora odnosno $\mathcal{A}$ ima α-LMP svojstvo ukoliko $\mathcal{A}$ vraća pokrivač $\mathcal{C}$ za kojeg vrijedi:

$$c(\mathcal{C}) + \alpha\lambda(\pi(U) - \pi(\mathcal{C})) \leq \alpha\text{OPT-PC}(\lambda) \tag{28}$$

Kako je OPT-PC$(\lambda) \leq$ OPT $+ \lambda b$ iz (28) dobiva se nejednakost

$$c(\mathcal{C}) \leq \alpha(\text{OPT} + \lambda(\pi(\mathcal{C}) - (\pi(U) - b)))$$

što, ukoliko se nađe λ takav da

$$\pi(\mathcal{C}) = \pi(U) - b, \tag{29}$$

implicira da je algoritam $\mathcal{A}$ α-aproksimacija. Za neke λ može se dogoditi $\pi(\mathcal{C}) < \pi(U) - b$ što je unutar α aproksimacije, ali je nedopustivo rješenje. Nadalje, za neke λ vrijedi $\pi(\mathcal{C}) < \pi(U) - b$ što je dopustivo rješenje, ali ne daje nikakvu garanciju na gornju granicu cijene pokrivača $\mathcal{C}$. Štoviše, za neke λ postoje pokrivači koji ne daju jednakost $\pi(\mathcal{C}) = \pi(U) - b$ [72].

Könemann i ostali u [56] predlažu metodu za rješavanje parcijalnog pokrivanja tako da koriste α-LMP algoritam za Prize-Collect-Set-Cover kao rutinu 'crne kutije' za aproksimaciju parcijalnog pokrivanja.

Njihov rezultat je iskazan sljedećim teoremom:

Teorem 5.1 ([56]). *Neka je dana instanca* Partial-Set-Cover *problema definirana sustavom skupova* $(U, \mathcal{S})$ *i težinskom funkcijom* $c\colon \mathcal{S} \to \mathbb{Q}_+$ *te*

profit funkcija $\pi\colon U \to \mathbb{Q}_+$ *i ograničenjem* b *sa svojstvom da postoji* α-LMP *algoritam za odgovarajuću* Prize-Collect-Set-Cover *inačicu problema. Tada, za svaki* $\epsilon > 0$ *može se pronaći dopustivo rješenje za danu instancu* Partial-Set-Cover *problema čije rješenje je najviše* $4/3\alpha + \epsilon$ *od optimalnog rješenja u polinomnom vremenu ovisno o veličinama* $|U|, |\mathcal{S}|^{1/\epsilon}$.

5.1.1 *Pristup 'crne kutije'*

Koristeći ILP formulaciju s Lagrangeovim multiplikatorom LP16 definira se za svaki $\lambda \geq 0$ instanca Prize-Collect-Set-Cover problema čije je optimalno rješenje OPT-PC(λ). Koristeći α-LMP algoritam za Prize-Collect-Set-Cover problem kao podrutinu, binarnim pretraživanjem nalazi se dovoljno blizu λ_1, λ_2 za koje λ_1 daje rješenje $\mathcal{C}_1 \subseteq \mathcal{S}$ tako da je ukupan profit od pokrivenih elemenata manji od $\pi(U) - b$, dok za λ_2, nalazi se pokrivač $\mathcal{C}_2 \subseteq \mathcal{S}$ u kojem je ukupni profit od pokrivenih točaka barem $\pi(U) - b$.

Rješenje $\mathcal{C}_1$ ne mora biti dopustivo, dok rješenje $\mathcal{C}_1$ jest dopustivo, ali može biti proizvoljno daleko od optimalnog rješenja. U tu svrhu, definira se novo rješenje $\mathcal{C}_3$ tako da rješenje $\mathcal{C}_2$ se proširi s pažljivo odabranim skupom iz rješenja $\mathcal{C}_1$. Strategija takvog odabira temelji se na pristupu od Levin i Segev [66] za parcijalne multirezove u stablu.

Bez dodatnih pretpostavki na strukturu sustava skupova ova strategija ne može dati bolji aproksimacijski faktor:

Lema 5.1 ([56],[72]). *Problem* Partial-Set-Cover *se ne može općenito bolje aproksimirati od* $4/3\alpha$ *koristeći Lagrangeovu relaksaciju i* α*-LMP algoritam* $\mathcal{A}$ *kao podrutinu 'crne kutije'.*

5.1.2 *Parcijalno pokrivanje sa svojstvom potpune uravnoteženosti*

Mestre u [72] daje aproksimacijski algoritam koji koristi 1-LMP algoritam iz [57] za Prize-Collect-Set-Cover problem opisan s LP16 u kojem ja matrica uvjeta A potpuno uravnotežena. Algoritam za parcijalno pokrivanje temelji se na činjenici da postoji kritična vrijednost λ koje ima svojstvo da za svaki infinitezimalno mali $\delta > 0$ vrijednosti $\lambda^- = \lambda - \delta$ i $\lambda^+ = \lambda + \delta$ daju rješenja koja pokrivaju manje odnosno više od $\pi(U) - b$. Neka je $\mathcal{A}$ oznaka za 1-LMP algoritam. Neka su $\mathcal{C}^- = \mathcal{A}(\lambda^-)$ i $\mathcal{C}^+ = \mathcal{A}(\lambda^+)$ pokrivači vraćeni od algoritma $\mathcal{A}$ i bez smanjenja općenitosti neka $\mathcal{C}^-$ pokriva manje od $\pi(U) - b$ profita, a $\mathcal{C}^+$ više od $\pi(U) - b$ profita. Definirajući *graf spajanja* između rješenja

$\mathcal{C}^+$ i $\mathcal{C}^-$ na temelju svojstva potpune uravnoteženosti može se odrediti pravilo kako spojiti pokrivače $\mathcal{C}^-$ i $\mathcal{C}^+$ da tvore dopustivo rješenje $\mathcal{C}^*$ za PARTIAL-SET-COVER problem za koje vrijedi ocjena

$$c(\mathcal{C}^*) \leq \left(1 + \frac{1}{3^{k-1}}\right) \text{DLP} + kc_{\max}, k \in \mathbb{Z}_+ \tag{30}$$

gdje je DLP frakcionalno optimalno rješenje duala LP15 i $c_{\max}$ predstavlja najveću cijenu skupa iz $\mathcal{S}$. Kompletan postupak spajanja se može pronaći u [72].

Nadalje, koristeći pristup od [34] i [40] pokazuje kako se ocjena (30) može iskoristiti za dizajn $(\rho + \epsilon)\text{OPT} + kc_{\max}$ aproksimacije za problem parcijalnog pokrivanja gdje se matrica incidencije A može prikazati kao kombinacija ρ potpuno uravnoteženih matrica za $\epsilon > 0$. Dodatno, aditivni član $kc_{\max}$ se može apsorbirati u aproksimaciji izdvajanjem $k/(\alpha - 1)$ najskupljih skupova u vraćenom rješenju i ponovnim rješavanjem reducirane inačice problema.

DEFINICIJA 26: Za binarnu matricu $A \in \{0,1\}^{m \times n}$ kaže se da je *ρ-odvojiva*[1] ukoliko postoje binarne matrice $A_1, A_2, \ldots, A_\rho$ tako da je $A = \sum_{q=1}^{\rho} A_q$ i svaka matrica B koja se dobije uzimanjem redaka iz $A_1, A_2, \ldots, A_q$ na način da i-ti redak od B je i-ti redak od A_j za neki $1 \leq j \leq \rho$ je potpuno uravnotežena.

Ključan rezultat iz [72] za probleme parcijalnog pokrivanja u kojem je matrica uvjeta ρ-odvojiva predstavlja temelj analize koje će biti predstavljeno u ovom poglavlju:

TEOREM 5.2 ([72]). *Neka je A ρ-odvojiva matrica. Tada postoji $(\rho + \epsilon)$-aproksimacija u polinomnom vremenu za problem parcijalnog pokrivanja opisan s LP13.*

Postupak za rješavanje PARTIAL-SET-COVER problema koji imaju svojstvo ρ-odvojivosti sažet je u *algoritmu 6*.

Algoritam započinje tako da rješava primal LP14 u *koraku 2*. U *koraku 3* slijedi konstrukcija potpuno uravnotežene matrice B koja se sastoji od redaka pripadnih ρ odvojivih matrica A_q. Binarnim pretraživanjem u unaprijed određenom intervalu (λ_L, λ_R) i korištenjem 1-LMP rutine može se naći kritična vrijednost λ^* (*korak 4*) i u skladu s time odrediti λ^-, λ^+ za koje vrijedi ili $\lambda^- < \lambda^* =: \lambda^+$ ili $\lambda^- := \lambda^* < \lambda^+$. Algoritam 1-LMP može biti Kolenova procedura za problem LP16 opisana u [57] gdje je matrica A potpuno uravnotežena. Neka za λ^- 1-

1 engl. ρ-separable

ALGORITAM 6 Algoritam parcijalnog pokrivanja za ρ-odvojive probleme

1: **procedure** ρ-SEPERABLE-PARTIAL-COVER$(U, \mathcal{S}, c, \pi, b)$
2: $(x^*, r^*) \leftarrow$ LP-SOLVE(A, c, π, b)
3: definiraj matricu B tako da $b_e = a_e^{q_e}$ za koje $a_e^{q_e\ T} x^* \geq \frac{1-r_e^*}{\rho}$ za $e \in U$ i $a_e^{q_e}$ odgovara retku od matrice A_{q_e} za element e
4: $\lambda^* \leftarrow$ SEARCH(λ_L, λ_R)
5: $\mathcal{C}^- \leftarrow$ 1-LMP$(\lambda^-), \mathcal{C}^+ \leftarrow$ 1-LMP(λ^+)
6: $\mathcal{C}^* \leftarrow$ MERGE$(C^-, C^+, \lceil \log_3 \frac{\delta}{\epsilon} + 1 \rceil)$
7: **if** $\mathcal{X} \neq$ NIL **then**
8: **return** $\mathcal{X} \cup \mathcal{C}^*$
9: **end if**
10: odaberi $\mathcal{X}$ kao $\frac{k}{\alpha-1}$ najskupljih skupova iz optimalnog rješenja
11: reduciraj instancu:
$U' \leftarrow U \setminus \{e : e \in \mathcal{X}\}$, $\mathcal{S}' = \mathcal{S} \setminus \{S : c(S) > \min_{S' \in \mathcal{X}} c(\mathcal{S}')\}$
$b' = b + \pi(\mathcal{X})$
12: **return** ρ-SEPARABLE-PARTIAL-COVER$(U', \mathcal{S}', c, \pi, b')$
13: **end procedure**

LMP vraća nedopustiv pokrivač $\mathcal{C}^-$, a za λ^+ 1-LMP vraća dopustiv pokrivač $\mathcal{C}^+$. Procedura MERGE$(\mathcal{C}^-, \mathcal{C}^+, k)$ nalazi dopustiv pokrivač $\mathcal{C}^*$ za problem parcijalnog pokrivanja za koje vrijedi ocjena (30). Kako je svako dopustivo rješenje za problem parcijalnog pokrivanja s matricom uvjeta B ujedno dopustivo i za problem parcijalnog pokrivanja s matricom uvjeta A, vrijedi:

$$\begin{aligned} c(\mathcal{C}^*) &\leq \left(1 + \tfrac{1}{3^{k-1}}\right) c(\rho x^*) + k c_{\max} \\ &= \left(1 + \tfrac{1}{3^{k-1}}\right) \rho c(x^*) + k c_{\max} \\ &\leq (\rho + \epsilon)\,\mathrm{OPT} + k c_{\max} \end{aligned}$$

gdje prva nejednakost dolazi od dopustivosti $(\rho x^*, r^*)$ rješenja, a zadnja nejednakost zbog $k \geq \lceil \log_3 \frac{\delta}{\epsilon} + 1 \rceil$.

Kako bi se uklonio aditivni član $k c_{\max}$ u aproksimaciji, instanca problema se reducira na temelju *koraka 10-12*. Optimalno rješenje reducirane instance je $\mathrm{OPT}' = \mathrm{OPT} - c(\mathcal{X})$.

U *koraku 8* rješenje $\mathcal{X}$ se vraća zajedno s $\mathcal{C}^*$ koje je α aproksimacija:

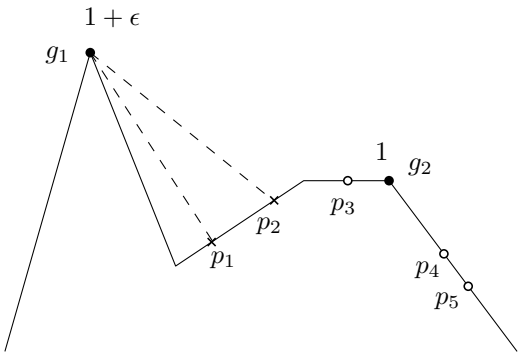
$$\alpha \mathrm{OPT}' + k c'_{\max} + c(\mathcal{X}) \leq \alpha \mathrm{OPT}' + k \frac{\alpha - 1}{k} c(\mathcal{X}) + c(\mathcal{X}) = \alpha \mathrm{OPT}.$$

Nažalost, ne zna se koji skupovi iz optimalnog rješenja čine $\mathcal{X}$ stoga se algoritam u *koracima 10-12* mora izvršavati za svaki odabir $\mathcal{X}$ iz $\mathcal{S}$ i najbolje rješenje se zatim vraća. Broj poziva algoritma za odabir najboljeg rješenja je $\binom{|U|}{\frac{k}{\alpha}-1}$.

5.2 PROBLEM PARCIJALNOG ČUVANJA 1.5D TERENA

Problem parcijalnog čuvanja 1.5D terena može se formulirati kao problem parcijalnog pokrivanja gdje elemente čine klijenti s 1.5D terena, dakle, $U = N$, a skupovi $S_i = \{\mathcal{V}(g) : g \in G\}$ čine familiju pokrivača $\mathcal{S}$ na kojima je definirana težinska funkcija $w : \mathcal{S} \to \mathbb{Q}_+$. Relacija incidencije elemenata-skupova opisana je matricom vidljivosti $A = [a_{p,g}] \in \{0,1\}^{m \times n}$ gdje $m = |U|$ i $n = |\mathcal{S}|$ u kojoj vrijednost $a_{p,g} = 1$ ako i samo $g \sim p$. Varijabla x_g pokazuje pripada li čuvar g u rješenje $\mathcal{C}$, varijabla r_p ukazuje je li klijent p nepokriven. Vrijednost $b \in \mathbb{Q}_+$ predstavlja ograničenje za nepokrivanje terena i za svakog klijenta $p \in N$ dana je vrijednost profita $\pi(p)$.

PRIMJEDBA 10: Parcijalno čuvanje 1.5D terena se ogleda u primjeru ilustriranom na *slici 22* u kojoj je prikazana situacija gdje čuvar g_1 može biti proizvoljno skup, stoga čuvar g_2 pokriva 3 klijenta p_3, p_4, p_5 dok klijenti p_1, p_2 nisu čuvani premda ih g_1 vidi. Optimalno rješenje parcijalnog čuvanja je 1 dok čuvanje svih klijenata košta $2 + \epsilon$ za $\epsilon > 0$ proizvoljno velik.



Slika 22: Parcijalno čuvanje 1.5D terena u kojem je $b = 2$ i $\pi(p) = 1$, te težine čuvara $w(g_1) = 1 + \epsilon, w(g_2) = 1$

5.2.1 *Parcijalno jednostrano čuvanje 1.5D terena*

Za matricu A kaže se da je *matrica jednostrane vidljivosti* ukoliko odgovara matrici incidencije čuvara-klijenata problema pokrivanja opisanog s linearnim programom LP5 za neku instancu jednostranog problema čuvanja 1.5D terena.

Parcijalna inačica jednostranog problema čuvanja 1.5D terena je problem pokrivanja u kojoj je matrica incidencije upravo matrica jednostrane vidljivosti. Dovoljno je pokazati da je ona 2-odvojiva i aproksimacijski faktor slijedi direktno iz *Teorema 5.2*.

LEMA 5.2. *Bilo koja matrica jednostrane vidljivosti je* 2*-separabilna.*

Dokaz. Neka je A matrica jednostrane vidljivosti i pretpostavimo, bez smanjenja općenitosti, da A ima oblik $[\ C_1 \quad C_2\ ]$ gdje stupci od C_1 odgovaraju lijevim čuvarima iz G_L i stupci od C_2 odgovaraju desnim čuvarima iz G_R.

Matrica A se može zapisati kao $A = A_L + A_R$ gdje $A_L = [\ C_1 \quad 0\ ]$ i $A_R = [\ 0 \quad C_2\ ]$. Pretpostavimo da je matrica A' formirana tako da se uzimaju retci od A_L i A_R. Neka je N_L skup klijenata kojima odgovaraju retci u matrici A_L i N_R skup klijenata kojima odgovaraju retci iz A_R. Permutiranjem redaka od A' na način da se klijenti iz N_L nalaze prije klijenata iz N_R daje permutiranu blok matricu

$$A'' = \begin{bmatrix} D_L & 0 \\ 0 & D_R \end{bmatrix}$$

čiji se blokovi D_L i D_R sastoje od redaka koji odgovaraju klijentima iz N_L i stupcima koji odgovaraju G_L, odnosno retcima iz N_R i stupcima iz G_R. Po *Lemi 1.6* matrice D_L i D_R su potpuno uravnotežene (odgovaraju relacijama vidljivosti u problemima lijevog čuvanja za (G_L, N_L) odnosno desnog čuvanja za (G_R, N_R)) stoga permutiranjem redaka i stupaca matrice A'' formira se nova blok matrica

$$A''' = \begin{bmatrix} D'_L & 0 \\ 0 & D'_R \end{bmatrix}$$

u kojoj su blokovi D'_L i D'_R u standardom pohlepnom obliku. Iz definicije standardnog pohlepnog oblika primjećuje se da je A''' ujedno u standardnom pohlepnom obliku. Kako je matrica A''' formirana permutacijom redaka i stupaca iz A slijedi da je A potpuno uravnotežena matrica.

□

TEOREM 5.3. *Za parcijalni problem jednostranog čuvanja 1.5D terena postoji* $(2+\epsilon)$*-aproksimacijski algoritam u polinomnom vremenu za* $\epsilon > 0$.

5.2.2 *Diskretno parcijalno čuvanje 1.5D terena*

Neka je dan 1.5D teren T te skup čuvara G s težinskom funkcijom $w\colon G \to \mathbb{Q}_+$ i skup klijenata N s funkcijom profita $\pi\colon N \to \mathbb{Q}_+$ te ograničenje na nepokrivenost b. Problem DISCRETE-PARTIAL-1.5D-TERRAIN-GUARD traži utvrđivanje čuvara najmanje težine X tako da je profit od nečuvanih točaka najviše b. Problem se može svesti na parcijalno jednostrano čuvanje 1.5D terena. Pretpostavlja se da $G \cap N = \emptyset$. Svaki se čuvar $g \in G$ zamijeni s lijevim i desnim čuvarom. Na takav način definira se problem parcijalnog jednostranog čuvanja 1.5D terena za lijeve čuvare G_L i desne čuvare G_R čije je rješenje $2+\epsilon$ od optimalnog rješenja parcijalnog jednostranog čuvanja 1.5D terena. Kako $G_L = G_R = G$, vraćeno rješenje je dopustivo za problem koje je dodatno faktor 2 gore od njegovog optimalnog rješenja. U slučaju $G \cap N \neq \emptyset$ potrebno je izdvojiti čuvare koji imaju velik frakcionalni doprinos i njih zaokružiti. Tako reducirana instanca predstavlja instancu problema parcijalnog jednostranog čuvanja s 2-odvojivom matricama uvjeta. Slijedeći analogiju iz dokaza za *Teorem 3.4* dobiva se faktor $5+\epsilon$.

