Optimizacija raspodjele kontejnera na slagalištu lučkoga kontejnerskog terminala

Maglić, Livia

SVEUČILIŠTE U RIJECI
POMORSKI FAKULTET U RIJECI

Livia Maglić

OPTIMIZACIJA RASPODJELE KONTEJNERA NA SLAGALIŠTU LUČKOGA KONTEJNERSKOG TERMINALA

DOKTORSKA DISERTACIJA

Rijeka, 2015.

SVEUČILIŠTE U RIJECI
POMORSKI FAKULTET U RIJECI

Livia Maglić

OPTIMIZACIJA RASPODJELE KONTEJNERA NA SLAGALIŠTU LUČKOGA KONTEJNERSKOG TERMINALA

DOKTORSKA DISERTACIJA

Mentor: prof. dr. sc. Čedomir Dundović

Rijeka, 2015.

UNIVERSITY OF RIJEKA
FACULTY OF MARITIME STUDIES

Livia Maglić

OPTIMISATION OF CONTAINER RELOCATION PROBLEM IN PORT CONTAINER TERMINAL

DOCTORAL DISSERTATION

Rijeka, 2015.

Mentor rada: prof. dr. sc. Čedomir Dundović

Doktorska disertacija obranjena je dana 05.02.2016. na Pomorskom fakultetu Sveučilišta u Rijeci, pred povjerenstvom u sastavu:

1. Red. prof. dr. sc. Svjetlana Hess, predsjednica i članica,
2. Red. prof. dr. sc. Čedomir Dundović, mentor i član,
3. Red. prof. dr. sc. Zvonko Kavran, vanjski član.

SAŽETAK

U radu je istražen operativni logistički problem razmještaja kontejnera (BRP/CRP) na slagalištu lučkoga kontejnerskog terminala. Cilj istraživanja bio je razviti optimizacijski model koji rješava problem razmještaja kontejnera te omogućava brže i lakše donošenje odluka na operativnoj razini planiranja na kontejnerskom terminalu. Predloženi optimizacijski model izrađen je primjenom genetskog algoritma koji na osnovu 4 definirana pravila premještaja određuje poziciju premještaja kontejnera unutar odjeljka. Rezultat optimizacije za promatrani odjeljak predstavlja set pravila premještaja kojim se postiže minimalan broj radnji premještaja kontejnera unutar odjeljka. Model je testiran na osnovu 4 veličine odjeljka i 3 različita scenarija u pogledu udjela popunjenosti odjeljaka. Za svaku veličinu i scenarij popunjenosti odjeljka slučajno je stvoren jedan početni raspored kontejnera unutar odjeljka. S ciljem utvrđivanja različitosti rješenja s aspekta dobivenog seta pravila (kromosoma rješenja), odjeljak zadane veličine i scenarija popunjenosti te istog početnog rasporeda testiran je prilikom deset pokretanja modela. Dakle, model je ukupno pokrenut 120 puta, a kao zaključak se nameće to da je za isti početni raspored odjeljka zadane veličine i scenarija popunjenosti odjeljka moguće dobiti jednako kvalitetno ili optimalno rješenje primjenom različitog seta pravila tj; premještanjem kontejnera na različite kontejnerske pozicije.

Dodatno testiranje predloženog modela provedeno je s ciljem usporedbe rezultata modela s rezultatima drugih autora koji su za izradu modela koristili metaheurističke i heurističke metode. Testiranje predloženog modela provedeno je za 4 veličine odjeljka i jedan scenarij s udjelom popunjenosti od približno 90% na uzorku od 40 slučajno stvorenih različitih početnih rasporeda odjeljaka, dakle ukupno je ostvareno 160 pokretanja modela. Dobiveni rezultati pokazuju učinkovitost predloženog modela s maksimalnim postignutim poboljšanjem od 16,59 %.

Izrađena je i analiza osjetljivosti predloženog modela na promjenu parametara genetskog algoritma, na osnovu koje je zaključeno da su svi parametri rada genetskog algoritma postavljeni tako da omogućavaju postizanje optimalnih ili najkvalitetnijih rješenja.

KLJUČNE RIJEČI: lučki kontejnerski terminal, slagalište, logistika, problem razmještaja kontejnera, optimizacija, genetski algoritam

SUMMARY

This paper analyses the operational logistical problem in the container storage yard, called the block relocation problem or container relocation problem (BRP/CRP). The aim of the paper was to develop an optimization model that solves BRP/CRP and enables faster and easier decision-making at the operational planning level at the container terminal. The proposed optimization model is based on genetic algorithm which determines the storage location of moving container based upon 4 relocation rules. The result of optimization, for a given yard bay, is based upon a set of relocation rules which minimize the number of relocation movements. The model was tested for a 4 different sizes and 3 different scenarios of a bay occupancy. For each size and yard bay occupancy one initial layout of containers in the yard bay was generated randomly. In order to determine the diversity of obtained results in terms of chromosome solution, given yard bay defined size, occupancy and the same initial layout was tested ten times. Thus, the model was running total of 120 times and we can conclude that for a given yard bay defined with size, occupancy and initial layout, it can obtain high-quality solution or optimal solution by applying different sets of rules. Additional testing of the proposed model was conducted to compare the results of the model with results of the other authors who solved CRP/BRP problem by a metaheuristics and heuristics methodology. Testing of the proposed model was conducted for 4 different yard bay sizes and one scenario of yard bay occupancy with share of about 90% on a sample of 40 randomly generated different initial yard bay layout. Thus model was running for total of 160 times. The results shows the effectiveness of the proposed model with the maximum achieved improvement of 16.59%. A sensitivity analysis of proposed model is made on which it was concluded that all the parameters of genetic algorithm are set to ensure high-quality or optimal solution.

KEY WORDS: port container terminal, storage yard, logistics, container relocation problem, optimisation, genetic algorithm

1. UVOD

Lučki kontejnerski terminali (LKT) važna su prometna čvorišta u transportu kontejnera i glavna poveznica između pomorskog i kopnenog prometa. Pored prometne funkcije, kontejnerski terminali imaju funkcionalne značajke logističkih centara s nizom različitih usluga koje se pružaju korisnicima. Lučki kontejnerski terminali mogu se s tehničkog stajališta promatrati kao proizvodni sustav unutar kojeg se odvijaju različiti logistički procesi. Zbog tehnoloških posebnosti tih procesa opravdano je taj sustav podijeliti u tri podsustava: obalni prekrcajni podsustav, podsustav slagališta i kopneni prekrcajni podsustav .

Slagališni podsustav smatra se jednim od najvažnijih podsustava lučkoga kontejnerskog terminala. Osnovni razlog je taj što se na slagalištu odvijaju brojni procesi koji uključuju razne resurse terminala (slagališni prostor, slagališne dizalica, kamione za unutarnji transport itd.) te što slagališta pojedinih svjetskih kontejnerskih terminala zauzimaju i do 75% ukupne površine terminala. Stoga je, da bi slagališni podsustav ostvario svoju funkcionalnost, potrebno koordinirano i optimizirano upravljanje pojedinim logističkim procesima na slagalištu terminala. Sukladno navedenom, formuliran je problem i predmet istraživanja.

1.1 Problem i predmet istraživanja

Opće je poznato da lučki kontejnerski terminal ima obilježja sustava unutar kojeg se, među ostalim procesima, odvija i prijevozni proces kontejnera. Kako je na terminalu praktički nemoguće ostvariti apsolutnu sinkronizaciju dolazaka prijevoznih sredstava, kontejnere koji pristignu na terminal potrebno je skladištiti na slagalištu terminala. Slagalište kontejnerskog terminala jedan je od podsustava kontejnerskog terminala s namjenskim prostorom za privremeno slaganje kontejnera koji čekaju otpremu kamionom, vlakom ili brodom. Skladištenje kontejnera omogućava izvođenje operacija prekrcaja, neovisno o vremenu otpreme kontejnera te je od presudne važnosti za funkcionalnost terminala. Slaganjem kontejnera tj. njihovim skladištenjem kompenzira se vremenska razlika u različitom dolasku prijevoznih sredstava i time otklanja potreba za njihovom sinkronizacijom.

U većini svjetskih kontejnerskih terminala kontejneri na slagalištu slažu se u blokove. Osnovni razlozi ogledaju se u tome što brodovi prevoze sve veći broj kontejnera koji je potrebno skladištiti i što su površine za slaganje kontejnera ograničene. U nastojanju da se poveća kapacitet slagališta, a sukladno tehničko- tehnološkim značajkama slagališnih dizalica, kontejneri se slažu do nekoliko redova u visinu. No, kako informacije o složenim kontejnerima na slagalištu često puta nisu točne niti pravovremene te kako je slagalište dinamičan sustav u koji pristižu novi kontejneri, javljaju se slučajevi kada je potrebno premještati kontejnere radi njihove dopreme na brod. Takvi slučajevi induciraju posebne logističke probleme u slagališnom podsustavu koji se, ovisno o tehnološkim obilježjima procesa, u znanstvenoj teoriji dijele na:

- problem razmještaja (*block relocation problem – BRP* ili *container relocation problem- CRP*)
- problem preslagivanja (*premarshalling problem - PMP*)
- problem raspoređivanja (*remarshalling problem - RMP*).

Problem razmještaja (*block relocation problem*- BRP/CRP) podrazumijeva sukcesivni premještaj kontejnera u slobodne stupce(*stacks*) unutar postojećeg odjeljka (*bay*) radi oslobađanja ciljnog kontejnera koji je prvi u nizu za otpremu. Cilj je otpremiti sve kontejnere unutar odjeljka s minimalnim brojem premještaja. Ovaj problem klasificiran je kao operativni problem kod kojeg se istovremeno odvijaju radnje premještaja i otpreme kontejnera.

Problem preslagivanja (*premarshalling- PMP*) svodi se na premještaj kontejnera u slobodne stupce unutar postojećeg odjeljka. Cilj je presložiti kontejnere unutar odjeljka s minimalnim brojem premještaja kako bi ih bilo moguće u što kraće vrijeme otpremiti tj. osigurati nesmetani slijed procesa. Kod ovog problema vrši se samo premještaj kontejnera što znači da broj kontejnera unutar odjeljka ostaje isti. Ovaj problem klasificiran je kao taktički problem.

Problem raspoređivanja (*remarshalling- RMP*) odnosi se na premještaj kontejnera na način da se prikupljaju iz jednog, ili više ishodišnih odjeljaka (*source bays*), i slažu na odredišne pozicije unutar jednog ili više odjeljka u istom bloku ili u novi blok. Cilj je kao i u prethodnom slučaju osigurati optimalni redoslijed otpreme kontejnera u skladu s planom ukrcaja broda. Kao i kod prethodno opisanog problema vrši se samo premještaj kontejnera. Ovaj problem klasificiran je kao taktički problem.

Osnovna razlika između problema BRP/CRP te PMP i RMP je ta što se kod problema BRP/CRP uz premještaj kontejnera obavlja i otprema kontejnera te što predstavlja operativni problem.

Svi navedeni problemi nastaju zbog stohastičnosti dolazaka jedinica u podsustav i nemogućnosti predviđanja potpuno točnog redoslijeda opreme kontejnera sa slagališta na ukrcaj. Također, svi navedeni problemi imaju za cilj optimizirati proces distribucije kontejnera između podsustava slagališta i obalnog prekrcajnog podsustava. Svaki od navedenih problema predstavlja zasebni logistički problem koji se može primijeniti ovisno o zahtjevima koji se postavljaju u odnosu na cilj optimizacije, prostornu konfiguraciju slagališta i tehnološki proces.

Iz prethodnog razmatranja može se definirati predmet istraživanja i problemski zadatak. U ovom doktorskom radu problem istraživanja je proces premještaja i otpreme kontejnera unutar odjeljka na slagalištu lučkoga kontejnerskog terminala. Predmet istraživanja je slagalište lučkog kontejnerskog terminala opremljeno RTG slagališnim dizalicama.

1.2 Znanstvena hipoteza i pomoćne hipoteze istraživanja

Na temelju definiranog problema i predmeta istraživanja, postavlja se radna hipoteza istraživanja koja glasi:

Moguće je optimizirati broj radnji premještaja kontejnera unutar odjeljka na slagalištu lučkoga kontejnerskog terminala primjenom genetskog algoritma.

Temeljem osnovne znanstvene hipoteze postavljaju se pomoćne hipoteze kako slijedi:

1. Moguće je dobiti optimalno rješenje za problem BRP/CRP za različite početne rasporede kontejnera unutar odjeljka.
2. Moguće je dobiti optimalno rješenje za problem BRP/CRP za različite veličine i udjele popunjenosti odjeljaka.
3. Model raspodjele kontejnera moguće je prilagoditi različitim zahtjevima korisnika.
4. Korištenjem modela raspodjele kontejnera ubrzava se i olakšava proces donošenja odluka planerima na slagalištu LKT.

1.3 Svrha i ciljevi istraživanja

Svrha istraživanja je rješavanje problema BRP/CRP na slagalištu lučkoga kontejnerskog terminala.

Cilj doktorskog rada je izrada optimizacijskog modela raspodjele kontejnera na slagalištu lučkoga kontejnerskog terminala koji će doprinijeti bržem i lakšem donošenju odluka na operativnoj razini planiranja te povećanju produktivnosti slagališta i konkurentnosti lučkog kontejnerskog terminala.

Da bi se ostvarila postavljena svrha i cilj istraživanja postavljeni su sljedeći zadaci istraživanja:

- analizirati dosadašnje teorijske spoznaje iz područja optimizacije logističkih procesa na slagalištu lučkih kontejnerskih terminala
- definirati funkciju optimizacije na temelju koje će biti definiran algoritam modela
- definirati pravila i ograničenja za rješavanje problema BRP/CRP
- provesti eksperimentalno testiranje modela s obzirom na različite veličine, početne rasporede i udjele popunjenosti odjeljaka te različite parametre rada genetskog algoritma
- usporediti dobivene rezultate s rezultatima drugih autora koji istražuju optimizaciju problema BRP/CRP.

1.4 Pregled dosadašnjih istraživanja

Kako bi se detaljno proučilo i utvrdilo područje istraživanja pretražene su sve dostupne baze znanstvenih radova i knjiga: Science Direct, Web of Science (WoS), IEEE Xplore,Digital Library, Scopus, Springer Link, Amazon i dr. Nakon opsežnog pretraživanja navedenih baza ustanovljeno je da je literatura koja istražuje, analizira i rješava problem ovog doktorskog rada relativno oskudna. U ovom dijelu rada prikazan je detaljan kronološki pregled najvažnijih radova koji su ostvarili znatan doprinos u problematici optimizacije radnji premještaja kontejnera na slagalištu lučkih kontejnerskih terminala.

Prvi znanstveni rad na tu temu napisali su Sculli et al. (1988), a u radu predstavljaju simulacijski model u kojem istražuju utjecaj broja stupaca, broja redova i različitih tipova kontejnera na ukupan broj radnji s kontejnerima na slagalištu lučkoga

kontejnerskog terminala. Na temelju dobivenih rezultata zaključuju da najveći utjecaj na ukupan broj radnji s kontejnerima imaju različiti tipovi kontejnera.

Gupta et al. (1992) dokazuju da problem BWP (*Blocks-world planning*) u teoriji kompleksnosti spada u nedeterministički polinomijalni problem (NP težak problem).

Castilho et al. (1993) uvode opće izraze za izračun ukupnog očekivanog broja radnji premještaja prilikom otpreme svih kontejnera iz odjeljka.

Kim (1997) izračunava ukupan očekivan broj radnji premještaja prilikom otpreme svih kontejnera iz odjeljka. Vrijeme otpreme kontejnera iz odjeljka nije unaprijed poznato, već svaki kontejner ima jednaku vjerojatnost da će biti sljedeći za otpremu.

Avriel et al. (1998) istražuju problem CRP s ciljem smanjenja ukupnih troškova premještaja kontejnera na brodu. Problem je definiran kao dinamički, dakle kontejneri se otpremaju u više odredišnih luka, te se u njima odvijaju procesi ukrcaja i iskrcaja kontejnera. Izrađeni model temelji se na metaheurističkim metodama i metodi binarnog programiranja.

Autori Kim et al. (1999) u svom znanstvenom radu istražuju dinamiku problema premještaja kontejnera te prezentiraju formulu za tumačenje odnosa između visine slaganja i očekivanog broja premještaja kontejnera unutar odjeljka. U radu je određena odgovarajuća maksimalna visina slaganja kontejnera sa svrhom minimizacije broja premještaja kontejnera.

Kim (2000) predlaže stablo odlučivanja kao podršku u određivanju premještaja kontejnera u stvarnom vremenu. Performanse ovog stabla odlučivanja uspoređene su s metodama dinamičkog programiranja. Kao rezultat istraživanja pokazalo se da su metode dinamičkog programiranja efikasnije odnosno da se njihovom primjenom dolazi do odluke u kraćem vremenskom periodu.

Petering (2004) razvija matematičku formulaciju problema BRP.Petering (2005) razvija heuristički algoritam za rješavanje problema BRP koji je korišten kao baza za LA (*look ahead*- gledaj naprijed) algoritma.

Zhang (2005) istražuje problem BRP te izrađuje model u koji uvodi heurističko pravilo koje prilikom premještaja kontejnera smješta kontejner u najniži stupac unutar odjeljka (TLP- *tier lowest position*).

Murty et al. (2005) u studiji izrađuju sustav za podršku odlučivanju za kontejnerski terminal u Hong Kongu te realni problem iz prakse dijele na pet problema koji među ostalima obuhvaćaju i problem premještaja kontejnera. Kako bi minimizirali broj

premještaja autori definiraju indeks premještaja kontejnera (*Reshuffle Index* - RI) koji uvode u postojeće matematičke modele minimizacije broja premještaja kontejnera.

Yang et al. (2006) u znanstvenom radu definiraju opći koncept problema premještaja kontejnera te prezentiraju model optimizacije u koji je uvrštena pretpostavka da jednom premješten kontejner ne može biti ponovo premješten. Navedeno, ograničava primjenu modela u stvarnim situacijama.

Kim et al. (2006) istražuju BRP problem te u model uvode vremena otpreme kontejnera sa slagališta u skladu s dodijeljenim prioritetom, a s ciljem minimizacije broja premještaja kontejnera. U prezentiranom modelu predstavljaju metaheurističku metodu „grananja i ograđivanja"(B&B- *branch and bound*) .

Aydin et al. (2008) istražuju osnovni i prošireni problema BRP te razvijaju heuristički algoritam za rješavanje problema velikih dimenzija. Taj algoritam nazivaju DH (*difference heuristics* - diferencijalna heuristika) algoritam. Također, uvode i naziv *cleaning move*. *Cleaning move* (potez oslobađanja) odnosi se na radnju premještaja onog kontejnera koji blokira otpremu ciljnog kontejnera.

Caserta (2008) izrađuje matematički model za rješavanje problema BRP i dokazuje da pripada u razred NP teških problema.

Lee et al. (2009) u znanstvenom radu predlažu metaheurističku metodu koja se bazira na problemu PMP koji podrazumijeva premještaj kontejnera prije početka procesa otpreme. S obzirom na početno rješenje njihova metoda prvo traži najbolji mogući raspored kontejnera, a zatim minimizira broj premještaja kontejnera uz primjenu binarnog cjelobrojnog programiranja.

Wan et al. (2009) u radu prezentiraju model optimizacije vremena otpreme kontejnera sa slagališta s ciljem minimizacije ukupnog vremena otpreme. Matematički model izrađen je primjenom metode cjelobrojnog programiranja.

Kefi (2010) pomoću simulacije dodjeljuje pozicije za smještaj kontejnera uz minimalne troškove, a u zavisnosti o očekivanim neproduktivnim manipulacijama s kontejnerima.

Wu et al. Predlažu algoritam usmjerenog (zrakastog) pretraživanja(*beam search* -BS) za rješavanje problema BRP. Dobiveni rezultati pokazuju da njihov algoritam uspješno pronalazi optimalno rješenje za probleme malih dimenzija te rješenje blizu optimalnog za probleme velikih dimenzija.

Caserta et al. (2010) iznose pregled doprinosa u području rukovanja kontejnerima te definiraju problem BRP, njegovu korelaciju i značaj za produktivnost slagališta lučkih kontejnerskih terminala.
Caserta et al. (2011) predstavljaju model za rješavanje BRP problema, koristeći metaheurističku koridor metodu. Model u vrlo kratkom vremenu uspješno rješava problem malih dimenzija.
Caserta et al. (2011) primjenjuje metaheurističku koridor metodu za rješavanje BRP problema. Dobiveni rezultati pokazuju da je primjenom koridor metode za problem malih dimenzija moguće, za vrlo kratko vrijeme izvođenja algoritma, dobiti optimalno rješenje u pogledu broja premještaja kontejnera.
Forester et al. (2012) predstavljaju model za rješavanje problema CRP u kojem kontejneri, koji se nalaze u odjeljku, imaju dodijeljene grupne prioritete. Model je izrađen primjenom heurističkih metoda donje granice i razapinjajućeg stabla. Dobiveni rezultati znatno su bolji od rezultata dobivenih od Caserta et al. (2011).
Hussein et al. (2012) u svom radu prezentiraju model koji uspješno rješava varijantu problema BRP, gdje su kao ograničenje uzete težine kontejnere(BRP-W). Model je izrađen primjenom genetskog algoritma te uspješno rješava probleme velikih dimenzija.
Jovanović et al. (2014) predlažu novi heuristički pristup u kojem prilikom odlučivanja o poziciji smještaja premještanog kontejnera uzimaju u obzir i značajke sljedećeg kontejnera koji će biti premještan. Ova ideja predstavlja poboljšanje Min-max heurističkog pristupa za rješavanje problema BRP. Dobiveni rezultati pokazuju da uvedeno poboljšanje učinkovito umanjuje ukupan broj premještaja kontejnera.
U većini navedene literature zbog složenosti problema premještaja kontejnera, za izradu modela koriste se razne heurističke i metaheurističke metode koje omogućavaju dobivanje dobrog rješenja u vrlo kratkom vremenu izvođenja modela. Tako se i u ovom istraživanju za izradu modela koji rješava BRP problem koriste te metode.

1.5 Znanstvene metode istraživanja

Prilikom istraživanja korištene su sljedeće znanstvene metode: indukcija i dedukcija, analiza i sinteza, komparacija, klasifikacija, kompilacija, deskripcije, generalizacije i specijalizacije.

Kako lučki kontejnerski terminal ima obilježja sustava, prilikom istraživanja korištena je i metoda teorije sustava.

Glavni dio istraživanja, koji se odnosi na optimizaciju raspodjele kontejnera, u podsustavu slagališta zasniva se na heurističkoj i metaheurističkoj metodi. Za potrebe testiranja modela na računalu je provedena metoda simulacije. Simulacijska metoda bazirana je na: različitim dimenzijama odjeljaka, početnom rasporedu kontejnera složenih unutar jednog odjeljka te udjelu popunjenosti odjeljka. Također, kako bi se sa sigurnošću potvrdilo dobivanje optimalnog rješenja, simulacijskom metodom testirani su različiti parametri rada genetskog algoritma.

Tijekom dokazivanja znanstvene hipoteze korištene su induktivna, deduktivna i komparativna metoda. Komparativna metoda korištena je za analizu dobivenih optimalnih rezultata za različite varijante odjeljaka.

Programski kod modela izrađen je u računalnom okruženju *Netbeans IDE 7.4* u programskom jeziku *Java*. Potom je u *Netbeans IDE 7.4* implementirano *Javino* radno okruženje za genetski algoritam i programiranje (*Java Genetic Algorithm and Programming* – JGAP) u kojem je izrađen genetski algoritam modela za rješavanje problema BRP/CRP. Za kreiranje programskih skripti za generiranje početnih rasporeda kontejnera unutar odjeljka (*bay*) korišten je programski jezik R.

1.6 Struktura doktorskog rada

Ovaj doktorski rad sastoji se od osam poglavlja i četiri privitka. U uvodnom poglavlju definiran je problem i predmet istraživanja, postavljena je znanstvena hipoteza te određena svrha i ciljevi istraživanja. Prikazan je pregled dosadašnjih istraživanja u području optimizacije logističkih problema na slagalištu lučkoga kontejnerskog terminala te korištenih znanstvenih metoda istraživanja.

U drugom poglavlju definirane su značajke i tipovi kontejnera koji dominiraju u pomorskom prometu s kratkim pregledom razvoja kontejnerskih brodova i kontejnerskog prometa.

Treće poglavlje definira strukturu i logističke procese unutar pojedinih podsustava lučkoga kontejnerskog terminala. Uz navedeno, u ovom dijelu rada unutar svakog podsustava terminala detaljno su opisani i analizirani logistički procesi na koje se

primjenjuju optimizacijske metode te je dan kratki prikaz inteligentnih tehnologija koje doprinose optimizaciji tih procesa.

U četvrtom poglavlju opisno i ilustrativno je prikazana konfiguracija slagališta, analizirane su relevantne značajke slagališne zone, načini slaganja kontejnera, parametri za vrednovanje rada slagališta, slagališna prijevozno- prekrcajna sredstva te njihova učestalost uporabe na pojedinim svjetskim kontejnerskim terminalima.

U petom poglavlju predstavljen je izrađeni model optimizacije koji rješava problem BRP/CRP.

U šestom poglavlju opisani su uvjeti pod kojima se provodi simulacijsko testiranje rada modela. U tu svrhu u programskom jeziku R generirani su početni rasporedi kontejnera za različite veličine odjeljaka. Programska skripta i primjeri početnih rasporeda prikazani su u privitku.

U sedmom poglavlju analizirani su rezultati dobiveni temeljem eksperimentalnog testiranja provedenog na računalu. Također, rezultati dobiveni predloženim modelom uspoređeni su s rezultatima drugih autora te je izvršena analiza osjetljivosti rada modela.

U zaključku su predstavljeni objedinjeni rezultati istraživanja na temelju kojih je potvrđena postavljena hipoteza te su date preporuke za daljnja istraživanja.

Nakon iznesenog sadržaja dat je popis korištenih referenci, tablica, slika, grafikona i kratica te su priložena četiri privitka. Privitak 1 prikazuje dijelove izrađenog programskog koda optimizacijskog modela. Privitak 2 prikazuje programsku skriptu na osnovu koje su generirani početni rasporedi kontejnera unutar odjeljaka. Privitak 3 prikazuje primjer slučajno generiranih početnih rasporeda kontejnera unutar odjeljka veličine 6x7 te udjela popunjenosti od približno 90%. Privitak 4 predstavlja ilustrativni prikaz dobivenih rezultata za različite veličine odjeljka. Za svaki odjeljak prikazan je dobiven rezultat primjenom jednog seta pravila.

2. RAZVOJ POMORSKOG KONTEJNERSKOG PROMETA

2.1 Značajke i tipovi kontejnera u pomorskom prometu

Idejnim tvorcem zamjene tradicionalnog načina prijevoza tereta kontejnerima smatra se Malcom Mcleann koji je 1955. godine osnovao tvrtku čija je glavna ideja bila prevoziti kamionske prikolice s njihovim teretom na brodovima. Potom je uvidio kako bi bilo jednostavnije i brže da postoji jedan kontejner u kojem se nalazi teret koji bi mogao biti podignut s vozila na brod, bez potrebe iskrcaja cijelog sadržaja. Tako je 1956.godine na brodu *Ideal*-X prvi put prevezeno 58 kontejnera. Dakle, kontejneri su nastali kao i sve prijevozne jedinice kao rezultat potrebe da se olakša i ubrza prijevoz i prekrcaj tereta te da se poveća sigurnost robe prilikom prijevoza.

Naziv kontejner(*container*) potječe od engleske riječi *contain,* a znači sve ono što u sebi može sadržavati nešto drugo. Prema njemačkom institutu za standardizaciju(DIN-*Deutsches Institut für Normung)* kontejner se definira kao sredstvo za prijevoz tereta koje [1]:

- ima trajne značajke te je dovoljno čvrsto da se može ponovo upotrebljavati
- je tako konstruirano da omogućava prijevoz tereta bez oštećenja, jednom ili više prijevoznih grana
- je opremljeno elementima koji omogućavaju lako rukovanje, posebice kada se obavlja prekrcaj s jednog na drugo prijevozno sredstvo
- je tako konstruirano da omogućava lako punjenje i pražnjenje i
- ima unutrašnju zapreminu od najmanje 1 m^3.

Danas se najviše koriste ISO[1]kontejneri koji su nastali za potrebe pomorskog i intermodalnog prijevoza i u potpunosti su standardizirani. ISO kontejneri poznati su još kao intermodalna sredstva, napravljeni tako da mogu mijenjati prijevozna sredstva (brod, cesta, željeznica). Proizvode se kao suhi (*Dry*), kocke (*Cube*), platforme i otvoreni kontejneri.

Procjenjuje se da danas postoji oko 20.000 tipova kontejnera koji se razlikuju po svojim specifičnostima i obilježjima[2]. Međunarodna organizacija za standardizaciju kontejnera u skladu s propisanim standardima Instituta za standardizaciju (DIN ISO 436,

[1] *International Organization for Standardization*- Međunarodna organizacija za standardizaciju

1996.) podijelila je kontejnere koji se najčešće koriste u pomorskom prijevozu na sljedeće:

- standardni suhi (*General purpose standard dry containers*)
- za rasute terete (*Bulk container*)
- za određene vrste tereta (*Named cargo containers*)
- temperaturni (*Thermal containers*)
- otvoreni prema gore, otvor na vrhu (*Open-top containers*)
- platforme (*Flattracks*)
- cisterne (*Tanks*)
- ventilacijski (*Air/surface containers*).

Osim navedenih vrsta postoje i brojne podvrste kontejnera, a razlog njihova postojanja je taj što kontejnerski proizvođači nastoje pronaći što prikladnije kontejnere za određene specijalizirane terete.

Standardni suhi kontejneri poznati su i kao višenamjenski. To su kontejneri zatvorenog tipa, što znači da imaju sve stranice, dok se vrata nalaze na čelu ili boku kontejnera.

Kontejneri za prijevoz rasutog tereta na samom vrhu imaju tri grotla za ukrcaj, svako grotlo je promjera otprilike 450 mm, a postoje i otvori na samim vratima. U te kontejnere teret se krca slobodnim padom ili pod tlakom[3].

Kod kontejnera za prijevoz određene vrste tereta značajne su dvije podskupine: sklopivi kontejneri i kontejneri za prijevoz živih životinja.

Temperaturni kontejneri dijele se na : izolacijske, rashladne (frigo) i grijane. Izolacijski kontejneri, zbog debljine stjenki kontejnera od iznosa do 100 mm, imaju manji skladišni prostor u odnosu na standardne kontejnere[3]. Kontejneri s grijanim/rashladnim uređajem posjeduju sustav za hlađenje koji se snabdijeva električnom energijom, stoga treba voditi računa o blizini električnih priključaka.

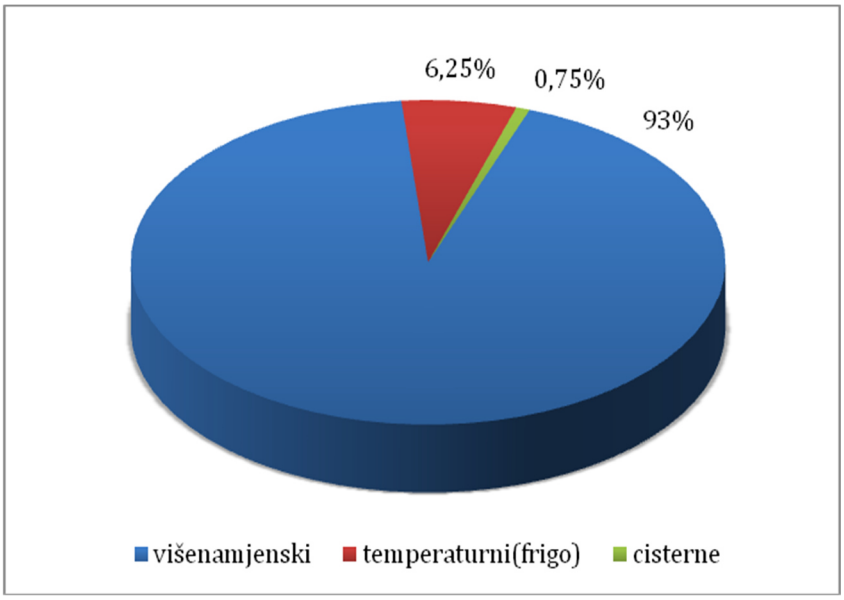
Kontejneri s otvorom na vrhu su suhi kontejneri s pomičnim krovom, a spadaju u skupinu poluotvorenih kontejnera. Na mjestu čeličnog krova nalazi se cerada koja se može maknuti za potrebe prekrcaja ili ventilaciju tereta. Ovi kontejneri prikladni su za sve tipove teških i vangabaritnih tereta.

Kontejneri platforme su kontejneri otvorenog tipa s jakom čeličnom konstrukcijom te fiksnim ili sklopivim stranicama. Namijenjeni su za ukrcaj teškog i vangabaritnog tereta, a postoje u veličinama od 20' i 40'[2].

Kontejneri cisterne namijenjeni su za prijevoz tekućina i komprimiranih plinova. Sastoje se od dva osnovna elementa: cisterne (prostor za smještaj tereta) i čeličnog okvira (zaštita cisterne od opterećenja i učvršćivanje na prijevozna sredstva).

Ventilacijski kontejneri su suhi kontejneri zatvorenog tipa s otvorima za prirodnu ventilaciju. Namijenjeni su za transport organskih tereta s visokim udjelom vlage.

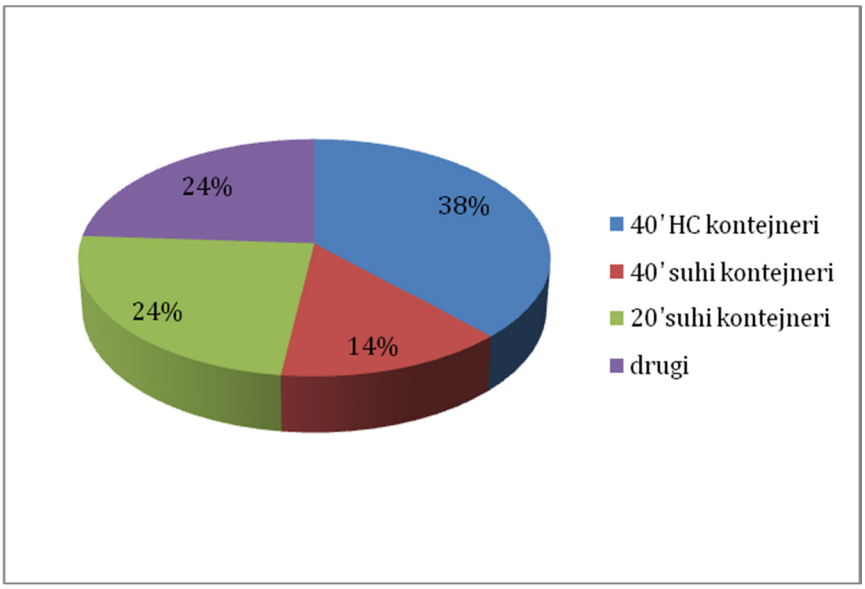
Prema podacima *Drewry Maritime Research-a* veličina flote kontejnera u svijetu u 2013. godini iznosila je oko 34,5 milijuna TEU[3], a sačinjena je od raznih tipova kontejnera od kojih najveći udio pripada standardnim suhim (višenamjenskim) kontejnerima (Grafikon 1).



Grafikon 1 Tipovi kontejnera u svjetskoj kontejnerskoj floti u 2013. godini- prema vrsti tereta (u TEU)

U međunarodnom prijevozu najviše se upotrebljavaju standardni tipovi ISO kontejnera, a to su redom: 20', 40' te 45' kontejneri (Grafikon 2). Njihova širina iznosi 8', dok je visina kontejnera 8.5' i 9.5' (*High-cubes* kontejneri).

[2] Stopa (*feet*) je mjerna jedinica za duljinu izvan Međunarodnog sustava, a jednaka je 0.3048 metara.
[3] *Twenty-foot equivalent unit* – jedinica ekvivalentna duljini 20-stopnog kontejnera.