Konačan rezultat je iskazan teoremom:

TEOREM 5.4. *Za problem* DISCRETE-PARTIAL-1.5D-GUARD *postoji* $(4+\epsilon)$*-aproksimacijski algoritam ukoliko* $G \cap N = \emptyset$*. Inače se dobije* $(5+\epsilon)$*-aproksimacijski algoritam.*

5.2.3 *Parcijalno čuvanje 1.5D terena*

U parcijalnom pokrivanju čitavog 1.5D terena u oznaci PARTIAL-1.5D-TERRAIN-GUARD treba se pokriti dio terena T tako da najviše duljine b terena je nepokriveno. Koristeći diskretizacijski postupak iz *Teorema 3.4* dovoljno je svakom klijentu iz $p \in M$ pridjeliti dužinu esencijalnog segmenta, $\pi(p) = l(\overline{st})$ za $p \in \overline{st}$. U realističnoj postavci, dijelovi terena mogu imati različite vrijednosti profit funkcije. Uz pretpostavku da postoji posebna rutina (*oracle*) koji će davati profit za esencijalni segment analiza vrijedi i za proizvoljno definirane profit funkcije. Dis-

kretizacijski postupak reducira problem parcijalnog pokrivanja čitavog terena na diskretnu inačicu parcijalnog pokrivanja te kao posljedica slijedi:

TEOREM 5.5. *Za* PARTIAL-1.5D-TERRAIN-GUARD *problem postoji* $(4+\epsilon)$*-aproksimacijski algoritam.*

5.3 OSVRT NA PARCIJALNO ČUVANJE 1.5D TERENA

Doprinos

U ovom poglavlju formuliran je problem parcijalnog čuvanja 1.5D terena i dan prvi aproksimacijski algoritam konstantne aproksimacije za nj. Temelj analize predstavlja pokazivanje 2-odvojivosti matrice jednostrane vidljivosti, koje zbog [72] ima $(2+\epsilon)$-aproksimacijski algoritam. Pokazuje se kako se parcijalno jednostrano čuvanje 1.5D terena može iskoristiti za nalaženje $(4+\epsilon)$-aproksimacijskog algoritma za diskretnu inačicu parcijalnog čuvanja 1.5D terena i parcijalno čuvanje čitavog 1.5D terena. Rezultati su objavljeni u [25]. Ovo poglavlje pokazuje praktičnu primjenu rezultata za probleme parcijalnog pokrivanja sa svojstvom ρ-odvojivosti razvijenih u [72].

Daljni rad

Problem parcijalnog čuvanja 1.5D terena s višestrukim pokrivanjem ostaje otvoreno pitanje. Po svemu sudeći rezultat iz [56] zahtjeva postojanje α-LMP za odgovarajući PRIZE-COLLECT-SET-MULTI-COVER problem. Jedna od ideja bi svakako bila probati iskoristiti svojstvo potpune uravnoteženosti kao u slučaju [72] i pokušati dati analogon tog rezultata.

Dio II

IMPLEMENTACIJSKI MODEL ALGORITAMA ČUVANJA 1.5D TERENA

6

BRZI RJEŠAVATELJI ZA PROBLEME PAKIRANJA I POKRIVANJA

U ovom poglavlju proučava se brza implementacija aproksimativnog rješavatelja linearnih programa posebne klase problema pakiranja i pokrivanja. Pored slijedne implementacije istražuje se paralelna CUDA implementacija za aproksimativni rješavatelj izvorno opisan u [31]. Motivacija za promatranje CUDA platforme kao računalnog resursa za masivni paralelizam temeljeno na nitima izvršenja dijelom je motivirano rezultatima koje su promatrani za množenje matrica u višenitnom okruženju u [5]. CUDA implementacija aproksimativnog rješavatelja uspoređena je sa slijednom implementacijom na jednoprocesorskom sustavu te se obje implementacije uspoređuju sa simpleks metodom iz GLPK (GNU Linear Programming Kit) biblioteke. Rezultati su u pripremi za publiciranje u [69].

Sadržaj

6.1 UVOD U RJEŠAVANJE PROBLEMA PAKIRANJA I POKRIVANJA

Problemi pakiranja/pokrivanja je široko poznata klasa kombinatornih optimizacijskih problema koje se može opisati kao linearni program:

$$\max_{x\in\mathbb{R}^n}\{c^T x\colon Ax \leq b, x \geq 0\} = \min_{y\in\mathbb{R}^m}\{b^T y\colon A^T y \geq c, y \geq 0\} \qquad (31)$$

gdje je A nenegativna $m \times n$ matrica, b i c su nenegativni vektori. Pod pojmom *primalnog linearnog programa* u ovom kontekstu podrazumijeva se maksimizacijski problem i pridruženi minimizacijski problem kao *dualni linearni program*. Opravdanost pisanja (31) je temeljeno na *Teoremu 1.2*. Kako su elementi matrice A, vektora b i c svi nenegativni brojevi, primal LP se zove *problem pakiranja* i dualni LP kao *problem pokrivanja*.

6.1.1 *Rješavanje problema linearne optimizacije*

Linearna optimizacija postavila je čvrste temelje u mnogim problemima koje proizlaze iz operacijskih istraživanja u zadnjih 70 godina. Khachiyan [50] i Karmarkar [49] pokazali su da problem linearnog progamiranja pripada klasi P i providjeli algoritam u polinomnom vremenu za nj. Dodatno u [39] pokazuje se da linearno programiranje pripada PC klasi za koje se čini da su inherentno slijedni, u smislu da je nalaženje njihova rješenja krajnje slijedni postupak. Bez obzira na relativnu neparalelizibilnost rješavanja problema linearnog programiranja, postoje istraživanja na tom području [80] koji zapravo pokazuju da se određeni dijelovi metoda za rješavanje tih problema mogu efikasno paralelizirati.

Jedno od motivacija korištenja *grafičkog programiranja opće namjene* (GPGPU) u implementaciji paralelnih algoritama je upravo korištenje grafičkih procesora kao matematičkog koprocesora, u smislu, intenzivnija računanja koje bi zahtjevalo veći broj slijednih koraka mapiraju se na GPU platformu i koristi veliki broj optimiranih niti izvršavanja (engl. *threads of execution*) za pristupanje i računanje na podacima, gdje i dalje glavna kontrola toka programa je pridružena jednom samo procesoru. Dovoljno sazrijevanje API-a za korištenje GPGPU platformi je jedan od velikih razloga popularizacije istih. Platforme koje implementiraju takav način paralelnog računanja mogu se naći u osobnim računalima koji dolaze s grafičkom karticom NVIDIA ili ATI familije te u novije vrijeme GPGPU računanje služi kao servis unutar HPC

sustava. O iskorištenosti GPGPU paradigme kao servisa u grozd računalima može se pogledati u [21].

U nastavku se daje pregled paralelnih implementacija potpomognute grafičkim procesorom za metode rješavanja problema linearne optimizacije.

6.1.1.1 *Simpleks metoda*

Jedno od prvih rezultata implementacije simpleks metode potpomognute grafičkim koprocesorom predstavljeno je u [38] za implementaciju revidirane simpleks metode koristeći Cg i OpenGL grafičke biblioteke. Implementacija se prilagođava grafičkom cjevovodu i programski model je grafički orijentiran. Razvojem CUDA API-a pojavile su se nekoliko implementacija simpleks metode: Spampinato i Elster [80] su implementirali CUDA verziju revidirane simpleks metode koristeći pritome NVIDIA CUBLAS i NVIDIA LAPACK grafički ubrzane rutine matričnog računa koje dolaze u *CUDA Toolkit*-u [1]. Lalami i ostali [63] daju CUDA implementaciju dvostruke preciznosti simpleks metode u kojima ne koriste CUDA grafički ubrzane rutine matričnog računa. Njihova implementacija cilja na LP sustave u kojima je matrica uvjeta gusta.

6.1.1.2 *Metode unutrašnje točke*

Koristeći CUDA biblioteku Jung i O'Leary [48] daju implemenataciju LP rješavatelja temeljenog na metodi unutrašnje točke. Računalno zahtjevnije dijelove rješavatelja kao što je slaganje matrica, Cholesky faktorizacija te slijedna i povratna supstitucija mapiraju na GPU. Nedavno su Smith i ostali [79] predložili GPU implementaciju metode unutrašnje točke koristeći strategiju matrično slobodnih metoda. Algoritam se može prilagoditi za korištenje *oracle* rutina ukoliko A nije eksplicitno dana.

6.1.1.3 *Aproksimacijske sheme za linearne programe*

Ponekad LP rješavatelj može biti sastavni dio nekog drugog algoritma (u našem slučaju, algoritma zaokruživanja LP-a) i optimalno rješenje se može zamjeniti s aproksimativnim rješenjem unutar neke garancije na dobit efikasnosti.

Luby i Nissan [67] predstavili su brzi paralelni aproksimacijski algoritam za LP probleme pakiranja i pokrivanja. Njihov algoritam radi u polilogaritamskom vremenu ϵ^{-4} koristeći linearni broj procesora.

Young [85] opisuje slijedni i paralelni algoritam koji aproksimiraju općenitije inačice - problem miješanog pakiranja i pokrivanja. Njegov sekvencijali algoritam radi u $O(md \log(m)/\epsilon^2)$ vremenu, gdje je m broj uvjeta, a d najveći broj uvjeta u kojoj se pojavljuje bilo koja varijabla, dok je složenost paralelnog algoritma polilogaritamska u veličini inputa s faktorom $1/\epsilon^4$ koristeći linearni broj procesora. Ukupni broj operacija paralelnog algoritma je usporediv sa sekvencijalnim. Kourogiannakis i Young [61] su dali randomizirani aproksimacijski algoritam za linearne programe pakiranja i pokrivanja koji radi u $O(D + (n+m)\log D/\epsilon^2)$ očekivanom vremenu, gdje D označava broj elemenata različitih od nule u matrici uvjeta A. Valja napomenuti kako je ovaj algoritam asimptotski brži od algoritma koji je implementiran u ovom poglavlju, ali njihov randomizirani pristup ne može riješavati probleme ukoliko je broj uvjeta eksponencijalan.

6.2 APROKSIMACIJSKA SHEMA ZA PROBLEME POKRIVANJA I PAKIRANJA

U ovom poglavlju proučava se brza implementacija $(1+\epsilon)$-aproksimacijskog algoritma za probleme pakiranja i pokrivanja, odnosno algoritam koji vraća primalno i dualno rješenje čiji je omjer cijena unutar $1+\epsilon$ faktora (a samim time, unutar $1+\epsilon$ faktora od optimalnog rješenja). Algoritam koji se implementira je izvorno dan u Garg i Koenemann [31] koji je općenito formuliran za rješavanje problema toka robe[1]. Primalne i dualne LP formulacije njihovih problema mogu imati *eksponencijalnu veličinu*, tj. ili u broju varijabli primala odnosno u broju uvjeta ekvivalentnog dualnog problema. Pokazali su da bez obzira na takve okolnosti, aproksimativno rješenje se može i dalje naći u polinomnom vremenu ukoliko postoji *oracle* rutina u polinomnom vremenu koje vraća "najviše narušeni uvjet" među eksponencijalno mnogo uvjeta. Upravo je ovo prednost njihova pristupa. Štoviše, prijavljeno je kako algoritam radi u $O(\epsilon^{-2} m \log m) \cdot T(\text{ORACLE})$ gdje je m broj redaka, $\epsilon^{-2} m \log m$ gornja granica na broj iteracija algoritma i $T(\text{ORACLE})$ označava asimptotsko vrijeme potrebno za *oracle* rutinu po iteraciji.

1 engl. *multicommodity flow problems*

Algoritam predstavlja Lagrange relaksacijski tip i koristi tehnike ažuriranja primalnih i dualnih varijabli iterativno najvećim mogućim korakom kako bi se ostvario što bolji napredak algoritma u svakom koraku. Svaka iteracija algoritma je bazirana računanju minimalnog elementa polja i računanju ažuriranja primalnog vektora x i dualnog vektora y. Paralelna CUDA implementacija koja bude predstavljena je upravo bazirana na činjenici da se takve operacije mogu efikasno paralelizirati u CUDA paralelnom okruženju.

Definirajmo prvo osnovne pojmove:

DEFINICIJA 27: Neka je $\epsilon > 0$ neka konstanta. Za dopustivi primal-dual par (x, y) za (31) kaže se da je $(1+\epsilon)$-aproksimacija ukoliko

$$b^T y \leq (1+\epsilon) c^T x$$

DEFINICIJA 28: FPTAS za (31) je algoritam koji nalazi $(1+\epsilon)$-aproksimaciju u polinomnom vremenu ovisno o veličini inputa m, n i aproksimacijskom faktoru $1/\epsilon$.

6.2.1 *Algoritam primalnih i dualnih ažuriranja*

Lagrangeova relaksacija linearnog programa pakiranja i pokrivanja dano je kao:

$$\max_{Ax \leq b, x \geq 0} c^T x = \max_{x \geq 0} \min_{y \geq 0} (c^T x + \sum_{i=1}^{m} y(i)(1 - A_i x / b(i))) \quad (32)$$

$$\min_{A^T y \geq c, y \geq 0} b^T y = \min_{y \geq 0} \max_{x \geq 0} (b^T y + \sum_{j=1}^{n} x(j)(1 - y^T A^j / c(j))) \quad (33)$$

gdje A_i i A^j označavaju i-ti redak odnosno j-ti stupac matrice A. Varijable $x(j), y(i)$ se mogu smatrati kao kazne koje se plaćaju za kršenje primalnih i dualnih uvjeta. Procedura iterativno konstruira par vektora (x_k, y_k) kao kazne koje se plaćaju za kršenje primalnih i dualnih uvjeta do iteracije k. Inicijalno, procedura započinje s $x_0 = y_0 = \theta > 0$ i postupno ažurira par vektora $(x_k, y_k) = (x_{k-1} + \lambda_k, y_{k-1} + \mu_k)$ za odgovarajući odabrane vektore λ_k, μ_k. U završnom koraku t, procedura završava s primalnim i dualnim parom rješenja koji može biti nedopustiv. Skalirajući to rješenje s vrijednostima $M(t) = \max_{i=1,2,\ldots,m}\{A_i x_t\}$ i $m(t) = \min_{j=1,2,\ldots,n}\{y_t^T A^j\}$ konstruira se dopustivo primalno i dualno rješenje $(x^*, y^*) = (x_t / M(t), y_t / m(t))$. Uvjet zaustavljivanja t

ovisi o tome kako su definirani parove ažuriranja (λ_k, μ_k) te moraju biti dovoljno veliki kako bi osigurali $b^T y^* / c^T x^* \leq 1 + \epsilon$.

6.2.2 *Pregled algoritma*

Na tragu ovog generičkog postupka iterativnih ažuriranja primala i duala opisuje se algoritam prezentiran u [31] i iskazana je analiza vremenske složenosti. Analiza korektnosti se može naći u [31]. Algoritam je temeljen na multiplikativnim i aditivnim ažuriranjima primalnih i dualnih varijabli odgovarajuće formulacije linearnog programa problema pakiranja i pokrivanja. Algoritam iterira dok ne dostigne uvjet zaustavljivanja. Skaliranje parametara osigurava uvjet dopustivosti rješenja te omeđenost rješenja primala i duala s faktorom $1 + \epsilon$. Parametar preciznosti ϵ eksplicitno utječe na vrijeme izvršavanja s $1/\epsilon^2$.

ALGORITAM 7 CPU implementacija aproksimacijske sheme

procedure PC-APX(A, b, c, ϵ)
INPUT: $A \in \mathbb{R}_+^{m \times n}, b \in \mathbb{R}_+^m, c \in \mathbb{R}_+^n, \epsilon > 0$
OUTPUT: primal-dual rješenje (x^*, y^*) tako da $b^T y^* / c^T x^* \leq 1 + \epsilon$

$\epsilon' \leftarrow 1 - 1/\sqrt{1+\epsilon}, \quad \delta \leftarrow (1+\epsilon')((1+\epsilon')m)^{-1/\epsilon'}$
$x_0(j) \leftarrow 0, \quad j = 1, \ldots, n$
$y_0(i) \leftarrow \delta / b(i), \quad i = 1, \ldots, m$
$D(0) \leftarrow m \cdot \delta, \quad P(0) \leftarrow 0$
while $D(k) < 1$ **do**
  $q \leftarrow \text{argmin}_j \, L_{y_{k-1}}(j)$
  $p \leftarrow \text{argmin}_i \, b(i)/A(i,q)$
  $x_k(q) \leftarrow x_{k-1}(q) + b(p)/A(p,q)$
  $y_k(i) \leftarrow y_{k-1}(i) \left(1 + \epsilon' \frac{b(p)/A(p,q)}{b(i)/A(i,q)}\right), \quad i = 1, \ldots, m$
  $P(k) \leftarrow P(k-1) + c(q)b(p)/A(p,q)$
  $D(k) \leftarrow D(k-1) + c(q)b(p)/A(p,q) \cdot \rho(y_{k-1})$
end while
$\rho \leftarrow \min_j L_{y_k}(j)$
$x(j) \leftarrow x_t(j) / \log_{1+\epsilon'}((1+\epsilon')/\delta), \quad j = 1, 2, \ldots, n$
$y(i) \leftarrow y_t(i)/\rho, \quad i = 1, 2, \ldots, m$
return (x, y)
end procedure

Koriste se sljedeće oznake: s $L_y(j) := \sum_{i=1}^{m} A(i,j)y(i)/c(j)$ neka je označena *duljina* stupca j s obzirom na dualnu varijablu y te minimalna duljina stupaca s $\rho(y) = \min_j L_y(j)$.

Procedura je iterativna. Neka x_{k-1} i y_{k-1} označavaju primalne odnosno dualne varijable na početku k-te iteracije. Isto tako, neka P_{k-1} označava vrijednost primalnog rješenja na početku k-te iteracije. S q je označen stupac od A minimalne duljine: $q := \arg\min_j L_{y_{k-1}}(j)$ te neka je p minimizator skupa vrijednosti $b(i)/A(i,q), i = 1,2,\ldots,m$. Pseudokod algoritma je dan u proceduri PC-APX u *Algoritmu 7*.

PRIMJEDBA 11: Prednost ovog algoritma što ga čini posebno atraktivnim je mogućnost rješavanja instanci problema pakiranja i pokrivanja koje su eksponencijalno velike u n, gdje n označava broj stupaca matrice A. Ideja je zamijeniti *korak 7* u *Algoritmu 7* sa *oracle* rutinom u koja radi u polinomnom vremenu za nalaženje stupca od A čija je duljina najmanja te u *koraku 15* ažurirati samo primalno rješenje koje odgovara tom stupcu. Garg i ostali su u [31] pokazali kakve *oracle* rutine se koriste za različite tipove problema toka robe.

VRIJEME IZVRŠAVANJA Vrijeme izvršavanja sekvencijalnog algoritma je $O(\epsilon^{-2}mD\log m)$ gdje je D broj elemenata različitih od nula u matrici A. Kako bi odredili gornju granicu na broj iteracija primjetite da u nekoj iteraciji k, algoritam odabire neke dualne varijable $y_k(i)$ i ažurira ih faktorom $1+\epsilon'$, a ostale dualne varijable s $1+z\epsilon' \leq 1+\epsilon'$, $z = \frac{b(p)/A(p,q)}{b(i)/A(i,q)}$. Uvjet zaustavljivanja zahtjeva $(1+\epsilon')/b(i) > y_t(i)$ što u najgorem slučaju algoritam mora to učiniti za svih m dualnih varijabli. Stoga, za dualne varijable vrijedi $(1+\epsilon')/b(i) > \delta/b(i)(1+\epsilon')^t$ što daje gornju granicu na ukupan broj iteracija kao $m\lceil 2/\epsilon' \log_{1+\epsilon'} m\rceil$.

Ovisno o tome je li matrica A dana u gustoj (*dense*) ili u rijetkoj (*sparse*) formi može se pronaći stupac minimalne duljine u $O(mn)$ vremenu odnosno $O(D)$ vremenu. Stoga, ukupno vrijeme potrebno za pronaći $(1-\epsilon')^{-2}$ aproksimaciju rješenja problema pakiranja i pokrivanja je $O(\epsilon'^{-1}Dm\log_{1+\epsilon'} m)$ za eksplicitno dane rijetke matrice uvjeta odnosno $O(\epsilon'^{-1}m^2n\log_{1+\epsilon'} m)$ za eksplicitno dane guste matrice uvjeta. Dakle, asimptotsko vrijeme izvršavanja PC-APX procedure za (31) je $O(\epsilon^{-2}Dm\log m)$ za $\epsilon = (1-\epsilon')^{-2}$ dovoljno malen.

THEOREM 6.1. *Postoji algoritam koji računa $(1+\epsilon)$-aproksimaciju za linearni program pakiranja i pokrivanja (31) u vremenu $O(\epsilon^{-2}mD\log m)$ gdje je m broj uvjeta i D je broj elemenata različito od nule u danoj matrici uvjeta.*

6.3 CUDA PLATFORMA

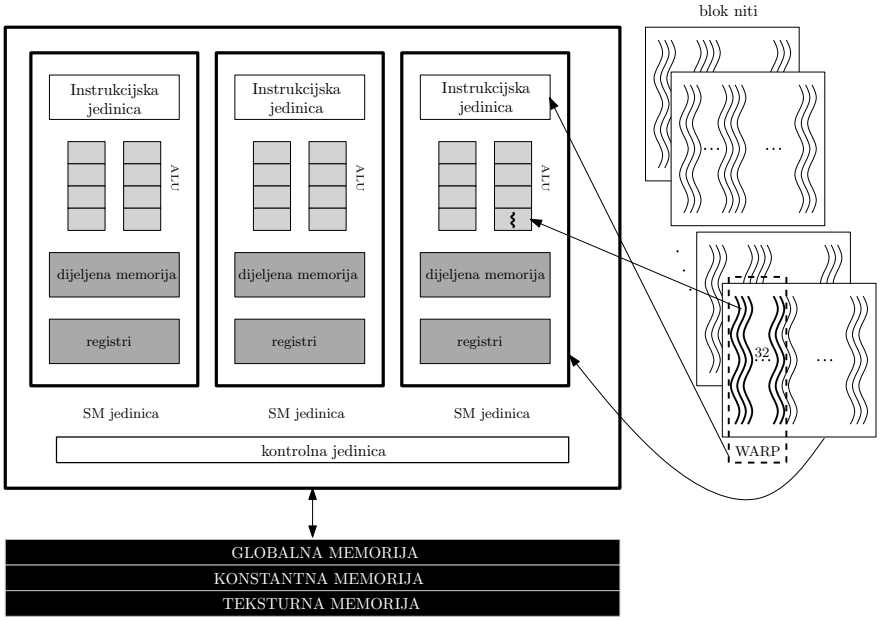
CUDA (engl. *Compute Unified Device Architecture*) predstavlja računalni pogon za razvijanje na NVIDIA grafičkim procesorima (GPU). Od prve pojave u 2007. god., razne industrijske i privredne grane ostvarili su dobit u performancama u razvijanju aplikacija u CUDA-i u usporedbi s prijašnjim *state-of-the-art* aplikacijama u nekoliko reda veličine. Upravo pojava CUDA-e dovelo je do razvoja brzih implementacija aplikacija opće namjene na GPU. U [59] daje pregled rezultata za algoritme obrade slika. Algoritmi na grafovima su prijavljeni na GPU u [43] za velike instance. Postoje radovi koje se tiču paralelnih primitiva kao što su prefiks sume, redukcije i sortiranje [59]. GPU su korišteni za numeričke algoritme [36] te u linearnoj optimizaciji [63] i [80].

NVIDIA unificirana arhitektura u modernim GPU podržavaju grafičko i numeričko računanje. Generalno, GPU se može promatrati kao masivna višenitna arhitektura[2] koja se sastoji od velikog broja procesorskih čvorova. Tipični grafički procesori široke primjene sastoje se od više *stream* procesora ili CUDA jezgri grupirane u jedinice koje se zovu *streaming multiprocesori* (SM). CUDA API omogućuje implementaciju paralelnog računanja na većem broju niti koji se izvršavaju na GPU jezgrama. API koristi koncept mreže (engl. *grid*) kako bi grupirao niti u logičke jedinice koje se zovu *blokovi niti*[3] slijedno mapirani za obrađivanje na SM-ovima prilikom izvršavanja.

Jezgre u SM podržavaju velik broj niti izvršenja u usporedbi sa standardnim CPU jezgrama koje su napravljene za optimalnu paralelnu obradu do maksimalnog broja CPU jezgri koje je u trenutku pisanja 8. Dijeljena memorija je posebna vrsta brze memorije koje se nalazi blizu SM i dostupna je svim nitima u SM i reda je veličine 46 kB po SM-u na modernim GPU. Štoviše, SM dodjeljuje skup privatnih registara svakoj niti i reda je veličine 32kB po SM-u na modernim GPU. Doduše, veličina dijeljenje memorije i broj registara po SM se razlikuje od arhitekture do arhitekture i do programera je je da prouči arhitekturu platforme koju koristi prije optimizacije programskom koda. Arhitekturalna organizacija prikazana je na *slici 23*.

Svaki SM na većini GPU arhitektura ima jednu instrukcijsku jedinicu. Stoga, sve niti mapirane na neki SM moraju izvršiti istu instrukciju svaki ciklus, ali na različitim podacima. Svaka logička grupa od

2 engl. *massively multithreaded architecture*
3 engl. *thread blocks*



Slika 23: CUDA GPU arhitektura

32 različite niti koje dijele instrukciju zove se *warp*. Niti koje pripadaju različitim warpovima mogu izvršavati različite instrukcije ali u različitom vremenskom okviru. Drugim riječima, grupa niti u warpu se ponaša kao SIMD (*Single Instruction Multiple Data*) jedinica. S druge strane, svaki warp u bloku se ponaša kao SMT (*Simultaneous Multithreading*) jedinica. Kao rezultat, CUDA proviđa jednostavne sinkronizacijske primitive koje stvaraju sinkronizacijske točke za sve niti unutar bloka, točnije, između različitih warpova u bloku, koju su pridjeljeni nekom SM-u sa sljedećom svrhom: ukoliko nit izvrši programski kod iza te točke, nijedna nit ne izvršava kod prije te točke. Na takav način, niti unutar bloka mogu sigurno dijeli međusobono rezultate.

Moderne GPU arhitekture se tipično zovu SIMT (*Single Instruction Multiple Threads*) i smatraju se kao relaksirani SIMD zbog gore navedenih razloga da niti u nekom warpu mogu izvršavati različite instrukcije. Dakle, različite instrukcije se ne mogu konkurentno izvršavati pa su vremenski serijalizirani na uštrb performanse. S druge strane, SIMT performanse uvelike ovise o raspoloživosti velikog broja niti koje se izvršavaju i brzog kontekstualnog prebacivanja između niti (warpova) kako bi se sakrila memorijska latenccija. Jedna od većih nedostataka SIMT-a je divergencija toka programa. Niti koje izvršavaju različite instrukcije u warpu mogu divergirati prema IF-THEN-ELSE naredbama. Na primjer, kada niti imaju isti tok - svi rade IF tok, a

nitko ELSE - tada sve niti imaju maksimalni protok. Nažalost, u situacijama kada jedan ili više niti trebaju vršiti ELSE, moraj čekati da niti izvrše IF tok.

Performanse CUDA implementacije uvelike ovisi i o korištenju memorijske hijerarhije. Kao što je spomenuto ranije, skup od 32-bitnih registara su ravnomjerno raspoređeni nitima u SM-u i koriste se kao *lokalna memorija* neke niti. Niti koriste 49kB *dijeljene memorije* po SM i eksplicitno je dostupno programeru. Postoji, *off-chip* globalna memorija velike veličine kojoj mogu pristupiti sve niti unutar mreže. Glavni nedostatak globalne memorije je što zahtjeva znatno više ciklusa za operacije čitanja i pisanja (*load/store*). Postoje posebni tipovi *read-only* pričuvne globalne memorije koje smanjuju broj potrebnih operacija čitanja i pisanja u memoriju koje se zovu *teksturna memorija* i *konstantna memorija*.

CUDA sučelje programiranja proviđa skoro Parallel Random Access Machine (PRAM) arhitekturu ukoliko se koristi isključivo globalna memorija (pogledati [43]). PRAM apstrakcija je često korištena kako bi se istražila teoretska paralelna performansa algoritama [46] i kao pretpostavku uzima neograničeni broj procesora i jediničnom latencijom zajedničke memorije iz bilo kojeg procesora. Stvarne sklopovske implementacije PRAM-a su dosada bile rijetke [45]. Hong i ostali su uspostavili direktno mapiranje CUDA nitnog modela s PRAM apstrakcijom u [43].

CUDA pretpostavlja CPU-GPU hibridnu programersku paradigmu. Potrebno joj je da CPU i GPU zajedno sudjeluju u obradi podataka. Terminološki, CPU zajedno sa svojim memorijskim prostorom se zove HOST, a GPU sa svojom memorijom se zove DEVICE. Za izvršavanje programskog koda potrebno je imati kontrolni tok na CPU, a računanja koje zahtjevaju paralelnu obradu ostvaruju se eksplicitnim pozivima *jezgrinih funkcija*[4] unutar kontrolnog toka programa. Prije nego se ostvaruju jezgrini pozivi svi potrebni podaci nad kojima će se raditi paralelno računanje moraju biti u memoriji DEVICE-a. Svaki jezgrin poziv eksplicitno definira skalabilnost izvršavanja programa defnirajući dimenzije mreže i blokova. CUDA API nudi dodatne mehanizme sinkronizacije između GPU i CPU programskog toka. Jedini način, koliko je poznato u trenutku pisanja, da se sinkroniziraju niti u čitavoj mreži jest da se paralelna obrada podijeli u više jezgrinih funkcija povezanih s HOST-DEVICE sinkronizacijom.

4 engl. *kernel functions*

6.4 PARALELNA IMPLEMENTACIJA APROKSIMACIJSKE SHEME

Proširivanje sekvencijalnog koda na CPU-GPU platformi podrazumijeva korištenje HOST platforme kao kontrolnu jedinicu računanja, a DEVICE platformu kao jedinicu za masivno paralelno računanje. Srž ovog algoritma leži u matrično-vektorskom računu i pretraživanju vektora što se može vrlo efikasno implementirati u CUDA-i (detaljnije o tome pogledati u [6]). Jezgra algoritma je iterativna procedura koja ažurira primalno i dualno rješenje. Iteracije algoritma su ovisne o prethodnim iteracijama pa one ostaju sekvencijalne.

6.4.1 *PRAM model algoritma*

Algoritam je opisan pseudokodom u *Algoritmu 8*. U PC-APX-CUDA proširuje se programski kod iz *Algoritma 7* tako da se označava dijelovi koji se paraleliziraju označeni kao jezgrine funkcije. Prije poziva jezgrinih funkcija podaci se kopiraju u DEVICE adresni prostor. Radi preglednosti, koristi se ista notacija za HOST i DEVICE varijable. Kako bi izbjegli često kopiranje varijabli iz DEVICE u HOST memorijski prostor i obratno koriste se jedna varijabla za provjeru uvjeta zaustavljivanja koja je uvijek prisutna u HOST memoriji (*zero-copy* varijabla). Čitavo računanje po iteraciji algoritma se obavlja na DEVICE-u. Sinkronizacija DEVICE i HOST jedinice osigurava CUDA mehanizam SYNC HOST-DEVICE.

6.4.2 *Opis jezgrenih funkcija*

kernel 1 - PREDOBRADA: Jezgrina funkcija inicijalizira vrijednosti primalnih i dualnih rješenja x, y i računa pomoćno polje P koje sprema minimalne vrijednosti od $b(i)/A(i,j), j = 1, 2, \ldots, n$. Za svaki stupac matrice A pridružuje se blok predefiniranog broja niti koji pristupaju elementima stupaca. U svakom bloku koristi se tehnika logaritamske redukcije kao standardna rutina za nalaženje minimalne vrijednosti po stupcu.

kernel 2 - RAČUNANJE VEKTORA R: Jezgrina funkcija treba obraditi sve stupce i retke matrice A. Kako bi se iskoristio masivni paralelizam, blok niti je pridjeljen stupcu j i računa se $L_y(j)$ sumacijskom

ALGORITAM 8 CUDA paralelna implementacija za PC-APX

procedure PC-APX-CUDA(A, b, c, ϵ)
INPUT: $A \in \mathbb{R}_+^{m\times n}, b \in \mathbb{R}_+^m, c \in \mathbb{R}_+^n, \epsilon > 0$
OUTPUT: primal/dual rješenje (x^*, y^*) za koje $b^T y^*/c^T x^* \leq 1+\epsilon$

$\epsilon' \leftarrow 1 - 1/\sqrt{1+\epsilon}, \quad \delta \leftarrow (1+\epsilon')((1+\epsilon')m)^{-1/\epsilon'}$
$\mathcal{D}(0) \leftarrow m \cdot \delta, \quad \mathcal{P}(0) \leftarrow 0$
alociraj i kopiraj A, b, c na DEVICE
alociraj x, y i inicijaliziraj na DEVICE
KERNEL 1: ▷ priprema
  $y_0(i) \leftarrow \delta/b(i), \quad i = 1, \ldots, m$
  $x_0(j) \leftarrow 0, \quad j = 1, \ldots, n$
  $P(j) \leftarrow \min_i b(i)/A(i,j), \quad j = 1, 2, \ldots, n$
end KERNEL
SYNC HOST-DEVICE
while $\mathcal{D}(k) < 1$ **do**
  KERNEL 2: ▷ računaj paralelno $R(j)$
    $R(j) \leftarrow L_y(j), j = 1, 2, \ldots, n$
  end KERNEL
  SYNC HOST-DEVICE
  KERNEL 3: ▷ nađi q koristeći logaritamsku redukciju
    $q \leftarrow \text{argmin}_j\, R(j)$
    $\rho \leftarrow R(q)$
  end KERNEL
  SYNC HOST-DEVICE
  KERNEL 4: ▷ ažuriraj primal/dual rješenja
    $x_k(q) \leftarrow x_{k-1}(q) + P(q)$
    $y_k(i) \leftarrow y_{k-1}(i)\left(1 + \epsilon' A(i,q)\frac{P(q)}{b(i)}\right), \quad i = 1, \ldots, m$
    $\mathcal{P}(k) \leftarrow \mathcal{P}(k-1) + c(q)P(q)$
    $\mathcal{D}(k) \leftarrow \mathcal{D}(k-1) + c(q)P(q) \cdot \rho$
  end KERNEL
  SYNC HOST-DEVICE
end while ▷ stop u iteraciji t
KERNEL 2&3 poziv za nalaženje ρ
SYNC HOST-DEVICE ▷ pronaći faktor skaliranja
KERNEL 5:
  $x_t(j) \leftarrow x_t(j)/\log_{1+\epsilon'}((1+\epsilon')/\delta), \quad j = 1, 2, \ldots, n$ ▷ skaliraj rješenje
  $y_t(i) \leftarrow y_t(i)/\rho, \quad i = 1, 2, \ldots, m$
end KERNEL
SYNC HOST-DEVICE
return (x_t, y_t)
end procedure

tehnikom logaritamske redukcije nitima iz bloka. Postupak se čini za svaki stupac $j = 1, 2, \ldots, n$.

kernel 3 - RAČUNANJE VRIJEDNOSTI ρ, q: Jezgrina funkcija je primarno osmišljena kako bi se sinkronizirali niti u jednom bloku, stoga ova jezgrina funkcija započinje s jednim blokom i najvećim mogućim brojem niti po bloku. Iznova, koristi se logaritamska redukcija za nalaženje vrijednosti q, ρ.

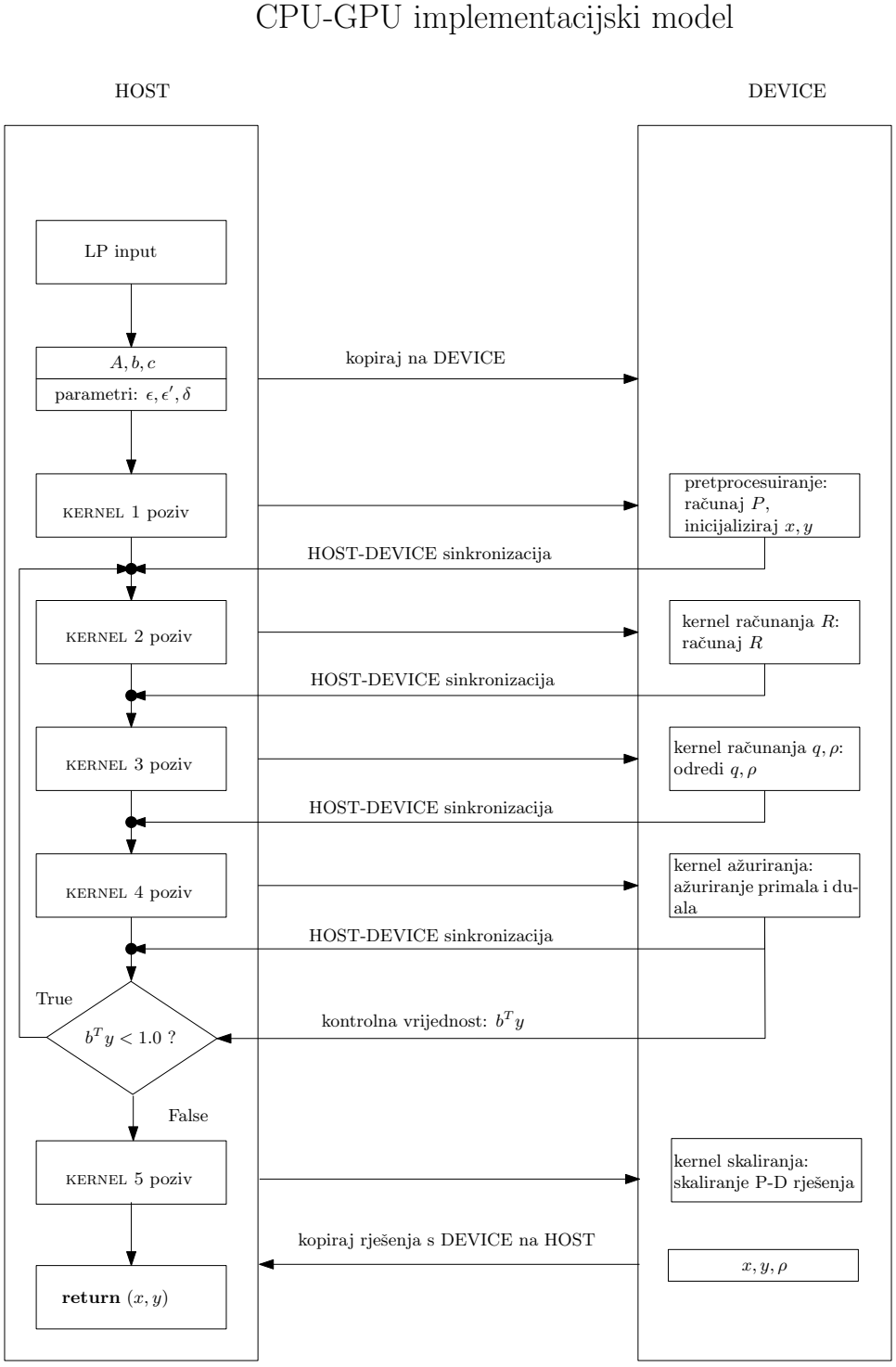
kernel 4 - AŽURIRANJE PRIMALNIH I DUALNIH RJEŠENJA: U ovom koraku, jezgrine funkcije mogu pristupati vrijednostima q, p koji su nužni za ažuriranje određenog elementa u primalnom rješenju. Isto tako, ažuriraju se sve dualne varijable koristeći dovoljan broj blokova s nitima tako da je svaka nit odgovorna za barem jednu varijablu dualnog rješenja y.

kernel 5 - SKALIRANJE PRIMAL-DUAL RJEŠENJA: Prije nego što se vrati rješenje, primalna i dualna rješenja x_t, y_t se moraju skalirati nakon terminacije iterativnog postupka. Kao i u prethodnoj jezgrinoj funkciji, koristi se mreža s dovoljnim brojem blokova u kojem svaka nit skalira primalno rješenje faktorom $1 / \log_{1+\epsilon'}((1+\epsilon')/\delta)$ i dualno rješenje s ρ koje odgovara posljednjem izračunatom dualnom rješenju y_t.

Nakon terminacije iterativnog postupka primalno i dualno rješenje se kopira nazad s DEVICE-a na HOST i vraća. Detaljniji opis DEVICE-HOST komunikacije je pokazano na *slici 24*.

6.4.3 *Vrijeme izvršavanja*

CUDA nitni model je prirodno temeljen na PRAM apstrakciji kao što se tvrdi u [43]: svaka nit predstavlja procesor kojem je pridjeljen jedinstveni identifikator i svojstven posao te postoji linearno mnogo N niti koji rade sinkronizirano. Skup tih procesora označava se s $\{P_0, P_1, \ldots, P_N\}$. Procesori mogu dijeliti (neograničenu) vanjsku memoriju u jediničnom vremenu i nema koncepta prinuđene memorijske koherencije nad nitima koje se izvršavaju. Ulaz i izlaz za model predstavlja N predmeta spremljenih u N memorijskih ćelija. Sve artmetičke operacije u procesorima zahtjevaju jedinično vrijeme i svaki procesor ima svoju lokalnu memoriju za spremanje i čitanje privatnih podataka. Model izvršava instrukcije u 3 faze: u svakom koraku, svaki



Slika 24: HOST/DEVICE iskorištenost

procesor čita iz memorije (lokalne ili zajedničke), napravi lokalno računanje i sprema rezultat u lokalnu ili zajedničku memoriju. Procesori izvršavaju trofazne PRAM instrukcije sinkronizirano. Ne postoji direktna interprocesorska komunikacija, već u posrednom obliku preko zajedničke memorije. Inicijalni procesor P_0 se sastoji od aktivacijskog registra koji definira maksimalni broj aktivnih procesora. Inicijalno, jedino je P_0 aktivan koji aktivira preostale procesore. Računanje traje dok P_0 ne završi, u trenutku kad svi ostali procesori su završili. Paralelna vremenska složenost je definirana vremenom izvršavanja P_0 procesora te prostorna složenost kao broj korištenih zajedničkim memorijskih ćelija.