Grafikon 2 Tipovi kontejnera u kontejnerskoj floti u 2012. godini- prema veličini

Međutim, posljednjih su godina na pojedinim kontejnerskim tržištima vidljive promjene sa aspekta veličine kontejnera, te se tako u željezničkom prometu u SAD[4]-u već dulje vrijeme koriste 53' kontejneri koji imaju iste prekrcajne zahtjeve i zauzimaju isti kapacitet na željezničkim vagonima kao i 40'kontejneri, no povećava se kapacitet po jedinici prijevoza. Tako se u pojedinim lukama Sjeverne Amerike obavljaju operacije prekrcaja kontejnera, gdje se tri 40' kontejnera prekrcavaju u dva 53' kontejnera. Osim SAD-a i u Europi se, u cilju povećanja efikasnosti transportnih lanaca i povećanja koeficijenta iskorištenja teretnog prostora kontejnera, javila inicijativa za razvoj i standardizaciju Europske intermodalne jedinice- EILU[5]. EILU je definirao Europski odbor za standardizaciju (CEN TC 119). Opravdani razlog za uvođenje EILU je u tome što su dimenzije kontejnera, budući da su se prvo pojavili u SAD-u, izražene u anglosaksonskom sustavu, dok su dimenzije Europalete izražene u metričkom sustavu pa se javlja nemogućnost racionalnog slaganja kontejnera, što se negativno odražava na koeficijent popunjenosti kontejnera. Standardne Euro palete su dimenzija: 800x1.000 mm i 800x 1.200 mm i ne mogu se efikasno slagati u kontejnere širine 8 stopa (2.435 mm). Osim toga, širina teretnog prostora kontejnera (u zavisnosti od tipa i vrste materijala od koga je napravljen) iznosi 2.250-2.300 mm. Uvođenjem nove širine kontejnera, koja iznosi 8.5 stopa (2.610 mm), omogućava se smještaj dvije standardne Euro palete usporedo jedna do druge. Uz navedene minimalne korekcije postojeće širine

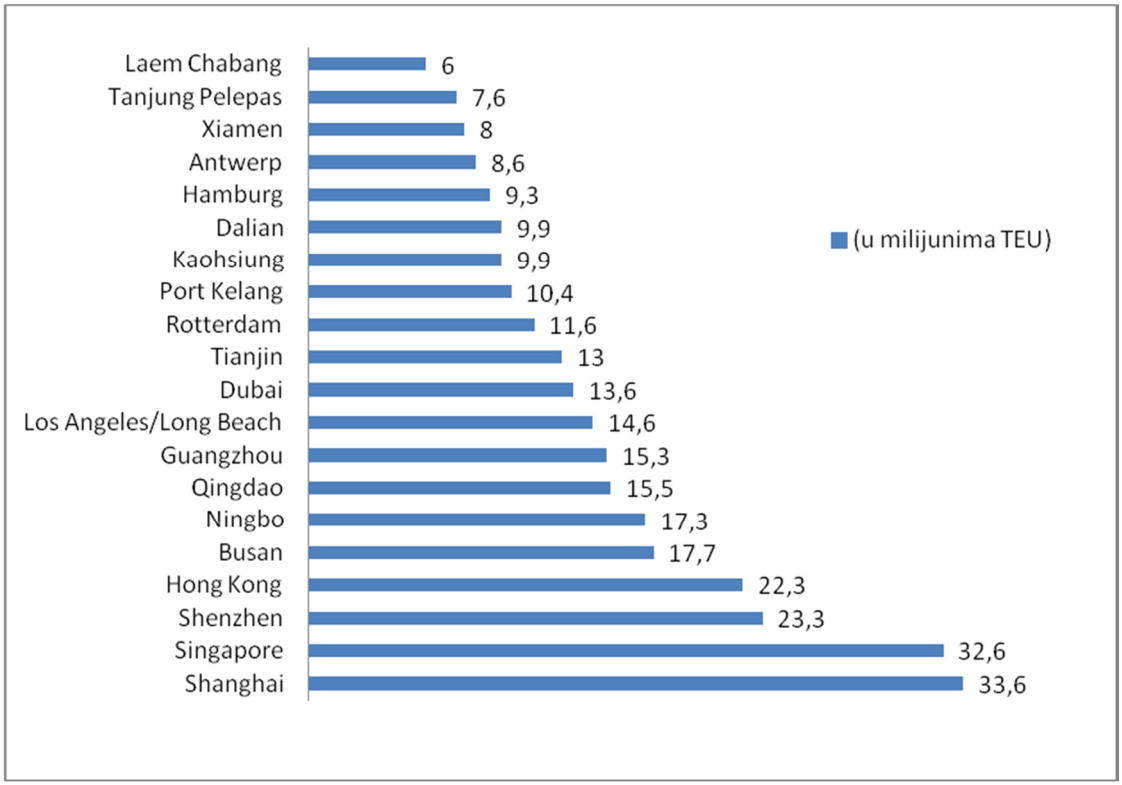
4 Sjedinjenje američke država (*United States of America*- USA)
5 *European Intermodal Loading Unit*

kontejnera postigao bi se i do 30% veći kapacitet za ukrcaj paletizirane robe od postojećih ISO kontejnera[4]. Povećanje širine intermodalnih jedinica ne bi prouzročilo velike preinake u cestovnom i željezničkom prometu, no najveće izmjene i troškove imali bi brodari koji bi trebali prilagoditi kontejnerske brodove novim intermodalnim kontejnerskim jedinicama. Može se zaključiti da je uz 60 godina kontejnerizacije i dominaciju tržišta SAD-a i Kine, promjena na svjetskoj razini nemoguća. Međutim, prema dosadašnjim analizama razvidno je da će se nove intermodalne jedinice postupno uvoditi na integriranog europskom transportnom tržištu.

2.2 Brodovi za prijevoz kontejnera

Sve se više različitih vrsta tereta transportira kontejnerima te upravo ta činjenica pridonosi konstantnom rastu i povećanju kontejnerskog prometa koji preuzima glavnu ulogu u svjetskoj trgovini.

Grafikon 3 prikazuje 20 najprometnijih svjetskih lučkih kontejnerskih terminala u 2013. godini s ukupnim prometom od 300,1 milijun TEU. Oko 26% od ukupnog svjetskog lučkog kontejnerskog prometa obavlja se u kineskim terminalima, a udio od 11% ostvaruje Singapur kao vodeći kineski kontejnerski terminal[5].



Grafikon 3 Najprometniji kontejnerski terminali u svijetu u 2013. godini

Izvor: www.worldshiping.com (travanj 2015.)

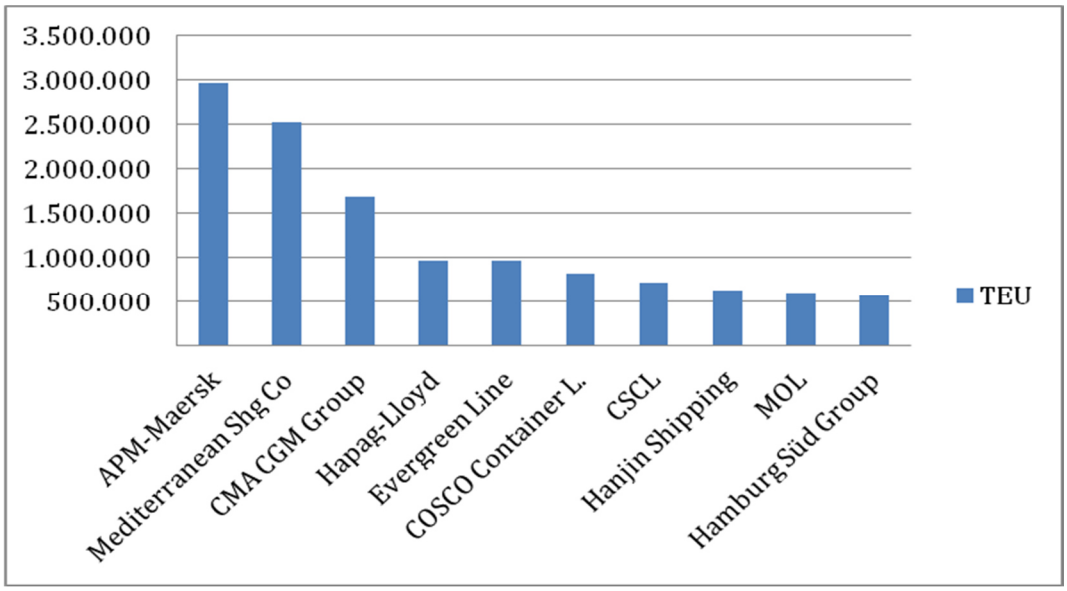
Zbog povećanja svjetske trgovinske razmjene kontejnera, koju prati konstantno povećanje kontejnerskog pomorskog prometa (smatra se da se danas oko 90% generalnog tereta prevozi morem u kontejnerima), dolazi do razvoja kontejnerskih brodova koji s većim kapacitetom omogućavaju prijevoz velikog broja kontejnera[6].
Iz Tablica 1 vidljivo je da je preko 50-ak godina prisutno neprekidno povećanje kontejnerskih brodova, tako da su danas najveći kontejnerski brodovi kapaciteta oko 18.000 TEU te spadaju u Triple E-klasu.

Tablica 1 Generacije razvoja kontejnerskih brodova

Generacije kontejnerskih brodova	**Godina pojave**	**Kapacitet (TEU)**	**Okvirni gaz (m)**	**Okvirna duljina (m)**	**Okvirna širina(m)**	**Redovi (širina) slaganja kontejnera**
Višenamjenski	1956.	500-800	9	200	20	8
Celularni	1970.	1.000- 2.500	10	215	20	10
Panamax	1980.	3.000- 4.500	12.5	250	32	13
Post Panamax	1988.	4.000- 5.000	13	285	40	15
Post Panamax Plus	2000.	6.000- 8.000	14.5	300	43	17
Novi Panamax	2006.	12.500	15.2	366	49	20
Triple E	2013.	18.000	15.5	400	59	23

Izvor:www.porttechnology.org (siječanj 2015.)

Danas svijetom plovi 5.975 kontejnerskih brodova ukupnog kapaciteta 19.119,262 TEU[7]. U nastavku je prikazan kapacitet flote deset najvećih kontejnerskih brodara u 2015.godini (stanje 25.03.2015.) čiji ukupni udio iznosi preko 50% ukupnog kapaciteta flote u svijetu (Grafikon 4).



Grafikon 4 Kapacitet flote 10 najvećih kontejnerskih brodara u 2015.godini

Izvor:www.alphaliner.com (veljača 2015.)

Budući da se predviđa daljnji porast kapaciteta i nosivosti kontejnerskih brodova, stvara se dodatni pritisak na operatere terminala jer sa stajališta ekonomske i tehničke efikasnosti, prednosti većih brodova mogu se realizirati samo uz istovremeno povećanje kapaciteta prekrcajnih sredstava i kontejnerskog terminala. Pored primjene suvremenih prekrcajnih sredstava većeg prekrcajnog kapaciteta, unapređenje performansi rada terminala može se postići i optimizacijom robnih tokova i radnji s kontejnerima na slagalištu terminala. Potreba za optimizacijskim metodama u planiranju procesa na kontejnerskim terminalima posljednjih je godina sve značajnija, a razlog tome je što su logistički problemi koji se javljaju dosegli razinu složenosti gdje daljnje unapređivanje zahtjeva primjenu znanstvenih metoda.

3. SUSTAV LUČKOG KONTEJNERSKOG TERMINALA

3.1 Funkcije i radnje na lučkom kontejnerskom terminalu

Poznato je da lučki kontejnerski terminal (LKT) predstavlja glavnu poveznica između pomorskog i kopnenog prometa. Dakle, uz prometnu funkciju terminal ima sljedeće funkcije: prekrcajnu, distribucijsku, slagališnu, gospodarsku i dr. Da bi se postiglo zadovoljenje navedenih funkcija, na terminalu se izvode sljedeće radnje s kontejnerima:

- ukrcaj/iskrcaj,
- slaganje(skladištenje),

- provjera točnosti informacija,
- nadzor i utvrđivanje šteta,
- provjera sadržaja,
- pružanje dodatnih usluga[8].

Radnja ukrcaja i iskrcaja kontejnera smatraju se jednima od ključnih logističkih procesa u radu lučkih kontejnerskih terminala. Navedene radnje obavljaju se na obali pomoću obalnih kontejnerskih dizalica (*Quay container cranes*- QCs) te na kopnu pomoću portalnih prijenosnika velikog raspona (*Transtainers*-RTG/RMG[6]), portalnih prijenosnika malog raspona (*Straddle Carriers- SC*), autodizalica (*Reachstackers*-RS), viličara (*forklifts*) te automatiziranih slagališnih dizalica (*Automatic Stacking Cranes*-ASC*).*

Nakon iskrcaja kontejnera, bilo na obali ili kopnu, kontejneri se kamionima, autodizalicama, viličarima, portalnim prijenosnicima malog raspona (*Straddle Carrier-SC*), AGV[7]-ima, L-AGV[8]-ima prevoze na slagalište.

Slaganje odnosno skladištenje kontejnera obavlja se na slagališnom prostor, a podrazumijeva pohranjivanje kontejnera do trenutka njegove ponovne upotrebe ili otpreme.

Kako bi se osiguralo da kontejneri sigurno stignu do svog odredišta, potrebna je provjera prateće dokumentacije. Sve informacije o kontejnerima šalju se putem interneta na brod na koji će biti ukrcani i u luku odredišta u kojoj će biti iskrcani.

Transportni lanac kontejnera uključuje brojne sudionike koji mogu uzrokovati oštećenja na kontejneru. Stoga je nadzor kontejnera neophodan, a obavlja se na ulazu i izlazu terminala. Nadzor omogućuje utvrđivanje odgovornog sudionika za nastalu štetu.

Temeljem slučajnog odabira baziranog na statistici, odabrani kontejner podliježe provjeri sadržaja, odnosno tereta koji se u njemu nalazi. Provjera se obavlja rentgenom

---

6 *Rubber Tyred Gantry Crane/Rail Mounted Gantry Crane*- portalni prijenosnik velikog raspona koji svoje kretanje ostvaruje na gumenim kotačima/na tračnicama.

7 *Automated Guided Vehicles*- su automatski vođena vozila bez posade koja se kreću pomoću automatskog upravljačkog sustava.

8 *Lift Automated Guided Vehicles* – novija generacija automatski vođenih vozila koja osim horizontalnog kretanja imaju sposobnost i vertikalnog podizanja kontejnera te smještaja na posebne okvire smještene u neposrednoj blizini kontejnerskih blokova.

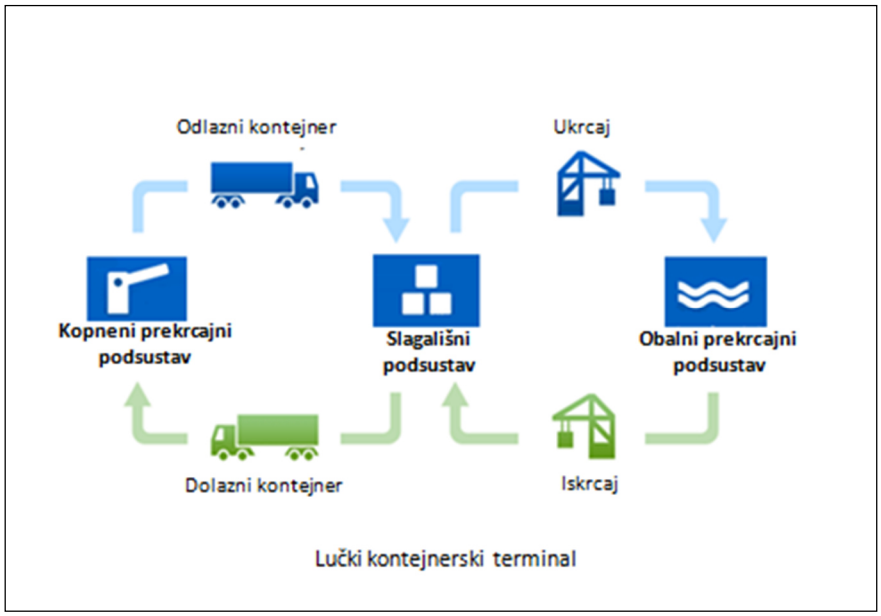
pomoću X-zraka[9] (*X-ray*) skeniranja. Ako skeniranje utvrdi sumnjiv predmet, kontejner se otvara i provodi se fizička inspekcija.

Danas se na terminalu smještaju brojna specijalizirana uslužna poduzeća koja pružaju brojne dodatne usluge (čišćenje, popravak i dr.).

3.2 Struktura i logistički procesi na LKT-u

Danas u svijetu postoje stotine lučkih kontejnerskih terminala koji se razlikuju prema koncepciji, namjeni, lokaciji i veličini, no ipak većina njih ima usporediv raspored svojih podsustava i objekata koji se nalaze u tim podsustavima. Tako Steenken kontejnerski terminal dijeli na tri podsustava (Slika 1):

- obalni
- slagališni i
- kopneni.



Slika 1 Podsustavi lučkog kontejnerskog terminala

Obalni podsustav namijenjen je radnjama ukrcaja i iskrcaja kontejnera s broda/na brod pomoću obalnih kontejnerskih dizalica. Obalni podsustav određen je: duljinom pristana (*berth*), brojem obalnih kontejnerskih dizalica, udaljenosti između operativne obale (*quay*) i slagališnih površina na terminalu (*storageyard ili storage area*)[9].

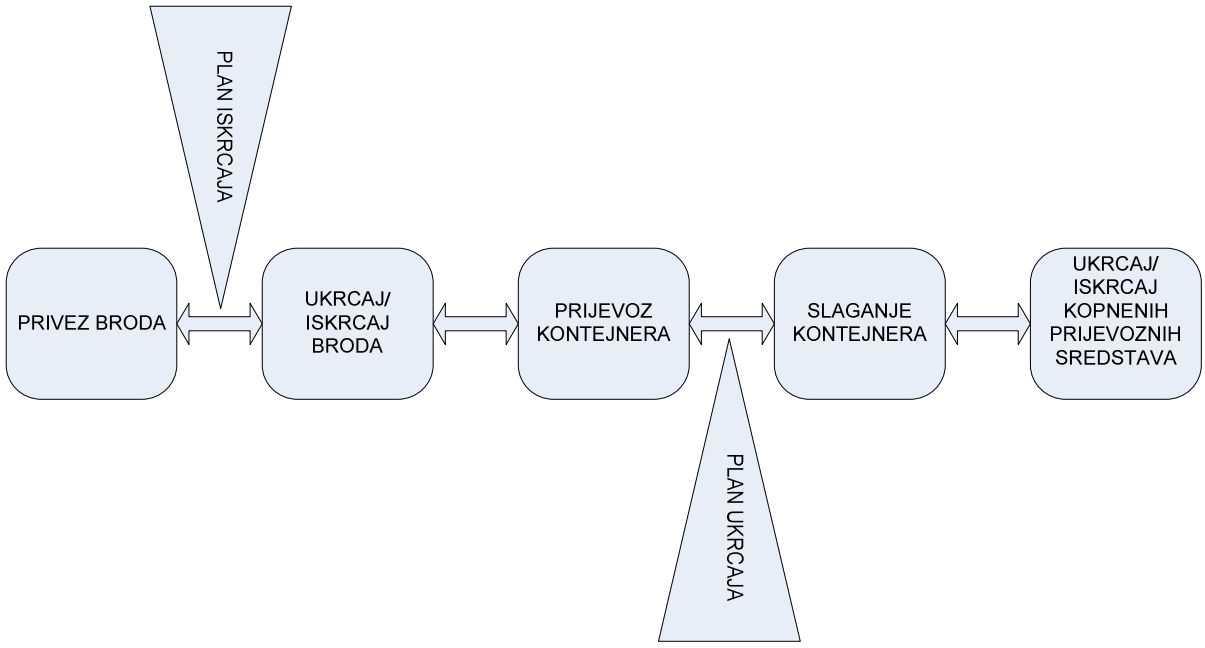
[9] X-zrake ili rentgenske zrake su valni oblik elektromagnetskog zračenja koji prenose fotoni. X-zrake prolaze u unutrašnjost kontejnera, a na kameri se prikazuje uzorak (sadržaj kontejnera) nastao od X-zraka.

Slagališni podsustav namijenjen je privremenom odlaganju, odnosno slaganju kontejnera pomoću slagališnih prekrcajnih sredstava.

Kopneni podsustav čine kopnene prometnice koje su od velikog značaja za povezivanje terminala i njegovog zaleđa te siguran, brz i pouzdan protok kontejnera.

Navedeni podsustavi imaju svoje specifičnosti te se unutar njih odvijaju različite radnje (vidi poglavlje 3.1.) s kontejnerima koje su osnova mnoštva logističkih procesa.

S obzirom na smjer kretanja kontejnera (način otpreme) razlikuju se: dolazni, tranzitni i odlazni kontejneri. Dolazni/uvozni kontejneri (*inbound/import containers)* pristižu brodom u luku te se potom prevoze dalje u unutrašnjost ili zaleđe zemlje. Nakon dolaska broda na terminal, brodu se dodjeljuje vez na pristanu opremljenom obalnim kontejnerskim dizalicama za iskrcaj kontejnera. Iskrcaj kontejnera s broda odvija se prema unaprijed definiranom planu iskrcaja (*unloading cargo plan*). Potom se kontejneri sredstvima unutarnjeg transporta odvoze na slagalište kako bi slagališna dizalica izvršila iskrcaj te uskladištila kontejnere na određeno vrijeme. Zadnja faza logističkog procesa dolaznih kontejnera odnosi se na ponovni ukrcaj kontejnera na kopnena prijevozna sredstva, odnosno njihovu otpremu do odredišta (Slika 2).



Slika 2 Logistički procesi na lučkom kontejnerskom terminalu

Izvor: Vis, I.F.A., de Koster, R., 2003. Transshipment of containers at a container terminal: an overview. European Journal of Operational Research 147, Elsevier

Tranzitni kontejneri, kontejneri namijenjeni provozu (*transshipment containers)* prevoze se na mjesto na terminalu s kojeg će ponovo biti ukrcani za daljnji prijevoz brodom (*feeder servis*[10]). Odlazni/izvozni kontejneri (*outbound/export containers*) na terminal pristižu kamionom ili vlakom te s logističkog aspekta imaju obrnut proces od dolaznih kontejnera[10].

3.3 Optimizacija logističkih procesa na LKT-u

Neprestano povećanje količine kontejnerskog prometa, veličine kontejnerskih brodova, prostorna ograničenja te velika konkurencija, stvaraju pritisak na rukovodstvo terminala kako bi što učinkovitije upravljalo svojim resursima. Da bi upravljanje postojećim resursima terminala bilo što učinkovitije, potrebno je optimizirati logističke procese u pojedinim podsustavima terminala.

Većina logističkih procesa koji se odvijaju na terminalu ne može biti predviđana na duže vremensko razdoblje, već je potrebno donošenje odluka u stvarnom vremenu. Primjerice, iako su podaci o kontejnerima koji pristižu kamionima na terminal unaprijed poznati (EDI [11]sustav), točno vrijeme kada će kontejneri stići na terminal nije poznato. Po dolasku kontejnera na terminal obavlja se kontrolni pregled kontejnera ne bi li se utvrdila oštećenja. Ukoliko se ona utvrde, kontejner se upućuje na popravak. I u ovom slučaju podaci pristigli EDI sustavom mogu se razlikovati, primjerice ako je oštećenje kontejnera uslijedilo kasnije, a podatak unutar sustava nije na vrijeme ažuriran. Navedeno uvelike utječe na planiranje i odvijanje svih logističkih procesa s kontejnerima na LKT-u.

Područja primjene optimizacijskih metoda su logistički procesi u podsustavima terminala, a sastoje se od čimbenika koji utječu na ukupnu produktivnost terminala, a to su[10]:

- planiranje plovila (obalni podsustav),
- logistika skladištenja i slaganja kontejnera (slagališni podsustav) i
- logistika transporta kontejnera (kopneni podsustav)

---

[10] *Feeder servis* je prijevoz kontejnera manjim kontejnerskim brodovima zaduženim za razvoz kontejnera od sabirnog terminala do luke odredišta.

[11] *Electronic Data Interchange*- je elektronički sustav razmjene podataka o kontejnerima, koji koristi specifični strukturirani format i zahtjeva minimalne manualne postupke.

3.3.1 Planiranje plovila

Planiranje plovila uključuje izradu plana: priveza broda, slaganja kontejnera na brod (plan ukrcaja/iskrcaja), raspodjele rada obalnih kontejnerskih dizalica. Prije dolaska broda u luku, brodu je potrebno dodijeliti vez (*Berth allocation problem*- BAP). Idealna dodjela veza brodu trebala bi biti realizirana prije dolaska prvog kontejnera namijenjenog ukrcaju na brod kojem se vez dodjeljuje. U obzir treba uzeti tehničke podatke o brodu (duljinu, gaz) i obalnoj kontejnerskoj dizalici te vrijeme koje će biti potrebno za ukrcaj/iskrcaj broda. Osnovni cilj optimizacije u dodjeli veza brodu je minimizirana ukupna udaljenost od obale do slagališta, što je u podudarnosti s maksimalnom učinkovitosti ukrcajnih/iskrcajnih operacija. Automatska i optimizirana dodjela veza brodu važna je u trenutku isteka planiranog vremena za ukrcaj/iskrcaj broda kada se brodu mora dodijeliti novi vez, kako bi ukrcao/iskrcao preostale kontejnere.

Plan slaganja tereta (*Stowage plan*) je ključan element planiranja plovila i uključuje plan ukrcaja i iskrcaja tereta (kontejnera). Izrada plana slaganja kontejnera u linijskoj plovidbi sastoji se od dva dijela[9] :

- općeg plana- koji sadrži samo informacije o broju kontejnera koje je potrebno ukrcati u pojedinoj odredišnoj luci i rezervaciju kontejnerskih pozicija na brodu,
- detaljnog plana- koji izrađuju planeri na terminalu i sadrži detaljne informacije o obilježjima kontejnera. Značajna obilježja o kontejnerima su: tip kontejnera, duljina kontejnera, luka iskrcaja i težina ili klasa težine kontejnera[10]. Sukladno navedenim obilježjima, kontejneru se dodjeljuje kontejnerska pozicija(*slot*) na brodu, vodeći pri tom računa o stabilnosti broda. Ovaj plan se izrađuje tek nekoliko dana prije dolaska broda u pojedinu luku duž linijske plovidbe.

Cilj optimizacije plana slaganja kontejnera u linijskoj plovidbi je minimizacija broja radnji s kontejnerima i maksimizacija iskorištenja brodskog teretnog prostora. Plan slaganja kontejnera, zajedno s napomenama za slaganje unesenim od posade broda, dostavlja se putem EDI sustava. Napomene se odnose na kontejnere posebnih obilježja, primjerice kontejner s opasnim teretom.

Plan slaganja kontejnera u korelaciji je s kontejnerima na slagalištu. Ukoliko kontejneri na slagalištu nisu složeni sukladno planu slaganja kontejnera te nisu odmah dostupni za otpremu, oni se moraju premještati. Premještaj kontejnera rezultira utroškom vremena otpreme kontejnera na relaciji slagalište-pristan i reduciranjem produktivnosti obalnih kontejnerskih dizalica. Da bi se postigla maksimalna produktivnost obalne kontejnerske dizalice, kontejneri koji pristižu na operativnu obalu trebali bi zadovoljiti dva preduvjeta: pristizat pravovremeno i prema unaprijed utvrđenom rasporedu, sukladno planu ukrcaja. Ukoliko se ne ostvari jedan ili oba preduvjeta, povećava se vrijeme čekanja dizalice (*cranewaitingtime*). Na vrijeme čekanja obalne kontejnerske dizalice utječe i to što su kontejneri na slagalištu složeni sukladno obilježjima te zbog toga imaju različite udaljenosti do pristana, odnosno obalne kontejnerske dizalice. Ukoliko se radi o kontejnerima s rashladnim sustavom (*frigo containers*) neposredno prije same otpreme sa slagališta potrebno ih je isključiti iz strujnog kruga, što dodatno produljuje vrijeme otpreme kontejnera sa slagališta. Navedeno utječe na vrijeme otpreme kontejnera. Isto tako, ukoliko se radi o slagališnim dizalicama i prijevoznim sredstvima s ručno upravljanim sustavom, izvedba dodatno ovisi o vozačevoj sposobnosti i odluci odabira prijevoznog puta. Osim navedenog, mogu se pojaviti određeni tehnički i operativni problemi koji jednako tako utječu na raspored i vrijeme otpreme kontejnera sa slagališta. Dakle, vrijeme otpreme kontejnera sa slagališta nemoguće je točno izračunati čak i ako se radi o uporabi automatiziranih slagališnih i prijevoznih sredstava.

Raspodjela rada obalnih kontejnerskih dizalica(*Quay crane allocation problem*-QCAP) podrazumijeva raspored rada obalnih dizalica na kontejnerskom brodu, sukladno njegovoj duljini i količini kontejnera koju je potrebno prekrcati.

3.3.2 Logistika skladištenja i slaganja kontejnera na LKT-a

Kontejneri koji morem ili kopnom pristižu na terminal, zbog gotovo nemoguće sihronizacije dolazaka prijevoznih sredstava na koje ih je potrebno ukrcati, skladište se na slagalištu terminala. Vrijeme zadržavanja kontejnera na terminalu (*dwell time*) ovisi o smjeru kretanja kontejnera, primjerice ukoliko se radi o tranzitnim kontejnerima njihovo vrijeme zadržavanja je najkraće (na azijskim terminalima od nekoliko sati do maksimalno dva dana), dok je optimalno vrijeme zadržavanja punih kontejnera od 3-7 dana[11]. Konstantnim rastom kontejnerskog prometa slagališne površine postaju

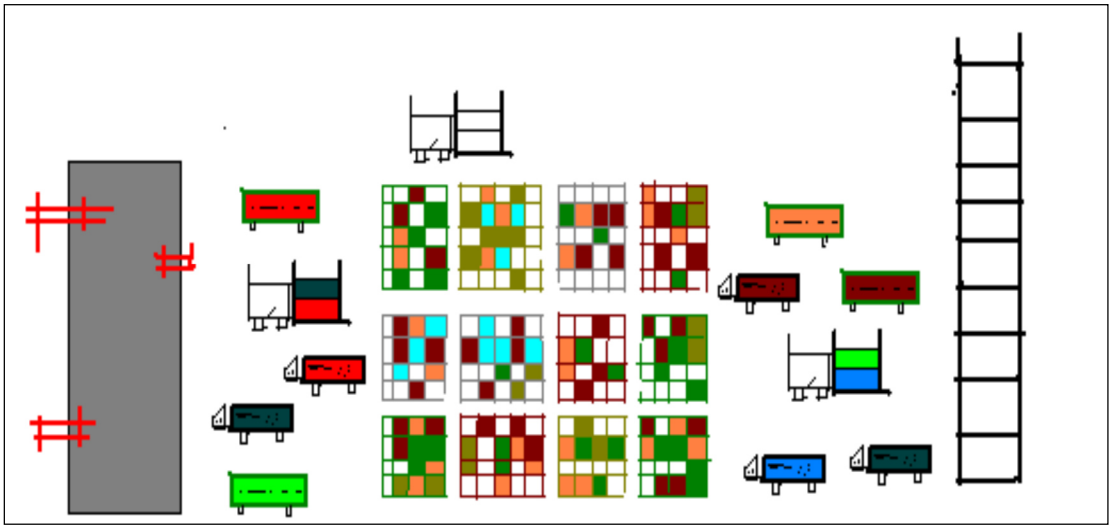
nedostatne, stoga, da bi terminali što racionalnije i učinkovitije iskoristili postojeće slagališne površine, izuzetne napore ulažu u skladišnu i slagališnu logistiku koja se bazira na primjeni optimizacijskih metoda. Zbog nedostatka slagališnog prostora te tehnološkog razvoja slagališnih prekrcajnih sredstava, kontejneri na slagalištu slažu se na sve veće visine. Međutim, navedeno slaganje uzrokom je velikog broja premještaja kontejnera, a razlog tomu je ograničena dostupnost tj; dostupni su samo oni kontejneri koji se nalaze na vrhu kontejnerskog bloka. Te neproduktivne pokrete s kontejnerima nemoguće je u potpunosti izbjeći, budući da je u trenutku slaganja kontejnera na slagalištu većina informacija o budućoj otpremi kontejnera nedostupna ili nedovoljna.

Uzroci premještaja kontejnera ogledaju se kroz : nepotpune i netočne podatke pojedinih kontejnera, netočne podatke o brodu na koji se ukrcava kontejner, netočno navedenoj luci iskrcaja te težini pojedinog kontejnera. Čak i ako su svi relevantni podaci za otpremu kontejnera u potpunosti ispravni, može doći do promjene broda kojim se kontejner otprema ili pak luke iskrcaja kontejnera što zahtjeva dodatni premještaj ciljnog kontejnera. Smatra se da na europskim terminalima 30 – 40% odlaznih kontejnera nema ispravne podatke o brodu na koji se krca i o luci odredišta. Za dolazne kontejnere situacija je još nepovoljnija. Za vrijeme iskrcaja kontejnera s broda, kopneno prijevozno sredstvo poznato je samo u 10 – 15% slučajeva otpreme kontejnere, a za preostale slučajeve način otpreme kontejnera odabire se u trenutku kad je kontejner složen na slagalište[12].

Logistika skladištenja i slaganja kontejnera ima za cilj minimizirati vrijeme ležanja kontejnera na slagalištu i vrijeme otpreme kontejnera sa slagališta. Obuhvaća dva glavna problema: gdje složiti dolazeći kontejner - dodjela kontejnerskih pozicija (*storage space allocation*- SAP) i kako presložiti kontejnere na slagalištu kako bi se izbjegao dodatni premještaj (BRP, PMP, RMP).

3.3.3 Logistika transporta kontejnera na LKT-u

Na kontejnerskom terminalu odvija se horizontalni i vertikalni transport kontejnera. Vertikalni transport se obavlja raznim vrstama dizalica. Horizontalni transport se dijeli prema području rada na podsustav: obalnog horizontalnog transporta (*waterside horizontal transport subsytem*) i kopnenog horizontalnog transporta (l*andside horizontal transport subsytem*)(Slika 3)[13].



Slika 3 Obalni i kopneni horizontalni podsustavi na LKT-u

Izvor: Kemme, N., 2013., Design and Operation of Automated Container Storage Systems, Springer, London

Optimizacija logistike transporta kontejnera podrazumijeva racionalno korištenje kopnenih prijevoznih i prekrcajnih sredstava unutar podsustava lučkoga kontejnerskog terminala.

Podsustav obalnog horizontalnog transporta djeluje kao poveznica između obalnog i slagališnog podsustava. Nakon iskrcaja kontejnera obalnom kontejnerskom dizalicom, kontejneri se sredstvima horizontalnog transporta prevoze od obale do slagališta i obrnuto. Osnovni cilj ovog podsustava je učinkovit, jednostavan i brz prijevoz kontejnera. Kako bi se postavljeni cilj ostvario, potrebno je donijeti odluke o: vrsti, broju, rasporedu i planiranoj ruti kretanja prijevoznog sredstva.

Prijevoz kontejnera na relaciji obala-slagalište (i obrnuto) moguće je obavljati raznim prijevoznim sredstvima različite nosivosti, fleksibilnosti, brzine, stupnja automatizacije i dr. Sredstva koja se koriste na navedenoj relaciji razlikuju se od terminala do terminala, no općenito postoje četiri vrste, a to su redom: SC, AGV, L-AGV, TTU (*truck trailer unit*- tegljač s poluprikolicom), MTS (*multi trailer system*- tegljač s više prikolica)[9]. Sredstva horizontalnog transportnog podsustava kategoriziraju se kao: pasivna i aktivna. Osnovna razlika je ta da aktivna sredstva osim obavljanja prijevoza imaju mogućnost i prekrcaja tereta (prijevozno-prekrcajna). Od navedenih vrsta jedino SC pripada prijevozno-prekrcajnim sredstvima koja imaju mogućnost slaganja kontejnera na nekoliko visina unutar kontejnerskog bloka. Uz SC potrebno je istaknuti i L-AGV koji

imaju mogućnost vertikalnog podizanja i odlaganja kontejnera na posebne okvire u neposrednoj blizini slagališta, no nisu u mogućnosti slagati kontejnere unutar kontejnerskog bloka. Ukoliko prijevoz obavljaju pasivna horizontalna sredstva tada je za prekrcaj kontejnera potrebno angažirati dodatno prekrcajno sredstvo, što produljuje vrijeme manipulacije s kontejnerom te umanjuje produktivnost terminala.
Otprema kontejnera sa slagališta odvija se prema unaprijed utvrđenom redoslijedu te je sukladna planu slaganja kontejnera na brod. Osim redoslijeda otpreme kontejnera važno je odrediti i vrijeme otpreme kako bi se postigla sinkronizacija horizontalnih transportnih sredstava i obalne kontejnerske dizalice te time onemogućila zagušenja na pristanu uzrokovana horizontalnim transportnim sredstvima.
Optimizacija obalnog horizontalnog transporta odnosi se na skraćivanje transportnog puta i sinhronizaciju sredstava horizontalnog transporta s prekrcajnim radnjama obalne kontejnerske dizalice. Cilj optimizacije je povećati produktivnost obalne kontejnerske dizalice. Produktivnost dizalice ne ovisi samo o radnom učinku, već i o neproduktivnim vremenima kao što su stanke tijekom smjene, zagušenja u horizontalnom transportu te ostale tehničke i operativne smetnje.

Podsustav kopnenog horizontalnog transporta od velike je važnosti za konkurentnost pojedinog LKT-a. Ne postojanje odgovarajuće, brze i pouzdane prometne infrastrukture zaleđa uvelike utječe na ukupni promet i uspješnost poslovanja lučkog kontejnerskog terminala. Povezivanje terminala sa zaleđem moguće je cestom, željeznicom i unutarnjim plovnim putovima.
Kontejneri koji „vanjskim“ kamionima (*external truck*- XT) pristižu na terminal podliježu pregledu koji se obavlja na ulazu (*gate in*), a obuhvaća kontrolu: težine, plombe i oštećenja kontejnera, a potom se nakon obavljenog pregleda podaci unose u EDI sustav[14]. Ukoliko je sve ispravno, vozač XT-a dobiva dozvolu ulaska na terminal te detaljne upute o lokaciji na koju mora prevesti kontejner. Nakon dolaska na naznačeno mjesto, vozač XT-a čeka na iskrcaj kontejnera. Vrijeme dolaska XT-a na slagalište nije moguće točno odrediti, a razlog tome je taj što se i nakon vremena dolaska na terminal ne može predvidjeti vrijeme koje će biti potrebno za pregled kontejnera i dokumentacije. Ukoliko dokumentacija nije ispravna, vozač XT-a se prosljeđuje administrativnoj službi, što dodatno produljuje vrijeme dolaska na slagalište. Iskrcaj kontejnera obavlja se slagališnom dizalicom. Što je XT bliže dizalici to će kontejner prije

biti iskrcan. Nakon iskrcaja kontejnera s XT-a, XT napušta slagališni podsustav voznom trakom namijenjenom za brzi izlaz, budući da je prazan pa nema pregleda kontejnera. Potom dolazi na izlaz (*gate out*), gdje službenik terminala provjerava je li to doista XT koji je i ušao na terminal, ukoliko je sve uredu XT napušta terminal.