Korištenje PRAM apstrakcije na CUDA nitnom modelu ima svoje nedostatke. Nedostaje koncept warpova, koncept memorijske hijerarhije i hijerarhija obradbe (niti izvršenja organizirani u blokove, blokovi u mrežu) koje u nekim slučajevima može dovesti do veće degradacije u performansama zbog divergencije niti u warpu ili raspršenog memorijskog pristupa. Isto tako korištenje specijaliziranih vrsta memorije (teksturne memorije ili konstantne memorije) umjesto globalne memorije može dati značajne beneficije. Postoje studije u radovima [43] i [59] koje promatraju ovakve situacije i adaptiraju PRAM model. Za našu aplikaciju PRAM model se pokazao dovoljno dobar što će se pokušati potkrijepiti eksperimentalnim rezultatima.

Iskazujemo konačan rezultat za algoritam PC-APX-CUDA:

THEOREM 6.2. *Postoji algoritam na PRAM stroju koji računa* $(1+\epsilon)$*-aproksimaciju linearnog programa pakiranja i pokrivanja oblika* (31) *u asimptotskom* $O(\epsilon^{-2}nm \log n \log m)$ *vremenu.*

Dokaz. Kako bi argumentirali tvrdnju teorema dovoljno je pobrojati vrijeme potrebno za pozive jezgrinih funkcija algoritma PC-APX-CUDA i množiti s gornjom granicom na broj iteracija algoritma. Pretpostavlja se da $\max\{m,n\} \leq N$ što je razumno za pretpostaviti zbog velikog broja dostupnih niti na CUDA platformi koje mogu raditi paralelno.

U *Kernelu 1* inicijalizacijski korak košta $O(1)$ vrijeme. Logaritamska redukcija na linearno mnogo procesora zahtjeva $O(\lg n)$ vremena kako bi se našao minimum. Računanje polja $R(j), j = 1, 2, \ldots, n$ PRAM zahtjeva $O(\lg m)$ vremena po svakom stupcu u *Kernelu 2*. Nalaženje minimalnog indeksa q polja R uzima $O(\lg n)$ vremena po iteraciji u *Kernelu 3*. Ažuriranje primalnih i dualnih rješenja zahtjeva $O(1)$ vremena u *Kernelu 4* kao i skaliranju u *Kernelu 5*. Iz *Teorema 6.1* poznato je

da najveći broj iteracija je ograničen s $O(\epsilon'^{-1} m \log_{1+\epsilon'} m)$ implicirajući ukupno vrijeme računanja $O(\epsilon'^{-1} nm \log_{1+\epsilon'} m)$, $\epsilon' = 1 - (1+\epsilon)^{-1/2}$. □

6.5 EKSPERIMENTI

U eksperimentalnom dijelu prikazuju se empirijski rezultati dobiveni na određenoj CUDA platformi s različitim skupom podataka.

6.5.1 *Specifikacija CUDA platforme*

Sva mjerenja su uzeta na platformi koja se sastoji od četverojezgrenog Intel procesora i5 s radnim taktom 2.8 MHz i 8 GB RAM memorije zajedno s grafičkom karticom TESLA C2070 s 6 GB DDR5 RAM memorije. Grafička kartica C2070 sadrži 448 masivno nitiranih procesorskih jezgri s radnim taktom 1.15 GHz koje može pokretati oko 30000 niti te visoka memorijska propusnost od 144 GB/s.

6.5.2 *Testovi i usporedbe*

Eksperimentalni rezultati se temelje na usporedbi sa simpleks metodom iz *GNU Linear Programming Kit* (GLPK) i slijedne implementacije algoritma PC-APX odnosno paralelne CUDA implementacije PC-APX-CUDA. Simpleks metoda računa LP optimalno gdje slijedna i paralelna CUDA implementacija aproksimativnog rješavatelja nalaze aproksimativno rješenje unutar $1+\epsilon$ faktora.

Simpleks metoda u GLPK paketu koristi dvofazni pristup: prva faza simpleks metode može riješavati odgovarajuće formirani LP za nalaženje početnog dopustivog rješenja za početak simpleks metode na originalnom LP-u u drugoj fazi. GLPK API proviđa mehanizme za rješavanje simpleks metoda na primalu ili dualu ili hibridno kako bi izbjegao mogućnosti cikliranja. Više o implementaciji simpleks metode za GLPK paket može se naći u [68].

MATRICE UVJETA RACIONALNOG TIPA Skup test inputa korištene u eksperimentu su kvadratne matrice različitog n broja redaka i stupaca te različite gustoće d (omjer broja elemenata matrice različito od nule i ukupnog broja elemenata matrice, $d = D/n^2$). Elementi različiti

od nule su racionalni brojevi izabrani s vjerovatnošću d s vrijednostima uniformno distribuirani na intervalu $[0,1]$.

Sve testne instance odgovaraju linearnim programima problema pakiranja i pokrivanja s gustoćama $d = \{1/4, 1/2, 3/4, 1\}$. Vektori b, c su postavljeni na vektor sa svim jedinicima jer se može pokazati da općeniti oblik linearnog programa pakiranja i pokrivanja se može ekvivalentno riješiti tako da se riješi transformirani linearni program oblika $\max\{c^T x \colon \tilde{A}x \leq \mathbf{1}, x \geq 0\} = \min\{b^T y \colon \tilde{A}^T y \geq \mathbf{1}, y \geq 0\}$ gdje je matrica $\tilde{A}(i,j) = A(i,j)/(b_i c_j), i, j = 1, 2, \ldots, n$ s primal-dual rješenjem (x^*, y^*). Polazni LP ima rješenje onda $(x^*(i)/c(i), y^*(i)/b(i))$, $i = 1, 2, \ldots, n$.

Za ovakav poseban tip inputa, može se raspisati asimptotska notacija *Teorema 6.2* i eksplicitno zapisati vremensko izvršavanje kao funkcije od n, d, ϵ:

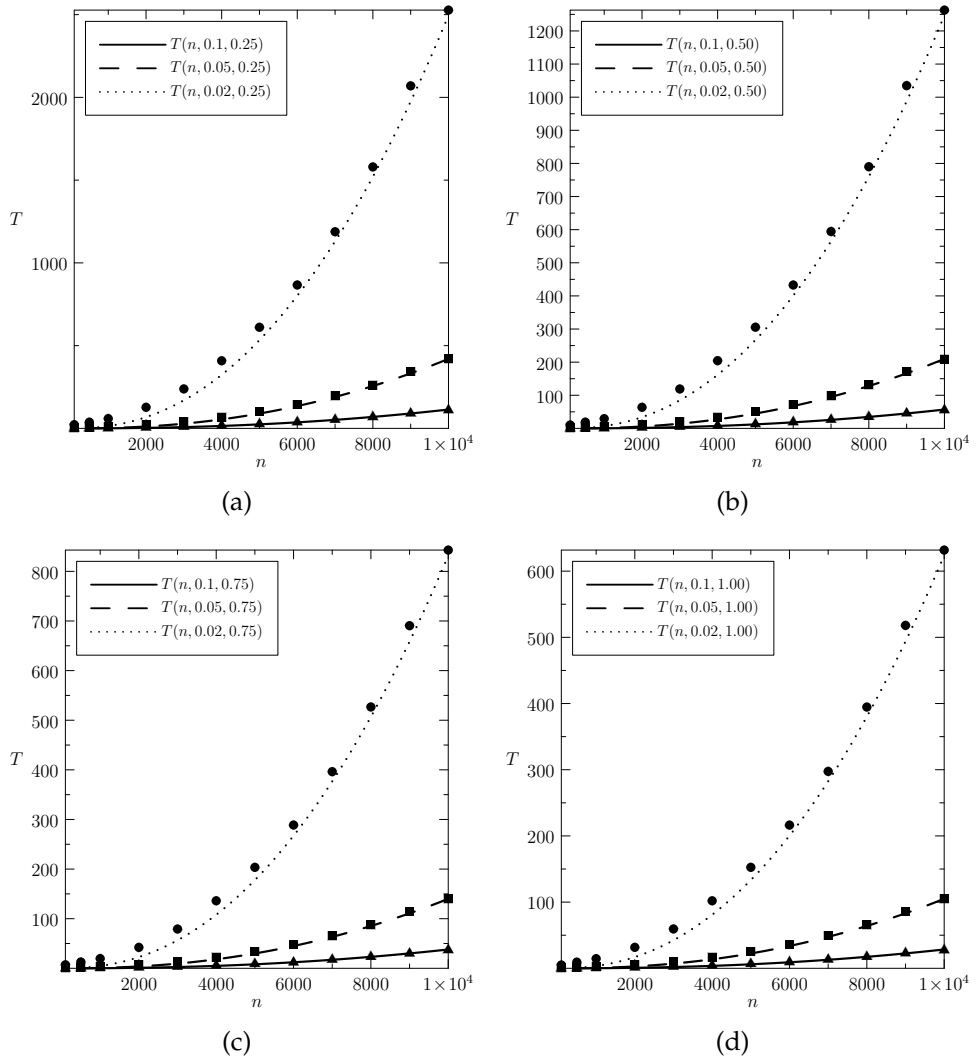
$$T(n, \epsilon, d) = C \cdot (d\epsilon')^{-1} n^2 \log_{1+\epsilon'} n \log_2 n, \quad \epsilon' = 1 - (1+\epsilon)^{-1/2}. \tag{34}$$

gdje je C konstanta fiksirana na $C = 4.9 \times 10^{-12}$, d je gustoća test ulaza $1/4 \leq d \leq 1$.

Na *slici 25* prikazano je kako pretpostavljena vremenska funkcija (34) dobro aproksimira mjerene rezultate za različite skupove podataka u 4 različita dijagrama koje odgovaraju vrijednostima od d.

U *tablici 1 – 4* redom su predstavljena ukupna vremena izvršavanja simpleks algoritma i slijedne (PC-APX) odnosno paralelne (PC-APX-CUDA) implementacije za različite ulazne veličine i različite gustoće matrice A. Unosi u tablici koje nedostaju ukazuju na vremena izvršavanja preko 1 sat. Predstavljeni su rezultati relativnog ubrzanja, dakle, omjera vremenskog izvršavanja paralelne implementacije u usporedbi sa sekvencijalnom implementacijom te isto tako usporedba sa simpleks metodom. Promatrajući tablice, može se primjetiti da je značajno ubrzanje paralelne implementacije ostvareno za tek relativno veće ulaze, točnije od veličina 1000x1000. Primjećuje se da je to i očekivano, jer CUDA kao paralelna arhitektura jest dizajnirana za masivni paralelizam (TESLA grafička kartica radi s redom veličine 30000 niti izvršenja paralelno i za značajnije ubrzanje potrebno ih je što više iskoristiti).

Kao što se može vidjeti, GLPK rješavatelj sa implementacijom simpleks metode rješeva optimalno LP problem čak i brže nego predstavljene implementacije koje aproksimiraju rješenje za LP instance manje od 1000x1000, posebno za male vrijednosti ϵ. Primjećuje se također



Slika 25: Eksperimentalno mjerena vremena su dobro predviđena funkcijom $T(n, \epsilon, d)$ za predstavljeni skup podataka.

kvadratni utjecaj faktora $1/\epsilon^2$ na vremensko izvršavanje PC-APX i PC-APX-CUDA algoritama. Što su veće instance ulaza PC-APX postaje brži, a samim time PC-APX-CUDA. Treba napomenuti da efikasnost simpleks metode uvelike ovisi o broju iteracija. Ukoliko je metoda pivotiranja korištena u simpleks metodi posebno uspješna na neke tipove ulaza, onda simpleks metoda može vratiti rezultat u nekoliko iteracija. S druge strane, broj iteracija može biti relativno velik (Klee i Minty [55] su dali primjer koji pokazuje vremensku složenost u najgorem slučaju koja može biti eksponencijalna), premda se u praksi rijetko događa. U usporedbi, PC-APX, PC-APX-CUDA algoritmi imaju strogu polinomnu granicu na broj iteracija što objašnjava relativno velike faktore ubrzanja koje se dobije za velike instance u usporedbi sa simpleks metodom.

MATRICE UVJETA BINARNOG TIPA Dodatno su testirane PC-APX-CPU i PC-APX-CUDA implementacije na specijalnog klasi input matrica čiji su elementi ili 0 ili 1. Binarni tip matrica se pojavljuju kao strukture podataka koje predstavljaju relaciju incidencije između skopova i elemenata u klasičnom problemu skupovnog pokrivača (SET-COVER).

Rezultati u *tablici 5–7* prikazuju vremena izvršavanja GLPK simpleks implementacije i implementacije PC-APX-CPU i PC-APX-CUDA algoritama na binarnom tipu matrice uvjeta za $d \in \{1/4, 1/2, 3/4\}$. Slučaj $d = 1$ se izostavlja zbog trivijalne prirode.

Iz eksperimenata se može primjetiti kako koristeći GPU implementaciju algoritma može se dobiti do 4 reda veličine uvećanja u usporedbi s vremenom potrebno za simpleks metodu u GLPK rješavatelju za dovoljno velike input instance i ϵ odgovarajuće malen. U eksperimentima GLPK rješavatelj je pokazao problem cikliranja pa smo koristili robusniju verziju metode: dvofaznu primal-dual simpleks metoda, koja načelno, počne riješavati dvofazni dualni linearni program i ukoliko nastane problem cikliranja prebacuje simpleks metodu na primalni linearni program.

RJEŠAVANJE VELIKIH INSTANCI Implementacija PC-APX-CUDA algoritma trenutno može aproksimativno rješiti instance linearnih programa problema pakiranja i pokrivanja do veličine 35000, granice koja je određena test platformom za naš specifičan skup podataka. U implementaciji, matrice i vektori u gustoj formi spremanja zahtjevaju barem $(n^2 + 2n)\cdot$`sizeof(float)` bajtova DEVICE memorije. Ova gornja

n	ϵ	GLPK simplex	PC-APX CPU	PC-APX CUDA	PC-APX-CPU / PC-APX-CUDA	GLPK-simplex / PC-APX-CUDA
100	0.1	0.006	0.412	0.998	0.41	0.01
100	0.05	0.006	1.538	3.737	0.41	0.00
100	0.02	0.006	9.203	22.488	0.41	0.00
500	0.1	0.868	12.411	1.691	7.34	0.51
500	0.05	0.868	46.488	6.344	7.33	0.14
500	0.02	0.868	278.728	37.560	7.42	0.02
1000	0.1	15.709	54.954	2.623	20.95	5.99
1000	0.05	15.709	206.069	9.880	20.86	1.59
1000	0.02	15.709	1236.360	59.023	20.95	0.27
2000	0.1	450.542	244.676	5.654	43.28	79.69
2000	0.05	450.542	918.324	21.190	43.34	21.26
2000	0.02	450.542	5515.120	127.389	43.29	3.54
3000	0.1	1935.900	582.581	10.427	55.88	185.67
3000	0.05	1935.900	2187.450	39.573	55.28	48.92
3000	0.02	1935.900	13120.120	238.472	55.02	8.12
4000	0.1	17467.100	1071.280	18.132	59.08	963.34
4000	0.05	17467.100	4024.790	68.040	59.15	256.72
4000	0.02	17467.100	24178.600	408.599	59.17	42.75
5000	0.1	24161.400	1719.790	27.083	63.50	892.14
5000	0.05	24161.400	6462.780	101.750	63.52	237.46
5000	0.02	24161.400	38764.200	610.820	63.46	39.56
6000	0.1			38.014		
6000	0.05			143.755		
6000	0.02			866.461		
7000	0.1			52.286		
7000	0.05			197.216		
7000	0.02			1188.190		
8000	0.1			69.388		
8000	0.05			262.144		
8000	0.02			1579.800		
9000	0.1			91.066		
9000	0.05			343.560		
9000	0.02			2069.620		
10000	0.1			111.050		
10000	0.05			419.760		
10000	0.02			2527.210		

Tablica 1: Eksperimenti s racionalnim matricama gustoće 25% ($d = 1/4$).

n	ϵ	GLPK SIMPLEX	PC-APX CPU	PC-APX CUDA	PC-APX-CPU / PC-APX-CUDA	GLPK-SIMPLEX / PC-APX-CUDA
100	0.1	0.011	0.189	0.480	0.39	0.02
100	0.05	0.011	0.708	1.718	0.41	0.01
100	0.02	0.011	4.237	10.301	0.41	0.00
500	0.1	1.649	6.213	0.842	7.38	1.96
500	0.05	1.649	23.285	3.154	7.38	0.52
500	0.02	1.649	139.657	18.862	7.40	0.09
1000	0.1	23.686	27.411	1.308	20.96	18.11
1000	0.05	23.686	102.805	4.906	20.96	4.83
1000	0.02	23.686	617.076	29.390	21.00	0.81
2000	0.1	415.878	122.222	2.822	43.30	147.35
2000	0.05	415.878	458.706	10.587	43.33	39.28
2000	0.02	415.878	2755.020	63.637	43.29	6.54
3000	0.1	2972.220	290.885	5.240	55.52	567.26
3000	0.05	2972.220	1087.010	19.752	55.03	150.48
3000	0.02	2972.220	6549.290	118.956	55.06	24.99
4000	0.1	11474.400	530.200	9.074	58.43	1264.53
4000	0.05	11474.400	2002.380	34.053	58.80	336.96
4000	0.02	11474.400	12068.900	204.558	59.00	56.09
5000	0.1	25921.100	850.370	13.535	62.83	1915.06
5000	0.05	25921.100	3210.990	50.821	63.18	510.05
5000	0.02	25921.100	19163.800	305.484	62.73	84.85
6000	0.1			19.147		
6000	0.05			72.071		
6000	0.02			432.943		
7000	0.1			26.240		
7000	0.05			98.750		
7000	0.02			594.442		
8000	0.1			34.725		
8000	0.05			131.065		
8000	0.02			790.065		
9000	0.1			45.872		
9000	0.05			171.690		
9000	0.02			1034.870		
10000	0.1			55.490		
10000	0.05			209.473		
10000	0.02			1262.910		

Tablica 2: Eksperimenti s racionalnim matricama gustoće 50% ($d = 1/2$).

n	ϵ	GLPK simplex	PC-APX CPU	PC-APX CUDA	PC-APX-CPU / PC-APX-CUDA	GLPK-simplex / PC-APX-CUDA
100	0.1	0.012	0.131	0.323	0.41	0.04
100	0.05	0.012	0.489	1.193	0.41	0.01
100	0.02	0.012	2.926	7.089	0.41	0.00
500	0.1	1.823	4.177	0.566	7.39	3.22
500	0.05	1.823	15.653	2.124	7.37	0.86
500	0.02	1.823	93.867	12.755	7.36	0.14
1000	0.1	25.070	18.352	0.878	20.91	28.56
1000	0.05	25.070	68.856	3.293	20.91	7.61
1000	0.02	25.070	413.291	19.614	21.07	1.28
2000	0.1	450.539	81.519	1.886	43.22	238.86
2000	0.05	450.539	306.196	7.086	43.21	63.58
2000	0.02	450.539	1835.580	42.192	43.51	10.68
3000	0.1	2911.700	191.776	3.394	56.50	857.85
3000	0.05	2911.700	724.266	13.162	55.03	221.22
3000	0.02	2911.700	4364.570	79.200	55.11	36.76
4000	0.1	11494.900	353.429	6.048	58.44	1900.69
4000	0.05	11494.900	1334.990	22.669	58.89	507.07
4000	0.02	11494.900	8045.030	136.153	59.09	84.43
5000	0.1	31548.200	566.500	9.025	62.77	3495.82
5000	0.05	31548.200	2140.310	33.889	63.16	930.93
5000	0.02	31548.200	12902.800	203.494	63.41	155.03
6000	0.1			12.071		
6000	0.05			47.881		
6000	0.02			288.768		
7000	0.1			17.499		
7000	0.05			65.850		
7000	0.02			396.318		
8000	0.1			23.156		
8000	0.05			87.383		
8000	0.02			526.697		
9000	0.1			30.054		
9000	0.05			114.435		
9000	0.02			690.574		
10000	0.1			37.070		
10000	0.05			139.800		
10000	0.02			842.920		

Tablica 3: Eksperimenti s racionalnim matricama gustoće 75% ($d = 3/4$).

n	ϵ	GLPK SIMPLEX	PC-APX CPU	PC-APX CUDA	PC-APX-CPU / PC-APX-CUDA	GLPK-SIMPLEX / PC-APX-CUDA
100	0.1	0.012	0.095	0.228	0.41	0.05
100	0.05	0.012	0.353	0.845	0.42	0.01
100	0.02	0.012	2.109	5.040	0.42	0.00
500	0.1	1.730	3.106	0.416	7.46	4.16
500	0.05	1.730	11.642	1.552	7.50	1.12
500	0.02	1.730	69.835	9.336	7.48	0.19
1000	0.1	25.134	13.755	0.652	21.10	38.55
1000	0.05	25.134	51.639	2.443	21.14	10.29
1000	0.02	25.134	310.193	14.573	21.29	1.72
2000	0.1	616.506	61.109	1.409	43.36	437.40
2000	0.05	616.506	229.490	5.277	43.49	116.83
2000	0.02	616.506	1378.690	31.689	43.51	19.45
3000	0.1	3542.230	145.133	2.659	54.58	1332.19
3000	0.05	3542.230	545.302	9.923	54.95	356.96
3000	0.02	3542.230	3277.610	59.487	55.10	59.55
4000	0.1	13676.500	267.447	4.532	59.01	3017.84
4000	0.05	13676.500	1005.910	16.984	59.23	805.26
4000	0.02	13676.500	6043.890	102.020	59.24	134.06
5000	0.1	37682.000	429.150	6.769	63.40	5567.25
5000	0.05	37682.000	1614.010	25.394	63.56	1483.89
5000	0.02	37682.000	9700.800	152.544	63.59	247.02
6000	0.1			9.595		
6000	0.05			36.032		
6000	0.02			216.330		
7000	0.1			13.286		
7000	0.05			49.660		
7000	0.02			297.456		
8000	0.1			17.385		
8000	0.05			65.490		
8000	0.02			394.680		
9000	0.1			22.967		
9000	0.05			85.875		
9000	0.02			518.036		
10000	0.1			27.470		
10000	0.05			104.840		
10000	0.02			631.798		

Tablica 4: Eksperimenti s racionalnim matricama gustoće 100% ($d = 1.0$).

n	ϵ	GLPK SIMPLEX	PC-APX CPU	PC-APX CUDA	PC-APX-CPU / PC-APX-CUDA	GLPK-SIMPLEX / PC-APX-CUDA
100	0.1	0.007	0.201	0.236	0.85	0.03
100	0.05	0.007	0.749	0.875	0.86	0.01
100	0.02	0.007	4.486	5.257	0.85	0.00
500	0.1	0.995	6.240	0.427	14.61	2.33
500	0.05	0.995	23.390	1.583	14.78	0.63
500	0.02	0.995	140.309	9.503	14.76	0.10
1000	0.1	20.543	27.637	0.659	41.94	31.17
1000	0.05	20.543	103.673	2.479	41.82	8.29
1000	0.02	20.543	622.279	14.809	42.02	1.39
2000	0.1	659.014	122.289	2.818	43.40	233.86
2000	0.05	659.014	459.327	10.588	43.38	62.24
2000	0.02	659.014	2758.470	63.411	43.50	10.39
3000	0.1	5103.360	290.687	5.296	54.89	963.63
3000	0.05	5103.360	1092.360	19.893	54.91	256.54
3000	0.02	5103.360	6563.190	119.589	54.88	42.67
4000	0.1	78224.600	535.272	9.205	58.15	8498.06
4000	0.05	78224.600	2012.530	34.587	58.19	2261.68
4000	0.02	78224.600	12090.900	207.629	58.23	376.75
5000	0.1	240766.000	859.343	13.658	62.92	17628.20
5000	0.05	240766.000	3230.270	51.084	63.23	4713.14
5000	0.02	240766.000	19430.400	307.566	63.17	782.81

Tablica 5: Eksperimenti na binarnim matricama s 25% gustoće ($d = 1/4$).

n	ϵ	GLPK SIMPLEX	PC-APX CPU	PC-APX CUDA	PC-APX-CPU / PC-APX-CUDA	GLPK-SIMPLEX / PC-APX-CUDA
100	0.1	0.01	0.10	0.337	0.29	0.02
100	0.05	0.01	0.36	0.374	0.97	0.02
100	0.02	0.01	2.17	5.163	0.42	0.00
500	0.1	1.64	3.10	0.425	7.29	3.86
500	0.05	1.64	11.63	1.587	7.33	1.03
500	0.02	1.64	69.79	9.437	7.40	0.17
1000	0.1	29.64	13.76	0.659	20.88	44.97
1000	0.05	29.64	52.196	2.493	20.94	11.89
1000	0.02	29.64	309.86	14.770	20.98	2.01
2000	0.1	728.94	61.13	1.415	43.20	515.15
2000	0.05	728.94	229.58	5.301	43.31	137.51
2000	0.02	728.94	1378.72	31.800	43.36	22.92
3000	0.1	5267.25	145.49	2.660	54.70	1980.17
3000	0.05	5267.25	546.79	9.959	54.90	528.89
3000	0.02	5267.25	3283.88	59.820	54.90	88.05
4000	0.1	84960.30	267.85	4.637	57.76	18322.61
4000	0.05	84960.30	1006.71	17.328	58.10	4903.12
4000	0.02	84960.30	6049.35	104.256	58.02	814.92
5000	0.1	252173.400	429.30	6.815	62.99	37002.70
5000	0.05	252173.400	1614.39	25.546	63.20	9871.35
5000	0.02	252173.400	9701.32	153.646	63.14	1641.26

Tablica 6: Eksperimenti na binarnim matricama s 50% gustoće ($d = 1/2$).

n	ϵ	GLPK SIMPLEX	PC-APX CPU	PC-APX CUDA	PC-APX-CPU / PC-APX-CUDA	GLPK-SIMPLEX / PC-APX-CUDA
100	0.1	0.01	0.06	0.158	0.41	0.06
100	0.05	0.01	0.24	0.582	0.41	0.02
100	0.02	0.01	1.44	3.467	0.41	0.00
500	0.1	2.13	2.07	0.283	7.33	7.52
500	0.05	2.13	7.77	1.050	7.40	2.03
500	0.02	2.13	46.63	6.282	7.42	0.34
1000	0.1	29.04	9.71	0.443	21.94	65.61
1000	0.05	29.04	34.42	1.643	20.95	17.68
1000	0.02	29.04	206.66	9.897	20.88	2.93
2000	0.1	762.29	40.75	0.947	43.03	804.97
2000	0.05	762.29	153.06	3.518	43.51	216.67
2000	0.02	762.29	919.38	21.179	43.41	35.99
3000	0.1	17482.00	96.86	1.775	54.57	9849.79
3000	0.05	17482.00	364.17	6.646	54.80	2630.56
3000	0.02	17482.00	2187.62	39.811	54.95	439.13
4000	0.1	98651.90	178.44	3.081	57.92	32021.42
4000	0.05	98651.90	670.82	11.601	57.82	8503.45
4000	0.02	98651.90	4031.42	69.337	58.14	1422.79
5000	0.1	256927.000	286.34	4.532	63.19	56694.25
5000	0.05	256927.100	1077.22	17.012	63.32	15102.60
5000	0.02	256927.100	6471.25	102.239	63.30	2513.00

Tablica 7: Eksperimenti na binarnim matricama s 75% gustoće ($d = 3/4$).

n	ϵ	d	PC-APX CUDA	n	ϵ	d	PC-APX CUDA
15000	0.10	1/2	139.451	15000	0.10	1/2	70.504
15000	0.05	1/2	526.314	15000	0.05	1/2	264.357
15000	0.10	3/4	93.086	15000	0.10	3/4	46.977
15000	0.05	3/4	350.922	15000	0.05	3/4	70.239
20000	0.10	1/2	229.166	20000	0.10	1/2	117.336
20000	0.05	1/2	863.677	20000	0.05	1/2	436.092
20000	0.10	3/4	152.745	20000	0.10	3/4	77.675
20000	0.05	3/4	577.988	20000	0.05	3/4	291.247
25000	0.10	1/2	281.679	25000	0.10	1/2	213.388
25000	0.05	1/2	1596.990	25000	0.05	1/2	803.009
25000	0.10	3/4	281.634	25000	0.10	3/4	142.444
25000	0.05	3/4	1066.630	25000	0.05	3/4	535.470
30000	0.10	1/2	599.481	30000	0.10	1/2	303.302
30000	0.05	1/2	2267.310	30000	0.05	1/2	1139.810
30000	0.10	3/4	400.400	30000	0.10	3/4	202.499
30000	0.05	3/4	1511.960	30000	0.05	3/4	7760.197
35000	0.10	1/2	879.068	35000	0.10	1/2	444.616
35000	0.05	1/2	3325.940	35000	0.05	1/2	1672.010
35000	0.10	3/4	586.140	35000	0.10	3/4	296.801
35000	0.05	3/4	2217.210	35000	0.05	3/4	1159.110

Tablica 8: Vrijeme izvršavanja za racionalne i binarne input matrice većih instanci

granica na veličinu inputa u memoriji može se malo reducirati tako da se koristi ekvivalentni linearni program s jediničnim vektorima b, c što povlači suvišnost spremanja vektora b i c za računanje i time granicu reduciramo na $n^2 \cdot$`sizeof(float)`. Vektori b, c eventualno trebaju u postprocesirajućoj fazi kako bi dobili rješenje polaznog linearnog programa. Boljoj memorijskoj iskorištenosti bi dodatno doprinjelo podaci binarne matrice spremljeni kao jedan bajt (čak i jedan bit je dovoljan). Kako su računanja obavljena u aritmetici dvostruke preciznosti, dodatni podaci za računanje zahtjevaju oko $5n \cdot$`sizeof(double)` DEVICE memorije. Ukupno, za paralelnu implementaciju potrebno je oko $n^2 \cdot$ `sizeof(float/byte)`$+ 5n \cdot$`sizeof(double)` DEVICE memorije.

U *tablici 8* dani su rezultati PC-APX-CUDA implementacije za racionalni tip matrica i binarni tip matrica do veličine 35000 za relativno guste matrice i $\epsilon \in \{0.1, 0.05\}$.

6.6 OSVRT NA IMPLEMENTACIJU LP RJEŠAVATELJA

Doprinos

U ovom poglavlju predstavljena je paralelna CUDA implementacija aproksimacijske sheme izvorno opisane u [31]. Na temelju eksperimenata utvrdila se dostatnost PRAM modela za teorijsku analizu vremenske složenosti algoritma u CUDA okruženju i prijavljeno je nekoliko reda veličine ubrzanje s obzirom na sekvencijalnu inačicu algoritma odnosno sa simpleks implementacijom iz GLPK paketa. Implementacija je pokazala izvedivost na velikim instancama problema pakiranja i pokrivanja što za GLPK nije bio slučaj.

Budući rad

Za probleme pakiranja i pokrivanja postoji randomizirani algoritam od Koufogiannakisa i Younga [60] za koje bi bilo interesantno probati efikasno paralelizirati na CUDA-i. Njihov algoritam je asimptotski brži i trenutni rezultati su vrlo obećavajući kao što je pokazano u [60]. Isto tako, Young u [85] predstavio paralelni algoritam za linearne programe problema miješanog pakiranja i pokrivanja. Napominjemo da su takvi problemi općenitiji i primjenjuje se u više realnih scenarija. Npr. nalaženje aproksimativnog rješenja sustava linearnih jednadžbi $Ax = b$, za nenegativnu matricu A i vektor b, tako da je

$Ax \geq b, Ax \leq (1+\epsilon)b$ se može promatrati kao poseban slučaj miješanog problema.

Izazov kod implementacije algoritama predstavljenih u [31] i [60] predstavlja i smanjenje broja iteracija, s time da se po iteraciji napravi više paralelnog posla.

7

IMPLEMENTACIJSKI MODEL ALGORITAMA ZA ČUVANJE 1.5D TERENA

U ovom poglavlju predstavlja se implementacijski model algoritama predstavljenih u *poglavljima 3 i 4*. Glavno težište implementacije predstavlja rješavanje formulacije linearnog programa odgovarajuće inačice problema čuvanja 1.5D terena. Linearni programi su problemi pokrivanja odnosno problemi ograničenog višestrukog pokrivanja čije se rješenje može dobiti iz već postojećih implementacija rješavatelja linearnih programa. Biblioteke koje implementiraju optimalne rješavatelje linearnih programa pokazuju relativno dobre performanse na relativno manjim instancama dok za velike instance nalaženje optimalnog rješenja zahtjeva znatno više vremena ili čak nemogućnost zbog memorijskih ograničenja. U tu svrhu, aproksimativni rješavatelji linearnih programa pokazuju dobre performanse za nalaženje aproksimativnih rješenja čak i za velike instance problema. Za predstavljene algoritme u *poglavljima 3 i 4* dopustivo rješenje vraćeno aproksimacijskom shemom može poslužiti u algoritmu umjesto optimalnog rješenja linearnog programa, što ima utjecaj na aproksimacijski faktor s dodatnim multiplikativnim članom $(1 + \epsilon)$ gdje $\epsilon > 0$ predstavlja parametar greške. Implementacija i primjena algoritama je u pripremi za publiciranje u [70].

Sadržaj

7.1 GENERIČKI ALGORITAM DISKRETNOG ČUVANJA 1.5D TERENA

Kompletan algoritam za težinsko čuvanje 1.5D terena odnosno čuvanje 1.5D terena s višestrukim pokrivanjem može se sažeti u algoritmu GENERIC-1.5D-TERRAIN-GUARD.

ALGORITAM 9 Generički algoritam za implementaciju aproksimacijskih algoritama rješavanja problema čuvanja na 1.5D terenima

procedure GENERIC-1.5D-TERRAIN-GUARD(T, G, N, w, d)
  $\mathcal{A} \leftarrow$ CALCULATE-VISIBILTY(T, G, N)
  $(A, w, d, u, m, n) \leftarrow$ LP-FORM($\mathcal{A}, w, d$)
  $(x^*, y^*) \leftarrow$ LP-SOLVE($A, w, d, u, m, n, \epsilon$)
  $X_0 \leftarrow$ FIND-GUARD-POINTS($G \cap N, x^*, \alpha$)
  $(G', N', w, d') \leftarrow$ REDUCE-INSTANCE(X_0, d)
  $(N'_L, N'_R, d'_L, d'_R) \leftarrow$ DECOMPOSE(G', N', d', x^*)
  $X_L \leftarrow$ LEFT-GUARDS(T, G', w, N'_L, d'_L)
  $X_R \leftarrow$ RIGHT-GUARDS(T, G', w, N'_R, d'_R)
  $X \leftarrow X_0 \cup X_L \cup X_R$
  return X
end procedure

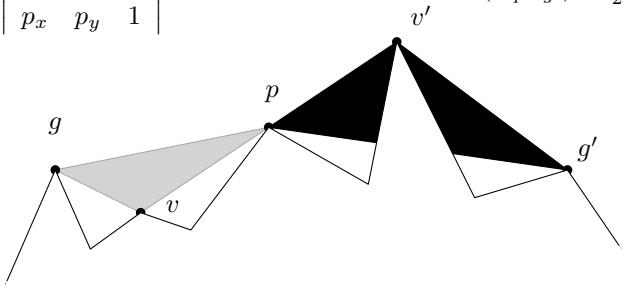
OPIS KORAKA Na temelju danog 1.5D terena T odnos između čuvara G i klijenata N na 1.5D terenu se može kodirati na sljedeći način:

$$\mathcal{A}(p,g) = \begin{cases} \texttt{-1}, & g \sim p, g < p \\ \texttt{0}, & g \not\sim p \\ \texttt{1}, & g \sim p, g > p \\ \texttt{-1/1}, & g = p \end{cases}$$

Računanje relacije vidljivosti implementira procedura CALCULATE-VISIBILTY koja za svakog klijenta $p \in N$ i za svakog čuvara $g \in G$ računa vidljivost tako da za svaki vrh v iz T koji se nalazi između g i p (ili obrnuto) utvrdi nalazi li se iznad spojnice $\overline{pg}$ ili ne. Računanje predznaka determinante koje formiraju točke p, v, g može provjeriti taj uvjet. Na *slici 26* je prikazan primjer u kojem $p \sim g$ jer svi trokuti Δ_{gvp} za vrhove v za koje $g < v < p$ imaju pozitivnu površinu, dok $p \not\sim g'$ jer postoji vrh v' takav da $p < v' < g'$ za kojeg trokut $\Delta_{pv'g'}$ ima negativnu površinu. Postupak zahtjeva za svaki par p, g

prolaženje $O(k)$ vrhova što ukupno traje $O(mnk)$ vremena. Računanje determinante je $O(1)$ vrijeme.

$$P(\Delta_{gvp}) = \frac{1}{2}\begin{vmatrix} g_x & g_y & 1 \\ v_x & v_y & 1 \\ p_x & p_y & 1 \end{vmatrix} > 0 \qquad P(\Delta_{pv'g'}) = \frac{1}{2}\begin{vmatrix} p_x & p_y & 1 \\ v'_x & v'_y & 1 \\ g'_x & g'_y & 1 \end{vmatrix} < 0$$



Slika 26: Računanje vidljivosti između točaka 1.5D terena preko determinante

Korak 3 definira pripadni linearni program

$$\min_{x \in \mathbb{R}^n} \{w^T x \colon Ax \leq d, x \leq u, x \geq 0\} \tag{35}$$

za kojeg se matrica uvjeta A konstruira preko $\mathcal{A}$ u $O(mn)$ vremenu kao $A(p,g) = \varphi(\mathcal{A}(p,g))$ gdje je φ preslikavanje dano s

$$\varphi(x) = \begin{cases} 1, & x = \texttt{1}, \texttt{-1}, \texttt{-1/1} \\ 0, & x = \texttt{0} \end{cases}$$

LP-FORM dodatno kao parametar uzima vektor težina čuvara w veličine n, zahtjev klijenata d veličine m te vektor ograničenja na broj dopuštenih kopija čuvara u veličine n.

Ukoliko je riječ o težinskom problemu čuvanja, uzima se $d = \mathbf{1}$ i zanemari uvjet $x \leq u$ (skr. $u = \text{NIL}$). U inačici problema čuvanja 1.5D terena s višestrukim pokrivanjem, uzima se $w = \mathbf{1}$ i $u = \mathbf{1}$. Slučaj kad $w = \mathbf{1}$ i $d = \mathbf{1}$ se kao pretpostavljeno smatra problem čuvanja 1.5D terena s višestrukim pokrivanjem.

U *koraku 4* implementira se rješavatelj linearnog programa za (35). Ukoliko je parametar greške $\epsilon = 0$ rutina LP-SOLVE nalazi optimalno rješenje, odnosno za $\epsilon > 0$ nalazi se $1 + \epsilon$ aproksimativno rješenje.

Slučaj kada čuvari mogu biti i klijenti rješava se procedurom FIND-GUARD-POINTS u *koraku 6* koje na temelju rješenja x^* i parametra α određuje skup čuvara-klijenata. Na temelju tog skupa u *koraku 6* polazna instanca problema se reducira na instancu kad čuvari ne mogu biti klijenti.

Procedura DECOMPOSE u *koraku 7* dijeli polazni problem na probleme lijevog i desnog čuvanja u ovisnosti je li riječ ili o težinskom problemu ili o problemu s višestrukim pokrivanjem.

U *koracima 8 i 9* nalaze se optimalna cjelobrojna rješenja za odgovarajuće probleme lijevog i desnog čuvanja 1.5D terena. Konačno rješenje definirano je u *koraku 10*.

Specifični detalji implementacije pojedinih koraka algoritma razrađuju se u nastavku.

7.2 IMPLEMENTACIJSKI MODEL TEŽINSKOG POKRIVANJA 1.5D TERENA

7.2.1 *Ulazi za algoritam težinskog pokrivanja*

Ulaz problemu predstavlja instanca 1.5D terena T s k vrhova, točke terena G veličine n na kojima se nalaze čuvari i točke terena N veličine m koji su klijenti. Težine čuvara G predstavlja polje racionalnih vrijednosti $w(i) > 0, i = 1, 2, \ldots, n$. Zahtjevi definirani za klijente su definirani poljem d. Iz definicije težinskog problema, postavlja se da je $d = \mathbf{1}$, dakle, vektor svih jedinica. Memorijsko zauzeće ulaza je $O(m + n + k)$. Kako je težinski problem minimizacija i $d = \mathbf{1}$ uvjet $u = \mathbf{1}$ je suvišan, stoga se uzima oznaka $u = \text{NIL}$.