Ukoliko na terminal pristiže prazan XT, on također podliježe sigurnosnoj kontroli nakon koje vozač podnosi zahtjev za ukrcaj kontejnera na XT. Nakon utvrđene ispravnosti dokumentacije, podaci o kontejneru koji se ukrcava na XT unose se u EDI sustav, a vozač XT se upućuje na mjesto ukrcaja dolaznog kontejnera. Po dolasku na površine slagališta dolaznih kontejnera, kamion se zaustavlja u voznoj traci predviđenoj za ukrcaj kontejnera te čeka ukrcaj kontejnera dizalicom. Što je XT dizalici to će kontejner prije biti ukrcan. Nakon pozicioniranja kontejnera na XT, vozač XT-a nastavlja prema voznoj traci namijenjenoj izlazu. Po izlasku, obavlja se nadzor kontejnera na XT-u te, ukoliko je sve ispravno, vozač predaje kopiju zahtjeva za ukrcaj kontejnera i napušta terminal. Cilj optimizacije kamionskog prijevoza je minimizacija praznog hoda i vremena prijevoznog procesa. Prema Steenkenu minimizaciju praznog hoda, moguće je postići kombiniranjem prijevoza odlaznih kontejnera s dolaznim kontejnerima, što znači da sredstvo koje prevozi odlazni kontejner, nakon njegova iskrcaja ukrcava dolazni kontejner. Dakle, s ovakvim scenarijem prazan hod kamiona bio bi u potpunosti onemogućen.

Većina europskih lučkih kontejnerskih terminala, uz cestu, svoje veze sa zaleđem ostvaruje putem željeznice. Ti kontejnerski terminali posjeduju vlastitu željezničku postaju na kojoj se obavlja ukrcaj i iskrcaj kontejnera na/s vagona. Kod ukrcaja kontejnera na vagone, potrebno je barem 24 sata ranije, od strane željezničkog operatera, najaviti količinu i vrijeme ukrcaja kontejnera, kako bi se na vrijeme isplaniralo ljudstvo i prekrcajna sredstva potrebna za ukrcaj. Najmanje 8 sati prije ukrcaja željeznički operater šalje ukrcajni manifest koji sadrži: brojeve kontejnera, PIN[12]-ove te brojeve vagona. Istovremeno se šalje EDI poruka za ukrcaj kontejnera na vagone, a sadrži: plan ukrcaja koji informira operatere na terminalu na koji vagon ide koji kontejner, sekvence vagona, broj putovanja i odredište vagona. Ukoliko je u mogućnosti željeznički operater šalje najavu ukrcaja kontejnera na vagone, prije nego su iskrcani sa broda[15]. Ukrcajni plan može biti izrađen od strane planera na željeznici i terminalu. Cilj optimizacije željezničkog prijevoza je umanjiti manevarske aktivnosti tijekom željezničkog prijevoza, dok je cilj planera na terminalu minimizirati broj

[12] *Personal Identification Number* - Osobni identifikacijski broj kontejnera (npr: 6633, 502840918i)

premještanja kontejnera na slagalištu kako bi se umanjilo vrijeme manipulacije kontejnera slagališnom dizalicom te duljina puta praznog hoda slagališne dizalice[10].
Prije iskrcaja kontejnera s vagona željeznički operateri šalju EDI poruku koja sadrži sve potrebne podatke kako bi željeznički planer mogao isplanirati iskrcaj. EDI poruka mora sadržavati sljedeće podatke: broj kontejnera, status kontejnera (pun/prazan), ISO tip, težinu, luku iskrcaja, broj vagona, vrstu vagona, sekvencu i nosivost vagona[15].

3.4 Primjena informacijskih tehnologija na lučkom kontejnerskom terminalu

Kontinuirano nastojanje rukovodstva kontejnerskih terminala da se smanjenjem troškova usluga poveća konkurentnost terminala, dovelo je do razvoja i primjene informacijskih tehnologija (*Information technology* - IT) u procesu poslovanja lučkog kontejnerskog terminala. Smatralo se da će se njihovim uvođenjem postići učinkovita koordinacija, organizacija i planiranje tehnoloških procesa na lučkim kontejnerskim terminalima. IT na kontejnerskim terminalima su tehnologije koje nadziru i upravljaju resursima terminala te omogućavaju protok informacija[16]. Početak razvoja tih tehnologija započinje 70-ih godina prošlog stoljeća, uvođenjem sustava za planiranje prekrcajnih radnji (*Terminal Operating System - TOS*) na lučkim kontejnerskim terminalima.
Sustav TOS pruža podatke o : kontejneru (veličina, tip, sadržaj itd.), resursima (zauzetost slagališnim površinama, lokaciji kontejnera i prijevozno prekrcajnim sredstvima na terminalu), ograničenjima (značajke pristana i prekrcajnih sredstava) i tehnološkim procesima (optimalno slaganje kontejnera)[17]. Detaljni i točni podaci o stanju na terminalu imaju važnu ulogu u optimizacije procesa na lučkim kontejnerskim terminalima.
Sustav za prekrcaj i praćenje kontejnera temeljen na optičkom sustavu čitanja tagova (*Optical Character Recognition* – OCR), diferencijalni globalni pozicijski sustav (*Differential Global PositioningSystem* - DGPS) te sustav za radio frekvencijsku identifikacija (*Radio Frequency Identification* – RFID) koriste se za identifikaciju kamiona i kontejnera na ulazu u terminal i za praćenje kretanja po terminalu. OCR sustav pretvara skenirane slike teksta u strojno-kodirani tekst, a koristi se za čitanje identifikacijske pločice kontejnera. Također, taj sustav instaliran je na ulazu u terminal (identificira kamione koji ulaze na terminal) i na prekrcajnim sredstvima (provjerava se da li se prekrcava ispravan kontejner). DGPS je poboljšani globalni pozicijski sustav

(Global PositioningSystem – GPS) koji pruža veću točnost pozicije, nego GPS. Kontejnerski terminali koriste DGPS kako bi utvrdili poziciju kontejnera na terminalu, a postavljaju ih na prijevozna i prekrcajna sredstva. Pozicija kontejnera mjeri se i memorira svaki puta kada se sa kontejnerom izvodi neka radnja. RFID sustav sastoji se od oznake/etikete (*Tag*) i čitača. Oznake predstavljaju elektronički čipovi s kodiranim podacima kojima se može pristupiti pomoću čitača. RFID sustav se koristi za utvrđivanje lokacije pojedinog kontejnera na terminalu. Primjerice, u luci Rotterdam RFID čitači nalaze se na obalnim i slagališnim kontejnerskim dizalicama s ciljem praćenja dopreme i otpreme kontejnera. Osnovna prednost uporabe ovog sustava je prikupljanje podataka o kontejnerima u stvarnom vremenu, što dovodi do učinkovitijeg upravljanja tehnološkim procesima na lučkim kontejnerskim terminalima.

Kontejnerski terminali implementacijom IT povećavaju sigurnost rada i tereta te korisnicima osiguravaju kvalitetniju uslugu i bržu manipulaciju kontejnerima. No ipak, najvažnije prednost uporabe IT je optimizirano odlučivanje na lučkim kontejnerskim terminalima koje rezultira povećanju konkurentnosti terminala na tržištu lučkih usluga.

4. SLAGALIŠTE KAO PODSUSTAV LKT-a

4.1 Funkcija slagališta

Slagalište je jedan od podsustava kontejnerskog terminala koji čine specijalizirane površine smještene na kopnenoj strani terminala koje služe za privremen smještaj kontejnera koji čekaju daljnju otpremu kamionom, vlakom ili brodom. Slagalište se nastavlja na pristan pri čemu je veličina određena raspoloživim prostorom, propusnom moći terminala, koeficijentom obrta na slagalištu te načinima slaganja kontejnera. Smatra se najvažnijim podsustavom za funkcionalnost terminala, a tome u prilog ide činjenica da većina svjetskih kontejnerskih terminala koristi više od 75% ukupne površine kao slagalište. S logističkog stajališta slagalište je ključan čvor ili točka u logističkoj mreži u kojoj se kontejner prihvaća ili prosljeđuje u nekom drugom smjeru unutar logističke mreže. Stoga je pri projektiranju slagališta potrebno posebnu pozornost usmjeriti na nosivost tla, ukupnu i korisnu površinu te propusnu moć slagališta. Glavne tehničke funkcije slagališta su slaganje (skladištenje) i distribucija kontejnera s ciljem prostornog i vremenskog uravnoteženja tokova kontejnera.
Neposredno prije skladištenja kontejnera na slagalište potrebno je donijeti mnogobrojne odluke i izvršiti niz radnji, a to su redom:

- Prostorna raspodjela kontejnera (dolazni, odlazni, tranzitni, prazni)
- Segregacija (sortiranje, razvrstavanje) kontejnera ovisno o njihovim obilježjima (veličini, težini, brodu, odredišnoj luci, brodaru, tipu kontejnera, frigo, opasni teret) itd.
- Doprema kontejnera
- Slaganje kontejnera
- Otprema kontejnera
- Kontrola prometovanja prijevoznih sredstava na slagalištu terminala[18].

Sve navedene radnje međusobno su zavisne jedna o drugoj i imaju veliki utjecaj na produktivnost rada slagališta koja je od velike važnosti za cjelokupan rad LKT-a.

4.2 Konfiguracija slagališta

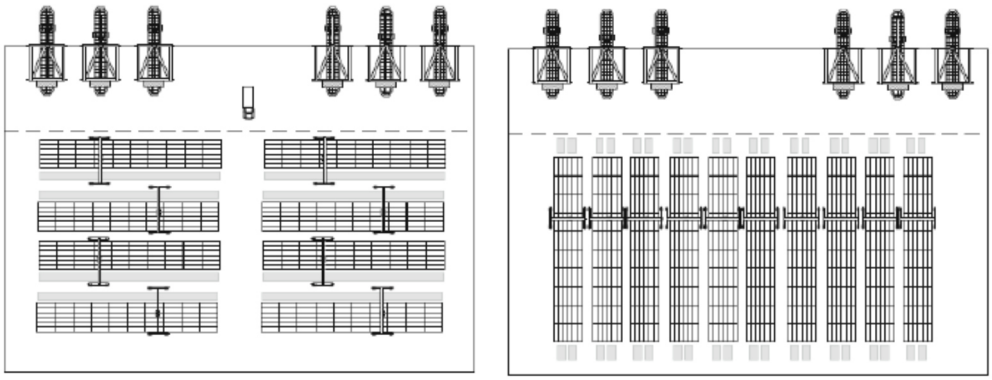
U svjetskim kontejnerskim lukama izdvajaju se dva načina slaganja kontejnera: direktno na tlo i na prikolicu. Sustav slaganja kontejnera na tlo naziva se „blok sustav“. Kod načina slaganja kontejnera na tlo, slagalište može biti postavljeno: paralelno(horizontalno) s brodom tj. linijom pristana i okomito(vertikalno) na brod odnosno liniju pristana(Slika 4)[9]. Konfiguracija slagališta zavisi o: regiji u kojoj se terminal nalazi, prometu, morfologiji terena, zahtjevima prijevoznih tvrtki te slagališnim sredstvima koja će se koristiti u radu terminala. Primjerice, mnogi automatizirani i poluatomatizirani terminali u sjevernoj Europi imaju okomito položeno slagalište na pristan, a razlog tome je jednostavna kontrola prometa na terminalu. Kontejnerski terminali diljem istočne Azije primjenjuju paralelan način slaganja kontejnera u odnosu na pristan. Primjerice, ukoliko se za rad na slagalištu koriste RTG dizalice, u 90% slučajeva slagalište će biti postavljeno paralelno u odnosu na pristan[19].

U nastavku sažeto su navedene glavne značajke pojedinih konfiguracija slagališta.

1. **Paralelni (horizontalni) izgled slagališta**
    - Kontejnerski blokovi (KB) smješteni su paralelno s linijom pristana.
    - Slagališna dizalica može se kretati od jednog do drugog KB-a.
    - KB za dolazne i odlazne kontejnere smještene su zasebno.
    - Prometne trake za kretanje IT i XT smještene su duž blokova.

2. **Okomiti (vertikalni) izgled slagališta**
    - KB smješteni su okomito s linijom pristana.
    - Broj slagališnih dizalica po KB-u je fiksan.
    - IT i XT ne mogu pristupiti slagališnim površinama, već se prekrcaj kontejnera vrši na kraju svakog bloka u tzv. točkama transfera (*transfer point areas*).
    - Dolazni i odlazni kontejneri slažu se zajedno u isti KB. Generalno, odjeljci smješteni u blizini obale namijenjeni su slaganju odlaznih kontejnera, dok su odjeljci smješteni bliže kopnu namijenjeni slaganju dolaznih kontejnera[20].

Svaki od načina slaganja kontejnera ima svoje prednosti i nedostatke. Tako je prednost slaganja kontejnera u „blok sustav“ u tome što zauzima manje prostora pa se primjenjuje na terminalima s manjim slagališnim površinama i velikim kontejnerskim prometom (pretežito na europskim i azijskim terminalima), dok je nedostatak nemogućnost direktnog pristupa svakom složenom kontejneru. Iz prethodno navedenog može se zaključiti da je prednost slaganja kontejnera na prikolicu direktan pristup svakom kontejneru u bilo kojem trenutku te minimalan broj manipulacija s kontejnerima, dok je nedostatak taj što zahtjeva velike slagališne površine pa se ovaj način slaganja primjenjuje na velikim kontejnerskim terminalima (najčešće na terminalima Sjeverne Amerike)(
Slika 5).



Slika 4 Izgled slagališta kontejnerskog terminala primjenom slaganja kontejnera na tlo

Izvor: Böse, J.W.,2011., Handbook of Terminal Planning, Springer, London

Slika 5 Slaganje kontejnera na prikolicu na slagalištu kontejnerskog terminala Norfolk

Izvor: Solmenikovs, A., 2006., Simulation Modelling and Research of Marine Container Terminal Logistics Chains, Riga

Površine kontejnerskog slagališta podijeljene su, ovisno o vrsti kontejnera, odnosno tereta koji se u njemu nalazi. U skladu s navedenim razlikuju se sljedeće specijalizirane površine za smještaj: frigo kontejnera, kontejnera s opasnim teretom klasificiranom prema IMDG kodu, punih kontejnera i praznih kontejnera. S obzirom na način otpreme kontejnera tj. smjer kretanja kontejnera (more i kopno) razlikuju se: dolazni (uvozni) i odlazni (izvozni) kontejneri te u skladu s time površine za smještaj dolaznih i odlaznih kontejnera.

Slaganje kontejnera obavlja se prema sljedećim kriterijima:

- luka iskrcaja (odredišna luka)
- vrsta tereta koji se nalazi u kontejneru
- vlasnik kontejnera (Maersk, Evergreen, Hanjin, APL, itd.)
- veličina i tip kontejnera (20' ili 40' i dr., frigo kontejner itd.)[8].

S obzirom na vrijeme zadržavanja kontejnera na slagalištu i tip kontejnera razlikuju se tri vrste slagališta:

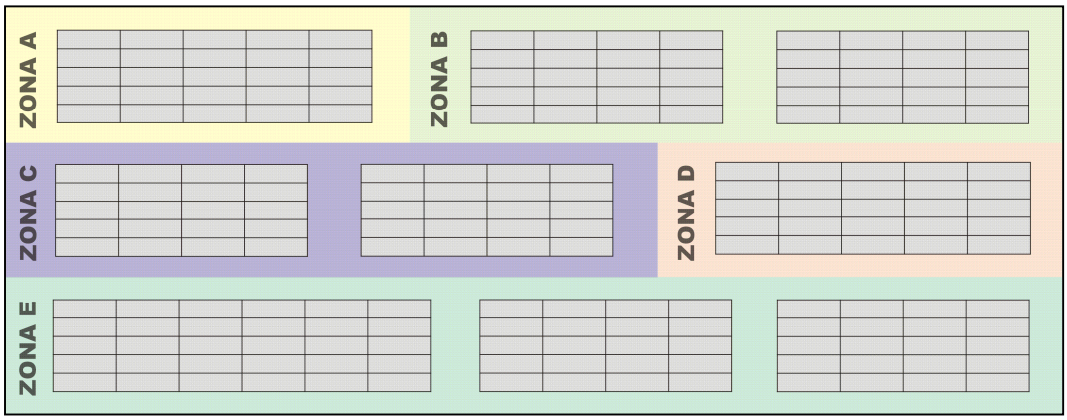
- kratkoročna
- dugoročna i
- specijalizirana.

Kratkoročna (*short term*) slagališta namijenjena su tranzitnim kontejnerima koji su na terminal pristigli brodom, a koji će u kratkom vremenskom razdoblju biti ukrcani na drugi brod.

Dugoročna (*long term*) slagališta namijenjena su kontejnerima koji čekaju carinsko oslobođenje ili nadzor.

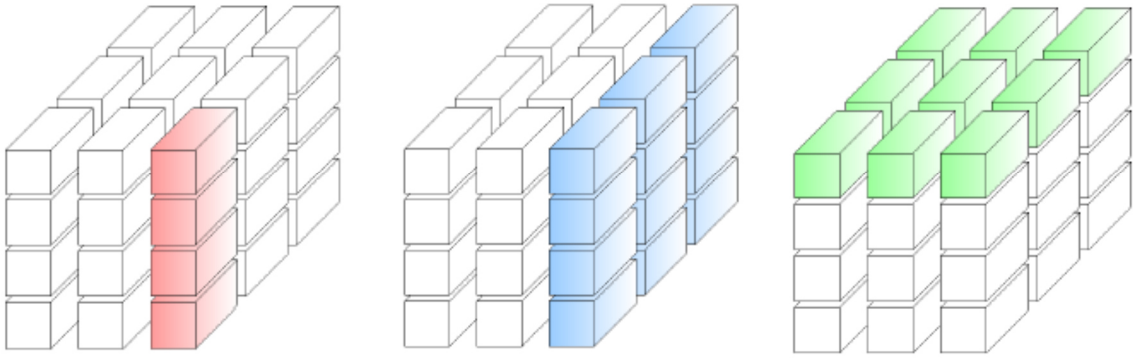
Specijalizirana slagališta rezervirana su za sljedeće tipove kontejnera: frigo, prazni, nestandardni teret (teret koji je viši od 2,36 m pa se pakira u *open top* kontejnere)(*out of gauge containers*), tekući rasuti teret i opasni teret[8].

Slagališne površine (*stacking area ili storage yard*) podijeljene su na zone u kojima su u istom redu smješteni jedan ili više kontejnerskih blokova (*container blocks ili yard blocks*)(Slika 6).



Slika 6 Slagalište KT podijeljeno na zone

Kontejnerski blok (KB) sastoji se od tri elementa: odjeljak (bay), stupac (*stack*)) i red (po visini) (*tier*). Lokacija kontejnera na slagalištu određena je: slovom ili brojem zone, brojem KB-a, odjeljkom, stupcem i redom po visini slaganja[21]. Kontejneri se slažu jedan povrh drugog tvoreći stupac. Slaganje stupaca u red jedan do drugog predstavlja širinu (*width*) KB-a. Stupac je određen brojem redova koji predstavljaju visinu (*height*) slaganja kontejnera. Slaganje stupaca u red (jedan iza drugog) naziva se odjeljkom, a predstavlja duljinu (*length*) KB-a (Slika 7)[22]. Iz navedenog razvidno je da se KB sastoji od tri elementa određena s tri dimenzije.



Slika 7 Sastavni elementi kontejnerskog bloka

Izvor: Tus, A., 2014, Heuristic Solution Approaches for Two Dimensional Pre-marshalling Problem, Wien

Tipični KB obično se sastoji od 6 do 8 odjeljaka u kojima se na tlu nalazi 40 do 60 kontejnerskih pozicija(*slots*)[13][21]. Površina kontejnerskih pozicija (Kp) ovisi o tehnologiji rada na slagalištu, primjerice ukoliko se za rad koriste RTG/RMG dizalice

[13] Dimenzije *slot*-a jednake su dimenzijama 20' kontejnera. Ukoliko se slaže 40' kontejner tada on zauzima dva stupca u istom redu bloka.

jedinična površina bit će manja, nego da se za rad koriste autodizalice/viličari koji zahtijevaju više prostora za manipulaciju s kontejnerima, a najčešće iznosi od 15 – 20m^2/TEU[23]. Broj Kp dobije se dijeljenjem ukupne površine slagališta s površinom Kp-a. Slaganje stupaca u red jedan do drugog predstavlja širinu KB-a, ukoliko KB ima 6 redova, 5 redova namijenjeno je slaganju kontejnera, a 1 red služi kao prometna traka za kretanje kamiona koji su u interakciji sa slagališnom dizalicom (navedeno ne vrijedi za autodizalice i portalne prijenosnike malog raspona)(Slika 8).

Slika 8 Kontejnerski blok slagališta opremljen RTG dizalicom

Izvor: www.konecranes.com (siječanj 2015.)

Preostale vozne traka smještene su na prostoru između KB-a. Također, kontejneri u KB-u slažu se u 5 do 7 redova po visini (*tiers*), ovisno o mogućoj visini slaganja slagališne dizalice. Broj blokova ovisi o kapacitetu terminala tj. o njegovoj veličini i slobodnom slagališnom prostoru.

4.3 Osnovni parametri vrednovanja rada slagališta

Osnovi parametri vrednovanja rada su: statički i dinamički kapacitet (*static and dynamic capacity*) slagališta i slagališnih dizalica , gustoća slaganja kontejnera (*stacking density*), vrijeme zadržavanja kontejnera na slagalištu (*dwell time*). Navedeni parametri zavise o: veličini slagališnih površina, o tehnologija rada (tehničke značajke slagališnog sredstva),

organizaciji rada (informacijski sustavi, programi za primjenu optimizacijskih metoda i dr.) i stručnosti kadra (slagališnih operatera i planera).

Statički i dinamički kapacitet predstavljaju sposobnost prihvaćanja određene količine kontejnera na slagalištu, s time da se statički kapacitet izražava jednokratno, dok se dinamički kapacitet izražava u određenoj vremenskoj jedinici . Potrebno je napomenuti da se s obzirom na organizaciju i uvjete rada razlikuju teorijski (najpovoljniji radni uvjeti) i stvarni (realni) kapacitet (odnosi se na realne uvjete rada - zastoji u radu i dr.).

Statički teorijski kapacitet slagališta je određen veličinom korisnih površina za jednokratni smještaj kontejnera te podrazumijeva popunjenost slagališta od 100%, a izračunava se prema izrazu (niti jedan od narednih izraze ne vrijedi za slaganje kontejnera na prikolice)[24]:

$$STK_S = n_{kp} \times n_{rv}[TEU] \tag{1}$$

gdje su:

STK_S- statički teorijski kapacitet slagališta

n_{kp}- broj kontejnerskih pozicija za smještaj 20' kontejnera

n_{rv}– prosječan broj redova po visini za smještaj 20' kontejnera (u obzir se uzima tip kontejnera koji se slaže, prazni ili puni te se uzima prosječna visina slaganja)

Statički stvarni kapacitet slagališta dobiva se prema izrazu[24]:

$$SRK_S = STK_S \times k_{ps}[TEU] \tag{2}$$

gdje je SRK_S - statički stvarni kapacitet slagališta $[TEU]$, ak_{ps}- koeficijent popunjenosti slagališta s vrijednostima [0,7; 0,8] (ako udio popunjenosti slagališta dostiže 70% ukupne korisne površine dolazi do zagušenja na slagalištu – odnosno povećava se broj premještaja kontejnera)

Maksimalni statički kapacitet je postignut u trenutku kada više nema dodatnog prostora za širenje.

Dinamički teorijski kapacitet slagališta na godišnjoj bazi dobiva se formulom[24]:

$$DTK_S = \left(n_{kp} \times n_{rv} \times t_r\right) \div t_z[TEU/god] \tag{3}$$

gdje su:

DTK_S– dinamički teorijski kapacitet slagališta

t_r- broj radnih dana u godini

t_z- prosječno vrijeme zadržavanja TEU kontejnera.

Dinamički stvarni kapacitet slagališta izračunava se isto kao i statički stvarni kapacitet slagališta samo se u formulu[24]:

$$DRK_S = DTK_S \times k_{ps}[TEU/god] \quad (4)$$

uvrštavaju vrijednosti dinamičkog teorijskog kapaciteta i prosječne vrijednosti koeficijenta popunjenosti slagališta, budući da se u obzir uzima određeno razdoblje promatranja.

Dinamički teorijski kapacitet slagališnih dizalica odnosi se na idealne uvjete rada (maksimalna proizvodnost dizalice, maksimalni broj radnih dana), a dobiva se temeljem izraza[24]:

$$DTK_d = n_d \times p_d \times t_{dt} \times t_r[TEU/god] \quad (5)$$

gdje su:

DTK_d - dinamički teorijski kapacitet dizalica

n_d - broj slagališnih dizalica

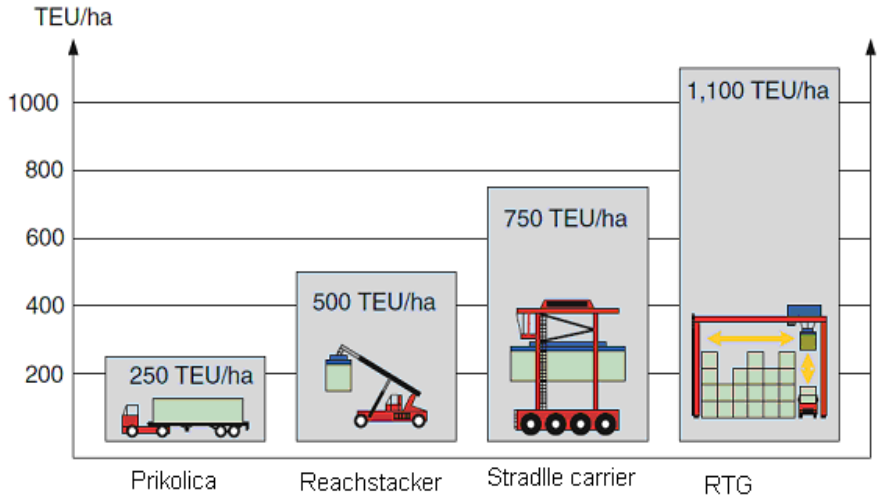
p_d – proizvodnost dizalice tj; broj prekrcanih TEU kontejnera po dizalici u 1 satu

t_{dt}– dnevno radno vrijeme terminala.

Dinamički stvarni kapacitet slagališnih dizalica(DRK_d) izračunava se isto kao dinamički teorijski kapacitet slagališnih dizalica, samo se koriste stvarni parametri rada. Primjerice, uzima se prosječna proizvodnost dizalica jer se podrazumijeva da sve slagališne dizalice nemaju istu proizvodnost i da ona nikako nije maksimizirana te realni broj radnih dana u godini.

Budući da je na većini svjetskih kontejnerskih terminala slagališni prostor postao nedovoljan, prisutna je tendencija slaganja kontejnera na velike visine, odnosno uporaba slagališnih dizalica sa velikom gustoćom slaganja. **Gustoća slaganja** kontejnera dobiva se dijeljenjem broja TEU kontejnera s veličinom slagališnog prostora koji zauzima[ha], a ovisni o vrsti tj, tehničkim značajkama slagališnog sredstva(Grafikon 5)[9]. U tom smislu, ukoliko se kontejneri na slagalištu slažu na kamionske prikolice, tada je gustoća slaganja najmanja, a iznosi 250 TEU/ha. Prednost slaganja kontejnera na prikolicu

ogleda se u direktnoj otpremi kontejnera, dok je nedostatak taj što se ovaj način slaganja može primjenjivati samo na velikim slagališnim površinama. Danas, mnoge moderne slagališne dizalice imaju veliku gustoću slaganja (350 -1.100 TEU) što doprinosi povećanju kapacitet slagališta. Iz grafikona je vidljivo da autodizalice imaju najmanju gustoću slaganja koja iznosi 250 TEU/ha, dok RTG dizalice imaju najveću gustoću slaganja od 1.100 TEU/ha. Međutim, velika gustoća slaganja kontejnera često puta rezultirati suvišnim i neproduktivnim manipulacijama s kontejnerima, posebice u slučaju kad raspored slaganja kontejnera nije u skladu s rasporedom otpreme kontejnera sa slagališta[12]. Stoga je potrebno voditi računa da broj neproduktivnih manipulacija s kontejnerima bude minimalna da ne dođe do zagušenja na slagalištu koje rezultira zastojima u radu terminala.



Grafikon 5 Gustoća slaganja kontejnera ovisno o vrsti slagališnih dizalica
(temeljeno na specifikacijama tvrtke *Kalmar*)
Izvor: Böse, J. W., 2011. Handbook of Terminal Planning, Springer, London

Također, vrlo važan parametar vrednovanja rada slagališta koji izuzetno utječe na produktivnost kontejnerskog terminala je **vrijeme zadržavanja kontejnera** na slagalištu. Vrijeme zadržavanja kontejnera na slagalištu je velika prepreka u radu terminala, budući da niti jedan terminal nema za cilj slagališne površine pretvoriti u skladište. Stoga rukovodstvo terminala nastoji postići optimalno moguće vrijeme zadržavanja kontejnera na slagalištu koje će u najmanjoj mogućoj mjeri utjecati na troškove premještaja kontejnera i troškove amortizacije skupe slagališne opreme. Od velike je važnosti da se skladištenje kontejnera, koje se pretežito odnosi na prazne kontejnere, izbalansira s privremenim slaganjem punih kontejnera na slagalištu. Prema

navedenom, vrijeme zadržavanja punih kontejnera na slagalištu trebalo bi se odnositi na period od 3-7 dana jer se taj period smatra optimalnim, dok vrijeme zadržavanja praznih kontejnera iznosi između 15-20 dana. Ukoliko je vrijeme zadržavanja kontejnera dulje od navedenog, tada se javlja potreba za povećanjem kapaciteta, u suprotnom terminal postaje manje konkurentan [11].

4.4 Izbor, vrste i tehničke značajke prijevozno-prekrcajnih sredstva za rad na slagalištu

Proces izbora prekrcajnih sredstava koja će se koristiti u radu kontejnerskog terminala je vrlo kompleksan, prvenstveno zbog velikog broja faktora utjecaja koji se pritom moraju uzeti u obzir.

Autor Roach te faktore kategorizira u 5 glavnih skupina:

- razvojni faktori utjecaja
- faktori eksploatacije prekrcajnih sredstava
- faktori održavanja prekrcajnih sredstava
- zahtjevnost rukovanja prekrcajnim sredstvima
- tehnološko- operativni faktori.

Svaka od navedenih skupina faktora sastoji se od više elemenata. Skupina razvojnih faktora utjecaja odnosi se na: iskorištenje raspoloživih razvojnih i slagališnih površina, pouzdanost rada pri povećanju stupnja efektivnog iskorištenja, opterećenje po jedinici površine, mogućnost komplementarnog korištenja s drugim prekrcajnim sredstvima. Skupina faktora eksploatacije prekrcajnih sredstava obuhvaća: troškove po satu rada i troškove goriva prekrcajnog sredstva. Skupina faktora održavanja obuhvaća: operativnu raspoloživosti, vrijeme između dva kvara, troškove održavanja, pogodnost za održavanje, kvalificirano radno osoblje za održavanje i opremu za održavanje. Zahtjevnost rukovanja prekrcajnim sredstvom odnosi se na broj rukovatelja sredstvom u smjeni i na potrebnu kvalifikaciju rukovatelja. Skupina tehnološko- operativnih faktora obuhvaća: tehnološke zahtjeve koji se javljaju u procesu manipulacije teretom, stupanj fleksibilnosti sredstva, brzinu kretanja sredstva pod punim opterećenjem, brzinu kretanja sredstva bez tereta, brzinu dizanja i spuštanja tereta i stupanj zadovoljenja sigurnosnih uvjeta.

Osim navedenih faktora, prilikom odabira slagališnih sredstava, treba uzeti u obzir postojeće i očekivane zahtjeve korisnika usluge, postojeći i očekivani promet u narednim razdobljima te tehnološka unapređenja rada.
Nekoliko je vrsta prijevozno- prekrcajnih sredstava koja se mogu koristiti za rad na slagalištu kontejnerskog terminala, a to su redom: RTG, RMG , SC, a u posljednje vrijeme sve se više primjenjuju ASC dizalice. Uz spomenute, na nekim slagalištima kontejnerskih terminala još se koriste viličari i autodizalice.

Viličari (*forklifts*) su specijalna transportno-manipulativna sredstva s ugrađenom vilicom za rad[25]. Viličari su prvenstveno zbog niske cijene, odnosno malog početnog ulaganja, ekonomično rješenje za rad na višenamjenskim i malim kontejnerskim terminalima (Slika 9). Njihova fleksibilnost omogućava im izvođenje svih operacija s kontejnerima na bilo kojem dijelu terminala. Na velikim kontejnerskim terminalima viličari se koriste za rukovanje praznim kontejnerima. Nedostaci uporabe viličara su: mala gustoća slaganja kontejnera te veliki radni prostor. Moderni viličari opremljeni su posebnim hvatanima za kontejnere (*spreader*) te mogu slagati i do 8 redova u visinu.

Slika 9 Viličar

Izvor: Mohseni, N. S., 2011., Developing a Tool for Designing a Container Terminal Yard,Delft University of Technology, Delft

Autodizalice (*Reachstackers*-RS) su mobilna prijevozno-prekrcajna sredstva s mehanizmom za dizanje i nagibanje dohvatnika, najčešće teleskopske izvedbe[25]. Prema konstrukciji slične su viličarima, no razlikuju se po načinu rada. Naime, RS-i podižu kontejnere pomoću koso postavljenog kraka na kojem se nalazi hvatač

kontejnera (*spreader)*, a mogu slagati do 8 kontejnera u visinu i 3 kontejnera u red, jedan iza drugog (Slika 10).

Slika 10 Autodizalica

Izvor: Mohseni, N. S., 2011., Developing a Tool for Designing a Container Terminal Yard,Delft University of Technology, Delft

No, kako bi se broj premještaja kontejnera sveo na minimum, slaganje kontejnera je često ograničeno na 2 kontejnera u dubinu i 3-4 reda u visinu[8]. RS može slagati 3-4 reda u visinu s gustoćom slaganja od 350-500 TEU/ha. Ova vrsta slagališnih sredstava posebice je pogodna za rad na višenamjenskim, malim i srednjim kontejnerskim terminalima. Prednosti uporabe autodizalica su mali investicijski i kapitalni trošak te mali troškovi rada.

Portalni prijenosnik malog raspona (SC dizalica) je jedan od najzastupljenijih prijevozno-prekrcajnih slagališnih sredstava na kontejnerskim terminalima (Slika 11).

Slika 11 SC dizalica

Izvor: Mohseni, N. S., 2011., Developing a Tool for Designing a Container Terminal Yard,Delft University of Technology, Delft

Opravdanost njegovog učestalog korištenja ogleda se kroz raznolikost izvođenja radnji s kontejnerom: ukrcaj, iskrcaj, slaganje i prijevoz kontejnera između slagališta i pristana. No; najvažniji razlog njegove zastupljenosti je efikasnost i fleksibilnost na ograničenom prostoru. Također, jedna od očitih prednosti SC-a je ta što ima sposobnost izravnog transporta kontejnera na relaciji slagalište - obala (i obrnuto), odnosno odlaganje kontejnera pod postolje dizalice, bez čekanja na pokret obalne dizalice. SC može obavljati razne vertikalne i horizontalne manipulacije s kontejnerima. Smatra se optimalnim sredstvom za rad na srednje velikim i velikim LKT-ima. Osim navedenih prednosti SC-i imaju i određene nedostatke poput velikih kapitalnih troškova, velikih troškova održavanja, veliki troškovi rada, veliko radno područje s malom gustoćom slaganja (u odnosu na RTG i RMG) te neadekvatnost za prijevoz kontejnera na velike udaljenosti duž terminala[9]. Portalni prijenosnici malog raspona slažu kontejnere u visinu jedan povrh drugoga, a između kontejnera složenih u redova smještene su vozne trake po kojima se kreću. Imaju mogućnost slaganja kontejnera do 4 reda u visinu. Zbog potrebnih traka za kretanje ova sredstva ostvaruju srednju gustoću slaganja, a iznosi od 500 TEU/ha(2-3 reda u visinu) do 750 TEU/ha(3-4 reda u visinu).

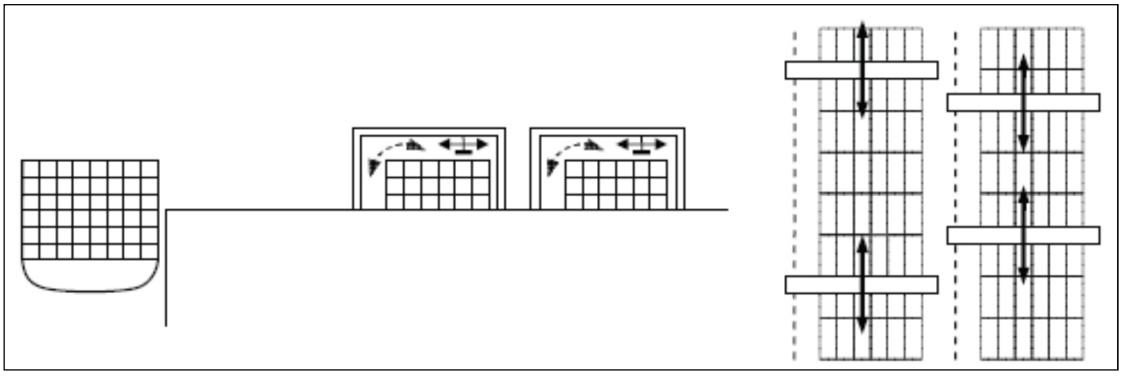
Portalni prijenosnik velikog raspona na kotačima (RTG dizalica) najčešće se upotrebljavaju na srednje velikim i velikim kontejnerskim terminalima, a primarno se

koriste za izvođenje vertikalnih kretnji s kontejnerima, prekrcavajući ih na kamione ili vagone (izvode horizontalne kretnje s kontejnerima). RTG dizalice imaju vrlo veliku fleksibilnost rada i veliki faktor gustoće slaganja kontejnera s mogućnošću slaganja do 6 redova u visinu + 1 rezervna traka i 5-7 redova u širinu+ 1 red koji predstavlja prometnu traku za kretanje IT i XT (Slika 12)[8]. U slučaju slaganja kontejnera do 4 reda u visinu, gustoća slaganja ovih slagališnih dizalica iznosi približno 1.000 TEU/ha[9]. Upravo zbog izvođenja kretnji putem kotača, radno područje ove vrste slagališne dizalice, izuzev slagališta, obuhvaća i otpremnu zonu (područje kopnenog podsustav terminala).