Rutinom LP-FORM formira se linearni program $(A, w, \mathbf{1}, \text{NIL}, m, n)$ problema pakiranja i pokrivanja oblika LP7. Alternativno, ulaz algoritmu može biti LP formulacija koja odgovara instanci problema DISCRETE-WEIGHTED-1.5D-TERRAIN-GUARD. U tom slučaju, LP-FORM provjerava u $O(mn)$ vremenu dopustivost tog LP-a, dakle, moraju vrijediti uvjeti $A\,\mathbf{1} \geq \mathbf{1}$ i za svaki $j = 1, 2, \ldots, n$ vrijedi $\sum_{i=1}^{m} A(i, j)y(i) = 0$ ako i samo ako $y(i) = 0$. Drugim riječima, svakog klijenta vidi barem jedan čuvar i svaki čuvar vidi barem jednog klijenta. Čuvar $g \in G$ za koje je $w_g = 0$ isključuje se iz instance problema.

7.2.2 *Rješavanje LP problema pakiranja i pokrivanja*

Ključni implementacijski izazov predstavlja brza implementacija LP rješavatelja za problem pakiranja i pokrivanja LP7 u proceduri LP-SOLVE. Na implementacijskoj razini dopušta se $1 + \epsilon$ aproksimativno rješenje LP-a koristeći ideje iz *Poglavlja 6* i provedena je analiza koliko

parametar utječe u konačan rezultat. Eksperimentalna usporedba s optimalnim rješavateljem dana je u *odjeljku 7.2.6*.

Procedura LP-SOLVE$(A, w, \mathbf{1}, \text{NIL}, m, n, 0)$ predstavlja implementaciju simpleks metode za traženje frakcionalnog optimalnog rješenja problema pakiranja i pokrivanja. Pokazano je da bez obzira koja se strategija pivotiranja odabire u simpleks metodi [55] uvijek postoji familija poliedara na kojoj simpleks metoda napravi eksponencijalno mnogo koraka. Prosječno vrijeme trajanja simpleks metode za LP probleme čije matrice uvjeta dolaze iz određenih vjerovatnosnih distribucija je pokazano $O(m^2 n)$ u [20]. Alternativno, za $\epsilon > 0$ procedura LP-SOLVE implementira aproksimacijsku shemu koja vrati primal-dual dopustiv par (x', y') koje je $1 + \epsilon$ aproksimacija u vremenu $O(\epsilon^{-2} n^2 m \log n)$ na RAM modelu računala. Kako ulaz problema može imati velike instance, ponajprije zbog definiranja ulaza preko diskretizacijskog postupka, procedura dodatno implementira paralelnu inačicu aproksimacijske sheme koja radi u $O(\epsilon^{-2} nm \log n \log m)$ vremenu na PRAM modelu računala koja modelira CUDA platformu.

7.2.3 *Reduciranje težinske instance*

Uz pretpostavku $G \cap N \neq \emptyset$ procedura FIND-GUARD-POINTS utvrđuje skup čuvara-klijenata X_0 koji imaju značajnu frakcionalnu vrijednost u primal rješenju x^* LP formulacije. Parametar je $\alpha = 1/5$. Instanca problema se reducira procedurom REDUCE-INSTANCE tako da se odstrane svi klijenti iz N koji su viđeni čuvarima od X_0 i ti čuvari-klijenti iz G, korak koji je implementiran u proceduri REDUCE-INSTANCE, a može se implementirati poništavanjem vrijednosti u A koja opisuje relaciju vidljivosti na T između G i N. Korak ukupno zahtjeva $O(mn)$ vremena. Ovaj korak algoritma osigurava da je skup preostalih čuvara G' i preostalih klijenata N' disjunktan.

7.2.4 *Rješavanje jednostranih problema težinskog čuvanja 1.5D terena*

Procedura DECOMPOSE$(G', N', \text{NIL}, x^*)$ vraća par (N'_L, N'_R) klijenata koji trebaju biti čuvani s lijeva odnosno s desna s čuvarima iz G'. Prema definiciji od N_L i N_R korak se može implementirati u $O(mn)$ vremenu. Procedure LEFT-GUARDS i RIGHT-GUARDS su implementacije kombinatornih algoritama koja vraćaju skup lijevih čuvara $X_L =$ WEIGHTED-LEFT-GUARDING(T, N'_L, G', w) odnosno skup desnih čuvara

$X_R = \text{WEIGHTED-RIGHT-GUARDING}(T, N'_R, G', w)$ gdje $d'_L = d'_R = \text{NIL}$ i rade u $O(mn)$ vremenu.

Konačno rješenje za WEIGHTED-1.5D-TERRAIN-GUARD predstavlja rješenje X koja je unija rješenja za čuvare-klijente i čuvare za lijevi odnosno desni problem. Operacija unije rješenja je izvediva u $O(n)$ vremenu.

7.2.5 *Aproksimabilnost težinskog čuvanja 1.5D terena na temelju implementacijskog modela algoritma*

U ovom dijelu istražit će se koliko parametar $\epsilon > 0$ u proceduri LP-SOLVE utječe na aproksimabilnost problema težinskog čuvanja 1.5D terena. Neka je (x', y') par primal-dual rješenja vraćeni LP-SOLVE procedurom koje vraća $1 + \epsilon$ aproksimaciju. Optimalno rješenje primala i duala predstavlja par (x^*, y^*).

Iz *Teorema 1.2* i *Teorema 1.3* dobiva se sljedeći niz nejednakosti:

$$\sum_{p \in N} y'_p \leq \sum_{p \in N} y^*_p = \sum_{g \in G} w_g x^*_g \leq \sum_{g \in G} w_g x'_g \leq (1+\epsilon) \sum_{p \in N} y'_p \leq (1+\epsilon) \sum_{p \in N} y^*_p \tag{36}$$

THEOREM 7.1. *Za problem* DISCRETE-WEIGHTED-1.5D-TERRAIN-GUARD *s m klijenata i n čuvara postoji algoritam koji vraća* $5(1+\epsilon)$ *aproksimaciju ukupnom vremenu* $O(mnk + \epsilon^{-2} n^2 m \log n)$ *na RAM stroju odnosno u vremenu* $O(mnk + \epsilon^{-2} mn \log n \log m)$ *ukoliko se odgovarajući linearni program računa na* PRAM *stroju gdje je* $\epsilon > 0$ *parametar greške.*

Dokaz. Uvedimo parametar greške $\epsilon > 0$ u analizu algoritma za jednostrano i obostrano čuvanje 1.5D terena. Pokažimo prvo da jednostrana inačica čuvanja 1.5D terena s čuvarima (G_L, G_R) i klijentima N ima $2(1+\epsilon)$ aproksimaciju. Neka je (x', y') primal-dual dopustiv par u $1+\epsilon$ aproksimaciji za LP5 gdje je $G_L = G_R = G$, te N_L i N_R definirani kao

$$N_L = \left\{ p \in N \mid \textstyle\sum_{g \in \mathcal{V}_L(p) \cap G} x'_g \geq \frac{1}{2} \right\},$$

$$N_R = \left\{ p \in N \mid \textstyle\sum_{g \in \mathcal{V}_R(p) \cap G} x'_g \geq \frac{1}{2} \right\}.$$

Iz dopustivosti primal-dual para vrijedi

$$\begin{aligned} w(X_L^*) &\leq 2\sum_{g\in G_L} w_g x'_g \\ &\leq 2(1+\epsilon)\sum_{p\in N_L} y'_p \\ &\leq 2(1+\epsilon)\sum_{p\in N_L} y^*_p \end{aligned}$$

Analogno,

$$w(X_R^*) \leq 2(1+\epsilon)\sum_{p\in N_R} y^*_p$$

Ukupno rješenje nije proizvoljno daleko od optimalnog rješenja:

$$\begin{aligned} w(X_L^*) + w(X_R^*) &\leq 2(1+\epsilon)(\textstyle\sum_{p\in N_L} y^*_p + \sum_{p\in N_R} y^*_p) \\ &\leq 2(1+\epsilon)\text{OPT} \end{aligned}$$

gdje OPT predstavlja optimalnu vrijednost za problem jednostranog čuvanja 1.5D terena.

Neka je (x', y') sada dopustivo primal-dual rješenje za LP7 kao $1+\epsilon$ aproksimacija i OPT predstavlja optimalnu vrijednost za Discrete-Weighted-1.5D-Terrain-Guard kad $G \cap N \neq \emptyset$. Nadalje, neka je X_0^* definiran analogno kao u (13) gdje umjesto optimalne vrijednosti x^* koristi se dopustivo rješenje x' iz $1+\epsilon$ aproksimacije. Neka je N' skup klijenata koji nije pokriven od x_0^*.

Ukoliko se uzme $5/4x'$ ono je dopustivo rješenje za LP5 prema (15). Problemi 1.5D-Terrain-Left-Guard i 1.5D-Terrain-Right-Guard vraćaju rješenja (X_L^*, X_R^*) za čuvanje N' čija cijena je ograničena odozgo s $5/2\sum_{g\in G'} w_g x'_g$.

Ograničenje na konačno rješenje je u ovisnosti o parametru greške $\epsilon > 0$:

$$\begin{aligned} w(X_0^*) + w(X_L^*) + w(X_R^*) &\leq 5\textstyle\sum_{g\in X_0^*} w_g x'_g + 5\sum_{g\in G'} w_g x'_g \\ &\leq 5(\textstyle\sum_{g\in X_0^*} w_g x'_g + \sum_{g\in G'} w_g x'_g) \\ &\leq 5(1+\epsilon)\textstyle\sum_{p\in N} y'_p \\ &\leq 5(1+\epsilon)\textstyle\sum_{p\in N} y^*_p \\ &\leq 5(1+\epsilon)\text{OPT} \end{aligned}$$

Ukupna vremenska složenost algoritma predstavlja vrijeme potrebno za sve korake iz algoritma specijaliziranog za težinsku inačicu. Slijedni koraci algoritma su izvedivi u $O(mn)$ vremenu osim računanja matrice vidljivosti koja zahtjeva $O(mnk)$ vremena. Vremenski najzahtjevniji dio algoritma čini LP-Solve procedura koja implementira

optimalni rješavatelj simpleks metode (GLPK simpleks implementacija) odnosno aproksimacijsku shemu. Dakle, ukupno vrijeme algoritma je asimptotski $O(mnk + \epsilon^{-2}n^2 m \log n)$ na RAM stroju odnosno $O(mnk + \epsilon^{-2}mn \log n \log m)$ na PRAM stroju. Memorijsko zauzeće algoritma je reda veličine $O(mn)$.

□

7.2.6 *Eksperimenti za* LP-Solve *proceduru*

U eksperimentalnom dijelu ispituju se performanse implementacijskog modela algoritma s naglaskom na LP-Solve proceduru. Uzorak testiranja predstavljaju kvadratne matrice koje odgovaraju relaciji vidljivosti problema čuvanja 1.5D terena.

PLATFORMA Mjerenja su uzeta na platformi koja se sastoji od četverojezgrenog Intel procesora *i5* s radnim taktom 2.8 MHz i 8 GB RAM memorije zajedno s grafičkom karticom TESLA C2070 s 6 GB DDR5 RAM memorije. Grafička kartica NVIDIA C2070 sadrži 448 masivno nitiranih procesorskih jezgri s radnim taktom 1.15 GHz koje može pokretati reda veličine 30000 niti izvršenja i visoka memorijska propusnost od 144 GB/s.

TEST ULAZI Instanca težinskog čuvanja 1.5D terena predstavljena je relacijom vidljivosti opisana matricom $\mathbf{A}_{n,h}$. Radi jednostavnosti, uzima se $w = \mathbf{1}$. Matrica uvjeta $\mathcal{A}_{n,h} \in \{-1, 0, 1\}^{n \times n}$ je generički definirana matrica čija sporedna dijagonalna vrpca je širine h i oblika je

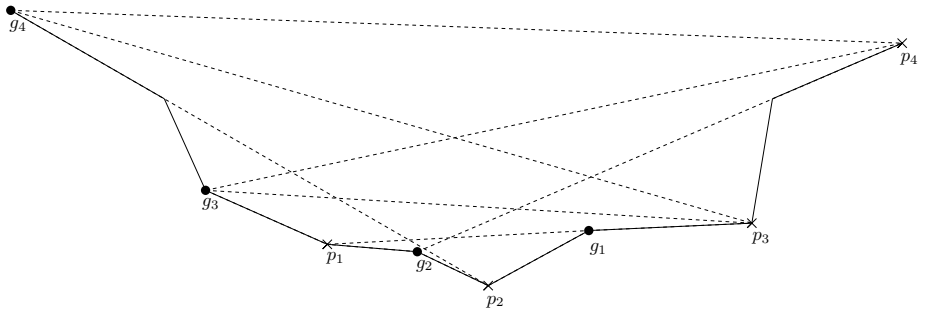
$$\mathcal{A}_{n,h} = \left[\begin{array}{cccccc} 1 & 1 & \dots & 1 & -1 & 0 \\ 1 & 1 & \dots & -1 & 0 & \vdots \\ 1 & 1 & & 0 & \vdots & 0 \\ 1 & -1 & \iddots & \vdots & 0 & -1 \\ -1 & 0 & \iddots & 0 & -1 & \vdots \\ 0 & 0 & \iddots & -1 & -1 & -1 \end{array}\right] \Big\} h$$

gdje retci poredani odozgo prema dolje odgovaraju klijentima poredanih na 1.5D terenu slijeva na desno odnosno stupci poredani slijeva na desno odgovaraju čuvarima poredani sdesna na lijevo.

Primjerice, za $n = 4$ i $h = 1$ matrica $\mathbf{A}_{4,1}$ predstavlja relaciju vidljivosti na terenu koji je implementiran na *slici 27*.

$$\mathcal{A}_{4,1} = \begin{bmatrix} 1 & 1 & -1 & 0 \\ 1 & -1 & 0 & -1 \\ -1 & 0 & -1 & -1 \\ 0 & -1 & -1 & -1 \end{bmatrix}$$

(a) Relacija jednostrane vidljivosti prikazana matricom $A_{4,1}$



(b) Isprekidana dužina predstavlja vidljivost između 2 točke

Slika 27: Teren s relacijom vidljivosti $A_{4,1}$.

Ulaz proceduri LP-SOLVE predstavlja šestorka $(A_{h,1}, \mathbf{1}, \mathbf{1}, \text{NIL}, n, n, \epsilon)$ gdje $A_{h,1}(i,j) = |\mathcal{A}_{n,1}(i,j)|$. Usporedba LP rješavatelja za LP-SOLVE proceduru dano je na *slici 28*. Veličina n predstavlja broj redaka i stupaca matrice $A_{n,1}$, GLPK SIMPLEX je vrijeme izvršavanja GLPK simpleks rutine za nalaženje optimalnog LP rješenja zajedno s brojem iteracija GLPK ITER, vrijeme izvršavanja sekvencijalne inačice aproksimacijske sheme je označeno s PC-APX CPU odnosno paralelna CUDA inačica s PC-APX CUDA. Broj iteracija aproksimacijske sheme je označeno PC-APX ITER. Parametar D/P predstavlja stvarnu aproksimacijsku grešku LP rješenja kao omjer dualne i primalne vrijednosti. Relativno ubrzanje između sekvencijalne inačice i paralelne CUDA inačice aproksimacijske sheme opisano je s $\frac{\text{PC-APX-CPU}}{\text{PC-APX-CUDA}}$ odnosno GLPK rješavatelja i CUDA implementacije s $\frac{\text{GLPK}}{\text{PC-APX-CUDA}}$.

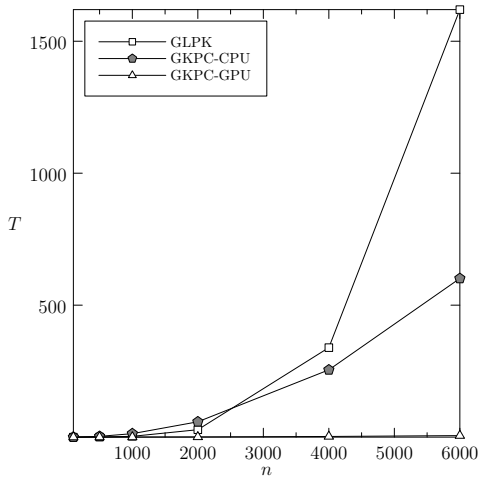
Tablica (a) prikazuje rezultate za $\epsilon = 0.1$ odnosno tablica (b) za $\epsilon = 0.05$. Oznaka N/A predstavlja nemogućnost izvođenja eksperimenta zbog ograničenosti memorije test platforme. Vidljivo je iz eksperimenata prednost GLPK rješavatelja na manjim instancama. Vremenska dobit od aproksimacijske sheme uočava se za razumno malen ϵ (grafički prikaz vremenske usporedbe je dano na slici (c) za $\epsilon = 0.1$

n	GLPK simplex	GLPK-simplex iter	PC-APX CPU	PC-APX CUDA	D/P	PC-APX iter	PC-APX CPU / PC-APX CUDA	GLPK simplex / PC-APX CUDA
100	0.0079	100	0.0926	0.0929	1.0663	1190	0.9966	0.0852
500	0.3969	500	2.9695	0.1819	1.0656	1588	16.3233	2.1817
1000	3.0434	1000	13.1324	0.3211	1.0660	1761	40.8929	9.4768
2000	28.1329	2000	58.1088	0.7762	1.0656	1936	74.8669	36.2462
4000	339.0830	4000	254.9180	2.4991	1.0659	2110	102.0035	135.6815
6000	1619.8100	6000	601.1990	5.7853	1.0659	2210	103.9187	279.9882
8000	N/A	N/A	1103.6612	10.1480	1.0657	2,285	108.7564	N/A
10000	N/A	N/A	1776.3554	15.7405	1.0659	2341	112.8522	N/A

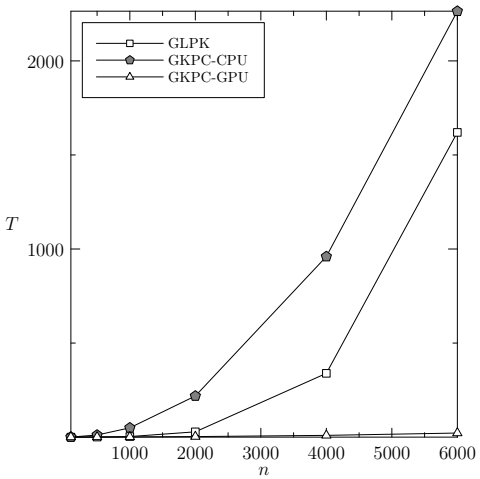
(a) Ekperimenti za $\epsilon = 0.1$

n	GLPK simplex	GLPK-simplex iter	PC-APX CPU	PC-APX CUDA	D/P	PC-APX iter	PC-APX CPU / PC-APX CUDA	GLPK simplex / PC-APX GPU
100	0.0079	100	0.3446	0.3358	1.0333	4473	1.0261	0.0236
500	0.3969	500	11.1625	0.6730	1.0332	5977	16.5859	0.5897
1000	3.0434	1000	49.4337	1.1912	1.0331	6634	41.4984	2.5548
2000	28.1329	2000	218.6430	2.9032	1.0331	7293	75.3116	9.6904
4000	339.0830	4000	959.8410	9.3651	1.0331	7953	102.4918	36.2073
6000	1619.8100	6000	2264.2600	21.7166	1.0331	8340	104.2640	74.5886
8000	N/A	N/A	4156.4800	38.0884	1.0331	8614	109.1272	N/A
10000	N/A	N/A	6652.6201	59.1534	1.0331	8827	112.4639	N/A

(b) Eksperimenti za $\epsilon = 0.05$



(c) Usporedba vremena za $\epsilon = 0.1$



(d) Usporedba vremena za $\epsilon = 0.05$

Slika 28: Eksperimenti na instanci 1.5D terena s matricom uvjeta $\mathbf{A}_{n,1}$

odnosno na slici (d) za $\epsilon = 0.05$). CUDA implementacija bilježi oko 100 puta ubrzanje u uspredbi sa sekvencijalnom implementacijom aproksimacijske sheme i ovisno o odabiru ϵ oko 200 puta za velike instance u usporedbi s GLPK simpleks metodom. Opravdanost tome leži u gustoj popunjenosti matrice uvjeta što zamjetno utječe na performanse CUDA paralelne implementacije. Velika popunjenost matrice uvjeta može se opravdati ukoliko je ulaz definiran diskretizacijskim postupkom čitavog 1.5D terena koji inducira dodatnih $O(k^2)$ točaka u složenost sustava. Sa smanjenjem ϵ evidentna je kvadratna ovisnost o broju iteracija aproksimacijske sheme. Aproksimacijska shema je pokazala manju memorijsku iskorištenost u usporedbi s GLPK rješavateljem što je pogodno za rješavanje velikih instanci 1.5D terena.

7.3 IMPLEMENTACIJSKI MODEL VIŠESTRUKOG POKRIVANJA 1.5D TERENA

7.3.1 *Ulazi za algoritam višestrukog pokrivanja 1.5D terena*

Kao i u prethodnom, ulaz problemu predstavlja instanca 1.5D terena T s k vrhova, točke terena G veličine n na kojima se nalaze čuvari i točke terena N veličine m koji su klijenti. Zahtjevi klijenata su definirani poljem nenegativnih cijelih brojeva d. Memorijsko zauzeće ulaza je $O(m + n + k)$.

Računanje jednostrane vidljivosti između čuvara G i klijenata N na terenu T i definiranju klijenata-čuvara $G \cap N$ izvodi se u $O(mnk)$ vremenu i sprema u matricu $\mathcal{A}$ procedurom CALCULATE-VISIBILTY. Rutina LP-FORM formira linearni program ograničenog višestrukog pokrivanja $(A, \mathbf{1}, d, \mathbf{1}, m, n)$ temeljen na LP8. Alternativno, ulaz algoritmu može biti i LP formulacija instance problema 1.5D-TERRAIN-MULTI-GUARD za koju LP-FORM provjeri u $O(mn)$ vremenu je li dopustiva, dakle, $A\,\mathbf{1} \geq d$.

7.3.2 *Rješavanje LP ograničenog problema višestrukog pokrivanja*

Procedura LP-SOLVE$(A, \mathbf{1}, d, \mathbf{1}, m, n, 0)$ implementira simpleks metodu za LP problem ograničenog višestrukog pokrivanja LP8 iz već postojeće biblioteke GLPK. Vektor $u = \mathbf{1}$ osigurava da najviše jedna kopija čuvara bude izabrana u rješenje.

Alternativno tome, aproksimacijska shema za probleme ograničenog višestrukog pokrivanja opisano u Fleischer [29] vraća $1 + \epsilon$ aprok-

simaciju i predstavlja aproksimativni rješavatelj za posebne vrste problema miješanog pakiranja i pokrivanja. Algoritam prati analogiju dizajna aproksimacijske sheme za probleme pakiranja i pokrivanja predstavljenih u [31]. Radi kompletnosti iskazat ćemo algoritam.

7.3.2.1 *Aproksimacijska shema za problem ograničenog višestrukog pokrivanja*

Fleischer [29] promatra sljedeću primal i dual formulaciju za problem pokrivanja s cjelobrojnim ograničenjem na varijablu pokrivanja:

$$\begin{aligned} P(x) &= \min_{x \in \mathbb{R}^n} \{c^T x \colon Ax \leq b, x \leq u, x \geq 0\} \\ D(y,z) &= \max_{y \in \mathbb{R}^m} \{b^T y - u^T z \colon A^T y - z \geq c, y \geq 0, z \geq 0\} \end{aligned} \tag{37}$$

gdje je matrica uvjeta nenegativna realna matrica veličine $m \times n$

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,n} \\ a_{2,1} & a_{2,2} & \dots & a_{2,n} \\ \vdots & \vdots & \dots & \vdots \\ a_{m,1} & a_{m,2} & \dots & a_{m,n} \end{bmatrix},$$

nenagativni cjelobrojni vektori cijene i zahtjeva

$$c = \begin{bmatrix} c_1 & c_2 & \dots & c_n \end{bmatrix}^T, b = \begin{bmatrix} b_1 & b_2 & \dots & b_n \end{bmatrix}^T,$$

te gornje pozitivno cjelobrojno ograničenje na primalnu varijablu odnosno vektor korekcije

$$u = \begin{bmatrix} u_1 & u_2 & \dots & u_n \end{bmatrix}^T, z = \begin{bmatrix} z_1 & z_2 & \dots & z_m \end{bmatrix}^T.$$

Algoritam 10 predstavlja aproksimacijsku shemu koja nalazi primal-dual rješenje $x^*, (y^*, z^*)$ tako da je $P(x^*) \leq (1+\epsilon) D(y^*, z^*)$:

Teorem 7.2 ([29]). *Algoritam 10 vraća $1+\epsilon$ aproksimaciju za (37) vremenske složenosti $O(\epsilon^{-2} n^2 m \log(c^T u))$.*

Za razliku od aproksimacijske sheme za probleme pakiranja i pokrivanja opisane u *poglavlju 6* algoritam iterira u tzv. α-fazama, u svakoj iteraciji, algoritam ažurira se primalno rješenje x tako da je $x(j) \leq \alpha$ osiguravajući tako da skaliranjem s α uvjet $x/\alpha \leq u$ je zadovoljen.

ALGORITAM 10 Aproksimacijska shema za ograničeno višestruko pokrivanje

procedure MULTI-COVER-APX(A, b, c, ϵ)
  INPUT: $A \in \mathbb{R}_+^{m\times n}, b \in \mathbb{Z}_+^m, c \in \mathbb{R}_+^n, u \in \mathbb{Z}_+^n, \epsilon > 0$
  OUTPUT: $(x^*, (y^*, z^*))$, $P(x^*)/D(y^*, z^*) \leq 1 + \epsilon$

  $\delta \leftarrow (1+\epsilon)((1+\epsilon)c^T u)^{-1/\epsilon}$
  $x(j) \leftarrow u(j)\delta, \quad j = 1, 2, \ldots, n$, $x^* \leftarrow x, \quad (y, z) = (\mathbf{0}, \mathbf{0})$
  $L_x(i) := \sum_j A(i,j)x(j)/b(i)$
  $\alpha^* \leftarrow \min_i L_x(i), \quad p \leftarrow \arg\min_i L_x(i)$
  while $c^T x < 1$ **do**
    $\alpha \leftarrow (1+\epsilon)L_x(p)$
    while $L_x(p) < \alpha$ and $c^T x < 1$ **do**
      $Q(p) = \{1 \leq j \leq n \colon x(j) < u(j)\alpha\}$
      $\eta \leftarrow \min_{j\in Q(p)} \frac{c(j)}{A(p,j)} \min\{1, \frac{u(j)\alpha - x(j)}{\epsilon x(j)}\}$
      $y(p) \leftarrow y(p) + \eta$
      $x(j) \leftarrow x(j)(1 + \epsilon\frac{\eta A(p,j)}{c(j)}), \quad j \in Q(p)$
      $z(j) \leftarrow z(j) + \eta A(p,j), \quad j \notin Q(p)$
      $p \leftarrow \arg\min_i L_x(i)$
    end while
    if $c^T x/\alpha < c^T x^*/\alpha^*$ **then**
      $x^* \leftarrow x, \alpha^* \leftarrow \alpha$
    end if
  end while
  $x^* \leftarrow x^*/\alpha^*, (y^*, z^*) \leftarrow \frac{\epsilon}{\ln(\frac{1+\epsilon}{\delta})}(y, z)$
  return $(x^*, (y^*, z^*))$
end procedure

Algoritam inicijalno započinje s $x(j) \leq u(j)\alpha/(1+\epsilon), j = 1,2,\ldots,n$ i $L_x(i) \geq \frac{\alpha}{1+\epsilon}, 1 \leq i \leq m$. Za sve one p za koje je $L_x(p) < \alpha$ algoritam uvećava vrijednost $L_x(p)$ održavajući $x(j) \leq \alpha$ sve dok $L_x(i) \geq \alpha$, $i = 1,2,\ldots m$. Primalno rješenje x se ažurira za one vrijednosti za koje $A(p,j) > 0$ i $x(j) < u(j)\alpha$ tako da barem jedna od varijabli $x(j)$ ili uveća za $1+\epsilon$ ili dostigne gornju granicu $\alpha u(j)$. Svako povećanje primalnih varijabli je uravnoteženo sa ažuriranjima dualnih varijabli tako da vrijednosti primala i duala mogu biti usporedivi. Sljedeća faza započinje s $\alpha := (1+\epsilon)\alpha$. Algoritam se zaustavlja kad $c^T x \geq 1$ gdje se primalno i dualno rješenje skalira s najviše narušenim uvjetom primala odnosno duala, kako bi se dobilo dopustivo rješenje. Vrijednosti skaliranja određuju $1+\epsilon$ aproksimaciju. Kompletan dokaz teorema sa analizom vremenske složenosti može se naći u [29]. Algoritam dopušta da matrica A bude implicitno dana preko posebne rutine *oracle* koja u polinomnom vremenu za dano primalno rješenje x i parametar α vraća i za koje je $L_x(i) < \alpha$ ili potvrđuje da ne postoji takav uvjet.

7.3.3 *Reduciranje instance s višestrukim pokrivanjem*

Uz pretpostavku $G \cap N \neq \emptyset$ procedura FIND-GUARD-POINTS nalazi skup klijenata-čuvara X_0 koji imaju značajnu frakcionalnu vrijednost u primal rješenju x^* LP formulacije za $\alpha = \frac{2}{5}\frac{d_{\min}}{d_{\min}+1}$.

Na temelju zaokruženih čuvara-klijenata X_0 REDUCE-INSTANCE procedura izračuna rezidualne zahtjeve d' u vremenu $O(mn)$. Instanca problema se zatim reducira u $O(mn)$ vremenu tako da se ažuriraju vrijednosti na relacijama vidljivosti s obzirom na X_0. Preostali čuvari G' i preostali klijenti N' zajedno s reduciranim zahtjevima d' definira instancu problema u kojem je $G' \cap N' = \emptyset$.

7.3.4 *Rješavanje jednostranog višestrukog pokrivanja*

Procedura DECOMPOSE(G', N', d', x^*) za svakog klijenta $p \in N'$ definira količinu zahtjeva klijenta $(d'_{p,L}, d'_{p,R})$ koji treba biti zadovoljen s lijeva odnosno s desna. Korak se može implementirati u $O(mn)$ vremenu. Poziv procedura LEFT-GUARDS za ulaz $(T, G', \mathbf{1}, N', d'_L)$ odnosno RIGHT-GUARDS za ulaz $(T, G', \mathbf{1}, N', d'_R)$ su implementacije kombinatornih algoritama za lijevo i desno višestruko pokrivanje 1.5D terena. Točnije, $X_L =$ LEFT-MULTI-GUARDING(T, G', N', d'_L) odnosno $X_R =$ RIGHT-MULTI-GUARDING(T, G', N', d'_R) su izračunati skupovi ču-

vara za problem lijevog čuvanja odnosno desnog čuvanja 1.5D terena s višestrukim pokrivanjem. Obje procedure računaju X_L, X_R u $O(mn)$ vremenu.

7.3.5 *Aproksimabilnost problema čuvanja 1.5D terena s višestrukim pokrivanjem na temelju implementacijskog modela algoritma*

Ovdje će se utvrditi aproksimabilnost problema čuvanja 1.5D terena s višestrukim pokrivanjem ukoliko je $\epsilon > 0$ za proceduru LP-Solve. Optimalno rješenje primala i duala predstavlja par (x^*, y^*). Neka je (x', y') par primal-dual rješenja vraćeni LP-Solve procedurom koje daje $1+\epsilon$ aproksimaciju za LP8.

Analogno nejednakosti (36) dobiva se sljedeći odnos između primala i duala za (37):

$$\sum_{p\in N} d_p y_p^* - \sum_{g\in G} z_g^* = \sum_{g\in G} x_g^* \leq \sum_{g\in G} x_g' \leq (1+\epsilon) \sum_{p\in N} (d_p y_p^* - \sum_{g\in G} z_g^*)$$

Neka je skup čuvara-klijenta X_0^* definiran preko primalnog dopustivog rješenja x'. Nadalje, pretpostavimo da je u definiciji $d'_{p,L}$ i $d'_{p,R}$ korišteno primalno dopustivo rješenje x'. Vrijedi

$$\begin{array}{rcl}
\sum_{g\in\mathcal{V}'_L(p)} x_{g,L} + \sum_{g\in\mathcal{V}'_R(p)} x_{g,R} & \geq & d_{p,L} + d_{p,R} \\
& = & \left\lfloor \frac{d_{\min}+1}{d_{\min}} \left(\sum_{g\in\mathcal{V}'_L(p)} x'_g + \frac{1}{2} x'_p \right) \right\rfloor \\
& & + \\
& & \left\lfloor \frac{d_{\min}+1}{d_{\min}} \left(\sum_{g\in\mathcal{V}'_R(p)} x'_g + \frac{1}{2} x'_p \right) \right\rfloor \\
& \geq & \lfloor d'_p + \frac{d'_p}{d_{\min}} \rfloor - 1 \\
& \geq & d'_p + 1 - 1 \\
& = & d'_p
\end{array}$$

po konstrukciji $d_{p,L}$ i $d_{p,R}$ za bilo koje cjelobrojno dopustivo rješenje x_L i x_R za LP8. Dakle, $d_{p,L} + d_{p,R} + |\mathcal{V}(p) \cap X_0^*| \geq d_p$ što znači da je $X_0^* \cup X_L^* \cup X_R^*$ i dalje dopustiv skup čuvara za problem.

Sljedeći analognu analizu iz dokaza *Teorema 4.1*, X_L^*, X_R^* se može omeđiti u ovisnosti o x':

$$|X_L^*| \leq (1+\beta) \sum_{g\in G'} x'_g \quad , \quad |X_R^*| \leq (1+\beta) \sum_{g\in G'} x'_g$$

za

$$\beta \geq \frac{\alpha(d_{\min}+1)^2}{2d_{\min}^2 - d_{\min}(d_{\min}+1)\alpha} + \frac{1}{d_{\min}}, \quad \beta \leq \frac{1}{\alpha} - 1$$

Ograničenje na konačno rješenje sadrži parametar pogreške ϵ:

$$\begin{aligned} |X_0^*| + |X_L^*| + |X_R^*| &\leq \frac{1}{\alpha}\sum_{g \in X_0^*} x'_g + 2\cdot(1+\beta)\sum_{g\in G'} x'_g \\ &\leq \max\{\frac{1}{\alpha}, 2\cdot(1+\beta)\}(1+\epsilon)(\sum_{p\in N'} d_p y'_p - \sum_{g\in G'} z'_g) \\ &\leq \max\{\frac{1}{\alpha}, 2\cdot(1+\beta)\}(1+\epsilon)\mathrm{OPT} \end{aligned}$$

Kako β ne ovisi o x', slijedeći analizu za γ iz *Teorem 4.1*, zaključujemo da je aproksimacijski omjer $5/2(1+d_{\min})/d_{\min}(1+\epsilon)$.

Slijedni koraci algoritma predstavljaju operacije koje su izvedive u $O(mn)$ vremenu. Računanje matrice vidljivosti u proceduri CALCULATE-VISIBILITY zahtjeva $O(mnk)$ vremens. Implementacijski model za proceduru LP-SOLVE temelji se na GLPK simpleks rješavatelju za $\epsilon = 0$ odnosno na aproksimativnom rješavatelju MULTI-COVER-APX za $\epsilon > 0$ koje radi u $O(mnk + \epsilon^{-2}m^2 n \log n)$ vremenu.

Zaključujemo sa teoremom:

THEOREM 7.3. *Za problem* 1.5D-TERRAIN-MULTI-GUARD *s n čuvara i m klijenata gdje je* $d_{\min}$ *najmanji zahtjev od klijenata, postoji algoritam koji vraća* $5/2(1+1/d_{\min})(1+\epsilon)$ *aproksimaciju u vremenu* $O(\epsilon^{-2}n^2 m \log n)$.

7.4 OSVRT NA IMPLEMENTACIJSKI MODEL ALGORITAMA

Doprinosi

U ovom poglavlju predložen je implementacijski model za težinsko čuvanje 1.5D terena odnosno čuvanje 1.5D terena s višestrukim pokrivanjem. Oba algoritma su opisana jedinstvenim generičkim algoritmom. Ključni implementacijski izazov problemu predstavlja brzo rješavanje linearnog programa koji odgovara problemu pakiranja i pokrivanja odnosno problemu ograničenog višestrukog pokrivanja. Na temelju ekperimenata zaključuje se da za male instance problema dovoljno je koristiti optimalni LP rješavatelj dok za veće instance paralelna aproksimacijska shema pokazuje bolji vremenski rezultat.

Budući rad

U implementacijskom modelu predložen je PRAM model aproksimacijske sheme za težinsku inačicu problema čuvanja 1.5D terena. Kako problem čuvanja 1.5D terena s višestrukim pokrivanjem sadrži LP formulaciju problema ograničenog višestrukog pokrivanja od interesa je ispitati adekvatnost PRAM modela za algoritam iz [29] na CUDA platformi.

8

ZAKLJUČAK

Ovaj doktorski rad bavi se problemima vidljivosti na 1.5D terenima. Predlaže poboljšane aproksimacije nekoliko inačica problema čuvanja 1.5D terena te model implementacije istih.

DIO I Glavni doprinos u problemu čuvanja 1.5D terena predstavlja aproksimacijski algoritam konstantne aproksimacije za inačicu problema u kojem čuvari imaju težine/cijenu i inačici problema u kojem neka točka terena koja treba biti čuvana ima zahtjev na broj čuvara koju je trebaju pokriti. Aproksimacijski algoritam za nekoliko inačica težinskog čuvanja 1.5D terena dano je u *poglavlju 3*, a problem s višestrukim pokrivanjem u *poglavlju 4*. Rezultat predstavlja prvo poopćenje klasičnog problema čuvanja 1.5D terena i u trenutku pisanja najbolju aproksimaciju. Treba napomenuti kako je nejasno kako dosadašnje algoritme za klasično čuvanje 1.5D terena se mogu prilagoditi da obuhvaćaju težinsku inačicu odnosno inačicu višestukog pokrivanja. Aproksimacijski rezultat obje inačice problema povezuje se s problemima pokrivanja u kojem je matrica uvjeta potpuno uravnotežena za koje već postoje primal-dual optimalni algoritmi [42]. Alternativno, u ovom radu se predlažu kombinatorni algoritmi za takve probleme. Težinsko čuvanje 1.5D terena promatra se dodatno s gledišta parcijalnog pokrivanja u *poglavlju 5* i pokazuje da se problem može svesti na inačicu problema u kojoj je matrica uvjeta 2-separabilna za koje onda slijede algoritmi iz [72]. Težinsko čuvanje 1.5D terena zajedno s parcijalnom inačicom objavljeno je u [25], te nakon toga, dana je prva konstanta aproksimacija za inačicu problema s višestrukim pokrivanjem u [26].

DIO II Porastom popularnosti CUDA paralelne platforme kao masivno paralelni računalni resurs te razvojem API-a i ekosustava koji ga prate otvorile su se nove mogućnosti u grafičkom programiranju

opće namjene. Proučavajući brze rješavatelje linearnih programa u ovom doktorskom radu predlaže se paralelna inačica aproksimacijske sheme prilagođena za CUDA platformu za linearne programe problema pakiranja i pokrivanja u *poglavlju 6* koji se može koristiti i za probleme pakiranja i pokrivanja s eksponencijalno mnogo uvjeta uz određene modifikacije. Na temelju eksperimenata bilježi se ubrzanje u usporedbi s RAM modelom implementacije. Implementacija je uspoređena sa simpleks metodom iz GLPK paketa te se primjećuje nekoliko reda veličine ubrzanje do na faktor aproksimacije. Rezultati su objavljeni u [69] i u trenutku pisanja se nalazi na recenziji. Predlaže se korištenje RAM i PRAM modela aproksimacijske sheme za implementaciju algoritama predstavljenih u *Dio I*.

Neki od neriješenih problema vezanih za čuvanje 1.5D terena:

- Za težinski problem 1.5D čuvanja s višestrukim pokrivanjem može li se s istom analizom doći do konstantne aproksimacije? Ključni izazov predstavlja nalaženje optimalnog algoritma za lijeve i desne probleme u kojima je matrica uvjeta potpuno uravnotežana.
- Za težinski problem čuvanja 1.5D terena ostaje riješiti može li se aproksimirati s aproksimacijskom shemom koristeći lokalnu pretragu kao u npr. [33].
- Može li se Mestreov pristup [42] za parcijalno pokrivanje s potpuno uravnoteženom matricom uvjeta proširiti i na problem parcijalnog višestrukog pokrivanja na 1.5D terenu?
- Problem k maksimalnog pokrivanja za 1.5D teren predstavlja odrediti najviše k čuvara s terena 1.5D tako da je teren maksimalno pokriven. Kao općenit problem pokrivanja ima već aproksimaciju $1 - 1/e$. Može li se kako problem proizvoljno aproksimirati koristeći ideje primjerice iz [33] je otvoren problem.

Primal-dual algoritmi za nalaženje aproksimativnog rješenja posebnih klasa linearnih programa predstavljaju alternativu klasičnim metodama za rješavanje linearnih programa. Osim analitičkog naglaska na smanjivanju ovisnosti vremena izvršavanja o parametru aproksimacije može li se na implementacijskoj razini ostvariti bolja paralelizacija tih rješavatelja ostaje implementacijski izazov.

LITERATURA

[1] CUDA 5 Toolkit, 2013. https://developer.nvidia.com/cuda-toolkit.