Slika 12 RTG dizalica

Izvor: Mohseni, N. S., 2011., Developing a Tool for Designing a Container Terminal Yard,Delft University of Technology, Delft

Kontejnerski terminali koji koriste RTG dizalice za rad na slagalištu najčešće slažu kontejnere paralelno u odnosu na zonu pristana, a razlog tomu je kraći prijevozni put između spomenutih područja (Slika 13)[8].



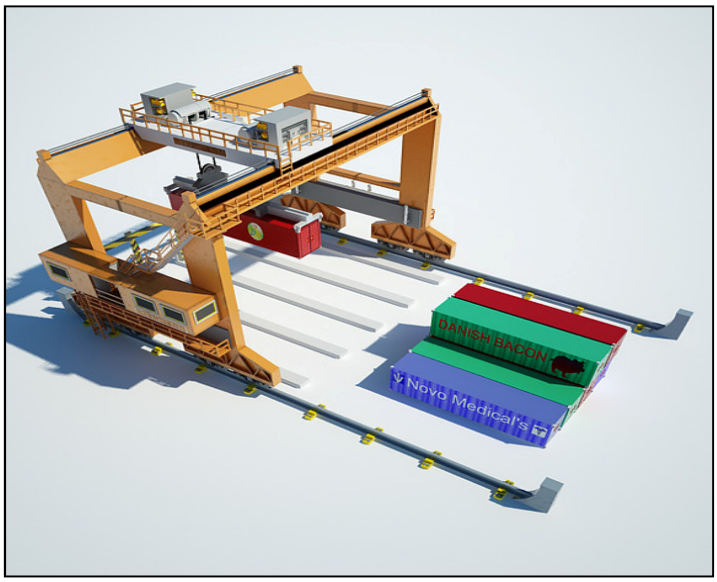
Slika 13 Prostorni prikaz radnog područja RTG dizalice na slagalištu kontejnerskog terminala

Izvor: Mohseni, N. S., 2011., Developing a Tool for Designing a Container Terminal Yard,Delft University of Technology, Delft

Zbog visokog faktora gustoće slaganja, odnosno velikog kapaciteta slaganja s obzirom na potrebnu veličinu slagališnog prostora, ove slagališne dizalice često se koriste za rad na terminalima u zemljama Dalekog Istoka i Europe, odnosno terminalima koji se susreću s nedostatkom slagališnog prostora. Razlog tome je što RTG dizalice u odnosu na RMG imaju mogućnost poprečnog kretanja između zona slagališta tj. mogućnost promjene radnog područja te sposobnost okretanja oko vertikalne osi jednog kotača ili jedne noge portala (u slučaj 8 kotača)[26]. Također, ove su dizalice u odnosu na RMG dizalice manjih dimenzija i težina.

Prema mišljenju mnogih stručnjaka, upotrebom ove vrste slagališnih dizalica, planeri na terminalu moraju voditi računa o mnoštvu taktičkih i operativnih problema poput raspodjele RTG dizalica i IT/XT, rasporeda slaganja kontejnera na kontejnerske pozicija i dr.[27].

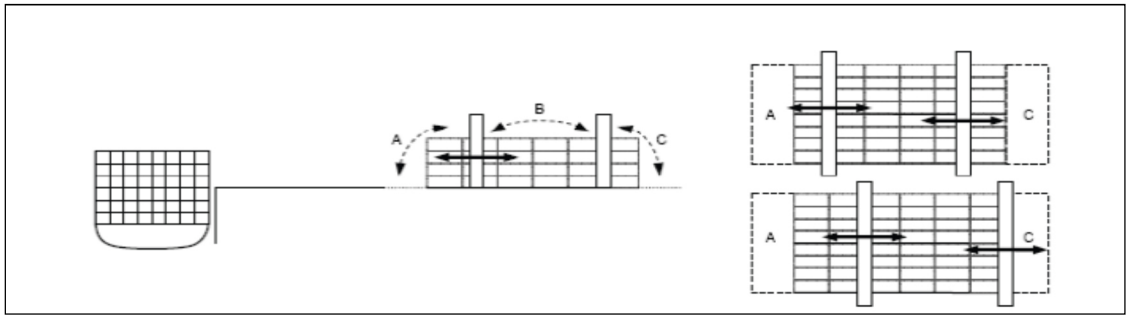
Portalni prijenosnik velikog raspona na tračnicama (RMG dizalice) su portalni prijenosnici velikog raspona koji svoje horizontalno kretanje izvode po tračnicama (Slika 14). Koriste se na velikim kontejnerskim terminalima poput Hong Konga, Singapura i dr.



Slika 14 RMG dizalica

Izvor: www.creativecrash.com (lipanj 2015.)

Automatizirane izvedbe RMG dizalica koriste u automatiziranim europskim terminalima poput Rotterdama i dr. Zahvaljujući tračnicama RMG dizalice u odnosu na RTG dizalice mogu bolje rasporediti opterećenje. Stoga su RMG dizalice pogodne za slagališta na kojima su opterećenja tla manja, primjerice na starim obalama. Također, potrebno je naglasiti da s aspekta slobode kretanja, ove dizalice u odnosu na RMG dizalice, imaju ograničeno kretanje po slagalištu terminala tj. mogu se kretati isključivo pravocrtno (linearno) duž blokova unutar samo jedne zone slagališta (Slika 15)[8].



Slika 15 Prostorni prikaz radog područja RMG dizalice na slagalištu kontejnerskog terminala

Izvor: Mohseni, N. S., 2011., Developing a Tool for Designing a Container Terminal Yard,Delft University of Technology, Delft

Upravo zbog specifičnog kretanja za instalaciju ovih slagališnih dizalica potrebna su mnogo veća investiranju od svih prethodno navedenih slagališnih sredstava.

RMG dizalice imaju veće brzine kretanja te mogu slagati do 12 redova u širinu i 8 redova u visinu. Faktor gustoća slaganja RMG dizalica iznosi preko 1.000 TEU/ha s mogućnošću slaganja do 4 kontejnera u visinu[8]. RMG dizalice u odnosu na RTG dizalice imaju brojne prednosti koje se ogledaju u: većoj pouzdanosti u radu, duljem vijeku trajanja, relativno malim troškovima održavanja i rada. Unatoč navedenim prednostima RMG dizalice u usporedbi s RTG dizalicama imaju dva bitna nedostatka, a to su da zahtijevaju veće investicijske troškove te posjeduju električni pogon, dok su RTG dizalice pogonjene diesel motorima. Također, navedeni nedostatak ujedno je i prednost RMG dizalica. Zahvaljujući spomenutoj vrsti pogona te su dizalice danas u potpunosti automatizirane (ASC dizalice). Na temelju analiziranih značajki slagališnih sredstava i Wiese-ova istraživanja objavljenog u 2009. godini, prikazana je učestalost uporabe pojedinih slagališnih sredstava na kontejnerskim terminalima (Tablica 2).

Tablica 2 Učestalost korištenja pojedinih slagališnih sredstava na kontejnerskim terminalima[9]

Vrsta slagališnog sredstva	**Broj terminala**	**Udio korištenja**
RTG	72	67,3%
SC	23	21,5%
Automatizirane RMG	7	6,5%
RTG/SC	2	1,9%
RTG/RMG	2	1,9%
RMG	1	0,9%
UKUPNO	**107**	**100,0%**

Izvor: Thoresen, C.A., 2010. Port Designer's Handbook. Thomas Telford, London

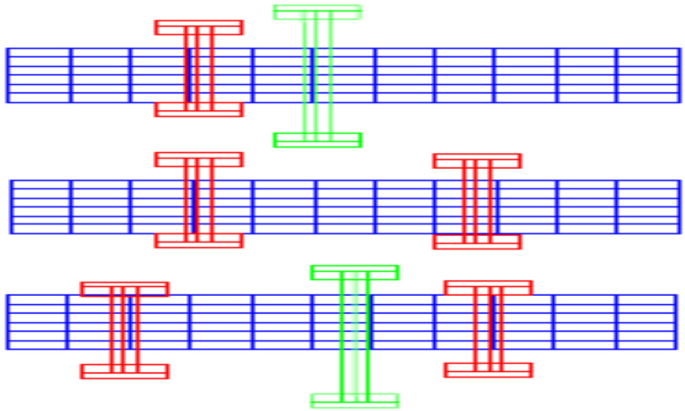
Iz prikazanog je razvidno da su najviše korištena slagališna sredstva RTG dizalica s ukupnim udjelom od 67,3%, dok su RMG dizalice sredstva s najmanjim udjelom korištenja od svega 0,9%. Razlozi takvih rezultata su ti što su RTG dizalice fleksibilnije, jeftinije i infrastrukturno manje zahtjevne.

ASC dizalice omogućavaju u potpunosti automatizirano izvođenje prekrcajnih radnji s kontejnerima na slagalištu kontejnerskog terminal te imaju najveću gustoću slaganja od svih prethodno navedenih slagališnih sredstava, a mogu slagati kontejnere do 10 redova po širini i 6 redova u visinu (Slika 16).ASC dizalice reduciraju operativne troškove rada,

imaju visok stupanj korištenja dizalice i slagališta (*crane and yard utilization rate*) te veliku proizvodnost. Mnogi moderni automatizirani LKT-i, da bi otklonili mogućnost tehničkog otkaza te povećali produktivnost rada terminala, obično koriste po dvije automatizirane RMG dizalice na istim tračnicama (*Twin* RMG- TWRMG) po jednom KB.

Slika 16 Tipična ASC dizalica(RMG tipa) na kontejnerskom terminalu Antwerpen
Izvor: www.porttechnology.org (travanj 2015)

Osim njih postoje i duple automatizirane RMG dizalice (*Double* RMG-DRMG) koje predstavljaju kombinaciju dviju automatiziranih RMG dizalica različitih visina koje imaju mogućnost prolaska jedna ispod druge (niža ispod više), a smještene su svaka na svojim tračnicama. Kontejnerski terminal Burchardkai u luci Hamburg 2012. godine u rad slagališta uvodi trostruku RMG dizalicu (*Triple* RMG-TRMG) koja se sastoji od para automatiziranih RMG dizalica koje se kreću po istim tračnicama te automatizirane RMG dizalice koja ih nadilazi, a kreće se po zasebnim tračnicama (Slika 17)[28].



Slika 17 Prikaz rada *Twin*, *Double* i *Triple* RMG dizalica

Općepoznato je da operateri na terminalu uporno nastoje pronaći nove tehnologije rada kako bi se povećao učinak i kapacitet terminala, stoga je logično za očekivati daljnji razvoj na području slagališnih sredstava.

5. OPTIMIZACIJA RASPODJELE KONTEJNERA NA SLAGALIŠTU LUČKOGA KONTEJNERSKOG TERMINALA

Otprema kontejnera sa slagališta odvija se prema određenom rasporedu sukladnom ukrcajnom planu broda. Stoga svi kontejneri na slagalištu imaju unaprijed određena vremena otpreme (*pickup time*). Kako bi se postiglo optimalno vrijeme otpreme svih kontejnera sa slagališta , kontejneri na slagalištu trebali bi biti složeni u skladu s LIFO (*Last-in, First-out*) metodom koja se temelji na pretpostavki da se na vrhu stupaca u odjeljku nalaze kontejneri koji najprije trebaju biti otpremljeni. Prethodno u radu navedeni su razlozi zbog kojih je gotovo nemoguće slagati kontejnere prema navedenoj metodi, a čija je posljedica blokiranje pojedinih ciljnih kontejnera. Da bi se ciljni kontejneri mogli otpremiti, potrebno je izvršiti radnje premještaja svih kontejnera koji se nalaze iznad ciljnih kontejnera. Uobičajena procedura je premještaj kontejnera unutar odjeljka, budući da premještaj kontejnera između odjeljaka znatno produljuje vrijeme otpreme kontejnera sa slagališta, povećava trošak rada slagališne dizalice, a, u konačnici, i vrijeme zadržavanja broda na vezu. Neovisno o tome obavlja li se premještaj unutar odjeljka ili između odjeljaka, najznačajniji utjecaj na ukupno vrijeme otpreme kontejnera ima broj premještaja kontejnera.

Kao što je navedeno u poglavlju 1.1. , pristup optimizaciji broja premještaja kontejnera unutar odjeljka moguć je u okviru problema BRP/CRP i PMP. U ovom modelu odabrana varijanta optimizacije je broj premještaja kontejnera unutar odjeljka u okviru problema BRP.

5.1 Opis problema BRP/CRP

Problem BRP/CRP pripada u kategoriju lako objašnjivih, no teško rješivih problema. Sam problem vrlo je jednostavan - promatra se jedan odjeljak unutar kojeg su slučajnim odabirom generirani početni rasporedi kontejnera, potrebno je otpremiti sve kontejnere iz odjeljka, sukladno naznačenim prioritetima otpreme koji predstavljaju vremena otpreme uz minimalan broj premještaja. Unutar odjeljka izvode se dvije radnje: premještaj i otprema kontejnera. Premještaj kontejnera jest radnja koja podrazumijeva premještaj kontejnera s vrha stupca na neki drugi stupac unutar odjeljka. Otprema kontejnera jest radnja koja podrazumijeva podizanje kontejnera s vrha stupca te

njegovo odlaganje/ukrcaj na kamion. Rješenje problema BRP/CRP rezultira otpremom svih kontejnera iz odjeljka odnosno potpuno praznim odjeljkom.

Modeli problema BRP/CRP razlikuju se ovisno o:

- vremenu dolaska kontejnera - statički i dinamički
- varijantama premještaja - neograničen i ograničen
- vrstama prioriteta otpreme – pojedinačni i grupni
- ciljevima optimizacije.

Problem BRP/CRP, sa stajališta vremena dolazaka kontejnera u odjeljak, može se promatrati kao statički ili dinamički problem. Statički problem podrazumijeva premještaj i otpremu kontejnera koji se već nalaze unutar promatranog odjeljka, dakle nema slaganja novih kontejnera u odjeljak, dok dinamički problem dozvoljava dolazak novih kontejnera u odjeljak. Statički problem predstavlja zatečeno stanje odjeljka u nekom promatranom vremenu, a dinamički problem predstavlja promjenjivo stanje odjeljka ovisno o vremenu.

Problem BRP/CRP s obzirom na varijante premještaja kontejnera unutar odjeljka može biti neograničen i ograničen. Neograničen premještaj kontejnera (*unrestricted variant*) dozvoljava premještaj bilo kojeg kontejnera unutar odjeljka, dok ograničen premještaj *(restricted variant*) dozvoljava isključivo premještaj onog kontejnera koji se nalazi iznad ciljnog kontejnera koji ide na otpremu.

Formulacija modela BRP/CRP problema, s obzirom na prioritet otpreme kontejnera, može biti pojedinačna ili grupna. Pojedinačni prioritet kontejnera podrazumijeva da svaki kontejner unutar odjeljka ima jedinstven prioritet otpreme, dok grupni prioritet predstavlja grupu kontejnera zajedničkih obilježja poput vremena otpreme, tipa kontejnera, broda, odredišne luke itd.

S obzirom na veličine optimizacije formulacija modela BRP/CRP može biti broj premještaja kontejnera, vrijeme prekrcajnih operacija, trošak premještaja kontejnera i duljina puta slagališnih dizalica. Za sve navedene veličine cilj optimizacije je minimum vrijednosne funkcije.

Prema autoru Caserta, problem BRP/CRP pripada u NP težak problem (*Non-deterministic Polynomial hard problem*- NP hard), što znači da ne postoji algoritam koji ga rješava u polinomijalnom vremenu. Stoga se, da bi se NP težak problem mogao

riješiti, teži primjeni metaheuristika čija je osnovna prednost brzina rješavanja problema velikih dimenzija. Osnova rada metaheuristika temeljena je na stohastičkim algoritmima, što znači da u procesu donošenja odluka, neka je od odluka donesena slučajnim odabirom. Kao rezultat primjene metaheuristika, dobiva se rješenje koje je blizu optimalnog rješenja, odnosno dovoljno kvalitetno rješenje ili čak optimalno rješenje.

Za postizanje cilja istraživanja postavljen je model koji optimizira ukupan broj premještaja te premještaj kontejnera po širini i visini, primjenom određenog redoslijeda pravila premještanja.

Model raspodjele kontejnera koji rješava problem BRP/CRP detaljno je prikazan u nastavku rada.

5.2 Osnovne postavke i ograničenja modela

Opće pretpostavke modela su:

- svi kontejneri su po tipu- suhi višenamjenski
- svi kontejneri su iste veličine – 20 stopni (1 TEU)
- svi kontejneri su puni
- smatra se da je prilikom slaganja kontejnera zadovoljen uvjet težine kontejnera
- svi kontejneri su odlazni i ukrcavaju se na kamione namijenjene unutarnjem transportu, a potom na brod
- operacije premještaj i otpreme kontejnera izvode se jednom RTG dizalicom
- RTG dizalica izvodi operacije premještanja i otpreme samo s jednim kontejnerom
- promatra se jedan odjeljak
- odjeljak je određen s dvije dimenzije :brojem stupaca (*stacks*) i brojem redova (*tiers*)
- statički problem (nema dolazaka i slaganja novih kontejnera u odjeljak)
- početni raspored kontejnera unutar odjeljka generiran je slučajnim odabirom kontejnerskih pozicija
- svaki kontejner unutar odjeljka ima utvrđen prioritet otpreme kontejnera, a prioriteti otpreme označeni su rednim brojevima; kontejner označen manjim rednim brojem ima veći prioritet otpreme, što znači da mora biti otpremljen prije kontejnera označenog većim rednim brojem

- ispod portala RTG dizalice smještena je vozna traka za kamione namijenjena jednosmjernom kretanju prometa
- prioriteti kontejnera su jedinstveni, ne postoje grupe kontejnera s istim prioritetom
- broj, visina i širina premještaja izračunava se samo za premještane kontejnere. Dok kontejneri koji idu direktno na otpremu nisu uzeti u obzir

Ograničenja modela jesu:

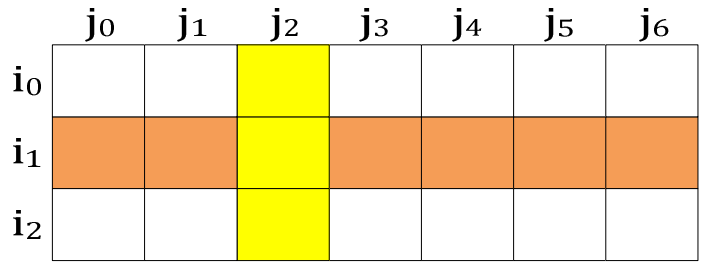
- maksimalna visina slaganja kontejnera unutar odjeljka sukladna je maksimalnoj visini slaganja najsuvremenije RTG dizalice, umanjenoj za rezervnu visinu koja je potrebna za izvođenje operacije premještaja i otpreme kontejnera
- širina odjeljka sukladna je rasponu najsuvremenije RTG dizalice te je umanjena za jednu širinu kontejnera koja služi kao vozna traka za kamione unutarnjeg transporta.
- Promatrani odjeljak ne smije popunjen u potpunosti, a razlog je taj što su za radnju premještaja potrebne slobodne pozicije unutar odjeljaka, stoga maksimalni udio popunjenosti odjeljka iznosi otprilike 90%.
- Dozvoljen je premještaj samo onih kontejnera koji se nalaze u istom stupcu u kojem se nalazi ciljni kontejner koji mora na otpremu, dok je premještaj ostalih kontejnera unutar odjeljka zabranjen (*Restricted* BRP/CRP- R BRP/CRP).

5.3 Ulazni i izlazni podaci modela

Osnovni ulazni podaci modela su:

- broj stupaca u odjeljku
- broj redova u odjeljku
- broj kontejnera unutar odjeljka
- početni raspored kontejnera unutar odjeljka te
- prioriteti otpreme kontejnera

Odjeljak je prikazan matričnim zapisom gdje *j* označava broj stupaca (širinu), a *i* predstavlja broj redova (visinu) odjeljka (Slika 18).



Slika 18 Prikaz odjeljka

Stupci unutar odjeljka broje se s lijeva prema desno. Stupac s minimalnim indeksom označen je s j_0. Najviša visina slaganja kontejnera u odjeljku označena je s i_0. Kontejnerska pozicija određena je brojem stupaca i redova, a predstavlja ju uređeni par *i* i *j* koji se označava oznakom (*i, j*). Kontejneri raspoređeni na kontejnerske pozicije imaju utvrđene prioritete otpreme koji predstavljaju redoslijed njihove otpreme iz odjeljka.

Izlazni podaci modela su:

- evolucija (generacija/iteracija) u kojoj je postignuto rješenje
- redoslijed primjene pravila premještaja
- vrijednost *fitness* funkcije
- ukupan broj premještaja unutar odjeljka
- ukupna premještaja po širini unutar odjeljka
- ukupna premještaja po visini unutar odjeljka
- vrijeme izvođenja modela.

5.4 Izrada modela raspodjele kontejnera na slagalištu lučkoga kontejnerskog terminala primjenom genetskog algoritma

Genetski algoritam (*Genetic algorithm* -GA) je jedna od metaheurističkih metoda koja se koristi za rješavanje složenih kombinatornih optimizacijskih problema. Ovu metodu prvi je predložio J. Holland davne 1975. godine, a predstavlja metodu pretraživanja i nalaženja rješenja koje je optimalno ili blizu optimalnog rješenja. To je prirodom inspiriran evolucijski algoritam koji se bazira na ideji genetike i evolucije u prirodi. On simulira proces genetske evolucije populacije pod utjecajem promjenjivog okruženja i genetskih operatora.

U ovom radu GA ostvaren je kroz tri razreda. Razred `GenetskiAlgoritam` predstavlja jezgru samog algoritma. Razred `MyFitnessFunction` predstavlja *fitness* funkciju algoritma u koju su ugrađene konstruktivne heuristike. Razred `MyConfiguration definira` genetske operatore, veličinu populacija, broj evolucija itd.

5.4.1 Osnovni elementi genetskog algoritma

Da bi se genetski algoritam mogao primijeniti na promatrani optimizacijski problem, potrebno je utvrditi njegove osnovne elemente: kromosome(jedinke), populaciju jedinki (populaciju kromosoma), uvjete preživljavanja jedinki (*fitness* funkciju) i njihova testiranja te genetske operatore[29].

5.4.1.1 Kromosom

Ključan čimbenik za stvaranje genetskog algoritma je izbor dobrog kromosoma. Kromosom se sastoji od skupa gena te predstavlja jedno rješenje problema BRP/CRP (broj premještaja, broj premještaja po širini i visini). Gen predstavlja jediničnu informaciju i može biti zapisan u obliku binarnog ili vrijednosnog koda. U ovom modelu gen je vrijednosno kodiran odnosno cjelobrojnog (*Integer*) tipa i predstavlja jedno od četiri heuristička pravila (detaljnije u nastavku rada) koja određuju poziciju premještaja kontejnera unutar odjeljka te poprimaju vrijednosti od 1 - 4. Svaki gen predstavlja jedno pravilo za određivanje pozicije premještaja kontejnera. Kodne vrijednosti gena dodijeljene su na sljedeći način: pravilo 1 – 1, pravilo 2 – 2, pravilo 3- 3 te pravilo 4 – 4. Navedeno je u algoritmu zapisano sa šest linija koda.

```
Gene[] sampleGenes = new Gene[brojGena];
for (int i = 0; i < brojGena; i++) {
sampleGenes[i] = new IntegerGene(conf, 1, 4);
}

Chromosom sampleChromosome= new Chromosome(conf,
sampleGenes);
conf.setSampleChromosome(sampleChromosome);
```

Analogno tome prikazan je slučajno generirani kromosom čiji prvi gen predstavlja pravilo 1, drugi gen predstavlja pravilo 1 i tako redom (Slika 19).

GEN 1	GEN 2	GEN 3	GEN 4	GEN 5	GEN 6	GEN 7	GEN 8	GEN 9	GEN10	…
1	1	2	3	4	2	4	3	3	2	…

Slika 19 Prikaz kromosoma cjelobrojnog tipa

Broj gena u kromosomu određen je umnoškom broja kontejnera i broja 10, a navedeno je zapisano sljedećom linijom koda:

```
int brojGena = brojKontejnera * 10;
```

gdje broj kontejner predstavlja ukupan broj kontejnera složenih unutar odjeljka, dok je broj 10 određen kao optimalan broj čijim će se umnoškom osigurati dovoljan broj gena u kromosomu, odnosno dovoljan broj pravila.

Primjerice, ukoliko se promatra odjeljak veličine 3x7 unutar kojeg je složeno 11 kontejnera, tada se za dobivanje rješenja problema BRP/CRP generira kromosom od 110 gena (Tablica 3).

Tablica 3 Prikaz broja gena prema broju kontejnera u odjeljku

Dimenzije odjeljka	Broj kontejnera	Broj gena
3x7	11	110
3x7	15	150
3x7	19	190
4x7	15	150
4x7	21	210
4x7	25	250
5x7	19	190
5x7	26	260
5x7	31	310
6x7	23	230
6x7	31	310
6x7	37	370

Iz tablice je razvidno da se u ovome modelu koristi dvanaest kromosoma s devet različitih duljina kromosoma tj; broja gena u kromosomu. Dakle, redoslijed pravila u kromosomu, iako je generiran slučajno, sadrži potencijalno rješenje problema BRP/CRP, odnosno redoslijed pravila koja će se primijeniti na premještaj kontejnera. Ukupan broj primijenjenih pravila ujedno predstavlja ukupan broj izvršenih radnji premještaja kontejnera.

5.4.1.2 Populacija

Populaciju predstavlja skup kromosoma u i-tom koraku algoritma (Slika 20). Populacija se još naziva i genotip, a predstavlja skup rješenja u trenutnoj evoluciji (generaciji/iteraciji) genetskog algoritma.

2	1	3	3	1	4	4	2	3	1 ...	Kromosom 1
3	2	3	1	1	2	4	2	3	4 ...	Kromosom 2
4	1	4	3	1	4	4	1	3	3 ...	Kromosom 3
1	1	3	4	1	3	4	2	3	4 ...	Kromosom 4

POPULACIJA

Slika 20 Prikaz populacije od 4 kromosoma

Dvije su glavne značajke populacije prilikom izrade genetskog algoritma: stvaranje početne populacije i utvrđivanje veličine populacije. Pravilno određivanje parametra veličine populacije od velike je važnosti jer izravno utječe na brzinu i kvalitetu postizanja rezultata.

Prilikom definiranja veličine populacije nastojalo se ostvariti sljedeće ciljeve: dobivanje optimalnih ili najboljih rješenja te relativno kratko vrijeme izvođenja modela. Kako bi se postiglo dobivanje navedenih rješenje, potreban je dovoljno velik prostor pretraživanja rješenja. S obzirom na složenost i vrijeme rješavanja problema BRP/CRP, taj prostor tj. veličina populacije određen je na 8.000 kromosoma(jedinki). Kao i u većini slučajeva, i u ovome modelu početna populacija je stvorena slučajnim odabirom. Tako stvorena populacija omogućava raznovrsnost gena u kromosomu, odnosno rješenja problema.

Definiranje svih kromosoma u populaciji metodom slučajnog odabira dobiva se na osnovu linije koda:

```
Genotype population = Genotype.randomInitialGenotype(conf);
```

5.4.1.3 Fitness funkcija

Fitness funkcija je evaluacijska funkcija kojom se utvrđuje kvaliteta pojedinog kromosoma (jedinke) kao rješenja određenog problema. U literaturi se još naziva funkcijom prikladnosti i funkcijom dobrote. Radi tako da uzima kromosom kao parametar te vraća vrijednost cijelog broja. Ta vrijednost pokazuje koliko je taj kromosom(rješenje) dobar u odnosu na ostale kromosome. Smatra se ključem selekcije kromosoma jer određuje koji će kromosom sudjelovati u stvaranju nove populacije, koja će biti bolja od prethodne. Kako bi se omogućio razvoj dobrih rješenja, vrijednost *fitness* funkcije(dobrota) dodijeljena rješenju mora izravno održavati njegovu kvalitetu, tj. *fitness* funkcija mora pokazati koliko dobro određeno rješenje ispunjava zahtjeve određenog problema.

Prilikom izrade *fitness* funkcije korišteni su parametri - broj redova, broj stupaca i broja kontejnera te varijabla - broj gena u kromosomu.

Fitness funkcija jednaka je jednostavnoj formuli, a u modelu je definirana sljedećom linijom koda:

```
fitness = brojPremjestaja * 1000 +
razlikaPremjestajaSirina + razlikaPremjestajaVisina;
fitness ukupno = 100000 - fitness;
```

gdje broj premještaja predstavlja promjenu kontejnerske pozicije pojedinog kontejnera unutar odjeljka, dok premještaj po širini i visini predstavlja razliku ostvarenog premještaja po širini i visini stupca, u odnosu na početnu kontejnersku poziciju premještanog kontejnera. Iz navedenog koda vidljivo da se *fitness* funkcija sastoji od tri varijable od kojih je za uspješnost rada modela najvažnija varijabla broja premještaja, stoga je i množena s 1.000.

Cilj ukupne *fitness* funkcije je pronalaženje minimalnog broja premještaja kontejnera i premještaja kontejnera po širini te izračun premještaja po visini. Kako je problem BRP/CRP minimizacijski problem za postizanje navedenog cilja, odnosno minimuma

ukupne *fitness* funkcije, potrebno je pronaći kromosom maksimalne vrijednosti *fitness* funkcije (maksimalne dobrote) . Što je dobrota kromosoma manja, kromosom ima manju vjerojatnost preživljavanja te se može reći da on, uistinu, ne predstavlja rješenje promatranog problema.

5.4.1.4 Genetski operatori

GA sastoji se od tri operatora:

- selekcije
- križanja
- mutacije

Selekcija je postupak izbora kromosoma (roditelja) između svih kromosoma u trenutnoj populaciji sa svrhom reprodukcije odabranih kromosoma. Selekcija se vrši temeljem vrijednosti *fitness* funkcije. Njezina osnovna uloga je očuvanje i prenošenje dobrih svojstava kromosoma (jedinki) u novu generaciju. Selekcija kromosoma odvija se sukladno statističkoj vjerojatnosti preživljavanja, pa tako kromosomi s većom vrijednosti *fitness* funkcije, imaju veću vjerojatnost „preživljavanja" tj. prelaska u sljedeću generaciju, no to se ne može tvrditi sa sto postotnom sigurnošću. S obzirom na prirodu optimizacijskog problema, u modelu se koristi tzv. *selekcija n najboljih kromosoma* koja je ujedno i najjednostavniji oblik selekcije. Četiri su osnovne faze rada *selekcije n najboljih kromosoma*:

- I. - preuzimanje kromosoma kandidata za sljedeću generaciju
- II. - rangiranje *n* najboljih kromosoma na temelju najvećih vrijednosti *fitness* funkcije
- III. - odabir *n* kromosoma s najvećom vrijednosti *fitness* funkcije za sljedeću generaciju
- IV. - uvrštavanje najboljih *n* kromosoma u sljedeću generaciju

U modelu broj najboljih kromosoma koji prelazi u sljedeću generaciju iznosi 50 % populacije. Najbolji kromosomi mogu se pojavljivati i do nekoliko generacija.
Navedeno se izvršava linijom koda:

```
BestChromosomesSelector best=new
BestChromosomesSelector(this, 0.5);
```

Također, dozvoljeno je i dupliciranje kromosoma koji se pojavljuju u generaciji, a izvršava se metodom:

```
best. setDoubletteChromosomesAllowed(true);
```

Osim navedenog, određeno je da u stvaranju nove generacije sudjeluje 60% kromosoma iz prethodne generacije, što znači da se u novoj generaciji pojavljuje 4.800 kromosoma iz prethodne generacije. Navedeno je zapisano linijom koda:

```
setSelectFromPrevGen(0.60);
```

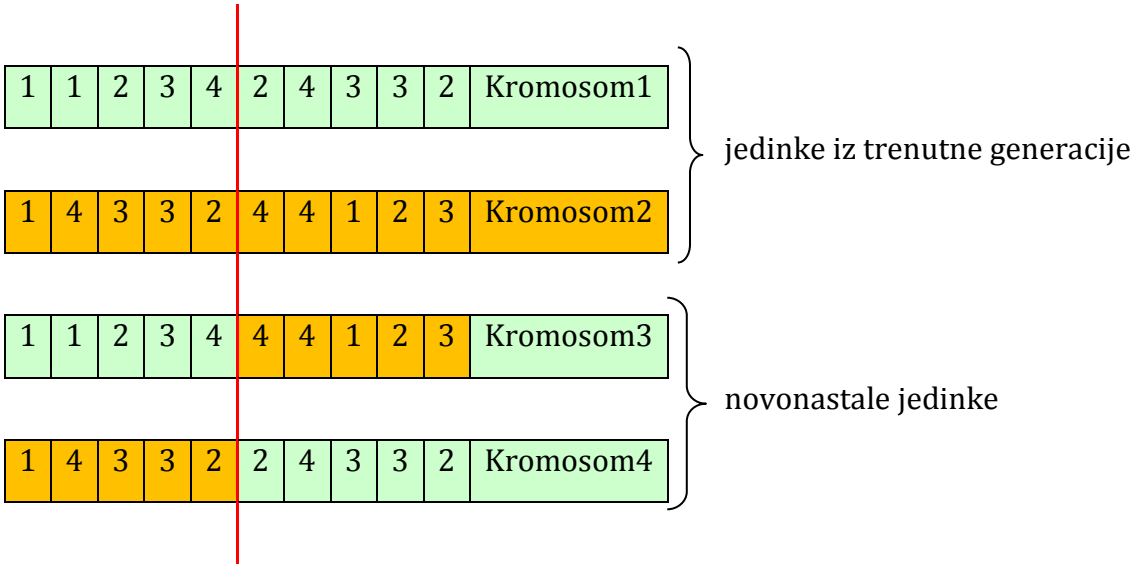
Dakle, u sljedećoj generaciji veličine populacije od 8.000 kromosoma sudjeluje 4.000 kromosoma koji su "preživjeli" selekciju i 800 kromosoma koji su odabrani slučajnim odabirom iz prethodne generacije te 3.200 novonastalih kromosoma. Kako i kromosomi s lošim rješenjima tj; malim vrijednostima *fitness funkcije* mogu sadržavati dobre gene, potrebno je i njima dozvoliti vjerojatnost za reprodukciju.

Križanje (rekombinacija) je proces kojim se iz dva kromosoma dobivaju novi kromosomi koji sadrže kombinacije gena kromosoma od kojih su nastali (tzv. kromosoma roditelja). Osnovni cilj križanja kromosoma je da se očuvaju dobra svojstva trenutno najkvalitetnijih kromosoma. Novonastali kromosom sadrži kombinaciju oba kromosoma roditelja te ima jednak broj gena kao i oni. U model je implementirano *križanje s jednom točkom prekida* koje je definirao J. Holland (1975), a provodi se kroz četiri osnovne faze:

- I. - preuzimanje svih kromosoma iz trenutne generacije
- II. - odabir n parova kromosoma koji će sudjelovati u križanju
- III. - slučajno odabiranje točke križanja za sve odabrane parove
- IV. - uvrštavanje novonastalih kromosoma u sljedeću generaciju.

Na samom početku operator križanja preuzima sve kromosome iz trenutne generacije, odnosno N/2 parova gdje je N veličina populacije. Konkretno u ovom algoritmu u trenutnoj generaciji ima 4.000 parova kromosoma. Zatim se bira određen broj parova kromosoma koji će sudjelovati u križanju. Koliko će broj parova kromosoma sudjelovati u križanju, ovisi o parametru koji se naziva *vjerojatnost križanja* (vidi objašnjenje na

str.66). U ovom algoritmu u križanju će sudjelovati 3.800 parova kromosoma odabranih slučajnim odabirom. Vjerojatnost da će kromosom biti odabran za križanje jednaka je za sve kromosome, neovisno o njihovoj vrijednosti *fitness* funkcije. Zatim se za svaki par kromosoma slučajnim odabirom određuje točka križanja (pozicija križanja) nakon koje se vrši izmjena gena u kromosomu. Način na koji se vrši izmjena, odnosno rekombinacija gena, prikazan je na slici 21. Nakon provedene rekombinacije gena, novonastali kromosomi uvrštavaju se u sljedeću generaciju.



Slika 21 Prikaz križanja kromosoma slučajnim odabirom u jednoj točki prekida

Nasuprot križanju, koje teži zadržavanju dobrih svojstva kromosoma, mutacija teži promjeni genetskog materijala.

Mutacija je proces slučajne promjene jednog ili više gena u kromosomu. Mutacija se koristi kako bi se povremeno unijela raznolikost genetskog materijala među kromosome u populaciji, a posebice u slučaju kada su oni vrlo slični. Ona omogućava raznolikost rješenja novonastale generacije u odnosu na postojeću te time usmjerava potragu u neistražen prostor rješenja. Dakle, mutacijom se pretražuje prostor rješenja, stoga ako cijela populacija "zaglavi" u lokalnom optimumu, mutacija će slučajnim pretraživanjem prostora pronaći bolje rješenje.

Mutacija korištena u izradi modela naziva se *jednostavna mutacija, a* provodi se u četiri osnovne faze:

- I . - preuzimanje svih kromosoma iz trenutne generacije
- II. - slučajno odabiranje kromosoma koji će mutirati
- III. - odabiranje gena koji će mutirati (za sve kromosome)

- IV. - uvrštavanje mutiranih kromosoma u sljedeću generaciju.

Prva faza podrazumijeva preuzimanje svih kromosoma iz trenutne generacije, nakon toga se u drugoj fazi odabiru kromosomi koji će mutirati. Ukupan broj kromosoma koji će sudjelovati u mutaciji, ovisio o parametru koji se naziva *vjerojatnost mutacije*. Vjerojatnost da će neki kromosom biti odabran za primjenu operatora mutacije jednaka je za sve kromosome u generaciji, neovisno o njihovoj vrijednosti *fitness* funkcije. U ovom algoritmu operator mutacije primjenjuje se na kromosom sa zadanom vjerojatnošću mutacije koja iznosi 0,01. Slijedom navedenog, u ovom algoritmu ukupno će mutirati 80 kromosoma određenih slučajnim odabirom.

Treća faza odnosi se na odabir gena koji će mutirati, a neposredno prije samog odabira gena potrebno je utvrditi duljinu kromosoma, odnosno broj gena u kromosomu, a zatim se slučajnim odabirom određuje gen koji će mutirati.

Vjerojatnost da će neki gen u kromosomu mutirati dobiva se sljedećim izrazom:

$$1 \div \text{broj gena u kromosomu} \tag{6}$$

Budući da broj gena u kromosomu nije konstantan već ovisi o broju složenih kontejnera unutar odjeljka, vjerojatnost odabira gena za mutaciju iznosi od 0,0026 do 0,0091 (vidi tablicu 3). Na slici u nastavku rada dan je ilustrativni prikaz jednostavne mutacije (Slika 22).

Kromosom 1 predstavlja kromosom na koji će se primijeniti operator mutacije. Dakle, za mutaciju je odabran gen 4 kromosoma 1 kodne vrijednosti 3 čija vrijednost nakon provedene mutacije iznosi 4.

1	1	2	3	4	2	4	3	3	2	Kromosom 1

1	1	2	4	4	2	4	3	3	2	Mutirani kromosom 1

Slika 22 Prikaz mutacije kromosoma slučajnim odabirom jednog gena

Iz prikazane slike vidljivo je da se operator mutacije provodi samo na jednom genu na način da se izmjeni kodna vrijednost gena tj; pravilo premještaja. Opisani postupak ponavlja se za sve kromosome, odnosno preostalih 79 kromosoma. U zadnjoj fazi provodi se postupak uvrštavanja mutiranih kromosoma u sljedeću generaciju.

5.4.1.5 Proces evolucije

Nakon što su definirani svi parametri, započinje proces evolucije. U ovom modelu zadani broj evolucija populacije je 20. Prilikom određivanja konačnog broja evolucija vodilo se računa da se omogući sljedeće:

- postizanje optimalnog ili najboljeg rješenja za sve promatrane odjeljke
- ponavljanje (potvrda) optimalnog/najboljeg rješenja promatranog odjeljka kroz nekoliko generacija
- postizanje rješenja u razumno kratkom vremenu

Nakon izvršavanja zadanog broja evolucija završava proces evolucije te se vrši vraćanje najboljeg kromosoma u populaciju(generaciju). Kod iz podpoglavlja 5.4.1.2. proširuje se s još pet linija koda.

```
private static int MAX_ALLOWED_EVOLUTIONS = 20;
for (int t = 0; t < MAX_ALLOWED_EVOLUTIONS; t++) {
population.evolve();
}
bestSolution = population.getFittestChromosome();
```

Metoda `getFittestChromosome()` vraća najbolji kromosom u populaciju (generaciju).

5.4.1.6 Kriteriji završetka/zaustavljanja

Poznato je da GA spada u stohastičku metodu pretrage dozvoljenog prostora rješenja, što znači da može beskonačno dugo vremena pretraživati prostor rješenja. Stoga je vrlo važno odrediti kriterije završetka (zaustavljanja). Ovaj model će završiti s radom u trenutku kada je dostignut maksimalni broj evolucija (generacija/iteracija). To je jedini kriterij završetka rada modela, a razlog tome je taj što se ne zna koje je najbolje rješenje(najmanji broj, širina i visina premještaja) za promatrani odjeljak, pa ga se određuje brojem generacija.

5.4.1.7 Ostali aspekti genetskog algoritma

Uz sve opisano u rad algoritma uvedena je elitistička strategija. *Elitistička strategija* (elitizam) odnosi se na politiku zamjene generacija, a njezina primjena skraćuje vrijeme rješavanja zadanog problema te omogućava očuvanje dobrih rješenja. Vrijeme rješavanja zadanog problema skraćuje se jer se u svaku generaciju bez primjene genetskih operatora te računanja *fitness* funkcije direktno propušta određen broj dobrih kromosoma.

Navedeno se ostvaruje metodom:

```
conf.setPreservFittestIndividual(true);
```

Na osnovu prethodno opisanih sastavnih elemenata GA, u sljedećem poglavlju detaljno je prikazan i opisan rad te općeniti pseudokod GA .

5.4.2 Prikaz rada genetskog algoritma

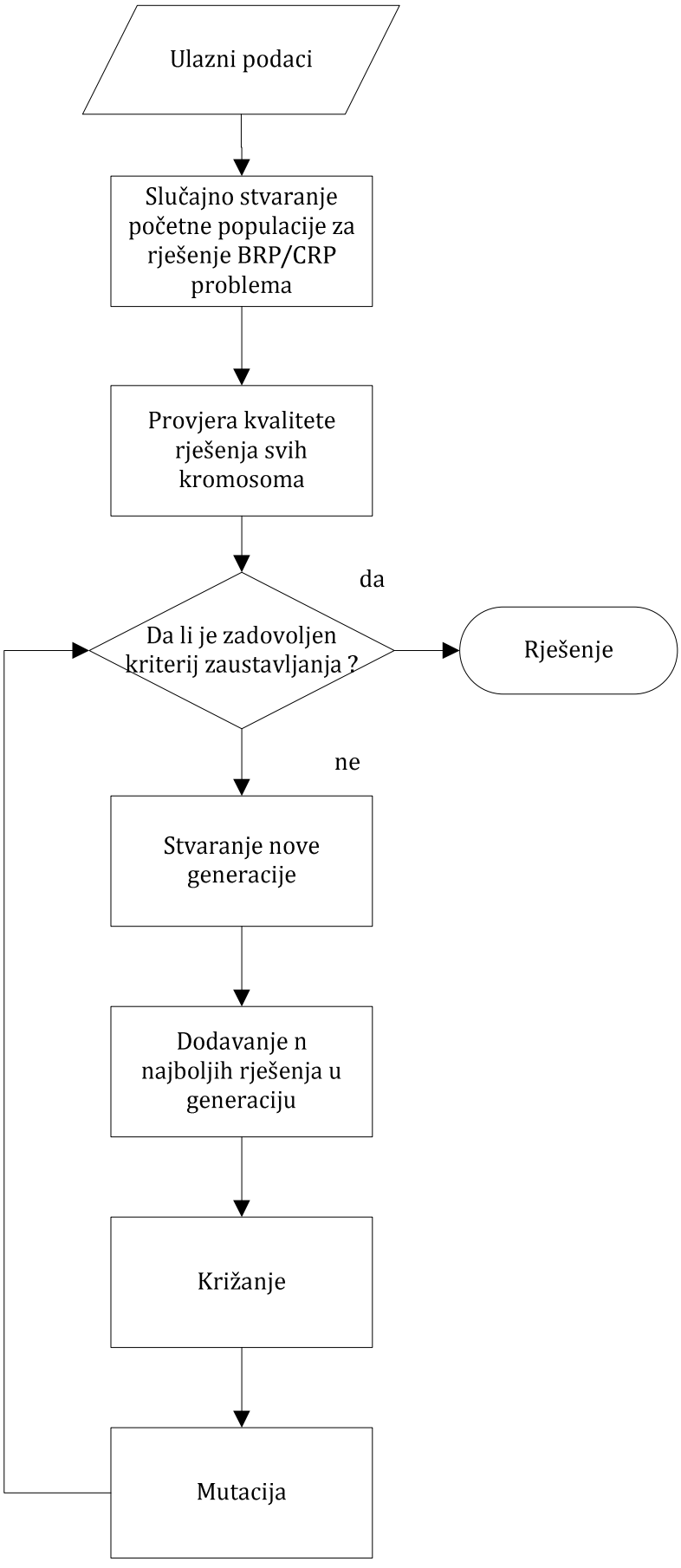
Rad GA temelji se na *bottom–up* (odozdo prema gore) pristupu. Što znači da se prvo kreira skup potencijalnih rješenja te se potom među njima odabire optimalno rješenje. Proces rada genetskog algoritma vrlo je jednostavan, a sastoji od tri osnovna koraka koji su opći i mogu se primijeniti na mnogobrojne optimizacijske probleme, a to su redom:

1. kreiranje početne populacije
2. evaluacija svih kromosoma u populaciji
3. ponavljanje (kreiranje nove generacije)

- selekcija
- križanje i mutacija
- uvrštavanje novonastalih kromosoma u populaciju

Na samom početku rada algoritma, slučajnim odabirom generira se početna populacija koja osigurava raznovrsnost kromosoma, odnosno potencijalnih rješenja BRP/CRP problema. U drugom koraku izračunava se kvaliteta rješenja, odnosno dobrota kromosoma u populaciji. Svaki kromosom prezentira rješenje problema koje se vrednuje temeljem evaluacijske funkcije odnosno *fitness* funkcije. Najboljim rješenjima smatraju se oni kromosomi koji imaju maksimalnu vrijednost *fitness* funkcije.

U trećem koraku se primjenjuje operator selekcije temeljem kojeg se za reprodukciju odabiru kromosomi koji imaju najbolja rješenja sukladna heuristici kvalitete te se formira nova generacija. Nakon toga neki od kromosoma sudjeluju u genetskim operatorima križanja i mutacije. Zatim se ponovo vraća na korak 2, temeljem kojeg se u populaciju uvrštavaju oni kromosomi koji imaju najbolja rješenja. Cijeli postupak se ponavlja sve dok nije izvršeno 20 evolucija, a kao konačno rješenje odabire se kromosom s minimalnom vrijednosti *fitness* funkcije. Opisani koraci rada genetskog algoritma prikazani su blok dijagramom (Slika 23).



Slika 23 Dijagram toka rada genetskog algoritma

5.4.3 Pseudokod rada genetskog algoritma

Na osnovu prikazanog dijagrama rada(Slika 23), izrađen je pseudokod rada genetskog algoritma.

1. [start] slučajnim odabirom stvori početnu populaciju kromosoma
2. [fitness] provjeri kvalitetu rješenja svih kromosoma
3. [nova populacija] stvori novu generaciju te ponavljaj naredne korake sve dok generacija ne bude kompletna
4. [selekcija] izvrši odabir dva najbolja kromosoma za reprodukciju
5. [križanje] na osnovu vjerojatnosti križanja izvrši križanje kromosoma kako bi nastali novi kromosomi(djeca)
6. [mutacija] na osnovu vjerojatnosti mutacije izvrši mutaciju na slučajno odabranom genu u kromosomu
7. [dodavanje] dodaj novonastale kromosome u novu generaciju
8. [zamjena] staru generaciju zamjeni novom generacijom
9. [testiranje] ako je postignut uvjet zaustavljanja,stani, ispiši dobiveno najbolje rješenje i vrati ga u generaciju
10. [petlja] vrati se na korak 2

5.4.4 Parametri rada genetskog algoritma

Parametri GA koriste se za upravljanje radom modela te utječu na brzinu rada modela odnosno vrijeme rješavanja zadanog problema. Dakle, parametri predstavljaju kontrolu rada GA, stoga je vrlo važno dobro ih podesiti. Temeljni parametri rada GA su:

- broj evolucija
- veličina populacije
- vjerojatnost mutacije

Broj evolucija (generacija/iteracija)(*ne*) je parametar koji ima direktan utjecaj na kvalitetu dobivenih rješenja i na vrijeme rješavanja zadanog problema (vrijeme izvođenja modela). Veći broj evolucija (generacija/iteracija) u pravilu znači bolje rješenje, ali i dulje vrijeme rješavanja. Budući da je cilj dobiti što bolje rješenje u što kraćem vremenu, maksimalni broj evolucija postavljen je na 20. Navedeno je u zapisano kodom:

```
private static int MAX_ALLOWED_EVOLUTIONS = 20;
```

Veličina populacije (*N*) također utječe na kvalitetu dobivanja rješenja, stoga je potrebno dobro uskladiti veličinu populacije s brojem evolucija (generacija/iteracija) kako bi se osiguralo postizanje dobrih rezultata. Primjerice, ako je broj kromosoma u populaciji malen, GA ima male mogućnosti izvođenja operatora križanja te malen prostor pretrage rezultata, što u konačnici znatno otežava pronalaženje optimalnog/najboljeg rješenja. S druge strane prevelik broj kromosoma u populaciji produljuje vrijeme izvođenja modela, no kako je cilj ovog modela postići optimalno/najbolje rješenje, vrijeme izvođenja je sekundarno, odnosno jedino je bitno da se optimalno/najbolje rješenje postigne u razumno kratkom vremenu. U ovom modelu veličina populacije je konstantna, što znači da se ne mijenja kroz nove iteracije, a iznosi 8.000 kromosoma te je zapisana sljedećom linijom koda:

```
conf.setKeepPopulationSizeConstant(true);
conf.setPopulationSize(8000);
```

Vjerojatnost mutacije (*pm*) je najvažniji parametar GA jer ona proširuje prostor pretrage rješenja i time sprečava zaglavljivanje u lokalnom optimumu te usmjeruje potragu ka dobivanju dobrih rješenja za promatrani problem. Ona određuje učestalost mutacije kromosoma.

Vjerojatnost mutacije u algoritmu određene je na 0,01, a zapisana je kodom:

```
addGeneticOperator(new MutationOperator(this, 100));
```

Radi preglednosti, korišteni standardni parametri u radu GA predstavljeni su tabličnim prikazom (Tablica 4).

Tablica 4 Vrijednosti temeljnih parametara u radu GA

Parametri	Oznaka
Broj evolucija (generacija)(ne)	20
Veličina populacije (N)	8.000
Vjerojatnost mutacije (pm)	0,01

Ostali parametri korišteni u izradi GA su:

- vjerojatnost križanja i
- duljina kromosoma

Vjerojatnost križanja (pc) nije standardni parametar jer provođenje operacija križanja nije obvezno u svakom GA. U ovome algoritmu koristi se operator križanja koji se provodi s vjerojatnošću križanja od 0,95, što znači da će 95% kromosoma iz populacije sudjelovati u križanju, a zapisano je kodom:

```
addGeneticOperator(new CrossoverOperator(this, 0.95));
```

Duljina kromosoma(d_k) je parametar koji je zadan unaprijed, konkretno u ovom modelu ovisi o broju kontejnera složenih unutar odjeljka i za promatrani odjeljak uvijek je ista. U modelu duljine kromosoma ovise o promatranom odjeljku, a iznose od 110-380 gena (vidi tablicu 3.). Utvrđivanje duljine kromosoma zapisano je sa šest linija koda:

```
{
int[] kromosom = new int[brojGena];
Gene[] genes= ic.getGenes();
for (int i = 0; i < genes.length; i++);
kromosom[i]=(Integer)genes[i].getAllele();
        }
```

Da bi se sa moglo potvrditi da odabrane veličine opisanih parametara zaista daju optimalna/najbolja rješenja, u nastavku ovog istraživanja bit će provedeno eksperimentalno testiranje s različitim veličinama parametara, odnosno ispitat će se utjecaj njihove veličine na kvalitetu dobivenih rješenja.

5.5 Implementacija konstruktivnih heurističkih pravila u genetski algoritam

Da bi se osiguralo postizanje dobrih rješenja u pogledu broja premještaja kontejnera, u modelu su uvrštene konstruktivne heurisitike. Inače, konstruktivne heuristike su konstruktivne metode koje postepeno grade rješenje koristeći pritom sva specifična znanja svakog pojedinačnog zadatka. Osnovna svrha primjene tih pravila je odabir pozicije za svaki premještani kontejner s ciljem da se reduciraju ili u potpunosti izbjegnu dodatna buduća premještaja kontejnera. Dakle, heuristička pravila se primjenjuju samo na kontejnere koji se premještaju. Jedno pravilo predstavlja jedan premještaj kontejnera.

U predloženi model uvrštena su četiri heuristička pravila, a to su redom:

Pravilo 1: **Premjesti kontejner u najbliži stupac u kojem svi kontejneri idu na otpremu kasnije od premještanog kontejnera**.[14]

Pravilo 2: **Premjesti kontejner u najbliži prazan stupac**.

Pravilo 3: **Premjesti kontejner u najbliži i najniži stupac**.[15]

Pravilo 4: **Premjesti kontejner u najbliži stupac koji nije popunjen do maksimalne visine slaganja.**

Potrebno je istaknuti da su sva opisana pravila jednake važnosti te da ne postoji slučaj da niti jedno pravilo nije primjenjivo jer su ovim pravilima obuhvaćeni svi mogući odabiri pozicija za premještaj kontejnera. Navedena pravila predstavljaju rješenja postavljenog problema. U poglavlju 5.5.3 grafički je prikazana njihova primjena u postizanju rješenja.

[14] Stupac koji ima najmanju udaljenost od stupca u kojem se nalazi kontejner koji se mora premjestiti.

[15] Jednostavnija inačica ovog pravila uzimajući u obzir samo najniži stupac prvi je put predstavljena u Zhang (2000). Zhang je to pravilo nazvao TLP - *tier lowest position*. Pri tome se ne uzima u obzir udaljenost odredišnog od ishodišnog stupca u kojemu se nalazi kontejner koji je potrebno premjestiti.

5.5.1 Uvjeti primjene konstruktivnih heurističkih pravila

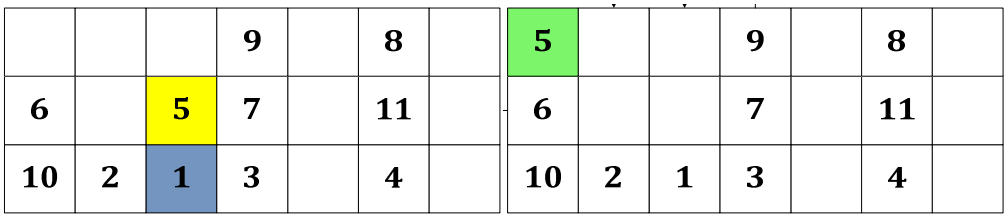
Proces primjene prethodno opisanih heurističkih pravila sastoji se od ispitivanja zadanih uvjeta za pojedino pravilo. Neposredno prije provjere uvjeta za primjenu pojedinog pravila te utvrđivanja potencijalnih rješenja, potrebno je izraditi listu u koju će se svrstavati sva potencijalna rješenja. Dakle, za svako pravilo izrađuje se lista rješenja u kojoj će se svrstavati svi stupci koji zadovoljavaju zadane uvjete opisane u nastavku rada.

Uvjeti koje je potrebno ispitati za primjenu **pravila 1** su:

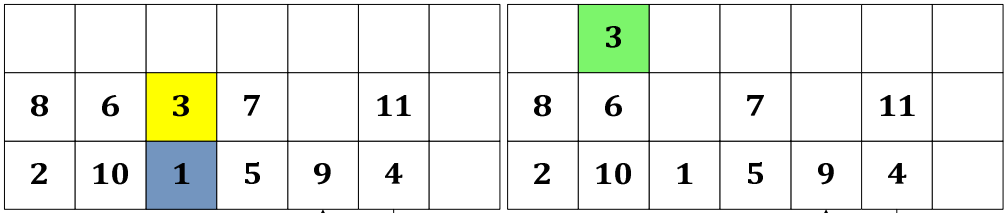
- nalazi li se u stupcu kontejner
- postoji li u stupcu slobodna pozicija s obzirom na ograničenje visine slaganja
- je li prioritet premještanog kontejnera veći od prioriteta kontejnera koji se trenutno nalaze u stupcu

Ako se temeljem navedenih uvjeta utvrdi da postoji takav stupac/stupci tada se on/oni svrstavaju u prethodno izrađenu listu mogućih rješenja. Zatim se kod svih stupaca koji predstavljaju moguća rješenja izračunava premještaj po širini i visini. Udaljenost po širini izračunava se od stupca u kojemu se nalazi kontejner koji se premješta (`indexJ`) do mogućeg stupca za premještaj kontejnera (`moguciJ`), s time da se kao konačno rješenje odnosno odredišni stupac, odabire onaj koji ima najmanji broj premještaja po širini. Ukoliko postoji više stupaca s istim minimalnim brojem premještaja po širini, tada se kao konačno rješenje odabire stupac s najmanjim indeksom. Primjena pravila 1 ilustrativno je prikazana na slici (Slika 24).

Scenarij 1



Scenarij 2



Slika 24 Dva moguća scenarija primjene pravila 1 u odjeljku veličine 3x7

U scenariju 1 potrebno je otpremiti kontejner prioriteta 1 složenog na Kp(2, 2) (označen sivoplavom bojom), no da bi se mogla izvršiti njegova otprema prethodno je potrebno izvršiti radnju premještaja kontejnera prioriteta 5, složenog na Kp(1, 2) (označen žutom bojom). Nakon ispitivanja navedenih uvjeta utvrđeno je da jedino prvi stupac (j=0) zadovoljava sva tri uvjeta, stoga se kao rješenje za premještaj kontejnera prioriteta 5 odabire Kp(0, 0) (označen zelenom bojom).

U scenariju 2 prije otpreme kontejnera prioriteta 1 složenog na Kp (2, 2) (označen sivoplavom bojom) vrši se premještaj kontejnera prioriteta 3 složenog na Kp (1, 2) (označen žutom bojom). Mogući stupci za slaganje kontejnera prioriteta 3 su indeksa 1, 3 i 4. Budući da stupci indeksa 1 i 3 imaju jednaku udaljenost od ishodišnog stupca, kao konačna pozicija za slaganje kontejnera prioriteta 3 (označen zelenom bojom) odabire se stupac manjeg indeksa odnosno stupac indeksa 1 i Kp (0, 1).

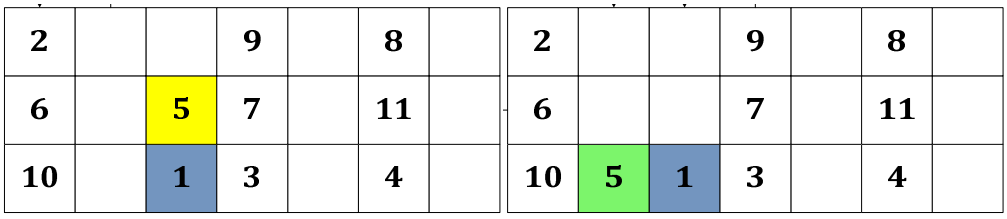
Kada je odabran stupac, odnosno pozicija za slaganje kontejnera koji se premješta, izvršava se izračun visine premještaja kontejnera.

Maksimalna moguća visina stupca jednaka je broju redova. U ovom modelu moguće maksimalne visine slaganja kontejnera ovise o odjeljku koji se promatra, a iznose 3, 4, 5 i 6 redova. Ukoliko je neki od stupaca popunjen do neke od navedenih maksimalnih visina, tada se u ukupan broj premještaja po visini ubraja i rezervna visina koja omogućava izvođenje radnje premještaja kontejnera.

Neposredno prije primjene **Pravila 2** potrebno je ispitati zadovoljava li se sljedeći uvjet :

- postoji li prazan stupac za slaganje kontejnera

Zadovoljenjem ovog uvjeta kao konačno rješenje tj; odredišni stupac odabire se stupac koji je najbliži ishodišnom stupcu u kojem je složen kontejner koji se premješta (Slika 25). I kod primjene ovog pravila, ukoliko postoji više stupaca jednakog minimalnog broja premještaja po širini, kao konačno rješenje odabire se onaj s najmanjim indeksom.



Slika 25 Primjena pravila 2 u odjeljku veličine 3x7

Iz slike je razvidno da je prije otpreme kontejnera prioriteta 1 složenog na Kp (2, 2) (označen sivoplavom bojom) potrebno izvršiti premještaj kontejnera prioriteta 5 složenog na Kp (1, 2) (označen žutom bojom) te se kao konačna pozicija za slaganje kontejnera prioriteta 5 odabire stupac indeksa 1 i Kp (2, 1) (označen zelenom bojom).

Pravilo 3 nešto je složenije te je stoga ispitivanje uvjeta njegove primjene podijeljeno na dva dijela. U prvom dijelu utvrđuju se visine stupaca, a u drugom uspoređuju se visine stupaca koji predstavljaju moguća rješenja.

Uvjeti koji se ispituju kako bi se omogućila primjena ovog pravila su:

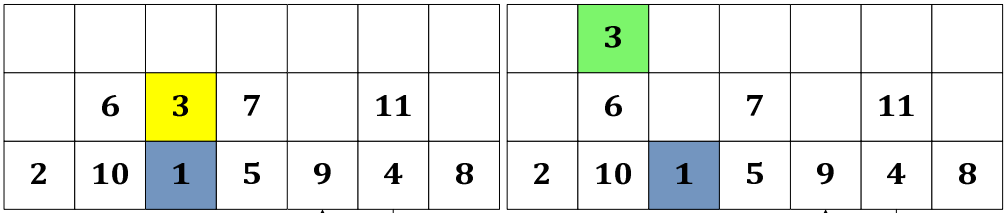
- nalazi li se u stupcu kontejner
- postoji li u stupcu slobodna pozicija s obzirom na ograničenje visine slaganja

Shodno navedenim uvjetima, utvrđuje se postojanje kontejnera u stupcu i slobodne pozicije u stupcu s obzirom na maksimalnu visinu slaganja, međutim, ne zna se koliko se točno kontejnera nalazi u stupcu, odnosno kolika je visina stupca.
Da bi se to utvrdilo potrebno je ispitati još jedan uvjet:

- koliko je redova u stupcu prazno

  Provjera uvjeta obavlja se linijom koda:

Iz prikazanog primjera na slici 26 vidljivo je da nakon provjere prethodno navedenih uvjeta stupci indeksa 1 i 3 predstavljaju moguća rješenja za premještaj kontejnera prioriteta 3 složenog na Kp (1, 2).



Slika 26 Primjena pravila 3 u odjeljku veličine 3x7

Budući da oba stupca imaju isti minimalni broj premještaja po širini, kao konačno rješenje odabire se stupac indeksa 1 i Kp (0, 1).

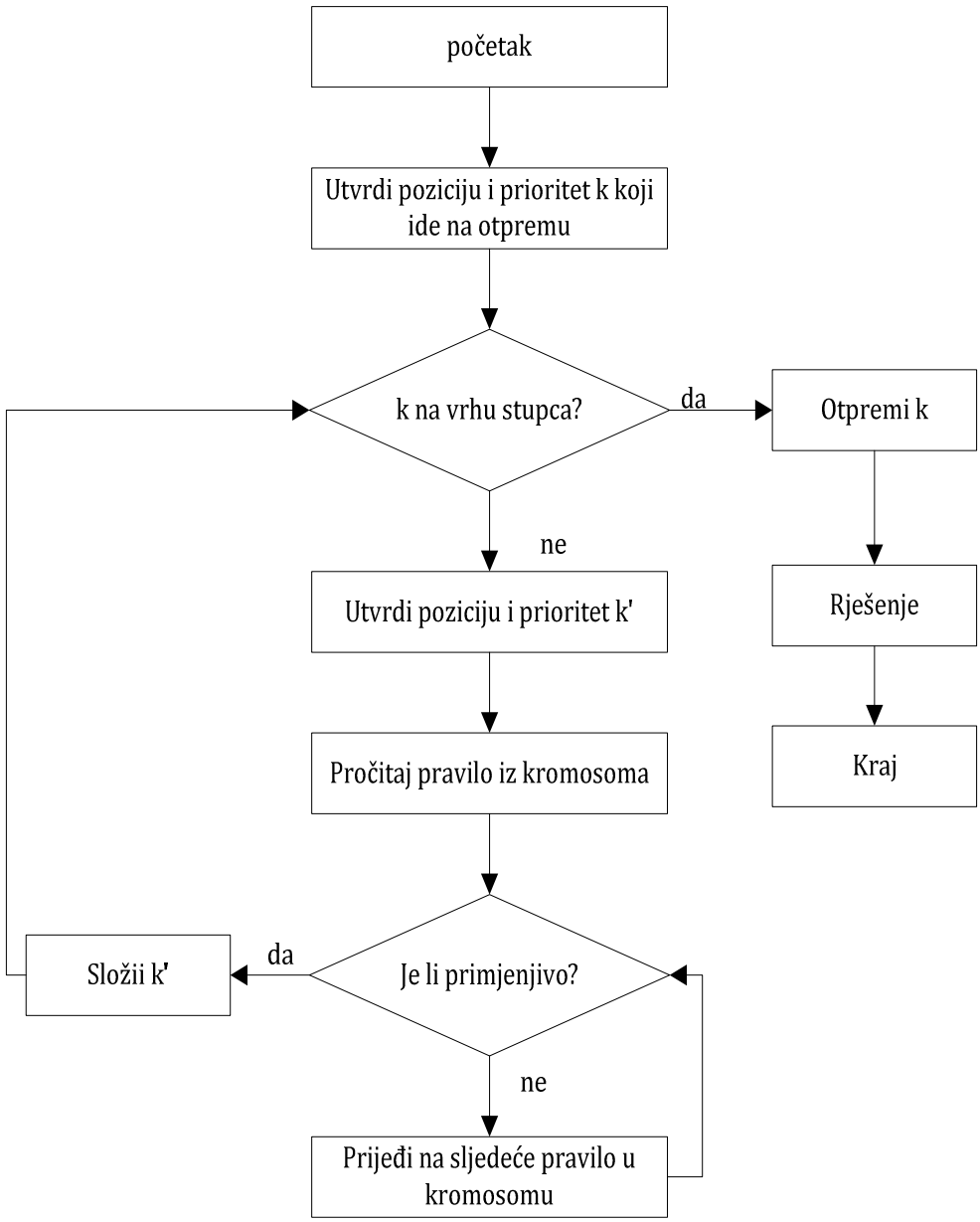
Prilikom primjena **Pravila 4** (Premjesti kontejner u najbliži stupac koji nije popunjen do maksimalne visine slaganja), potrebno je ispitati zadovoljavaju li se sljedeći uvjeti:

- nalazi li se u stupcu kontejner
- postoji li u stupcu slobodna pozicija s obzirom na ograničenje visine slaganja

Dakle, kod ovog pravila stupci koji predstavljaju moguća rješenja ne smiju biti prazni niti popunjeni do maksimalne visine slaganja. Ukoliko bi se uzeo u obzir raspored prikazan na slici 26, tada bi i kod primjene ovog pravila kao konačno rješenje bio odabran stupac indeksa 1. Budući da uvjeti za primjenu pravila 4 omogućavaju najveći broj dopustivih rješenja, u nekim situacijama ta rješenja mogu odgovarati konačnim rješenjima koja su sukladna primjeni pravila 1 i 3.

5.5.2 Logika rada modela u procesu rješavanja problema BRP/CRP

Nakon što su definirani svi uvjeti primjene heurističkih pravila, preostalo je prikazati logiku rada modela u procesu postizanja rješenja. Prethodno je navedeno da su rješenja problema BRP/CRP sadržana u četiri heuristička pravila tj; njihovom redoslijedu primjene. Stoga, kako bi se prikazala međusobna i logička povezanost slučajno generiranog kromosoma tj; početnog rješenja i heurističkih pravila koja predstavljaju konačne kontejnerske pozicije premještaja kontejnera, izrađen je dijagram logike rada modela (Slika 27).



Slika 27 Prikaz logike rada modela u procesu odabira pozicije premještaja kontejnera

Na početku je potrebno utvrditi poziciju i prioritet kontejnera (*k*) koji je potrebno otpremiti. Ukoliko se taj kontejner (*k*) nalazi na vrhu stupca, tada se izvršava direktna otprema s odjeljka, međutim, ako se iznad njega nalazi neki kontejner (*k'*) tada je potrebno izvršiti premještaj *k'*. Premještaj se obavlja sukladno redoslijedu gena (pravila) u kromosomu generiranom slučajnim redoslijedom.

Ako neko od pravila nije primjenjivo, prelazi se na sljedeći gen u kromosomu, odnosno na sljedeće pravilo i tako sve dok neko od pravila ne bude primjenjivo. Kada je pravilo primjenjivo, obavlja se premještaj kontejnera i ispisuje rješenje koje sadrži ukupan broj premještaja te premještaja po visini i širini.

5.5.3 Grafički prikaz rezultata

Za prikaz rezultata odabran je jedan od slučajno generiranih početnih rasporeda za odjeljak veličine 6x7 te udio popunjenosti odjeljka od približno 75% (Slika 28).

	29	**27**		**31**		**10**
	16	**22**	**28**	**26**		**7**
	9	**8**	**23**	**18**		**30**
	24	**5**	**17**	**11**		**21**
	2	**19**	**13**	**4**	**25**	**1**
	14	**15**	**6**	**12**	**20**	**3**

Slika 28 Početni raspored kontejnera unutar odjeljka

Na samom početku temeljem slučajnog odabira generira se kromosom koji predstavlja početno rješenje problema BRP/ CRP za promatrani odjeljak. Za navedeni primjer izrađen je tablični prikaz redoslijeda pravila slučajno generiranog kromosoma (Tablica 5). Pravila navedena u kromosomu određuju pozicije premještaja kontejnera.

Tablica 5 Slučajno generirani kromosom za rješavanje problema BRP/CRP za odjeljak veličine 6x7 i udio popunjenosti od približno 75%

4 3 1 2 2 3 4 1 1 4 1 2 2 2 1 1 1 2 3 3 3 3 2 2 3 3 1 2 3 4 2 1 2 3 1 2 1 4 2 1 2 4 3 2 1 3 3 2 1
2 2 3 3 3 4 3 2 3 3 2 3 2 3 3 2 3 3 1 1 2 1 2 2 1 4 1 2 1 3 3 2 4 3 2 2 3 2 4 2 3 4 3 4 1 2 4 1 3
3 2 3 1 2 3 2 2 1 4 1 1 3 1 3 3 3 2 3 3 3 4 3 2 1 1 2 3 4 2 3 4 2 2 1 1 1 3 3 4 2 3 3 2 2 2 2 2 2
2 4 2 2 3 3 1 2 2 1 3 2 2 1 1 2 2 2 2 2 4 3 3 3 3 2 2 3 1 2 3 3 3 2 3 2 1 3 1

Nakon što je izvršeno 20 evolucija, predloženi model ispisuje konačno rješenje sadržano u vrijednosti *fitness* funkciji. Rezultat *fitness* funkcije predstavlja ukupni broj premještaja te premještaj po širini i visini, dobiven primjenom seta pravila zapisanih u slučajno generiranom kromosomu.

Prvo je potrebno utvrditi poziciju i prioritet ciljnog kontejnera te da li iznad njega ima složenih kontejnera koji blokiraju njegovu direktnu otpremu sa slagališta. Ukoliko postoje kontejneri koji blokiraju otpremu ciljnog kontejnera, potrebno je utvrditi njihov

broj i Kp. Ciljni kontejner označen je sivoplavom bojom, kontejner/kontejneri koji ometa/ometaju otpremu označen/i je/su žutom bojom, a njihove konačne pozicije premještaja označene su zelenom bojom (Slika 29).

	29	27		31		10		29	27		31		
	16	22	28	26		7		16	22	28	26		
	9	8	23	18		30		9	8	23	18	7	
	24	5	17	11		21		24	5	17	11	10	
	2	19	13	4	25	1	21	2	19	13	4	25	
	14	15	6	12	20	3	30	14	15	6	12	20	3

Slika 29 Otprema kontejnera prioriteta 1

Da bi se kontejner prioriteta 1 složen na Kp (4, 6) mogao otpremiti, potrebno je izvršiti premještaj četiriju kontejnera prioriteta 10, 7, 30 i 21. Dakle, prvo se vrši premještaj kontejnera prioriteta 10. Iz tablice 5 je vidljivo da prvi gen ima vrijednost 4, što znači da se mora ispitati je li s obzirom na početni raspored pravilo 4 primjenjivo. Nakon provjere zadanih uvjeta proizlazi da je pravilo 4 moguće primijeniti te da potencijalna rješenja predstavljaju stupci indeksa 3 i 5. Budući da model radi tako da minimizira broj premještaja po širini, kao konačna pozicija premještaja kontejnera odabran je stupac indeksa 5 odnosno Kp (3, 5). Zatim se primjenom pravila vrši premještaj kontejnera prioriteta 7 te se kao konačna i jedina moguća pozicija odabire Kp (2, 5). Vidljivo je da se oba kontejnera premještaju sukladno redoslijedu pravila u slučajno generiranom kromosomu. Nakon toga slijedi premještaj kontejnera prioriteta 30 te ispitivanje uvjeta za primjenu pravila 1 zapisanog u trećem genu. Nakon ispitivanja uvjeta proizlazi da se pravilo 1 ne može primijeniti na premještaj kontejnera prioriteta 30, odnosno da u odjeljku ne postoji stupac u kojem svi složeni kontejneri imaju manji prioritet od kontejnera prioriteta 30, stoga je potrebno ispitati primjenu pravila 2 zapisanog u četvrtog genu kromosoma. Budući da je primjena pravila 2 moguća, kontejner prioriteta 30 premješta se u stupac indeksa o i Kp (5, 0). Preostao je još premještaj kontejnera prioriteta 21, koji nije moguće premjestiti prema pravilu 2 (peti gen), jer unutar odjeljka ne postoji niti jedan prazan stupac pa se primijenilo pravilo 3 koje je predstavljao šesti gen. Dakle, konačna pozicija premještaja kontejnera prioriteta

21 je Kp (0, 4) Nakon izvršenih premještaja kontejnera, kontejner prioriteta 1 ukrcava se na IT te otprema sa slagališta. Iz navedenog može se zaključiti da su prilikom otpreme kontejnera prioriteta 1 izvršene četiri radnje premještaja, u skladu sa pravilima prikazanim u tablici 6.