[2] A. Aggarwal. *The art-gallery theorem: its variations, applications, and algorithmic aspect*. PhD thesis, John Hopkins University, 1984.

[3] M. Aigner and G. M. Ziegler. *Proofs from THE BOOK*. Springer Publishing Company, Incorporated, 4th edition, 2009.

[4] D. Avis and G.T. Toussaint. An efficient algorithm for decomposing a polygon into star-shaped polygons. *Pattern Recognition*, 13(6):395–398, January 1981.

[5] F. Belić, Ž. Hocenski, and D. Ševerdija. Naïve Matrix Multiplication versus Strassen Algorithm in Multi-thread Environment. *Tehnički vjesnik*, 18(3):309–314, 2011.

[6] N. Bell and M. Garland. Efficient Sparse Matrix-Vector Multiplication on CUDA. NVIDIA Technical Report NVR-2008-004, NVIDIA Corporation, December 2008.

[7] B. Ben-Moshe, M. J. Katz, and J. S. B. Mitchell. A Constant-Factor Approximation Algorithm for Optimal 1.5D Terrain Guarding. *SIAM J. Comput.*, 36:1631–1647, 2007.

[8] C. Berge. Balanced Matrices. *Mathematical Programming*, 2:19–31, 1972.

[9] D. Bertsimas and J. N. Tsitsiklis. *Introduction to Linear Optimization (Athena Scientific Series in Optimization and Neural Computation, 6)*. Athena Scientific, February 1997.

[10] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, March 2004.

[11] H. Broennimann and M. Goodrich. Almost optimal set covers in finite VC-dimension. *Discrete & Computational Geometry*, 14:463–479, 1995. 10.1007/BF02570718.

[12] E. B. Burger and M. Starbird. *The Heart of Mathematics: An Invitation to Effective Thinking*. Key College Publishing, 2004.

[13] T. M. Chan and S. Har-Peled. Approximation algorithms for maximum independent set of pseudo-disks. In *Proceedings of the 25th annual symposium on Computational geometry*, SCG '09, pages 333–340, New York, NY, USA, 2009. ACM.

[14] M. Charikar, S. Khuller, D. M. Mount, and G. Narasimhan. Algorithms for facility location problems with outliers. In *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, SODA '01, pages 642–651, Philadelphia, PA, USA, 2001. Society for Industrial and Applied Mathematics.

[15] B. Chazelle. Triangulating a simple polygon in linear time. *Discrete Comput. Geom.*, 6(5):485–524, August 1991.

[16] C. Chekuri, K. L. Clarkson, and Sariel Har-Peled. On the set multi-cover problem in geometric settings. In *Proceedings of the 25th Annual Symposium on Computational Geometry*, SCG '09, pages 341–350. ACM, 2009.

[17] D. Z. Chen, V. Estivill-Castro, and J. Urrutia. Optimal guarding of polygons and monotone chains. In *Proceedings of the 7th Canadian Conference on Computational Geometry*, pages 133–138, 1995.

[18] V. Chvatal. A combinatorial theorem in plane geometry. *Journal of Combinatorial Theory*, 18:39–41, 1975.

[19] K. L. Clarkson and K. Varadarajan. Improved Approximation Algorithms for Geometric Set Cover. *Discrete Comput. Geom.*, 37:43–58, 2007.

[20] G. B. Dantzig and M. N. Thapa. *Linear Programming. 2. , Theory and extensions*. Springer series in operations research. Springer, New York, 2003.

[21] J. Duato, F. D. Igual, R. Mayo, A. J. J. Peña, E. S. Quintana-Ortí, and F. Silla. An Efficient Implementation of GPU Virtualization in High Performance Clusters. In Hai-Xiang Lin, Michael Alexander, Martti Forsell, Andreas Knüpfer, Radu Prodan, Leonel Sousa, and Achim Streit, editors, *Euro-Par 2009 – Parallel Processing Workshops*, volume 6043 of *Lecture Notes in Computer Science*, pages 385–394. Springer Berlin Heidelberg, 2010.

[22] A. Efrat and S. Har-Peled. Guarding galleries and terrains. *Inf. Process. Lett.*, 100(6):238–245, December 2006.

[23] S. Eidenbenz. *(In-)Approximability of visibility problems on polygons and terrains*. PhD thesis, ETH Zürich, 2000.

[24] S. Eidenbenz, C. Stamm, and P. Widmayer. Inapproximability Results for Guarding Polygons and Terrains. *Algorithmica*, 31:79–113, 2001.

[25] K. Elbassioni, E. Krohn, D. Matijević, J. Mestre, and D. Ševerdija. Improved Approximations for Guarding 1.5-Dimensional Terrains. *Algorithmica*, 60(2):451–463, 2011.

[26] K. Elbassioni, D. Matijević, and D. Ševerdija. Guarding 1.5D terrains with demands. *International Journal of Computer Mathematics*, 89(16):2143–2151, 2012.

[27] U. Feige. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM*, 45(4):634–652, July 1998.

[28] S. Fisk. A short proof of Chvátal's Watchman Theorem . *Journal of Combinatorial Theory, Series B*, 24(3):374 –, 1978.

[29] L. Fleischer. A fast approximation scheme for fractional covering problems with variable upper bounds. In *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, SODA '04, pages 1001–1010, Philadelphia, PA, USA, 2004. Society for Industrial and Applied Mathematics.

[30] G. N. Frederickson. Fast algorithms for shortest paths in planar graphs, with applications. *SIAM J. Comput.*, 16(6):1004–1022, December 1987.

[31] Naveen Garg and Jochen Könemann. Faster and Simpler Algorithms for Multicommodity Flow and Other Fractional Packing Problems. *SIAM Journal on Computing*, 37(2):630, 2007.

[32] S. K. Ghosh. Approximation algorithms for art gallery problems in polygons. *Discrete Applied Mathematics*, 158(6):718 – 722, 2010.

[33] M. Gibson, G. Kanade, E. Krohn, and K. Varadarajan. An Approximation Scheme for Terrain Guarding. In *Proceedings of the*

12th International Workshop and 13th International Workshop on Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX '09 / RANDOM '09, pages 140–148. Springer-Verlag, 2009.

[34] D. Golovin, V. Nagarajan, and M. Singh. Approximating the k-multicut problem. In *In Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 621–630, 2006.

[35] H. González-Banos. A randomized art-gallery algorithm for sensor placement. *Proceedings of the seventeenth annual symposium on Computational geometry - SCG '01*, pages 232–240, 2001.

[36] N. K. Govindaraju and D. Manocha. Cache-efficient numerical algorithms using graphics hardware. *Parallel Computing*, 33(10-11):663–684, November 2007.

[37] R. L. Graham. An Efficient Algorithm for Determining the Convex Hull of a Finite Planar Set. *Information Processing Letters*, 1(4):132–133, 1972.

[38] G. Greeff. The revised simplex algorithm on a GPU. Technical report, Univ. of Stellenbosch, 2005.

[39] R. Greenlaw, H. J. Hoover, and W. L. Ruzzo. *Limits to parallel computation: P-completeness theory*. Oxford University Press, Inc., New York, NY, USA, 1995.

[40] R. Hassin and D. Segev. Rounding to an integral program. In *In Proceedings of the 4th International Workshop on Efficient and Experimental Algorithms (WEA'05*, pages 44–54, 2005.

[41] D. S. Hochbaum. *Approximation algorithms for NP-hard problems*. PWS Publishing Co., Boston, MA, USA, 1997.

[42] A. J. Hoffman, A. Kolen, and M. Sakarovitch. Totally-balanced and greedy matrices. *SIAM Journal on Algebraic and Discrete Methods*, 6:721–730, 1985.

[43] S. Hong, S. K. Kim, T. Oguntebi, and K. Olukotun. Accelerating CUDA graph algorithms at maximum warp. *Proceedings of the 16th ACM symposium on Principles and practice of parallel programming - PPoPP '11*, page 267, 2011.

[44] J. E. Hopcroft, R. Motwani, and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation (3rd Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2006.

[45] M. Imai, Y. Hayakawa, H. Kawanaka, Wei Chen, Koichi Wada, Carla Denise Castanho, Y. Okajima, and H. Okamoto. A Hardware Implementation of PRAM and Its Performance Evaluation. In *Proceedings of the 15 IPDPS 2000 Workshops on Parallel and Distributed Processing*, IPDPS '00, pages 143–148, London, UK, UK, 2000. Springer-Verlag.

[46] J. Jaja. *An Introduction to Parallel Algorithms*. Addison-Wesley Professional, March 1992.

[47] D. Jang and S. Kwon. Fast Approximation Algorithms for Art Gallery Problems in Simple Polygons. *CoRR*, abs/1101.1346, 2011.

[48] J.H. Jung and D.P. O'Leary. Implementing an interior point method for linear programs on a CPU-GPU system. *Electronic Transactions on Numerical Analysis*, 28:174–189, 2008.

[49] N. Karmarkar. A new polynomial-time algorithm for linear programming. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing*, STOC '84, pages 302–311, New York, NY, USA, 1984. ACM.

[50] L.G. Khachiyan. A polynomial time algorithm for linear programming. *Doklady Akademiia Nauk SSSR*, 244:1093–1096, 1979.

[51] J. King. A 4-Approximation Algorithm for Guarding 1.5-Dimensional Terrains. In *Proceedings of the 13th Latin American Symposium on Theoretical Informatics*, pages 629–640, 2006.

[52] J. King. VC-dimension of visibility on terrains. In *Proceedings 20th Canadian Conference on Computational Geometry*, CCCG '08, pages 27–30, 2008.

[53] J. King. Fast vertex guarding for polygons with and without holes. *Comput. Geom. Theory Appl.*, 46(3):219–231, April 2013.

[54] J. King and E. Krohn. Terrain guarding is NP-hard. In *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '10, pages 1580–1593. SIAM, 2010.

[55] V. Klee and G. J. Minty. How good is the simplex algorithm? In O. Shisha, editor, *Inequalities*, volume III, pages 159–175. Academic Press, New York, 1972.

[56] J. Koenemann, O. Parekh, and D. Segev. A Unified Approach to Approximating Partial Covering Problems. In Yossi Azar and Thomas Erlebach, editors, *Algorithms – ESA 2006*, volume 4168 of *Lecture Notes in Computer Science*, pages 468–479. Springer Berlin Heidelberg, 2006.

[57] A. Kolen. *Location problems on trees and in the rectilinear plane*. PhD thesis, Matematisch Centrum, Amsterdam, 1982.

[58] A. Kolen and A. Tamir. Covering Problems. In *Discrete Location Theory*, chapter 6. ACM, 1990.

[59] K. Kothapalli, R. Mukherjee, M. S. Rehman, S. Patidar, P. J. Narayanan, and K. Srinathan. A performance prediction model for the CUDA GPGPU platform. *2009 International Conference on High Performance Computing (HiPC)*, pages 463–472, December 2009.

[60] C. Koufogiannakis and N. E. Young. Beating Simplex for Fractional Packing and Covering Linear Programs. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 494–504, 2007.

[61] C. Koufogiannakis and N. E. Young. Beating Simplex for Fractional Packing and Covering Linear Programs. *CoRR*, abs/0801.1987, 2008.

[62] A. Kröller, T. Baumgartner, S. P. Fekete, and C. Schmidt. Exact solutions and bounds for general art gallery problems. *J. Exp. Algorithmics*, 17(1):2.3:2.1–2.3:2.23, May 2012.

[63] M. E. Lalami, V. Boyer, and D. El-baz. Efficient Implementation of the Simplex Method on a CPU-GPU System. *PCO11*, pages 1999–2006, 2011.

[64] J.C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Norwell, MA, USA, 1991.

[65] D. Lee and A. Lin. Computational complexity of art gallery problems. *Information Theory, IEEE Transactions on*, 32(2):276 – 282, mar 1986.

[66] A. Levin and D. Segev. Partial multicuts in trees . *Theoretical Computer Science*, 369(1–3):384 – 395, 2006.

[67] M. Luby and N. Nisan. A Parallel Approximation Algorithm for Positive Linear Programming. In *ACM Symposium on Theory of Computing*, pages 448–457, 93.

[68] A. Makhorin. *GNU Linear Programming Kit, Reference Manual for GLPK Version 4.45*. Free Software Foundation, Inc., 2010.

[69] G. Martinović, D. Matijević, and D. Ševerdija. CPU-GPU Implementation of a Fast Approximation Scheme for Covering and Packing Problems. rukopis, 2013.

[70] G. Martinović, D. Matijević, and D. Ševerdija. Fast Implementation of Guarding 1.5D Terrains. rukopis, 2013.

[71] D. Matijević and D. Ševerdija. Problemi vidljivosti. *Osječki matematički list*, 10(1), 2010.

[72] J. Mestre. Lagrangian Relaxation and Partial Cover (Extended Abstract). In *Proceedings of the 25th Annual Symposium on Theoretical Aspects of Computer Science*, pages 539–550, 2008.

[73] N.. Mustafa and S. Ray. PTAS for geometric hitting set problems via local search. In *Proceedings of the 25th Annual Symposium on Computational Geometry*, SCG '09, pages 17–22. ACM, 2009.

[74] J. O'Rourke. *Art gallery theorems and algorithms*. Oxford University Press, Inc., New York, NY, USA, 1987.

[75] G. Safak. The Art-Gallery Problem : A Survey and an Extension, Master's Thesis. Master's thesis, School of Computer Science and Engineering Royal Institute of Technology, 2009.

[76] A. Schrijver. *Theory of linear and integer programming*. John Wiley & Sons, Inc., New York, NY, USA, 1986.

[77] T. C. Shermer. Recent results in art galleries. *Proceedings of the IEEE*, 1992.

[78] M. Sipser. *Introduction to the Theory of Computation*. International Thomson Publishing, 1st edition, 1996.

[79] E. Smith, J. Gondzio, and J. Hall. GPU acceleration of the matrix-free interior point method. In *Proceedings of the 9th international conference on Parallel Processing and Applied Mathematics - Volume Part I*, PPAM'11, pages 681–689, Berlin, Heidelberg, 2012. Springer-Verlag.

[80] D. G. Spampinato and A. C. Elstery. Linear optimization on modern GPUs. In *Proceedings of the 2009 IEEE International Symposium on Parallel&Distributed Processing*, IPDPS '09, pages 1–8, Washington, DC, USA, 2009. IEEE Computer Society.

[81] Š. Ungar. *Matematička analiza IV*. Matematički odjel PMF, Zagreb, 2001.

[82] J. Urrutia. Art gallery and illumination problems. *Handbook of computational geometry*, 2000.

[83] V. Vapnik and A. Chervonenkis. On the Uniform Convergence of Relative Frequencies of Events to Their Probabilities. *Theory of Probability & Its Applications*, 16(2):264–280, 1971.

[84] V. V. Vazirani. *Approximation algorithms*. Springer-Verlag New York, Inc., New York, NY, USA, 2001.

[85] N.E. Young. Sequential and Parallel Algorithms for Mixed Packing and Covering. In *42nd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 538–546, 2001.

UNAPRIJEĐENI APROKSIMACIJSKI ALGORITMI ZA ČUVANJE 1.5D TERENA

SAŽETAK

Ovaj doktorski rad proučava poboljšane aproksimacije za nekoliko inačica problema čuvanja 1.5D terena. Ulaz problema je poligonalna x-monotona linija na kojem je odabran skup čuvara i skup klijenata. Cilj je odrediti optimalni broj čuvara koji će pokriti sve klijente. U doktorskom radu su predstavljeni aproksimacijski algoritmi za varijante kad čuvari imaju pridružene težine/cijene odnosno kad klijenti zahtjevaju robusnije čuvanje, dakle, imaju zahtjev na broj čuvara koji ih moraju pokrivati. U inačici problema s težinama promatra se i problem parcijalnog pokrivanja. U svim inačicama problema predstavljen je prvi aproksimacijski algoritam konstantne aproksimacije i u trenutku pisanja predstavlja *state-of-the-art* rezultat za te probleme. Korištenjem tehnike zaokruživanja rješenja linearnog programa odgovarajućeg problema čuvanja 1.5D terena ispostavlja se da sve inačice problema promatranih u doktorskom radu mogu aproksimirati unutar faktora 5 do na neku proizvoljno malu aditivnu grešku.

Kako rješavatelj linearnog programa predstavlja osnovni dio predstavljenih algoritama za čuvanje 1.5D terena, promatra se slijedna i paralelna implementacija brzog aproksimativnog rješavanja linearnog programa posebne klase problema zvanih problemi pokrivanja i pakiranja. Eksperimentalnom analizom uspoređuje se aproksimacijska shema za rješavanje linearnih programa problema pakiranja i pokrivanja na nekoliko tipova ulaza na jednoprocesorskoj i CUDA omogućenoj platformi te s već postojećim rješavateljem linearnih programa. Za teorijsku analizu vremena izvršavanja CUDA implementacije uzima se PRAM model čija se korespondencija s CUDA platformom potvrđuje eksperimentima. Kao model implementacije u ovom radu se predlaže inkorporacija $(1+\epsilon)$ aproksimativnih rješavatelja za rješavanje linearnog programa kao podrutine za aproksimacijske algoritme čuvanja 1.5D terena što uvodi dodatni multiplikativni faktor pogreške $1+\epsilon$ u aproksimacijskim faktorima, ali daje eksplicitnu ovisnost o parametru $1/\epsilon$ u vremenima izvršavanja aproksimacijskih algoritama.

KLJUČNE RIJEČI: *aprokimacijski algoritmi, CUDA, čuvanje 1.5D terena, primal-dual algoritmi, problemi pakiranja i pokrivanja, problemi vidljivosti*.

IMPROVED APPROXIMATION ALGORITHMS FOR 1.5D TERRAIN GUARDING

ABSTRACT

Improved approximations for several versions of the 1.5D terrain guarding problem are presented in this thesis. The input for the problem is an x-monotone polygonal line, namely 1.5D terrain, with a given set of points from terrain called guards and a set of points from terrain called clients. The goal is to find the minimum cardinality set of guards which will guard all the clients on the terrain. The purpose of this thesis is to present approximation algorithms for the weighted version of the problem, i. e. the guards have associated weights/costs and the robust version respectively, i. e. the clients have a demand for the certain number of guards that must guard them. Additionally, we consider a partial variant of the weighted version of problem. In all the versions of problem the constant factor approximation algorithms are presented and in the time of writing it is the state-of-the-art result. Using a rounding technique for the solution of the corresponding linear programming relaxation of the 1.5D guarding problems it is shown that all the versions considered in this thesis can be approximated within factor 5 up to the small additive error.

In the second part of the thesis, the sequential and parallel implementations of the fast approximate linear programming solver for the covering and packing problems are presented. By experimental analysis we test the approximation scheme on different testbeds and compare to a linear programming solver toolkit. For the theoretical analysis of the running time, the PRAM abstraction suffices to model the CUDA implementation of the approximation scheme. In this thesis, the RAM and CUDA implementation of the approximation scheme as a subrutine is suggested to be incorporated in the implementation model of presented approximation algorithms for 1.5D terrain guarding problems. Using fast $(1+\epsilon)$-approximate solvers for solving linear programs, a multiplicative error $(1+\epsilon)$ is incured in approximation factors, though, on the other hand, the running time of approximation algorithms is explicitly dependent of $1/\epsilon$.

KEYWORDS: *1.5D terrain guarding, approximation algorithms, covering packing problems, CUDA, primal-dual algorithms, visibility problems.*

ŽIVOTOPIS

Domagoj Ševerdija je rođen 1983. god. u Vinkovcima, Hrvatska, gdje završava osnovnu školu. Prirodoslovno-matematičku gimnaziju završio je u Vinkovcima, 2002. godine. Iste godine upisuje nastavnički studij na Odjelu za matematiku, Sveučilišta J. J. Strossmayera u Osijeku, gdje diplomira 2007. god. te dobiva zvanje profesora matematike i informatike. Iste godine upisuje doktorski studij na Elektrotehničkom fakultetu, Sveučilišta J. J. Strossmayera u Osijeku, smjer: komunikacije i informatika. Trenutno radi kao asistent na Odjelu za matematiku u Osijeku od 2007.god.

PRILOG 1

Na

https://code.google.com/p/mpc-apx-solver/

nalazi se implementacija aproksimacijske sheme opisane u *poglavlju 6*. Implementacija se može koristiti za brzo aproksimativno izračunavanja problema koji se mogu formulirati kao linearni problemi pokrivanja i pakiranja. Implementacija dolazi u 2 verzije: C++ kod koji kao jednoprocesorska implementacija za rješavanje linearnog programa i CUDA 4.0+ programski kod ukoliko platforma na kojoj se kod izvršava ima CUDA omogućenu grafičku karticu.