Tablica 6 Prikaz pravila primijenjenih na premještaj kontejnera prije otpreme kontejnera prioriteta 1

GEN 1	GEN 2	GEN 4	GEN 6
4	3	2	3

Ukupno je ostvareno 14 premještaja po širini i 22 po visini, što znači da vrijednost *fitness* funkcije prilikom otpreme kontejnera prioriteta 1 iznosi 4.036. Od čega je razlika premještaja po širini jednaka 14, a po visini 22. Sljedeći kontejner za otpremu je kontejner prioriteta 2 složen na Kp (4, 1). Da bi se navedeni kontejner mogao otpremiti potrebno je premjestiti četiri kontejnera prioriteta 29, 16, 9 i 24. Konačni razmještaj prikazan je na desnom dijelu slike (Slika 30).

	29	27		31		
	16	22	28	26		
	9	8	23	18	7	
	24	5	17	11	10	
21	2	19	13	4	25	
30	14	15	6	12	20	3

24		27		31		
9		22	28	26		
16		8	23	18	7	
29		5	17	11	10	
21		19	13	4	25	
30	14	15	6	12	20	3

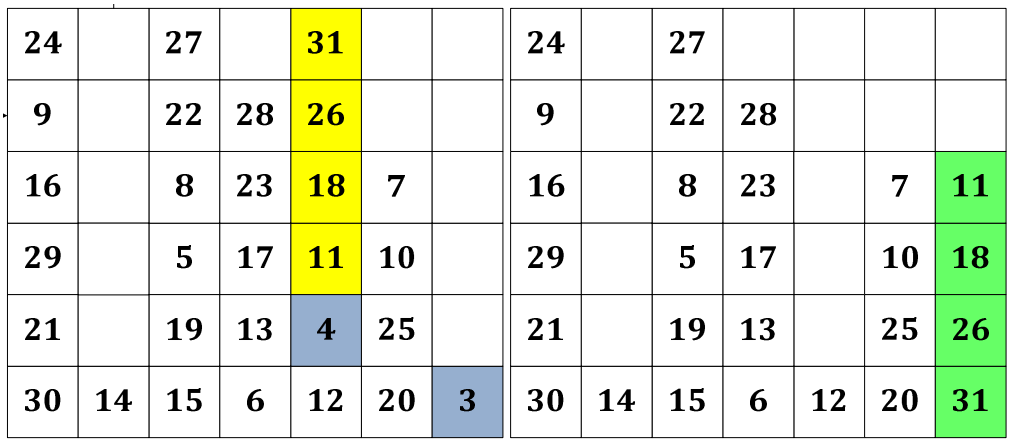
Slika 30:Otprema kontejnera prioriteta 2

Pravila koja su primijenjena, zapisana su u kromosomu koji je prikazan sljedećom tablicom.

Tablica 7 Prikaz pravila primijenjenih na premještaj kontejnera prije otpreme kontejnera prioriteta 2

GEN 7	GEN 8	GEN 9	GEN 10
4	1	1	4

Iz tablice je razvidno da je ukupno izvršeno četiri premještaja kontejnera. Sukladno konačno odabranim pozicijama premještaja kontejnera ostvareno je 4 premještaja po širini i 8 po visini, iz čega proizlazi da vrijednost *fitness* funkcije prilikom otpreme kontejnera prioriteta 2 iznosi 4.012. Kontejner prioriteta 3 koji je složen na Kp (5, 6) moguće je direktno otpremiti iz odjeljka, dok je prije otpreme kontejnera prioriteta 4 složenog na Kp (4, 4) potrebno izvršiti četiri premještaja kontejnera prioriteta 31, 26, 18, i 11. Konačni razmještaj prikazan je na desnom dijelu slike (Slika 31).



Slika 31 Otprema kontejnera prioriteta 3 i 4

Pravila koja su primijenjena na premještaj kontejnera predstavljena su tabličnim prikazom kromosoma (Tablica 8).

Tablica 8 Prikaz pravila primijenjenih na premještaj kontejnera prije otpreme kontejnera prioriteta 4

GEN 12	GEN 15	GEN 16	GEN 17
2	1	1	1

Neposredno prije otpreme kontejnera prioriteta 4 izvršeno je ukupno 4 premještaja s 8 premještaja po širini i 20 po visini, iz čega proizlazi da vrijednost *fitness* funkcije prilikom otpreme kontejnera prioriteta 4 iznosi 4.028. Od toga je duljina (udaljenost) premještaja po širini jednaka 8, a po visini 20.

Prije otpreme kontejnera prioriteta 5 složenog na Kp (3, 2), potrebno je izvršiti premještaj tri kontejnera prioriteta 27, 22 i 8. Konačni razmještaj prikazan je na desnom dijelu slike (Slika 32).

24		27				
9		22	28			
16		8	23		7	11
29		5	17		10	18
21		19	13		25	26
30	14	15	6	12	20	31

24						
9			28			
16			23		7	11
29	8		17		10	18
21	27	19	13	22	25	26
30	14	15	6	12	20	31

Slika 32 Otprema kontejnera prioriteta 5

Pravila koja su primijenjena na premještaj navedenih kontejnera prikazana su tablicom (Tablica 9).

Tablica 9 Prikaz pravila primijenjenih na premještaj kontejnera prije otpreme kontejnera prioriteta 5

GEN 19	GEN 20	GEN 21
3	3	3

Dakle, da bi se kontejner prioriteta 5 otpremio iz odjeljka izvršeno je 3 premještaja kontejnera s ukupno ostvarenih 4 premještaja po širini i 12 po visini. Temeljem navedenog proizlazi da vrijednost *fitness* funkcije prilikom otpreme kontejnera prioriteta 5 iznosi 3.016.

Sljedeći kontejner koji ide na otpremu je kontejner prioriteta 6 složen na Kp (5, 3). Potrebno je izvršiti premještaj četiriju kontejnera prioriteta 28, 23, 17 i 13. Konačne pozicije razmještaja prikazane su na desnom dijelu slike (Slika 33).

24						
9			28			
16			23		7	11
29	8		17		10	18
21	27	19	13	22	25	26
30	14	15	6	12	20	31

24						
9		13				
16		17			7	11
29	8	28		23	10	18
21	27	19		22	25	26
30	14	15		12	20	31

Slika 33 Otprema kontejnera prioriteta 6

Konačne pozicije navedenih kontejnera odabrane su primjenom pravila prikazanih u tablici (Tablica 10).

Tablica 10 Prikaz pravila primijenjenih na premještaj kontejnera prije otpreme kontejnera prioriteta 6

GEN 22	GEN 25	GEN 26	GEN 27
3	3	3	1

Kako bi se omogućila otprema kontejnera prioriteta 6 izvršeno je 4 premještaja sa 4 premještaja po širini te 7 po visini, iz čega proizlazi da vrijednost *fitness* funkcije prilikom otpreme kontejnera prioriteta 6 iznosi 4.011.

Kontejneri prioriteta 7 i 8 direktno se otpremaju iz odjeljka. Prije otpreme kontejnera prioriteta 9 složenog na Kp (1,0) potrebno je izvršiti premještaj kontejnera prioriteta 24. Konačna pozicija slaganja kontejnera prioriteta 24 prikazana je na desnom dijelu slike (Slika 34).

24													
9		**13**							**13**				
16		**17**			**7**	**11**	**16**		**17**				**11**
29	**8**	**28**		**23**	**10**	**18**	**29**		**28**		**23**	**10**	**18**
21	**27**	**19**		**22**	**25**	**26**	**21**	**27**	**19**		**22**	**25**	**26**
30	**14**	**15**		**12**	**20**	**31**	**30**	**14**	**15**	**24**	**12**	**20**	**31**

Slika 34 Otprema kontejnera prioriteta 7,8 i 9

Premještaj navedenog kontejnera izvršava se primjenom pravila 2 zapisanog u genu 28. Vrijednost *fitness* funkcije prilikom otpreme kontejnera prioriteta 9 iznosi 1.010, što znači da je ostvarena 1 radnja premještaja sa 3 premještaja po širini i 7 po visini.

Kontejner prioriteta 10 i 11 direktno se otpremaju iz odjeljka. Da bi se otpremio kontejner prioriteta 12 složen na Kp (5, 4), potrebno je izvršiti premještaj dvaju kontejnera prioriteta 23 i 22 (Slika 35).

		13							13				
16		17				11	16		17				
29		28		23	10	18	29		28	22			18
21	27	19		22	25	26	21	27	19	23		25	26
30	14	15	24	12	20	31	30	14	15	24		20	31

Slika 35 Otprema kontejnera prioriteta 10, 11 i 12

Razmještaj navedenih kontejnera izvršava se sukladno pravilima prikazanim u tablici (Tablica 11).

Tablica 11 Prikaz pravila primijenjenih na premještaj kontejnera prije otpreme kontejnera prioriteta 12

GEN 29	GEN 35
3	4

Vrijednost *fitness* funkcije prilikom otpreme kontejnera prioriteta 12 iznosi 2.004, što znači da su izvršene 2 radnje premještaja sa 2 premještaja po širini iznosi 2 i 2 po visini. Kontejner prioriteta 13 direktno se otprema iz odjeljka, dok je za otpremu kontejnera prioriteta 14 složenog na Kp (5, 1) potrebno izvršiti premještaj kontejnera prioriteta 27. Konačna pozicija slaganja navedenog kontejnera prikazana je u desnom dijelu slike(Slika 36).

		13											
16		17					16		17				
29		28	22			18	29		28	22			18
21	27	19	23		25	26	21		19	23		25	26
30	14	15	24		20	31	30		15	24	27	20	31

Slika 36 Otprema kontejnera prioriteta 13 i 14

Premještaj kontejnera prioriteta 27 izvršen je primjenom pravila 2 zapisanog u genu 36. Vrijednost *fitness* funkcije prilikom otpreme kontejnera prioriteta 14 iznosi 1.012, što znači da je izvršena jedna radnja premještaja s 3 premještaja po širini te 9 po visini .

Da bi se otpremio kontejner prioriteta 15 složen na Kp (5, 2), potrebno je izvršiti premještaj triju kontejnera prioriteta 17, 28 i 19. Konačni razmještaj navedenih kontejnera prikazan je u desnom dijelu slike (Slika 37).

16		**17**					**16**			**17**			
29		**28**	**22**			**18**	**29**			**22**			**18**
21		**19**	**23**		**25**	**26**	**21**	**19**		**23**		**25**	**26**
30		**15**	**24**	**27**	**20**	**31**	**30**	**28**		**24**	**27**	**20**	**31**

Slika 37 Otprema kontejnera prioriteta 15

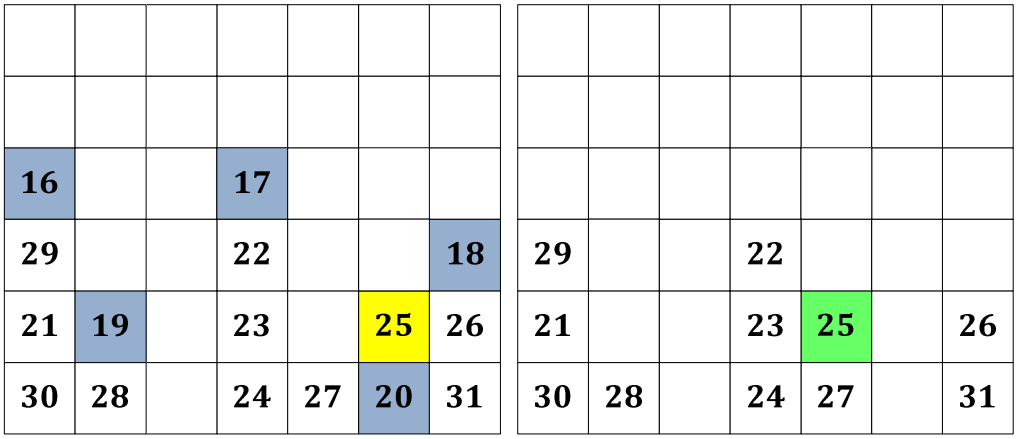
Razmještaj navedenih kontejnera izvršen je primjenom pravila prikazanih u tablici (Tablica 12).

Tablica 12 Prikaz pravila primijenjenih na premještaj kontejnera prije otpreme kontejnera prioriteta 5

GEN 37	GEN 38	GEN 39
1	2	3

Vrijednost *fitness* funkcije prilikom otpreme kontejnera prioriteta 15 iznosi 3.005, što znači da su izvršene 3 radnje premještaja s 3 premještaja po širini te 2 po visini.

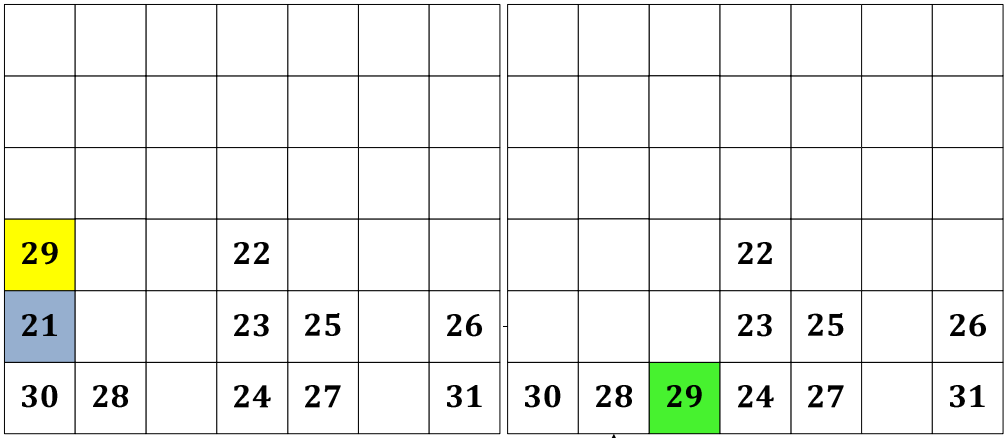
Kontejneri prioriteta 16, 17, 18 i 19 direktno se otpremaju iz odjeljka. No, da bi se otpremio kontejner prioriteta 20 složen na Kp (5, 5) prethodno je potrebno izvršiti premještaj kontejnera prioriteta 25 (Slika 38).



Slika 38 Otprema kontejnera prioriteta 16, 17, 18, 19 i 20

Premještaj kontejnera prioriteta izvršava se primjenom pravila 1 zapisanog u genu 40. Vrijednost *fitness* funkcije prilikom otpreme kontejnera prioriteta 20 iznosi 1.001, što znači da je izvršena 1 radnja premještaja s 1 premještajem po širini te 0 premještaja po visini, budući da je visina konačne Kp jednaka visini početne Kp.

Sljedeći kontejner za otpremu je kontejner prioriteta 21 složen na Kp (4, 0) (Slika 39).



Slika 39 Otprema kontejnera prioriteta 21

Međutim, da bi se navedeni kontejner otpremio potrebno je izvršiti premještaj kontejnera prioriteta 29, što je ujedno i posljednja radnja premještaja.

Premještaj kontejnera prioriteta 29 izvršava se primjenom pravila 2 zapisanog u genu 41. Vrijednost *fitness* funkcije prilikom otpreme kontejnera prioriteta 21 iznosi 1.004, što znači da je izvršena 1 radnja premještaja s 2 premještanja po širini i 2 po visini. Kontejneri prioriteta 22-31 direktno se otpremaju iz odjeljka. Temeljem prethodno detaljno opisanog postupka dobivanja rezultata za promatrani odjeljak, izrađen je tablični prikaz rezultata (Tablica 13).

Tablica 13 Strukturni prikaz rezultata za odjeljak veličine 6x7 i udio popunjenosti od približno 75%

Prioritet kontejnera	Kromosom/primijenjena pravila za premještaj	f(x)
1	4 3 2 3	4.036
2	4 1 1 4	4.012
3	direktna otprema	-
4	2 1 1 1	4.028
5	3 3 3	3.016
6	3 3 3 1	4.011
7	direktna otprema	-
8	direktna otprema	-
9	2	1.010
10	direktna otprema	-
11	direktna otprema	-
12	3 4	2.004
13	direktna otprema	-
14	2	1.012
15	1 2 3	3.005
16	direktna otprema	-
17	direktna otprema	-
18	direktna otprema	-
19	direktna otprema	-
20	1	1.001
21	2	1.004
22	direktna otprema	-
23	direktna otprema	-
24	direktna otprema	-
25	direktna otprema	-
26	direktna otprema	-
27	direktna otprema	-
28	direktna otprema	-
29	direktna otprema	-
30	direktna otprema	-
31	direktna otprema	-
UKUPNO	**4 3 2 3 4 1 1 4 2 1 1 1 3 3 3 3 3 3 1 2 3 4 2 1 2 3 1 2**	**28.139**

Iz tablice je vidljivo da je prilikom otpreme 31 kontejnera iz odjeljka, izvršeno 28 radnji premještaja s ukupno ostvarenih 48 premještaja po širini i 91 premještajem po visini te da vrijednost *fitness* funkcije iznosi 28.139. U tom smislu pravila koja su primijenjena na premještaj prikazana su u posljednjem retku drugog stupca.

6. EKSPERIMENTALNO TESTIRANJE MODELA

Testiranje rezultata modela obavljeno je na prijenosnom računalu 32-bitnog operativnog sustava *Windows 7 Professional* s procesorom Intel®Core™2DuoCPU T6500 2.10 GHz te 4 GB radne memorije.

Kako bi se usporedile performanse rada modela, definirane su različite veličine odjeljka, udjeli popunjenosti i početni rasporedi kontejnera. Zadane veličine odjeljka su sukladne specifikacijama tehnološki najsuvremenijih RTG dizalica, a to su redom: 3x7,4x7,5x7 i 6x7 s time da je parametar broj stupaca u odjeljku konstantan u sve četiri varijante s mogućnošću slaganja do 7 kontejnera u širinu. Parametar broja redova poprima vrijednosti slaganja od 3 do 6 kontejnera u visinu.

Broj složenih kontejnera unutar navedenih odjeljaka dobiven je iz izraza[30]:

$$N_k = WxH - (H - 1) \tag{7}$$

gdje je:

N_k- ukupan broj kontejnera složenih unutar odjeljka

W-ukupan broj stupaca u odjeljku

H- ukupan broj redova u odjeljku

Navedeni izraz određuje maksimalno moguć broj kontejnera unutar odjeljka, odnosno osigurava dovoljan broj slobodnih kontejnerskih pozicija za izvođenje premještaja kontejnera unutar odjeljka. Prema izrazu maksimalni udio popunjenosti odjeljka iznosi približno 90%. Također, model je testiran i za udjele popunjenosti odjeljka od približno 55% i 75%. Različiti početni raspored kontejnera s dodijeljenim prioritetima unutar odjeljka, stvoreni su na osnovu programske skripte koja je izrađena za potrebe ovog rada. Na slikama u nastavku na odabranim primjerima prikazana su tri scenarija popunjenosti odjeljka s ukupnim udjelima od 55% (označeno brojem 1), 75% (označeno brojem 2) i 90% (označeno brojem 3)(Slika 41).

3x7

(1)

		11	9		8	
6		10	7		5	
2		1	3		4	

(2)

	12		11	10	14	7
	8		15	5	9	13
	6		1	3	4	2

(3)

	12		11	10	14	7
18	8	19	15	5	9	13
16	6	17	1	3	4	2

4x7

(1)

			12	9		15
11			7	8		13
14			3	6		5
1			10	4		2

(2)

12		14	20		7	
15		19	2		18	13
10	4	16	8		3	21
1	11	6	17		9	5

(3)

12			20	25	7	
15	23	19	2	24	18	13
10	4	16	8	14	3	21
1	11	6	17	22	9	5

Slika 40 Početni raspored odjeljaka 3x7 i 4x7 s udjelom popunjenosti od približno 55%, 75% i 90%

S obzirom na broj kontejnera složenih unutar odjeljaka, izrađeno je 12 različitih odjeljaka. Za svaki odjeljak model je pokrenut 10 puta, što znači da je ukupan broj pokretanja modela jednak 120. Dodatno testiranje modela provedeno je za sve veličine odjeljaka s udjelom popunjenosti od približno 90%, a s ciljem usporedbe dobivenih rezultata s rezultatima modela predloženih od drugih autora.

Testiranje modela provedeno je u jednakim uvjetima kao što su ga provodili i autori koji se spominju u ovoj usporedbi rezultata. U programskom jeziku R za svaki je odjeljak kreirano još 39 početnih rasporeda kontejnera unutar odjeljka (vidi privitak 2).

5x7

(1)

	19					
	13	16		15		
7	10	12		14	18	
5	9	4		8	17	
3	1	2		6	11	

(2)

19		16		10		22
21		13		5		14
17	23	11	29	1		7
4	15	2	18	20		25
6	12	8	26	3	24	9

(3)

19		16		10		22
21	30	13	31	5		14
17	23	11	29	1	28	7
4	15	2	18	20	27	15
6	12	8	26	3	24	9

6x7

(1)

		21		22		
		19		12		
	23	15		10	16	
	8	11		7	14	20
	5	6		4	13	18
	2	3		1	9	17

(2)

	29	27		31		10
	16	22	28	26		7
	9	8	23	18		30
	24	5	17	11		21
	2	19	13	4	25	1
	14	15	6	12	20	3

(3)

		16		37	31	12
24	13	3		30	19	15
20	36	29	0	21	34	11
6	17	9	7	5	23	1
8	25	14	32	27	4	2
22	10	18	28	26	33	35

Slika 41 Početni raspored odjeljaka 5x7 i 6x7 s udjelom popunjenosti od približno 55%, 75% i 90%

Osim navedenog, provedeno je i eksperimentalno ispitivanje učinkovitosti rada modela u pogledu kvalitete dobivenih rezultata te vremena rada izvođenja modela. Učinkovitost modela testirana je s obzirom na različite veličine parametara broj evolucije, veličine populacije, vjerojatnost križanja i vjerojatnost mutacije. U nastavku rada detaljno su prikazani i analizirani svi dobiveni rezultati.

7. ANALIZA REZULTATA MODELA

U ovom poglavlju analiziraju se rezultati dobiveni predloženim modelom izrađenim primjenom GA. Analiza rezultata podijeljena je u tri dijela. U prvom se dijelu prikazuje po deset dobivenih različitih setova pravila s pripadajućim rezultatima za svih dvanaest varijanti odjeljaka. Potrebno je napomenuti da analizirani rezultati vrijede samo za početne rasporede odjeljka prikazanih na slikama 40 i 41. Drugi dio analize izrađen je za različite početne rasporede svih četiriju veličina odjeljka i udjela popunjenosti od približno 90%, a s ciljem dobivanja prosječnog broja premještaja te usporedbe kvalitete dobivenih rezultata s dobivenim rezultatima drugih autora. Na kraju je izrađena analiza osjetljivosti GA, odnosno provedeno je ispitivanje utjecaja različitih veličina parametara na kvalitetu dobivenih rezultata.

7.1 Analiza rezultata za odjeljak veličine 3x7

U ovom su dijelu rada prikazani i analizirani dobiveni rezultati prilikom deset pokretanja modela. Rezultati obuhvaćaju sve tri varijante odjeljka veličine 3x7, a prikazani su u narednim tablicama. Tablice se sastoje od sljedećih podataka: broja pokretanja modela, broja kontejnera, redoslijeda primijenjenih pravila, broja premještaja, premještaja po širini i visini, vrijednosti *fitness* funkcije, generacije u kojoj je dobiven rezultat te potrebnog vremena izvođenja modela. Rezultati prikazani u tablici 14 odnose se na odjeljak unutar kojeg je složeno 11 kontejnera (Tablica 14).

Iz tablice je vidljivo da se tijekom deset pokretanja modela kao rezultat dobiva deset različitih setova pravila. Unatoč tome, svaki put je dobiven isti optimalni rezultat čija vrijednost *fitness* funkcije iznosi 7.014, što znači da je izvršeno 7 radnji premještaja kontejnera sa 7 premještaja po širini i 7 premještaja po visini. Iz prikazane tablice može se zaključiti da je za postizanje optimalnog rezultata moguće primijeniti različit set pravila premještaja. Daljnjim pokretanjima modela moguće je dobivanje rezultata sa setovima pravila različitim od prikazanih, ali uz isti optimalni rezultat.

Prilikom svih deset pokretanja modela optimalni rezultat dobiven je u nultoj generaciji[16].

[16] Generacija rezultata odnosi se na broj evolucija populacije, a u modelu iznosi 20. Numeriranje započinje s nulom, što znači da je prva generacija rješenja označena s 0, a posljednja sa 19.

Tablica 14 Rezultati odjeljka veličine 3x7 i udjela popunjenosti od približno 55%

Broj pokretanja	Broj kontejnera	Redoslijed primijenjenih pravila[17]	Najčešće pravilo	Broj premještaja	Premještaj po širini	Premještaj po visini	Vrijednost fitness funkcije	Generacija (iteracija) rezultata	Vrijeme izvođenja modela
1	11	2 1 3 2 3 2 4	2	7	7	7	7.014	0	1'12"
2	11	2 3 3 2 4 2 1	2	7	7	7	7.014	0	1'13"
3	11	2 4 4 2 3 2 3	2	7	7	7	7.014	0	1'13"
4	11	2 3 1 2 1 2 1	1, 2	7	7	7	7.014	0	1'15"
5	**11**	**2 1 1 2 3 2 4**	**2**	**7**	**7**	**7**	**7.014**	**0**	**1'14"**
6	11	2 3 1 2 3 2 1	2	7	7	7	7.014	0	1'16"
7	11	2 3 4 2 3 2 3	2, 3	7	7	7	7.014	0	1'14"
8	11	2 3 3 2 1 2 1	2	7	7	7	7.014	0	1'17"
9	11	2 3 3 2 3 2 3	3	7	7	7	7.014	0	1'13"
10	11	2 3 4 2 4 2 1	2	7	7	7	7.014	0	1'14"

Navedeno znači da model vrlo brzo dolazi do rezultata za probleme malih dimenzija. Jedine razlike u radu modela uočavaju se kod vremena koje je potrebno modelu za postizanje optimalnog rezultata. U tom smislu, najkraće vrijeme izvođenja modela iznosi 1'12"a najdulje 1'17". Prosječno vrijeme izvođenja modela iznosi 1 minutu i 14 sekundi. Ukoliko bi se prilikom rješavanja ovog problema smanjio broj evolucija i/ili veličina populacije, utoliko bi navedena vremena izvođenja modela bila još kraća. Također, potrebno je istaknuti da na vrijeme izvođenja modela uvelike utječe početni rezultat, odnosno slučajnim odabirom stvoreni kromosom. Primjerice, ako slučajno generirani kromosom sadrži velik broj pravila koja u datom trenutku nisu primjenjiva na premještaj kontejnera, model stalno ispituje uvjete primjene pojedinog pravila te traži pravilo koje je primjenjivo, što dodatno produljuje vrijeme rada modela u procesu postizanja rezultata. Stoga se može zaključiti da rezultatu s najkraćim vremenom izvođenja modela prethodi „najbolji“ početni rezultat, odnosno kromosom koji je sadržavao veliki broj pravila koja su bila odmah primjenjiva na premještaj kontejnera.

[17] Kod ovog odjeljka za grafički prikaz rezultata odabran je set pravila dobivenih prilikom petog pokretanja modela (privitak 4).

Na kraju, potrebno je istaknuti da su sa stajališta optimalnosti svi prikazani rezultati jednako dobri, a da je u pogledu vremena izvođenja modela, odnosno računalnog vremena rješavanja modela, najbolji rezultat dobiven prilikom prvog mjerenja.
Sljedeći prikazani rezultati odnose se na odjeljak unutar kojeg je složeno 15 kontejnera (Tablica 15).

Tablica 15 Rezultati odjeljka veličine 3x7 i udjela popunjenosti od približno 75%

Broj pokretanja	**Broj kontejnera**	**Redoslijed primijenjenih pravila**	**Najčešće pravilo**	**Broj premještaja**	**Premještaj po širini**	**Premještaj po visini**	**Vrijednost *fitness* funkcije**	**Generacija rezultata**	**Vrijeme izvođenja**
1	15	2 2 3 2 3 4 2 1 3 3	2, 3	10	17	21	10.038	0	1'54"
2	15	2 2 3 2 4 4 2 4 3 3	2	10	17	21	10.038	0	1'50"
3	15	2 2 1 2 3 1 2 4 1 3	2	10	17	21	10.038	0	1'46"
4	15	2 2 1 2 4 1 2 3 1 3	2	10	17	21	10.038	0	1'44"
5	15	2 2 3 2 3 1 2 4 3 4	2	10	17	21	10.038	0	1'45"
6	15	2 2 4 2 3 4 2 3 4 4	2, 4	10	17	21	10.038	0	1'46"
7	15	2 2 3 2 3 4 2 3 1 4	2	10	17	21	10.038	0	1'48"
8	15	2 2 4 2 4 4 2 3 3 4	2, 4	10	17	21	10.038	0	1'48"
9	15	2 2 4 2 1 1 2 4 1 3	2	10	17	21	10.038	0	1'47"
10	15	2 2 4 2 1 4 2 3 1 4	2	10	17	21	10.038	0	1'46"

Jednako kao i u prethodnom primjeru i kod ovog se odjeljka svakim pokretanjem modela dobiva različit redoslijed pravila uz postizanje istog optimalnog rezultata čija vrijednosti *fitness* funkcije iznosi 10.038. Znači da se izvodi 10 radnji premještaja kontejnera s ukupno 17 premještaja po širini i 21 premještaj po visini. Model već u nultoj iteraciji postiže optimalan rezultat, a navedeno upućuje na činjenicu da je reduciranjem broja generacija moguće zadržati istu kvalitetu te skratiti vrijeme izvođenja modela. Vrijeme izvođenja modela varira ovisno o setu pravila , a iznosi od 1'44" do 1'54". Odnosno, prosječno vrijeme izvođenja modela iznosi 1 minutu i 47 sekundi.

Posljednji analizirani rezultati odjeljka veličine 3x7 odnose se na udio popunjenosti odjeljka od približno 90%, a prikazani su u tablici 16.

Tablica 16 Rezultata odjeljka veličine 3x7 i udjela popunjenosti od približno 90%

Broj pokretanja	Broj kontejnera	Redoslijed primijenjenih pravila	Najčešće pravilo	Broj premještaja	Premještaj po širini	Premještaj po visini	Vrijednost *fitness* funkcije	Generacija rezultata	Vrijeme izvođenja
1	19	3 3 2 4 2 4 2 3 2 1 2 2 2	2	13	26	32	13.058	0	2'33"
2	19	4 3 2 4 2 1 2 1 2 1 2 2 2	2	13	26	32	13.058	0	2'30"
3	19	4 4 2 4 2 4 2 3 2 1 2 2 2	2	13	26	32	13.058	0	2'26"
4	19	4 3 2 3 2 4 2 4 2 1 2 2 2	2	13	26	32	13.058	0	2'27"
5	19	3 3 2 3 2 1 2 3 2 1 2 2 2	2	13	26	32	13.058	0	2'27"
6	19	3 1 2 3 2 1 2 1 2 1 2 2 2	2	13	26	32	13.058	0	2'24"
7	19	3 4 2 4 2 1 2 4 2 1 2 2 2	2	13	26	32	13.058	0	2'27"
8	19	3 3 2 3 2 1 2 1 2 1 2 2 2	2	13	26	32	13.058	0	2'29"
9	19	4 4 2 3 2 4 2 3 2 1 2 2 2	2	13	26	32	13.058	0	2'27"
10	19	3 3 2 3 2 4 2 3 2 1 2 2 2	2	13	26	32	13.058	0	2'30"

Kao i kod prethodno analiziranih odjeljaka svakim pokretanjem modela postiže se isti optimalni rezultat čija vrijednost *fitness* funkcije iznosi 13.058. Iz vrijednosti *fitness* funkcije može se zaključiti da je prilikom otpreme 19 kontejnera iz odjeljka izvršeno 13 radnji premještaja kontejnera, s ukupno 26 premještaja po širini i 32 premještaja po visini. Optimalni rezultat i dalje se postiže u nultoj generaciji, dok prosječno vrijeme izvođenja modela iznosi 2 minute i 28 sekundi.

Iz tablica 14, 15 i 16 može se zaključiti da model, neovisno o udjelu popunjenosti odjeljka, uvijek postiže optimalno rješenje te da ga postiže u nultoj iteraciji uz relativno kratko vrijeme izvođenja. Sa stajališta učestalosti primjene pojedinog pravila, a na

osnovu prikazanih tablica, može se zaključiti da je za premještaj kontejnera unutar odjeljka veličine 3x7 ukupno najčešće primjenjivano pravilo broj 2.

7.2 Analiza rezultata za odjeljak veličine 4x7

U tablicama u nastavku prikazani su dobiveni rezultati za tri varijante udjela popunjenosti odjeljka veličine 4x7. U tom smislu tablica 17 prikazuje rezultate odjeljka s udjelom popunjenosti od približno 55 %.

Tablica 17 Rezultati odjeljka veličine 4x7 i udjela popunjenosti od približno 55%

Broj pokretanja	**Broj kontejnera**	**Redoslijed primijenjenih pravila[18]**	**Najčešće pravilo**	**Broj premještaja**	**Premještaj po širini**	**Premještaj po visini**	**Vrijednost *fitness* funkcije**	**Generacija rezultata**	**Vrijeme izvođenja**
1.	15	2 2 2 3 1 3 1 4 4 1	1, 2	10	11	16	10.027	0	1'53"
2.	15	2 2 2 4 4 3 4 4 1 4	4	10	11	16	10.027	0	1'54"
3.	15	2 2 2 3 4 1 1 3 1 4	1, 2	10	11	16	10.027	0	1'55"
4.	15	2 2 2 4 4 3 4 3 4 1	4	10	11	16	10.027	0	1'52"
5.	15	2 2 2 3 4 1 4 3 4 1	2, 4	10	11	16	10.027	0	1'51"
6.	15	2 2 2 3 1 4 1 3 4 4	2, 4	10	11	16	10.027	0	1'54"
7.	15	2 2 2 3 1 1 1 3 4 1	1	10	11	16	10.027	0	1'54"
8.	**15**	**2 2 2 1 1 4 4 1 1 1**	**1**	**10**	**11**	**16**	**10.027**	**0**	**1'51"**
9.	15	2 2 2 1 1 1 4 4 1 1	1	10	11	16	10.027	0	1'51"
10.	15	2 2 2 3 1 3 1 3 4 4	2, 3	10	11	16	10.027	0	1'51"

Iz tablice je razvidno da se svakim pokretanjem modela postiže optimalan rezultat čija vrijednost *fitness* funkcije iznosi 10.027, što znači da je izvršeno 10 radnji premještaja kontejnera s ukupno 11 premještaja po širini i 16 premještaja po visini. Također, može se zaključiti da se optimalan rezultat postiže primjenom različitih setova pravila premještaja.

[18] Kod ovog odjeljka za grafički prikaz rezultata odabran je set pravila dobivenih prilikom osmog pokretanja modela (vidi privitak 4).

Prilikom svih deset pokretanja modela rezultat je dobiven u nultoj generaciji (iteraciji). Najkraće vrijeme izvođenja modela iznosi 1'51", a ostvareno je prilikom petog, osmog i devetog pokretanja modela. Prosječno vrijeme izvođenja modela iznosi 1 minutu i 53 sekunde.

I kod odjeljka s udjelom popunjenosti od približno 75% predloženi model svakim pokretanjem dobiva isti optimalni rezultat od ukupno 14 radnji premještaja kontejnera s 35 premještaja po širini i 41 premještaja po visini, iz čega proizlazi da vrijednost *fitness* funkcije prilikom otpreme 21 kontejnera iz odjeljka iznosi 14.706 (Tablica 18).

Tablica 18 Rezultati odjeljak veličine 4x7 i udjela popunjenosti od približno 75%

Broj pokretanja	**Broj kontejnera**	**Redoslijed primijenjenih pravila**	**Najčešće pravilo**	**Broj premještaja**	**Premještaj po širini**	**Premještaj po visini**	**Vrijednost *fitness* funkcije**	**Generacija rezultata**	**Vrijeme izvođenja**
1.	21	2 4 3 2 4 3 4 4 4 1 2 1 2 1	4	14	35	41	14.076	2	3'06"
2.	21	2 4 1 2 4 3 4 4 4 1 2 1 2 1	4	14	35	41	14.076	4	3'05"
3.	21	2 4 3 2 1 1 4 3 4 1 2 1 2 1	1	14	35	41	14.076	3	3'32"
4.	21	2 4 3 2 1 3 1 3 4 1 2 1 2 1	1	14	35	41	14.076	2	3'10"
5.	21	2 4 1 2 4 1 1 4 4 1 2 1 2 1	1	14	35	41	14.076	2	3'24"
6.	21	2 4 3 2 4 1 4 3 4 1 2 1 2 1	1, 2, 4	14	35	41	14.076	3	3'29"
7.	21	2 4 3 2 4 1 1 3 4 1 2 1 2 1	1	14	35	41	14.076	4	3'01"
8.	21	2 4 3 2 1 3 4 4 4 1 2 1 2 1	1, 2, 4	14	35	41	14.076	3	3'36"
9.	21	2 4 3 2 4 3 4 3 4 1 2 1 2 1	2, 4	14	35	41	14.076	3	3'10"
10.	21	2 4 3 2 1 1 1 4 4 1 2 1 2 1	1	14	35	41	14.076	3	3'11"

Do sada je optimalan rezultat uvijek dobiven u nultoj iteraciji, međutim za problem ovih dimenzija modelu je potreban veći broj iteracija kako bi se postigao optimalan rezultat. U prosjeku potrebne su tri iteracije za postizanje optimalnog rezultata, uz prosječno vrijeme izvođenja modela u trajanju od 3 minute i 16 sekundi.

Model zadržava jednaka obilježja rada i kod odjeljka s udjelom popunjenosti od približno 90% (Tablica 19).

Tablica 19 Rezultati odjeljka veličine 4x7 i udjela popunjenosti od približno 90%

Broj pokretanja	Broj kontejnera	Redoslijed primijenjenih pravila	Najčešće pravilo	Broj premještaja	Premještaj po širini	Premještaj po visini	Vrijednost *fitness* funkcije	Generacija rezultata	Vrijeme izvođenja
1.	25	3 3 3 2 4 3 4 4 1 3 3 3 2 3 2 1 3 2 2 1	3	20	45	56	20.101	3	4'21"
2.	25	3 3 4 2 4 3 1 4 1 3 4 4 2 3 2 4 3 2 2 1	4	20	45	56	20.101	1	3'55"
3.	25	4 3 3 2 4 3 4 4 1 4 3 3 2 3 2 4 3 2 2 1	3	20	45	56	20.101	2	3'51"
4.	25	4 3 4 2 1 3 1 4 1 4 4 4 2 3 2 4 1 2 2 1	4	20	45	56	20.101	0	3'59"
5.	25	3 3 3 2 4 3 1 4 1 4 3 3 2 1 2 4 4 2 2 1	3	20	45	56	20.101	0	3'55"
6.	25	4 4 3 2 1 3 4 4 1 3 3 4 2 3 2 1 3 2 2 1	3	20	45	56	20.101	1	4'03"
7.	25	3 3 3 2 4 3 4 4 1 4 3 3 2 3 2 1 3 2 2 1	3	20	45	56	20.101	2	3'58"
8.	25	3 4 3 2 1 3 4 4 1 3 4 3 2 1 2 4 3 2 2 1	3	20	45	56	20.101	0	3'52"
9.	25	3 3 3 2 4 3 1 4 1 3 3 3 2 1 2 4 3 2 2 1	3	20	45	56	20.101	4	3'59"
10.	25	4 3 3 2 1 3 1 4 1 4 4 4 2 3 2 4 3 2 2 1	4	20	45	56	20.101	3	4'08"

Dobiveni rezultat predstavlja optimalno rješenje, a za promatrani odjeljak iznosi 20.101. Iz dobivenog rezultat proizlazi da je prilikom otpreme 25 kontejnera ostvareno 20 radnji premještaja kontejnera uz 45 premještaja po širini i 56 po visini. Od ukupno deset pokretanja modela, samo je tri puta u nultoj iteraciji dobiven optimalan rezultat. Rezultat je najkasnije dobiven u četvrtoj iteraciji. U prosjeku potrebne su dvije iteracije za dobivanje optimalnog rezultata, dok prosječno vrijeme izvođenja modela iznosi 4 minute i 12 sekunde.

Iz tablica 17, 18 i 19 može se zaključiti da model postiže optimalan rezultat uz primjenu različitih setova pravila te da je kod odjeljaka s udjelima popunjenosti od približno 75% i 90% za dobivanje rezultata potreban veći broj iteracija, dok je vrijeme izvođenja

modela gotovo zanemarivo produljeno. Kod odjeljaka veličine 4x7 podjednako često su primjenjivana pravila 1, 2 i 3.

7.3 Analiza rezultata za odjeljak veličine 5x7

U ovom dijelu rada prikazani su dobiveni optimalni rezultati za odjeljak veličine 5x7 te udjele popunjenosti od 55%, 75% i 90%. U narednoj tablici prikazani su rezultati dobiveni tijekom deset pokretanja modela, a odnose se na odjeljak veličine 5x7 i udio popunjenosti od približno 55% (Tablica 20).

Model i dalje postiže isti rezultat prilikom svih deset pokretanja. Dobiveni rezultat predstavlja optimalan rezultat s vrijednošću *fitness* funkcije u iznosu od 14.078. Iz vrijednosti *fitness* funkcije proizlazi da je prilikom otpreme 19 kontejnera ukupno ostvareno 14 radnji premještaja s 22 premještaja po širini i 56 po visini. Broj potrebnih iteracija za postizanje rezultata varira od 1 do 11, iz čega slijedi da su za postizanje rezultata u prosjeku potrebne 4 iteracije. Prosječno vrijeme rada modela iznosi 2 minute i 47 sekundi.

Tablica 20 Rezultati odjeljka veličine 5x7 i udjela popunjenosti od približno 55%

Broj pokretanja	**Broj kontejnera**	**Redoslijed primijenjenih pravila[19]**	**Najčešće pravilo**	**Broj premještaja**	**Premještaj po širini**	**Premještaj po visini**	**Vrijednost *fitness* funkcije**	**Generacija rezultata**	**Vrijeme izvođenja**
1.	19	2 3 3 1 2 3 3 2 1 2 1 4 2 1	1	14	22	55	14.078	5	2'49"
2.	19	2 3 3 1 2 1 4 2 3 2 1 4 2 3	2	14	22	56	14.078	2	2'45"
3.	19	2 3 3 1 2 1 3 2 1 2 1 4 2 3	2	14	22	56	14.078	4	2'45"
4.	**19**	**2 1 1 1 2 3 1 2 1 2 1 4 2 3**	**1**	**14**	**22**	**56**	**14.078**	**3**	**2'47"**
5.	19	2 3 3 1 2 3 1 2 3 2 1 1 2 3	1, 2	14	22	56	14.078	6	2'49"
6.	19	2 3 3 1 2 1 3 2 3 2 1 1 2 1	1	14	22	56	14.078	5	2'46"
7.	19	2 3 1 1 2 1 3 2 3 2 1 4 2 3	2	14	22	56	14.078	5	2'47"

[19] Kod ovog odjeljka za grafički prikaz rezultata odabran je set pravila dobivenih prilikom četvrtog pokretanja modela (privitak 4).

8.	19	2 3 1 1 2 1 3 2 3 2 1 1 2 3	1, 2	14	22	56	14.078	3	2'46"
9.	19	2 1 1 1 2 3 3 2 3 2 1 1 2 1	1	14	22	56	14.078	1	2'48"
10.	19	2 3 3 1 2 3 4 2 1 2 1 4 2 1	2	14	22	56	14.078	11	2'51"

Predloženi model postiže optimalno rješenje i kod odjeljka unutar kojeg je složeno 26 kontejnera. Optimalno rješenje ima vrijednost *fitness* funkcije u iznosu od 16.100, što znači da je ukupno izvršeno 16 radnji premještaja kontejnera s 37 premještaja po širini i 63 premještaja po visini (Tablica 21). Optimalno rješenje postiže se primjenom bilo kojeg od deset prikazanih setova pravila.

Tablica 21 Rezultati odjeljka veličine 5x7 i udjela popunjenosti od približno 75%

Broj pokretanja	**Broj kontejnera**	**Redoslijed primijenjenih pravila**	**Najčešće pravilo**	**Broj premještaja**	**Premještaj po širini**	**Premještaj po visini**	**Vrijednost *fitness* funkcije**	**Generacija rezultata**	**Vrijeme izvođenja**
1.	26	2 1 1 1 2 3 4 1 1 2 2 4 2 1	1	16	37	63	16.100	5	5'59"
2.	26	4 1 3 1 1 3 4 2 1 2 1 1 2 1 3 3	1	16	37	63	16.100	4	4'38"
3.	26	1 4 3 1 4 3 4 2 3 2 1 3 2 1 1 3	1, 3	16	37	63	16.100	4	4'25"
4.	26	4 1 3 1 1 3 4 2 1 2 1 1 2 1 1 3	1	16	37	63	16.100	6	4'36"
5.	26	4 4 3 1 1 3 4 2 3 2 1 1 2 4 3 3	1, 3, 4	16	37	63	16.100	2	4'07"
6.	26	1 4 1 1 4 3 4 2 1 2 1 3 2 4 3 3	1	16	37	63	16.100	8	4'07"
7.	26	1 4 3 1 1 3 4 2 3 2 1 3 2 1 3 3	3	16	37	63	16.100	3	4'42"
8.	26	1 4 3 1 1 3 4 2 3 2 1 3 2 4 3 3	3	16	37	63	16.100	4	4'54"
9.	26	1 1 1 1 4 3 4 2 3 2 1 3 2 1 3 3	1	16	37	63	16.100	5	4'41"
10.	26	1 4 3 1 4 3 4 2 3 2 1 1 2 1 1 3	1	16	37	63	16.100	6	4'10"

U prosjeku je potrebno pet iteracija kako bi se postigao optimalan rezultat, uz prosječno vrijeme rada modela od 4 minute i 42 sekunde. Kao i kod prethodno analiziranih odjeljaka model i kod odjeljka popunjenosti od približno 90 % postiže uvijek isti

optimalan rezultat čija vrijednost *fitness* funkcije iznosi 25.137 (Tablica 22). Dakle, prilikom otpreme 31 kontejnera izvršava se 25 radnji premještaja kontejnera s 49 premještaja po širini i 88 po visini. Navedeni rezultat postiže se primjenom različitih setova pravila premještaja.

Tablica 22 Rezultati odjeljka veličine 5x7 i udjela popunjenosti od približno 90%

Broj pokretanja	**Broj kontejnera**	**Redoslijed primijenjenih pravila**	**Najčešće pravilo**	**Broj premještaja**	**Premještaj po širini**	**Premještaj po visini**	**Vrijednost *fitness* funkcije**	**Generacija rezultata**	**Vrijeme izvođenja**
1.	31	4 3 3 3 3 3 4 4 2 3 1 1 4 4 2 4 2 2 3 1 2 3 1 4 2	3	25	49	88	25.137	5	6'09"
2.	31	1 3 3 3 3 4 4 4 2 3 3 4 4 1 2 3 2 2 3 1 2 1 1 4 2	3	25	49	88	25.137	0	6'05"
3.	31	1 4 3 3 3 4 3 3 2 3 3 1 4 1 2 4 2 2 3 4 2 3 1 2 3	3	25	49	88	25.137	2	6'11"
4.	31	1 3 3 3 1 4 3 3 2 3 1 4 1 1 2 3 2 2 1 1 2 3 3 3 1	3	25	49	88	25.137	0	5'53"
5.	31	4 4 3 3 1 4 4 3 2 3 1 1 1 4 2 3 2 2 3 4 2 3 3 2 1	3	25	49	88	25.137	1	6'05"
6.	31	4 3 3 3 3 3 4 3 2 3 3 1 3 4 2 3 2 2 3 1 2 3 3 1 1	3	25	49	88	25.137	3	6'00"
7.	31	1 4 3 3 3 3 4 3 2 3 3 1 1 4 2 1 2 2 3 1 2 3 1 2 4	3	25	49	88	25.137	4	6'34"
8.	31	4 3 3 3 3 3 4 4 2 3 3 1 1 1 2 3 2 2 1 1 2 1 1 2 3	3	25	49	88	25.137	0	6'08"
9.	31	4 1 3 3 3 3 4 4 2 3 1 4 3 4 2 3 2 2 1 1 2 1 1 3 2	3	25	49	88	25.137	2	6'15"
10.	31	1 4 3 3 4 4 3 3 2 3 1 1 3 4 2 1 2 2 3 1 2 1 4 3 1	3	25	49	88	25.137	3	6'22"

Prosječno vrijeme izvođenja modela iznosi 6 minuta i 6 sekundi, dok su za postizanje optimalnog rezultata u prosjeku potrebne dvije iteracije.

Iz tablica 20, 21 i 22 vidljivo je da predloženi model postiže optimalan rezultat primjenom različitih setova pravila te da su za premještaj kontejnera unutar odjeljka najčešće primjenjivana pravila 1 i 3.

7.4 Analiza rezultata za odjeljak veličine 6x7

U nastavku rada tabličnim prikazima predstavljeni su dobiveni rezultati za odjeljak veličine 6x7 i udjele popunjenosti od približno 55%, 75% i 90%. Sukladno tome, tablica 23 prikazuje rezultate za udio popunjenosti odjeljka od približno 55%.

Tablica 23 Rezultati odjeljka veličine 6x7 i udjela popunjenosti od približno 55%

Broj pokretanja	Broj kontejnera	Redoslijed primijenjenih pravila[20]	Najčešće pravilo	Broj premještaja	Premještaj po širini	Premještaj po visini	Vrijednost *fitness* funkcije	Generacija rezultata	Vrijeme izvođenja
1.	23	2 1 4 3 1 2 1 3 2 4 3 4 1 1 2 3 2 1	1	18	18	35	18.053	4	3'37"
2.	23	2 1 3 1 4 2 1 4 2 1 3 3 4 1 2 4 2 3	1, 2	18	18	35	18.053	4	3'48"
3.	23	2 3 4 3 1 2 4 4 2 3 4 1 1 1 2 3 2 4	2, 4	18	18	35	18.053	3	3'32"
4.	23	2 4 1 4 1 2 4 4 2 1 3 4 1 1 2 3 2 3	1, 2, 4	18	18	35	18.053	2	3'30"
5.	23	2 3 3 3 1 2 4 4 2 1 3 1 4 4 2 4 2 3	2, 3, 4	18	18	35	18.053	6	3'36"
6.	23	2 4 4 1 4 2 3 3 2 4 3 1 4 1 2 4 2 1	4	18	18	35	18.053	4	3'33"
7.	23	2 3 4 1 4 2 3 1 2 3 1 1 1 4 2 4 2 1	1, 2	18	18	35	18.053	4	3'41"
8.	23	2 4 3 1 1 2 3 3 2 3 4 3 1 4 2 4 2 3	3	18	18	35	18.053	4	3'34"
9.	23	2 3 3 3 1 2 1 3 2 3 3 1 1 1 2 4 2 1	1, 3	18	18	35	18.053	4	3'30"
10.	23	2 4 4 3 4 2 1 3 2 1 1 1 4 1 2 1 2 3	1	18	18	35	18.053	4	3'37"

Iz tablice je vidljivo da predloženi model i dalje svakim pokretanjem dobiva isti rezultat čija vrijednost *fitness* funkcije iznosi 18.053, što znači da je izvršeno 18 radnji premještaja kontejnera s ukupno 18 premještaja po širini i 35 premještaja po visini. Dobivanje rezultata u prosjeku se postiže u četvrtoj iteraciji uz prosječno vrijeme rada algoritma od 3 minute i 36 sekundi.

Kod odjeljka s udjelom popunjenosti od približno 75% situacija je nešto drugačija. Naime, nakon deset pokretanja algoritma dobivena su tri različita rezultata iz čega se

[20] Kod ovog odjeljka za grafički prikaz rezultata odabran je set pravila dobivenih prilikom sedmog pokretanja modela (privitak 4).

može zaključiti da model ne uspijeva pronaći optimalan rezultat, već da kao konačni rezultat dobiva najbolje moguće rješenje (Tablica 24).

Tablica 24 Rezultati odjeljka veličine 6x7 i udjela popunjenosti od približno 75%

Broj pokretanja	Broj kontejnera	Redoslijed primijenjenih pravila	Najčešće pravilo	Broj premještaja	Premještaj po širini	Premještaj po visini	Vrijednost fitness funkcije	Generacija (iteracija) rezultata	Vrijeme izvođenja modela
1.	31	3 3 2 1 4 1 1 4 2 3 1 1 3 4 1 4 3 4 1 2 4 1 2 1 2 1 3 2	1	28	48	91	28.139	9	7'23"
2.	31	4 4 2 3 4 4 4 4 2 1 1 4 3 4 1 3 3 3 1 3 2 4 3 2 2 1 4 2	4	28	49	89	28.138	8	7'12"
3.	31	3 3 2 3 4 4 4 4 2 3 1 1 3 4 1 4 3 4 1 2 3 1 2 4 2 1 1 2	1, 3, 4	28	53	88	28.141	12	7'12"
4.	31	4 3 2 3 4 1 4 4 2 3 1 1 3 4 4 4 3 4 1 2 4 3 2 1 2 1 1 2	4	28	49	89	28.138	6	7'06"
5.	31	3 1 2 3 4 4 4 4 2 3 1 4 3 4 4 3 3 3 1 1 2 4 1 2 2 4 4 2	4	28	49	89	28.138	3	7'18"
6.	31	3 4 2 3 4 1 4 4 2 3 1 1 3 3 1 3 3 3 4 2 4 3 2 4 2 1 3 2	3	28	48	91	28.139	5	8'15"
7.	31	3 4 2 1 4 4 4 4 2 3 1 1 3 4 4 4 3 4 4 2 4 1 2 1 2 3 3 2	4	28	49	89	28.138	9	7'14"
8.	31	4 4 2 1 4 1 1 4 2 1 1 1 3 4 4 4 3 4 1 2 1 3 2 1 2 3 3 2	1	28	48	90	28.138	3	6'59"
9.	31	3 4 2 3 4 1 1 4 2 1 1 1 3 3 4 3 3 3 1 2 4 3 2 4 2 3 3 2	3	28	48	91	28.139	6	7'17"
10.	31	3 3 2 3 4 4 4 4 2 1 1 1 3 3 4 3 3 4 1 2 1 3 2 1 2 3 4 2	3	28	48	91	28.139	8	6'50"

Što se tiče broja radnji premještaja kontejnera, model uvijek pronalazi optimalno rješenje međutim, ne pronalazi optimalno rješenje u pogledu broja premještaja po širini. Za promatrani odjeljak broj premještaja po širini varira od 48 do 53.

Vrijednost *fitness* funkcije najboljeg rezultata iznosi 28.138, a najlošijeg 28.141. Iz navedenog je razvidno da su razlike između postignutih rezultata vrlo male, međutim za postizanje optimalnog broja premještanja po širini potrebno je modificirati predloženi model. Kako model svaki puta dobiva različito rješenje, kao konačno rješenje ovog odjeljka uzima se prosječna rješenje čija vrijednost *fitness* funkcije iznosi 28.139. Budući da je broj radnji premještaja kontejnera, primarna varijabla *fitness* funkcije, sa stajališta optimalnosti rješenja može se zaključiti da ovaj model postiže optimalan broj radnji premještaja kontejnera s obzirom na promatrani odjeljak te da sa stajališta broja premještaja po širini i visini postiže najbolje moguće rješenje. Prosječan broj iteracija

potrebnih za dobivanje rezultata je sedam, a prosječno vrijeme rada modela iznosi 7 minuta i 13 sekundi.

Kao i kod prethodno analiziranog odjeljka, model gotovo svakim pokretanjem dobiva različite vrijednosti *fitness* funkcije. Bitno je napomenuti da se i dalje postiže isti broj radnji premještaja, dok se broj premještaja po širini i visini mijenja. Međutim, kvaliteta dobivenih rezultata i dalje je zadovoljavajuća uz vrlo mala odstupanja od najboljeg postignutog rezultata. Dakle, i kod ovog odjeljka model pronalazi najbolje moguće rješenje.

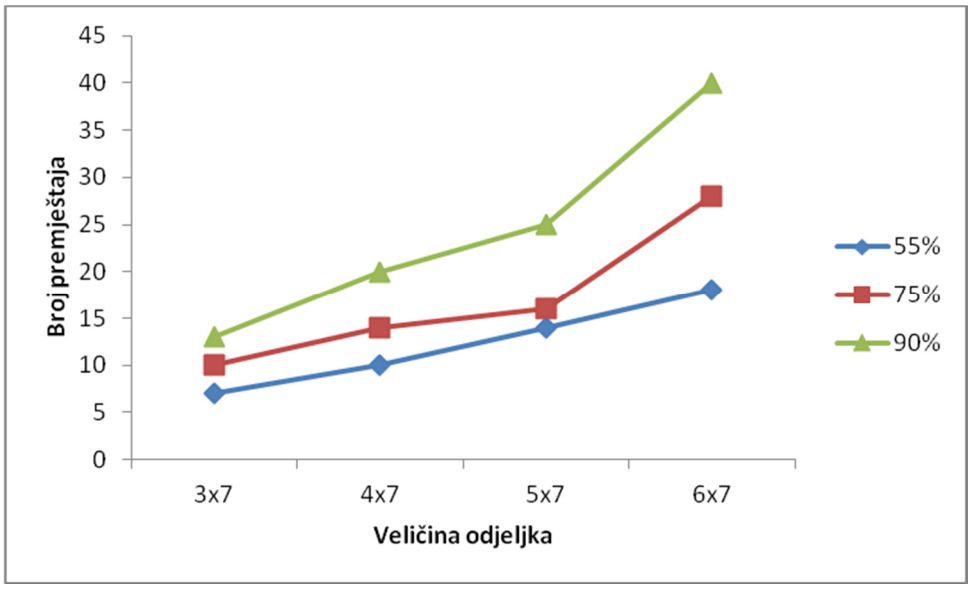
Tablica 25 Rezultati odjeljka veličine 6x7 i udjela popunjenosti od približno 90%

Broj pokretanja	Broj kontejnera	Konačni rezultat	Najčešće pravilo	Broj premještaja	Premještaj po širini	Premještaj po visini	Vrijednost *fitness* funkcije	Generacija rezultata	Vrijeme izvođenja
1.	37	3 1 3 3 3 3 1 3 3 4 3 2 4 3 1 3 4 4 4 3 3 4 4 2 3 4 2 1 3 1 1 1 2 3 2 1 1 1 2 3	3	40	94	157	40.251	11	10'51"
2.	37	3 1 3 4 3 3 1 4 3 3 4 2 3 3 1 3 4 4 3 3 3 4 1 2 3 3 2 4 3 4 1 2 1 3 2 1 1 1 2 1	3	40	93	158	40.251	9	9'51"
3.	37	3 4 3 4 3 3 4 3 3 3 3 2 3 1 4 3 4 3 3 3 4 3 1 4 2 3 3 1 2 1 1 2 1 4 2 3 1 3 2 1	3	40	88	155	40.243	11	10'10"
4.	37	3 4 3 3 3 3 4 3 4 4 3 2 4 1 3 4 3 1 3 4 3 4 4 2 3 4 2 1 3 4 1 2 1 1 2 1 1 1 2 3	3	40	93	159	40.252	18	10'12"
5.	37	3 4 3 4 3 3 1 3 4 3 4 2 4 3 4 4 3 1 4 3 4 4 4 2 3 3 2 4 3 1 1 2 1 3 2 1 1 1 2 1	3	40	93	159	40.252	15	10'02"
6.	37	3 4 3 3 3 3 1 3 4 3 4 2 1 3 4 3 4 3 3 4 4 4 1 2 4 3 3 2 1 4 1 2 4 1 2 3 1 3 2 4	3	40	89	159	40.248	12	10'02"
7.	37	3 4 3 3 3 3 1 3 4 4 3 2 4 3 3 4 3 4 3 3 3 4 1 2 4 1 1 2 4 1 3 2 3 3 2 3 1 1 2 4	3	40	94	163	40.257	15	10'01"
8.	37	3 4 3 3 3 3 1 3 3 3 3 2 3 4 1 3 4 3 4 3 4 3 1 4 2 1 3 1 2 4 1 2 1 1 2 4 3 3 2 4	3	40	88	155	40.243	17	10'20"
9.	37	3 3 3 4 4 3 3 4 4 1 4 2 4 3 3 3 4 1 3 4 4 3 4 2 3 1 1 1 1 1 2 3 3 2 2 1 1 1 2 3	3	40	96	165	40.261	16	10'11"
10.	37	3 4 3 3 3 3 4 4 3 3 4 2 3 3 3 3 4 4 3 3 4 4 4 2 4 1 2 1 3 1 1 2 3 1 2 1 1 1 2 3	3	40	93	158	40.251	12	10'11"

Najmanja ostvarena vrijednost *fitness* funkcije je 40.243, s 88 premještaja po širini i 155 premještaja po visini, a najveća je 40.261 s 96 premještaja po širini i 165 premještaja po visini. Iz tablice 25 proizlazi da prosječna vrijednost *fitness* funkcije, dobivena temeljem deset testiranja, iznosi 40.251. Kod ovog odjeljka problem postaje složeniji pa je u prosjeku potrebno četrnaest iteracija kako bi se postigao rezultat, odnosno dulje prosječno vrijeme rješavanja koje iznosi 10 minuta i 07 sekundi.

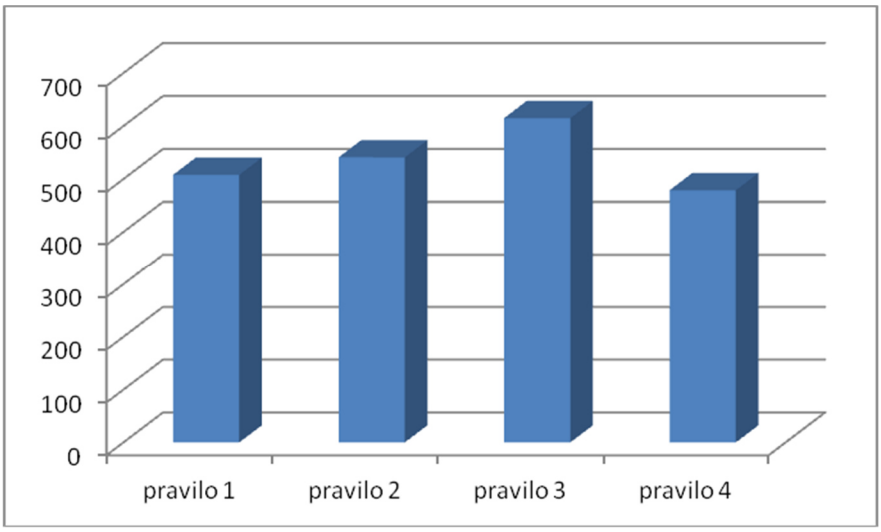
Iz prikazanih tablica 23, 24 i 25 uočava se da za premještaj kontejnera su ukupno najčešće primjenjivana pravila 1, 3 i 4. Također, iz tablica se može zaključiti da se kod promatranih odjeljka ne uspijeva pronaći optimalno rješenje, već najbolje moguće rješenje.

Na osnovu prethodno provedenih analiza može se zaključiti da broj premještaja kontejnera uvelike ovisi o početnom rasporedu kontejnera, odnosno o broju „loše“ složenih kontejnera te da se broj premještaja kontejnera povećava s veličinom i udjelom popunjenosti odjeljka (Grafikon 6) što na kraju rezultira povećanjem vrijednosti *fitness* funkcije.



Grafikon 6 Utjecaj udjela popunjenosti odjeljka na ukupan broj premještaja kontejnera

Poznato je da se radnja premještaja kontejnera odvija prema nekome od četiriju definirana pravila. Uzimajući u obzir učestalost primjene pojedinog pravila, može se zaključiti da su tijekom 120 pokretanja modela za premještaj kontejnera najčešće primjenjivana pravila 2 i 3 (Grafikon 7).



Grafikon 7 Najčešće primjenjivana pravila na premještaj kontejnera unutar odjeljka

Temeljem provedene analize može se zaključiti da predloženi model dobiva uspješno minimiziran broj premještaja i broj premještaja po širini, s obzirom na set primijenjenih pravila te uspješno izvršava izračun premještaja po visini. Također, vrlo važno je i saznanje da je izrađeni rješiv u vrlo kratkom vremenu te da svakim pokretanjem, ovisno o promatranom odjeljku, postiže optimalno ili najbolje rješenje.

7.5 Komparativna analiza rezultata predloženog modela s rezultatima modela drugih autora

U ovom dijelu analize izrađena je usporedba dobivenih rezultata predstavljenog modela s rezultatima drugih autora. Autori modela s kojima je izvršena usporedba dobivenih rezultata za izradu modela također su koristili heurističke i metaheurističke metode. Da bi se rezultati uopće mogli uspoređivati, prethodno je bilo potrebno uskladiti parametre modela te utvrditi uvjete u kojima će biti provedeno eksperimentalno testiranje. Eksperimentalno testiranje provedeno je pod jednakim uvjetima kao što su ga provodili i drugi autori.

Eksperimentalno testiranje provedeno je za:

- veličine odjeljaka 3x7, 4x7, 5x7, 6x7
- udjele popunjenosti od približno 90%
- četrdeset različitih početnih rasporeda kontejnera unutar odjeljka generiranih slučajnim odabirom.

Dobiveni rezultati modela prikazani su u tablici 26.

Tablica 26 Rezultati predloženog modela za odjeljke veličine 3x7,4x7,5x7,6x7
i udio popunjenosti od približno 90%

Redoslijed početnog rasporeda	Broj premještaja				Vrijeme izvođenja algoritma			
	3X7	4X7	5X7	6X7	3X7	4X7	5X7	6X7
1.	13	20	25[21]	40	2'33"	4'21"	6'06"	10'51"
2.	9	15	26	30	2'16"	3'38"	6'26"	8'31"
3.	8	11	21	33	2'20"	3'13"	5'54"	8'27"
4.	10	17	16	30	2'22"	3'32"	5'03"	8'54"
5.	9	18	19	36	1'42"	3'43"	5'15"	9'18"
6.	2	10	22	33	2'12"	3'09"	5'44"	8'55"
7.	7	11	24	25	2'13"	3'08"	6'02"	7'49"
8.	8	17	17	23	3'13"	3'42"	5'07"	7'15"
9.	10	14	24	33	2'26"	3'21"	5'58"	8'51"
10.	11	11	19	27	2'11"	3'12"	5'26"	8'06"
11.	7	17	18	24	2'33"	3'36"	5'11"	7'46"
12.	12	10	11	34	2'03"	2'59"	4'35"	9'12"
13.	6	15	10	34	2'36"	3'28"	4'25"	9'52"
14.	12	14	10	36	2'34"	3'18"	4'13"	9'13"
15.	7	16	12	27	2'11"	3'27"	4'39"	8'26"
16.	4	17	12	28	2'37"	3'32"	4'54"	8'36"
17.	10	15	13	31	2'23"	3'19"	4'41"	8'51"
18.	10	16	21	29	2'53"	3'31"	5'36"	8'45"
19.	8	10	15	39	2'27"	3'00"	4'55"	9'38"
20.	10	10	18	32	2'17"	2'53"	5'26"	8'54"
21.	8	13	28	28	2'14"	3'22"	6'46"	8'28"
22.	8	16	14	27	2'11"	3'30"	5'04"	8'28"
23.	6	13	12	29	2'15"	3'18"	4'34"	8'17"
24.	7	11	21	28	2'19"	3'04"	5'29"	8'34"

[21] Dobiveni rezultati prikazani u prvom retku tablice odnose se na početni raspored prikazan na slici 40 i 41 (odjeljak 3X7, 4X7, 5x7, 6X7(3)).

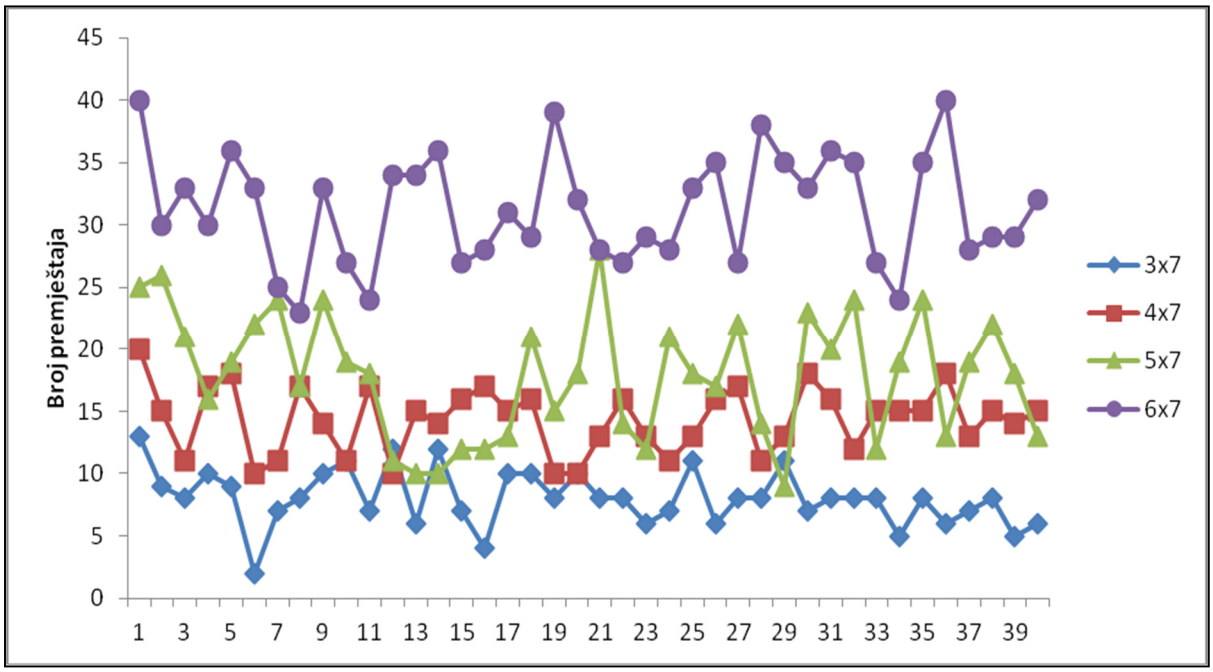
25.	11	13	18	33	2'03"	3'14"	5'18"	8'52"
26.	6	16	17	35	2'10"	3'33"	5'12"	8'43"
27.	8	17	22	27	2'18"	3'26"	5'51"	8'23"
28.	8	11	14	38	2'28"	3'06"	4'46"	9'47"
29.	11	13	9	35	2'09"	3'21"	4'02"	9'17"
30.	7	18	23	33	2'20"	3'39"	6'13"	9'27"
31.	8	16	20	36	2'19"	3'27"	5'59"	9'49"
32.	8	12	24	35	2'20"	3'08"	6'15"	10'53"
33.	8	15	12	27	2'01"	3'34"	4'32"	9'02"
34.	5	15	19	24	2'15"	3'20"	5'36"	7'54"
35.	8	15	24	35	1'57"	3'32"	5'50"	10'39"
36.	6	18	13	40	2'13"	3'32"	4'46"	11'04"
37.	7	13	19	28	2'12"	3'10"	5'55"	9'45"
38.	8	15	22	29	2'13"	3'19"	5'44"	9'17"
39.	5	14	18	29	2'10"	3'20"	5'15"	9'52"
40.	6	15	13	32	2'15"	3'31"	4'49"	8'20"
PROSJEK	**8,35**	**14,32**	**17,85**	**30,57**	**2'01"**	**3'22"**	**5'13"**	**9'01"**

U prvom stupcu tablice 26 naveden je broj početnog rasporeda dobiven na osnovu programske skripte izrađene u programskom jeziku R (vidi privitak 2). Primjer generiranih početnih rasporeda prikazan je u privitku 3. Stupci (2-5) prikazuju broj izvršenih premještaja kontejnera s obzirom na početni raspored i veličinu odjeljka, odnosno prosječan broj premještaja kontejnera postignut prilikom četrdeset mjerenja.

Stupci(6-9) prikazuju vrijeme potrebno za izvođenje algoritma te prosječno vrijeme izvođenja modela za pojedini odjeljak. Na osnovu rezultata prikazanih u tablici izrađen je grafički prikazi broja premještaja kontejnera (Grafikon 8).

Rezultati dobiveni autora Zhang i Murty et al. preuzeti su od autora Wu i Ting (2010) koji u svom radu "A Beam Search Algorithm for Minimizing Reshuffle Operations at Container Yards" prikazuju rezultate od navedenih autora te ih uspoređuju sa svojima.

Osim ovih autora, dobiveni rezultati predloženog modela uspoređeni su s rezultatima autora Jovanović et al. (2014) te autora Caserta et al (2012) prikazanim u radu "A chain heuristics for the Blocks Relocation Problem".



Grafikon 8 Broj premještaja kontejnera za odjeljke veličine 3x7, 4x7, 5x7 i 6x7

Slijedom navedenog, izrađen je tablični prikaz rezultata po autorima. Dobiveni rezultati prikazuju prosječan broj radnji premještaja kontejnera za odjeljke veličine 3x7, 4x7, 5x7 i 6x7 (Tablica 27). Usporedba vremena rada modela nije izvršena, budući da podaci o točnim vremenima rješavanja modela izrađenih od strane drugih autora nisu evidentirani.

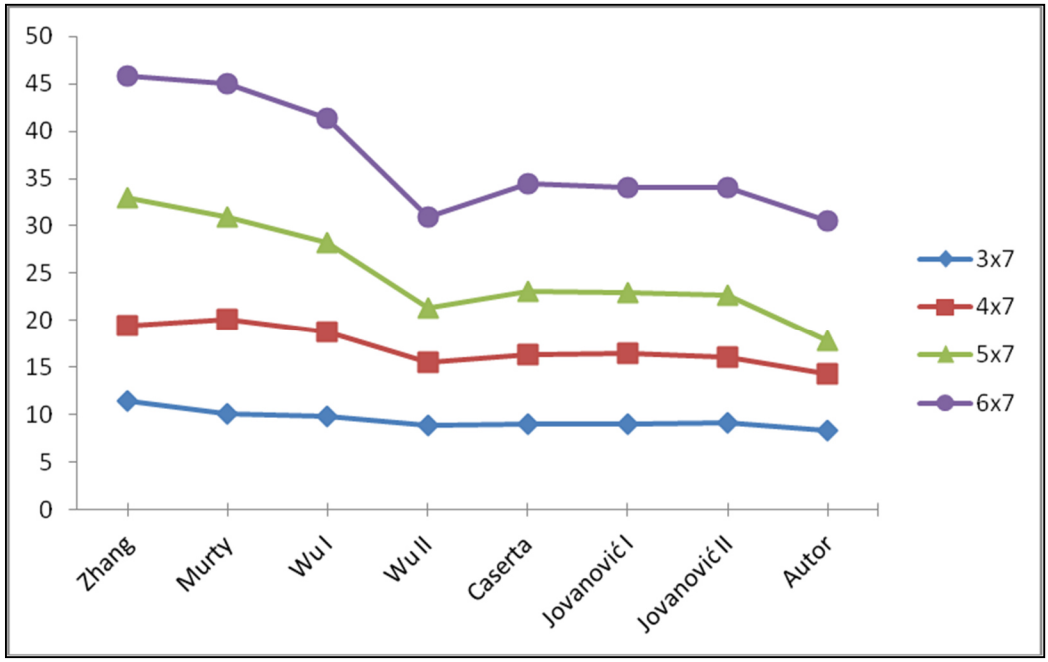
Tablica 27 Usporedni prikaz rezultata drugih autora za odjeljak veličine 3x7, 4x7, 5x7, 6x7 i udjela popunjenosti od 90%

Autor	**Heuristike korištene za izradu modela**	**Prosječan broj premještaja**			
		3X7	**4X7**	**5X7**	**6X7**
Zhang	Pravilo najnižeg stupca (*Tier lowest position*- TLP)	11,50	19,40	33,00	45,80
Murty et al.	Pravilo indeksa premještaja (*Reshuffle index*- RI)	10,10	20,10	30,90	45,00
Wu i Ting	Prošireno pravilo indeksa premještanja s pravilom „gledaj unaprijed"(*Reshuffle index with look ahead*- RIL)	9,88	18,80	28,30	41,30
Wu i Ting	Usmjereno (zrakasto) pretraživanje (*Beam search*)	8,95	15,50	21,40	31,00

Caserta et al.	Min-max heuristika (*Min-max heuristics*)	9,02	16,35	23,15	34,40
Jovanović i Voß	Heuristika lanca (*Chain Heuristics*)	9,05	16,45	22,92	34,07
Jovanović i Voß	Heuristika lanca F (*Chain Heuristics F*)	9,17	16,07	22,70	34,07
Livia Maglić	**Genetski algoritam (*Genetic Algorithms*)**	**8,35**	**14,32**	**17,85**	**30,57**
POBOLJŠANJE REZULTATA(%)		**6,70**	**7,61**	**16,59**	**1,39**

Potrebno je istaknuti da su za usporedbu kvalitete rezultata predloženog modela odabrani modeli autora koji su postigli najbolje rezultate u rješavanju problema BRP/CRP.

Na osnovu rezultata prikazanih u tablici 27, može se zaključiti da predloženi model daje kvalitetnije rezultate od modela drugih autora te da se na drugom mjestu po kvaliteti rezultata nalaze rezultati autora Wu i Ting. Dobiveni rezultati uspoređeni su s rezultatima autora Wu i Ting, a njihov udio poboljšanja prikazan je u posljednjem retku tablice. Vidljivo je da je kod svih testiranih odjeljaka predloženim modelom postignuto poboljšanje rezultata te da je kod odjeljka veličine 5x7 postignuto najveće poboljšanje od 16,59%, odnosno da je ukupan broj premještaja kontejnera umanjen za 3 premještaja. Najmanje poboljšanje zabilježeno je kod odjeljka veličine 6x7, a iznosi 1,39%. Na grafikonu u nastavku rada ilustrativno je prikazan prosječan broj premještaja po pojedinom autoru (Grafikon 9).



Grafikon 9 Prosječan broj radnji premještanja kontejnera po autoru

7.6 Analiza osjetljivosti modela na promjenu parametara genetskog algoritma

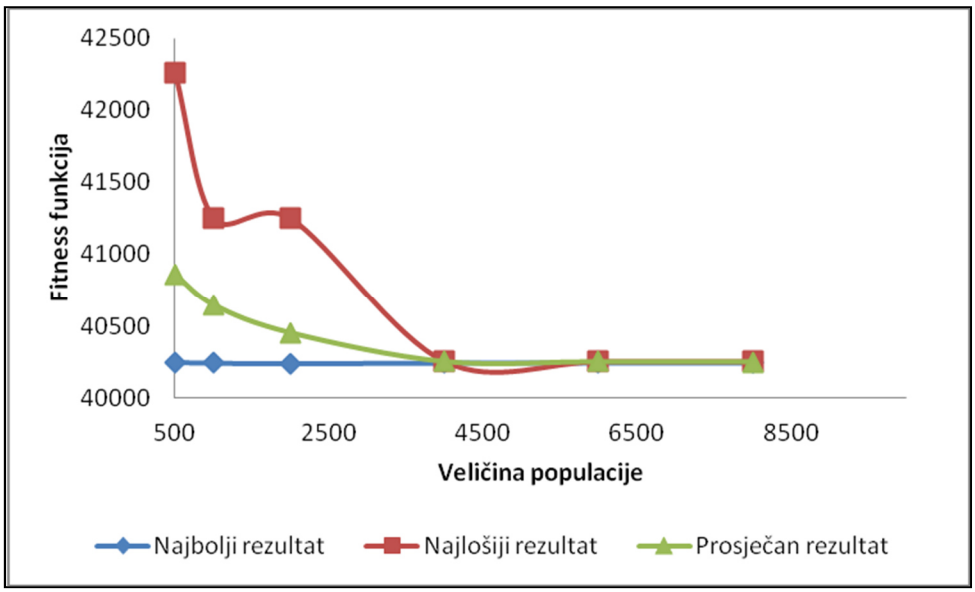
Analiza osjetljivosti modela obuhvaća analizu dobivenih rezultata s obzirom na promjenu parametara GA, a izrađena je kako bi se utvrdila moguća poboljšanja dobivenih rezultata modela. Odjeljak na kojem je provedena analiza osjetljivosti je veličine 6x7 i udjela popunjenosti odjeljka od približno 90%. Testiranje algoritma obavljeno je s različitim veličinama populacije, evolucije, križanja i mutacije. U tablicama u nastavku rada prikazani su dobiveni rezultati modela prilikom testiranja pojedinog parametra. Prikazani rezultati predstavljaju rezultate dobivene prilikom pet pokretanja modela. Prvo provedeno testiranje odnosi se na različite veličine populacije, dok veličina parametra broja evolucija nije mijenjana. Kod već provedenog testiranja i analize rezultata, koja je izrađena za veličinu populacije od 8.000 kromosoma, za usporedbu rezultata uzeto je prvih 5 mjerenja prikazanih u tablici 25. Testiranje je provedeno na dodatnih pet različitih veličina populacije, a dobiveni rezultati prikazani su u tablici.

Tablica 28 Utjecaj veličine populacije na vrijednosti *fitness* funkcije i vrijeme izvođenja modela za 20 evolucija

Veličina populacije	**Najbolji rezultat prema vrijednosti *fitness* funkcije**	**Najlošiji rezultat prema vrijednosti *fitness* funkcije**	**Prosječan rezultat prema vrijednosti *fitness* funkcije**	**Prosječno vrijeme izvođenja**
500	40.249	42.260	40.859	27"
1.000	40.242	41.252	40.647	57"
2.000	40.236	41.251	40.451	1'54"
4.000	40.243	40.253	40.250	4'33"
6.000	40.243	40.253	40.250	6'29"
8.000	40.243	40.252	40.249	10'05"

Iz tablice je vidljivo da veličina populacije uvelike utječe na kvalitetu najlošijih rezultata *fitness* funkcije, odnosno da se njenim povećanjem dobiva kvalitetniji rezultat. Primjerice, ukoliko se usporede rezultati za veličinu populacije od 500 i 4.000 kromosoma, tada proizlazi da je kod veličine populacije od 4.000 kromosoma vrijednost *fitness* funkcije umanjena za 2.000 odnosno 2 premještaja. Zanimljiva činjenica je da je sa stajališta najboljeg rezultata *fitness* funkcije, kod veličine populacije od 2.000 kromosoma postignut najkvalitetniji rezultat. Međutim, ukoliko se u obzir uzmu

prosječne vrijednosti *fitness* funkcije, tada se povećanjem populacije ne dobiva kvalitetniji rezultat u pogledu ukupnog broja premještaja kontejnera, ali se dobiva kvalitetniji rezultat u pogledu broja premještaja po širini i visini. Budući da je sa statističkog gledišta uzorak od pet mjerenja zaista malen, daljnjim mjerenjima moguće je dobiti još kvalitetniji rezultat s obzirom na prosječnu vrijednosti *fitness* funkcije. S obzirom na procesor i radnu memoriju računala maksimalna veličina populacije za koju se moglo izvršiti testiranje je 8.000 kromosoma. Da bi se prikazani rezultati što bolje predočili izrađen je grafički prikaz usporedbe rezultata (Grafikon 10).



Grafikon 10:Osjetljivost *fitness* funkcije na promjenu veličine populacije

Iz prikazanog grafikona uočava se da su najveća odstupanja rezultata zabilježena kod populacije veličine od 500 kromosoma, na osnovu čega se može zaključiti da je to premala veličina populacije i da je kvaliteta dobivenih rezultata nezadovoljavajuća. Nasuprot tomu, najmanja odstupanja i najbolji rezultat *fitness* funkcije ostvaren je kod populacije veličine od 8.000 kromosoma. Stoga se može zaključiti da uz odabranu veličinu populacije od 8.000 kromosoma GA zasigurno dobiva najkvalitetnije rezultate, premda je prosječno vrijeme za dobivanje rezultata najduže i iznosi 10 minuta i 05 sekundi.

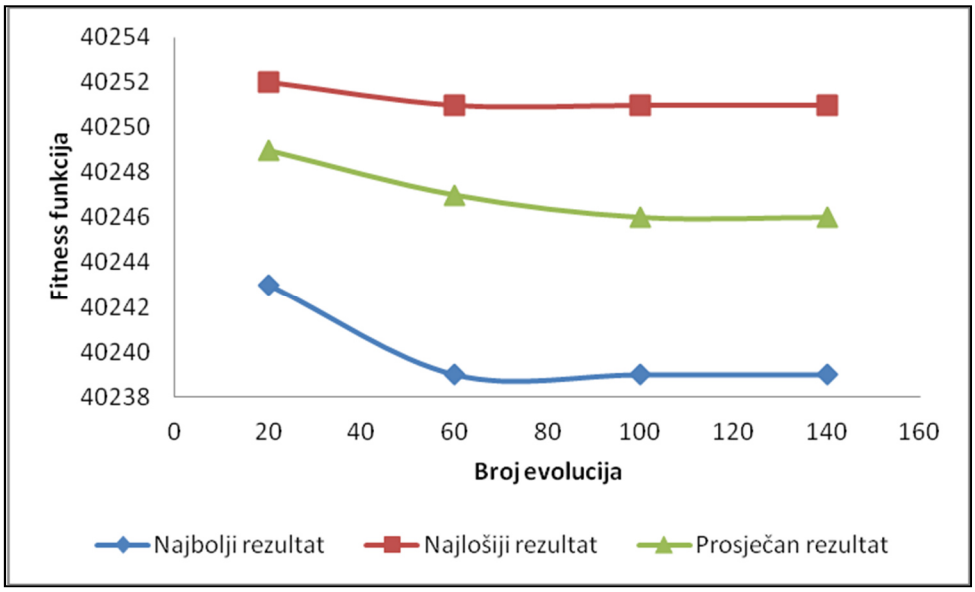
Sljedeće provedeno testiranje odnosi se na broj evolucija (generacija/iteracija) populacije. Prilikom utvrđivanja broja evolucija, na kojem će biti sprovedeno testiranje GA, vodilo se računa o vremenu izvođenja modela. Slijedom navedenog testiranjem je

obuhvaćeno 20, 60, 100 i 140 evolucija populacije, a dobiveni rezultati prikazani su tablicom (Tablica 29).

Tablica 29 Utjecaj broja evolucija na vrijednosti *fitness* funkcije i vrijeme izvođenja modela za veličinu populacije od 8.000 kromosoma

Broj evolucija	**Najbolji rezultat prema vrijednosti *fitness* funkcije**	**Najlošiji rezultat prema vrijednosti *fitness* funkcije**	**Prosječan rezultat prema vrijednosti *fitness* funkcije**	**Prosječno vrijeme izvođenja**
20	40.243	40.252	40.249	10'05"
60	40.239	40.251	40.247	26'12"
100	40.239	40.251	40.246	39'49"
140	40.239	40.251	40.246	57'04"

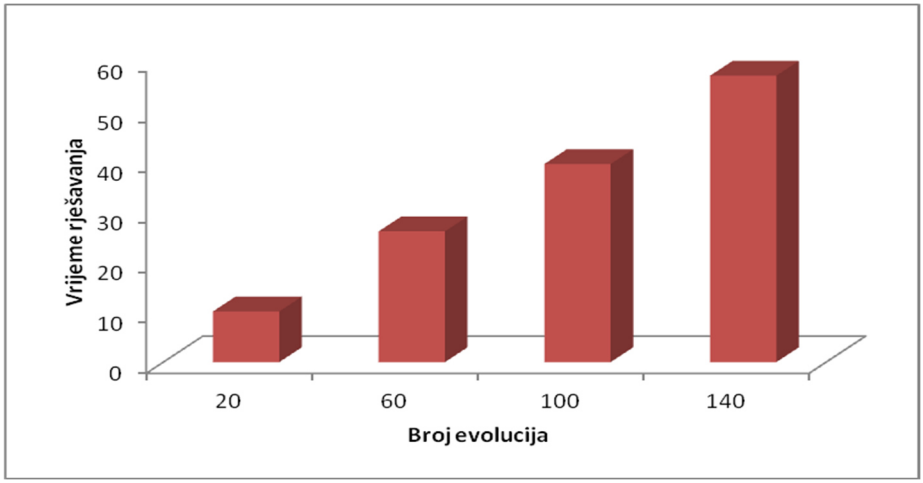
U grafikonu je u nastavku prikazana ovisnost rezultata *fitness* funkcije o promjeni parametra broja evolucija (Grafikon 11).



Grafikon 11 Osjetljivost *fitness* funkcije na promjenu broja evolucija

Iz predočenih rezultata uočeno je da se sa stajališta najboljih rezultata *fitness* funkcije, povećanjem broja evolucija postiže kvalitetniji rezultat, dok je kod najlošijeg rezultata zabilježeno minimalno poboljšanje u kvaliteti dobivenog rezultata. Sa stajališta najboljeg, najlošijeg i prosječnog rezultata vrijednosti *fitness* funkcije najkvalitetniji rezultati postignuti su prilikom 100 i 140 evolucija populacije. Zanimljiva činjenica je da se tijekom 60 evolucija postiže jednak najbolji i najlošiji rezultat kao što je to slučaj za

populacije veličine 100 i 140. Također, sa stajališta prosječne vrijednosti *fitness* funkcije najlošiji rezultat dobiven je tijekom 20 evolucija. Osim toga, vidljivo je da broj evolucija ne utječe toliko na kvalitetu dobivenih rezultata te da su dobiveni rezultati sa stajališta prosječne vrijednosti *fitness* funkcije vrlo bliski najboljem ostvarenom rezultatu. Međutim, broj evolucija uvelike utječe na prosječno vrijeme rada algoritma što je vidljivo na grafikonu(Grafikon 12).



Grafikon 12 Utjecaj broja evolucija na vrijeme izvođenja modela

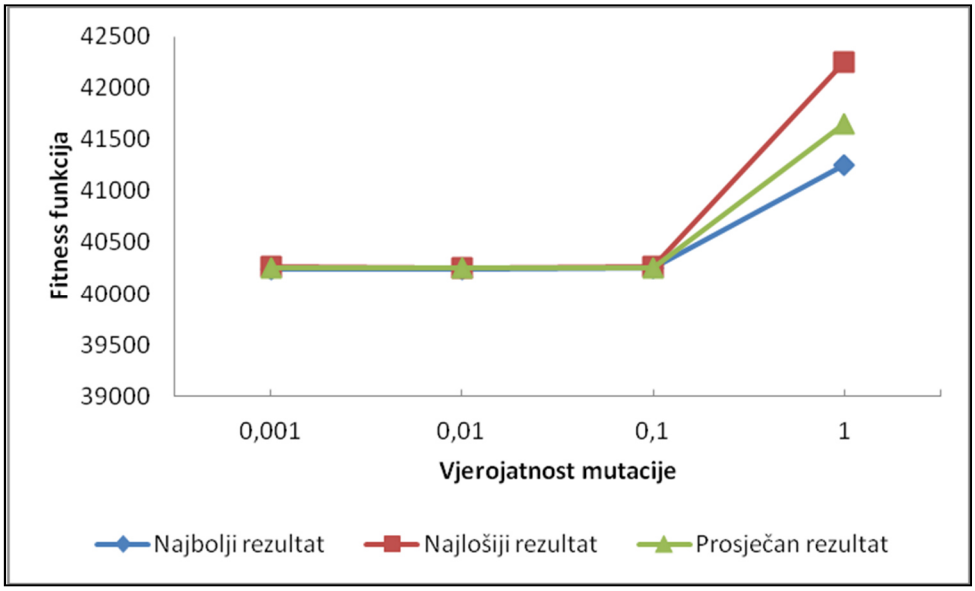
Usporedi li se prosječno vrijeme izvođenja modela 20 i 140 evolucija tada proizlazi da razlika prosječnog vremena izvođenja modela iznosi otprilike 47 minuta, što znatno produljuje rad modela. Stoga se na temelju prikazanih grafikona može zaključiti da se sa 20 evolucija postižu zadovoljavajući rezultati s obzirom na kvalitetu rezultata i vrijeme rada modela.

Posljednja dva testiranja provedena su za različite vjerojatnosti mutacije i križanja. U tablici 30 prikazani su najbolji, najlošiji i prosječni rezultati *fitness* funkcije dobiveni testiranje četiri različite vjerojatnosti mutacije s postotkom od 0,001, 0,001, 0,1 i 1. Također, prilikom svih pet testiranja pojedine vjerojatnosti mutacije mjereno je i vrijeme izvođenja modela čiji je prosjek iskazan u posljednjem stupcu tablice.

Tablica 30 Utjecaj vjerojatnosti mutacije na prosječan rezultat *fitness* funkcije i vrijeme izvođenja modela za veličinu populacije od 8.000 kromosoma i 20 evolucija

Vjerojatnost mutacije(%)	**Najbolji rezultat prema vrijednosti *fitness* funkcije**	**Najlošiji rezultat prema vrijednosti *fitness* funkcije**	**Prosječan rezultat prema vrijednosti *fitness* funkcije**	**Prosječno vrijeme izvođenja**
0,001	40.244	40.265	40.253	7'10"
0,01	40.243	40.252	40.249	10'05"
0,1	40.253	40.262	40.256	10'38"
1	41.250	42.255	41.652	13'49"

Na osnovu podataka iz tablice izrađen je grafički prikaz utjecaja vjerojatnosti mutacije na kvalitetu dobivenih rezultata *fitness* funkcije (Grafikon 13).



Grafikon 13 Osjetljivost *fitness* funkcije na različite vjerojatnosti mutacije

Iz grafičkog prikaza razvidno je da su, s gledišta najboljeg rezultata, odstupanja svih dobivenih rezultata vrlo mala u odnosu na najbolji dobiven rezultat (40.243). Iz navedenog može se zaključiti da model, unatoč promjeni postotka vjerojatnosti mutacije, svaki puta uspijeva pronaći rezultat vrlo blizu najboljeg dobivenog rezultata. Međutim, uzmu li se u obzir odstupanja između najboljeg, najlošijeg i prosječnog rezultata, uočava se da su ona najmanja kod vjerojatnosti mutacije od 0,01%, a najveća kod vjerojatnosti mutacije od 1%. Također, kod navedene vjerojatnosti mutacije najlošiji rezultat *fitness* funkcije iznosi 42.255, što znači da se kod testiranog odjeljka (6x7, 37 kontejnera) broj

radnji premještaja povećava s 40 na 42 premještaja kontejnera, što kvalitetom predstavlja nedovoljno dobro rješenje. Kod ostalih postotaka vjerojatnosti mutacije i najlošiji dobiveni rezultati imaju zadovoljavajuću kvalitetu rješenja. Najbolja dobivena prosječna vrijednost *fitness* funkcije je kod vjerojatnosti mutacije od 0,01%, što znači da se postavljanjem ove vjerojatnosti mutacije dobivaju najkvalitetnija rješenja, dok je za postizanje rezultata potrebno prosječno vrijeme od 10 minuta. Naime, na osnovu svih dobivenih prosječnih rezultata može se zaključiti da je model osjetljiv na promjenu vjerojatnosti mutacije.

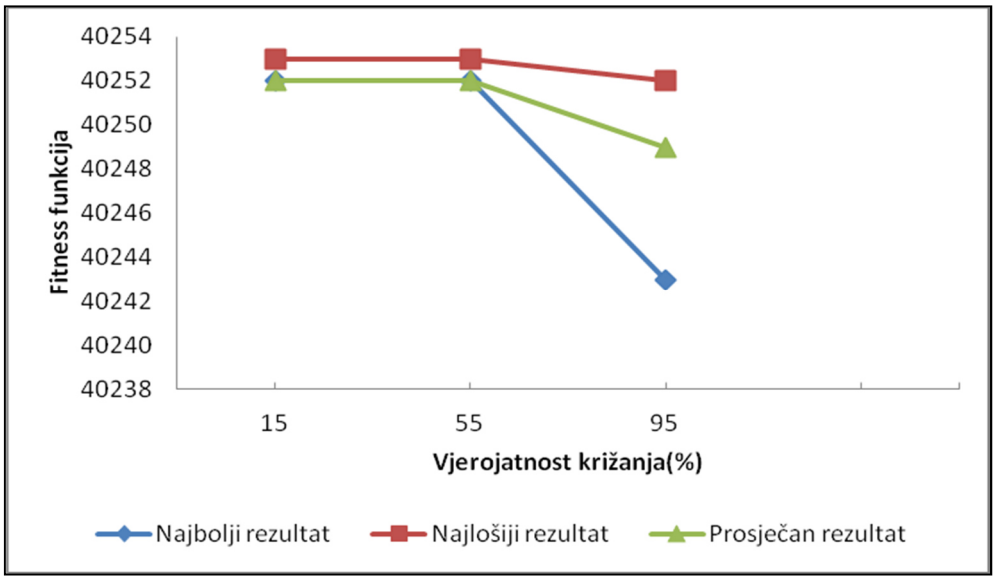
Tablica 31 objedinjuje rezultate *fitness* funkcije i prosječnog vremena izvođenja algoritma dobivenog testiranjem vjerojatnosti križanja s udjelima od 15%, 55%, 95% .

Tablica 31 Utjecaj vjerojatnosti križanja na prosječan rezultat *fitness* funkcije i vrijeme izvođenja modela za veličinu populacije od 8.000 kromosoma i 20 evolucija

Vjerojatnost križanja (%)	**Najbolji rezultat prema vrijednosti *fitness* funkcije**	**Najlošiji rezultat prema vrijednosti *fitness* funkcije**	**Prosječan rezultat prema vrijednosti *fitness* funkcije**	**Prosječno vrijeme izvođenja**
15	40.252	40.253	40.252	7'17"
55	40.252	40.253	40.252	9'05"
95	40.243	40.252	40.249	10'05"

Temeljem podataka iz tablice 31 izrađen je grafički prikaz utjecaja vjerojatnosti križanja na kvalitetu rezultata *fitness* funkcije (Grafikon 14).

Iz grafikona je razvidno da se sa vjerojatnošću križanja od 95% dobiva kvalitetniji najbolji i prosječni rezultat *fitness* funkcije. Premda je potrebno istaknuti da sa stajališta kvalitete rezultata, sva tri testirana udjela vjerojatnosti postižu kvalitetom zadovoljavajuće rezultate. Iz navedenog može se zaključiti da model nije osjetljiv na promjenu parametra vjerojatnosti križanja te da se povećanjem udjela vjerojatnosti prosječno vrijeme izvođenja modela minimalno produljuje.



Grafikon 14 Osjetljivost *fitness* funkcije na različite vjerojatnosti mutacije

Na kraju, na osnovu svih provedenih testiranja parametara može se zaključiti da su svi parametri rada GA postavljeni tako da se osigurava postizanje optimalnih ili najkvalitetnijih rješenja u relativno kratkom vremenu.

8. ZAKLJUČAK

Razvojem kontejnerskih brodova i rastom kontejnerskog prometa raste i potreba za skladištenjem sve većeg broja kontejnera, odnosno povećanjem postojećih kapaciteta slagališta. Stoga u brojnim svjetskim kontejnerskim terminalima postojeći slagališni kapaciteti postaju nedostatni. Rukovodstvo kontejnerskih terminala ulaže velike napore kako bi povećalo slagališne kapacitete, međutim, u mnogim terminalima zbog prostornih ograničenja to je moguće postići jedino povećanjem: gustoće slaganja, visine slaganja i udjela popunjenosti slagališta. Gustoću slaganja kontejnera na slagalištu moguće je povećati jedino upotrebom slagališne dizalice velike gustoće slaganja. Povećanjem visine slaganja i udjela popunjenosti kontejnera poveća se broj radnji premještaja kontejnera. Osim navedenog, uzrokom radnji premještaja nepotpune se, netočne i nedostupne informacije o vremenu otpreme kontejnera. Poznato je da radnje premještaja kontejnera pripadaju u neproduktivne radnje koje prouzrokuju dodatne troškove u radu i utječu na konkurentnost terminala na tržištu lučkih usluga. Stoga je za uspješno poslovanje lučkog kontejnerskog terminala izuzetno važno optimizirati broj radnji premještaja kontejnera. Na temelju utvrđenih činjenica, zaključuje se da je tematika ovog doktorskog rada aktualno i perspektivno područje za daljnja istraživanja.

U ovome istraživanju analizirana je metodologija problema BRP/CRP na slagalištu lučkoga kontejnerskog terminala. Definirane su osnovne postavke ove problematike i predloženog rješenja. Od osnovnih postavki potrebno je istaknuti da su veličine promatranih odjeljaka definirane sukladno tehničko-tehnološkim značajkama najsuvremenijih RTG dizalica te da su udjeli popunjenosti odjeljaka određeni kako bi simulirali moguća "zagušenja" odjeljka te ispitala učinkovitost rada predloženog modela u zadanim uvjetima. Također, nužno je istaknuti da simulirani odjeljci predstavljaju statičan sustav, odnosno da se predloženim modelom rješava problem premještaja i otpreme složenih kontejnera unutar odjeljka, odnosno da nema slaganja „novih" kontejnera u odjeljak. Navedeno je ujedno ograničavajući čimbenik primjene modela jer predloženi model ne simulira stvarnu situaciju u potpunosti, budući da je slagalište kontejnerskog terminala izrazito dinamičan sustav. Iako su na slagalištu kontejnerskog terminala moguće situacije kada pojedini odjeljci kontejnerskog bloka u kraćem promatranom razdoblju imaju obilježja statičnog sustava.

Ovaj problem u znanstvenoj teoriji pripada skupini slagališnih logistički problema koji se nastoje riješiti optimizacijskim metodama. Problem optimizacije razmještaja kontejnera obrađivan je u malom broju znanstvenih radova. Prvi rad na tu temu napisan je 1988. godine. U posljednje vrijeme autori znanstvenih radova za rješavanje navedenog problema koriste matematičke, statističke, heurističke i metaheurističke metode. Pretraživanjem dostupnih znanstvenih baza nije pronađen niti jedan znanstveni rad koji problem razmještaja kontejnera rješava primjenom genetskog algoritma. Stoga se u ovome doktorskom radu optimizacijski problem rješava primjenom genetskog algoritma, što predstavlja jedan od znanstvenih doprinosa ovog rada.

Formulirani genetski algoritam posebno je prilagođen optimizacijskom problemu s ciljem uspješnog rješavanje problema BRP/CRP na slagalištu lučkoga kontejnerskog terminala. Da bi se postigli što kvalitetniji rezultati, u model su integrirana konstruktivna heuristička pravila koja omogućavaju pronalazak kvalitetnijih rješenja.

Obilježje predloženog optimizacijskog modela je da se svakoj radnji premještaja kontejnera pridruži neko od četiriju definirana pravila koja određuju poziciju kontejnera koji se premješta. Rješenje dobiveno optimizacijskim modelom sadržano je u vrijednosti *fitness* funkcije koja predstavlja ukupan broj ostvarenih radnji premještaja te premještaj po širi i visini odjeljka, kao i kromosomu koji predstavlja set pravila koja su primijenjena na premještaj kontejnera. Na ovaj se način planerima slagališta olakšava donošenje odluka o premještaju kontejnera te se ubrzava proces njihove otpreme sa slagališta, budući da dizaličar može obavljati radnje premještaja i otpreme na temelju dobivenog seta pravila.

Potrebno je napomenuti da je, s obzirom na veličinu odjeljka, rad modela testiran na problemima malih i srednjih dimenzija odjeljaka. Razlog tome je što se u modelu simulira stvarna veličina odjeljka na kojemu se kao prekrcajno sredstvo koristi RTG dizalica, što nužno ne znači da model ne daje dovoljno kvalitetne rezultate i za probleme velikih dimenzija odjeljaka. Međutim, kako se velikim dimenzijama odjeljka smatraju odjeljci veličine 20x20, 20x30 itd., navedeni odjeljci nisu uzeti u obzir prilikom testiranja jer se smatraju nerealnim veličinama za rad RTG dizalica. Da bi se utvrdila učinkovitost rada modela, model je testiran na 4 različite veličine odjeljka te su za svaku od 4 veličine

izrađena 3 različita scenarija s obzirom na udio popunjenosti odjeljka. Za svaku veličinu i scenarij popunjenosti odjeljka slučajno je stvoren jedan početni raspored kontejnera unutar odjeljka. S ciljem utvrđivanja različitosti rješenja s aspekta dobivenog seta pravila(kromosoma rješenja), odjeljak zadane veličine i scenarija popunjenosti te istog početnog rasporeda testiran je prilikom deset pokretanja modela. Dakle, model je ukupno pokrenut 120 puta, a kao zaključak se nameće to da je za isti početni raspored odjeljka zadane veličine i scenarija popunjenosti odjeljka moguće dobiti jednako kvalitetno ili optimalno rješenje primjenom različitog seta pravila tj; premještanjem kontejnera na različite kontejnerske pozicije.

Dodatno testiranje modela provedeno je s ciljem usporedbe uspješnosti rada modela u odnosu na modele drugih autora. Testiranje je provedeno za odjeljke veličine 3x7, 4x7, 5x7 i 6x7 i jedan scenarij popunjenosti s udjelom od približno 90%. Za svaki navedeni odjeljak slučajno je generirano 40 različitih početnih rasporeda kontejnera unutar odjeljka, dakle ukupno je ostvareno 160 pokretanja modela. Navedene veličine odjeljka, scenarij popunjenosti te ukupan broj početnih rasporeda kontejnera unutar odjeljka jednaki su uvjetima u kojima su testirani modeli ostalih autora. Iz provedene analize i komparacije rezultata s rezultatima drugih autora, utvrđeno je da izrađeni model optimizacije, ovisno od veličini odjeljka, postiže sljedeće rezultate:

- za odjeljak veličine 3x7 postignuto je poboljšanje rezultata od 6,70%
- za odjeljak veličine 4x7 postignuto je poboljšanje rezultata od 7,61%
- za odjeljak veličine 5x7 postignuto je poboljšanje rezultata od 16,59% i
- za odjeljak veličine 6x7 postignuto je poboljšanje rezultata od 1,39%.

Dobiveni rezultati pokazuju da se optimizacijskim modelom za promatrane odjeljke postiglo znatno poboljšanje (do 16,59%) u odnosu na dosad najbolji postignut rezultat autora Wu et al. Otuda proizlazi i glavni znanstveni doprinos ovog rada. Rezultati dobiveni testiranjem različitih početnih rasporeda, veličina i udjela popunjenosti odjeljka pokazuju da je predloženim modelom ostvarena minimizacija ukupnog broja premještaja kontejnera, čime je potvrđena temeljna znanstvena hipoteza. Dobiveni rezultati potvrđuju znanstvenu hipotezu u svim testiranim slučajevima, čime se garantira uspješnost primjene modela na svim slagalištima kontejnerskih terminala koja za rad koriste RTG dizalicu.

Uz navedeno, provedena je jednostruka analiza osjetljivosti rada modela na promjenu parametara veličina populacije, broj evolucija, vjerojatnost mutacije i vjerojatnost križanja s ciljem ispitivanja promjene rezultata modela, u ovisnosti o promjenama vrijednosti parametara. Za svaku vrstu i vrijednost parametra genetskog algoritma izvršeno je pet pokretanja modela. Analiza osjetljivosti rada modela testirana je na odjeljku veličine 6x7 te scenariju s udjelom popunjenosti od približno 90%. Na osnovu provedene analize zaključuje se da parametar vjerojatnosti mutacije najviše utječe na kvalitetu rezultata modela te da su svi parametri rada genetskog algoritma postavljeni tako da osiguravaju postizanje optimalnih ili najkvalitetnijih rješenja u relativno kratkom vremenu.

Daljnja istraživanja mogu ići u više smjerova. Budući da su za rješavanje ovog problema formulirani modeli čije je vrijeme rješavanja od svega 1 sekundu, iznimno je važno da se prvi smjer istraživanja odnosi na poboljšanje brzine izvođenja same optimizacije. Drugi bi bio da daljnja istraživanja ispitaju rad predloženog modela za probleme velikih dimenzija odjeljaka, uz eventualno razvijanje novih operatora. Treći smjer odnosio bi se na implementiranje *look ahead* strategije u izrađeni model te istraživanje i testiranje utjecaja na kvalitetu dobivenih rezultata. Četvrti smjer mogao bi imati za cilj proširenje modela na način da se istovremeno optimiziraju radnje premještaja kontejnera unutar više odjeljaka, što predstavlja zamjetno složeniji problem. Peti smjer odnosio bi se na testiranje modela u uvjetima rada RTG dizalice koja istovremeno izvodi prekrcajne radnje s dva kontejnera. Šesti smjer mogao bi imati za cilj proširenje modela na 3D problem.

POZIVNE BILJEŠKE

[1] http://www.containerhandbuch.de/

[2] http://www.prometna-zona.com/

[3] Belamarić, G., 2014., Tehnologija prijevoza kontejnera., Pomorski fakultet u Splitu, Split.

[4] Radić, J., 2013., Compatibility of means and equipment of intermodal transport, Fakultet prometnih znanosti, Sveučilište u Zagrebu, Zagreb.

[5] http://www.worldshipping.org/

[6] Bojanić, V., 2012., Metodologija planiranja rada obalskih kontejnerskih dizalica pri pretovaru bodova i barži, Fakultet tehničkih nauka, Novi Sad.

[7] http://www.alphaliner.com/top100/

[8] Mohseni, N. S., 2011., Developing a Tool for Designing a Container Terminal Yard, Delft University of Technology, Netherland.

[9] Böse, J. W., 2011. Handbook of Terminal Planning. Springer, London.

[10] Solomenikovs, A., 2006., Simulation Modelling and Research of Marine Container Terminal Logistics Chains, Transport and Telecommunication Institute, Riga.

[11] Nazari, D., 2005., Evaluating container yard layout: A simulation approach Rotterdam: Erasmus University, Rotterdam.

[12] Steenken, D., Voß, S., Stahlbock,R. 2004., Container terminal operations and operations research- a classification and literature review, OR Spectrum, vol. 26, no. 1, pp. 3–49.

[13] Kemme, N., 2013., Design and Operation of Automated Container Storage Systems, Springer, London.

[14] Huynh, N. N., 2005., Methodologies for Reducing Truck Turn Time at Marine Container Terminals, Center of Transportation Research, Univesity of Texas, Austin.

[15] AGCT- Operativni priručnik za korisnike usluga AGCT-a., Adriatic Gate Container Terminal, Rijeka, 2014.

[16] Zehendner, E., 2013., Operations managemet at container terminals using advanced information technologies, Ecole Nationale Supérieure des Mines de Saint-Étienne, Gardanne, France.

[17] Tijan, E., Agatić, A., Hlača, B., 2010., Evolucija informacijsko-komunikacijskih tehnologija na kontejnerskim terminalima, Pomorstvo, vol. 24, no. 1, pp. 27–40.

[18] Guan, C. Q., 2009., Analysis of Marine Container Terminal Gate Congestion, Truck Waiting Cost and System Optimization, New Jersey Institute of Technology, New Jersey.

[19] Wiese , J., 2012., Quantitative Decision Support for the Layout Design of Container Terminal, Universitat Paderborn, Paderborn.

[20] Alcalde, E. M., 2014, Strategies for improving import yard performance at container marine terminals. Universitat Politecnica de Catalunya, Barcelona.

[21] Kun, L. M., 2010., Efficient algorithms for designing yard storage templates for export containers, University of Hong Kong, Hong Kong.

[22] Tus, A., 2014, Heuristic Solution Approaches for Two Dimensional Pre-marshalling Problem, Vienna University of Technology, Faculty of Informatics, Vienna.

[23] Sinha, D., 2011., Container Yard Capacity Planning: A causal Approach, Indian Institut of Foreign Trade, New Delhi.

[24] Salminen, J. B., 2013., Measuring the Capacity of a Port SYstem: A Case Study on a Southeast Asian Port, Massachusetts Institute of Technology, Massachusetts.

[25] Dundović, Č., 2005., Prekrcajna sredstva prekidnog transporta. Rijeka: Sveučilište u Rijeci, Pomorski fakultet u Rijeci.

[26] Bojanić, G., 2012., Formulacija matematičkih modela za pretovarne sisteme i razvoj optimizacione metode za planiranje kontejnerskih pretovara kod vozova, Fakultet tehničkih nauka, Novi Sad.

[27] Henesey, L. E., 2006., Multi-Agent Systems for Container Terminal Management, Department of Systems and Software Engineering, School of Engineerin, Blekinge Institute of Technology, Ronneby.

[28] Klaws,J., Stahlbock,R., Voß,S., 2011., Container Terminal Yard Operations – Simulation of a Side-Loaded Container Block Served by Triple Rail Mounted Gantry Cranes, in Computational Logistics, J. W. Böse, H. Hu, C. Jahn, X. Shi, R. Stahlbock, and S. Voß, Eds. Springer Berlin Heidelberg, 2011, pp. 243–255.

[29] Barukčić, M., Hederić, Ž., Jović, F., 2008., Prilagodba genetskog algoritma učinkovitijoj minimizaciji djelatnih gubitaka elektroenergetske mreže, Tehnički vjesnik, vol. 15, no. 3, pp. 11–19.

[30] Wu,K. C., Ting, C. J., 2010., A beam search algorithm for minimizing reshuffle operations at container yards, Proceedings of the 2010 International Conference on Logistics and Maritime Systems, Busan.

BIBLIOGRAFIJA

Knjige

1. Bauk, S.I., 2011. Kvantitativne metode optimizacije u funkciji naučnog menadžmenta, Ekonomska laboratorija za istraživanje tranzicije Podgorica, Podgorica.
2. Böse, J.W., 2011. Handbook of Terminal Planning. Springer, London.
3. Dundović, Č., 2005. Prekrcajna sredstva prekidnog transporta, Sveučilište u Rijeci, Pomorski fakultet u Rijeci, Rijeka.
4. Dundović, Č., 2002. Lučki terminali. Sveučilište u Rijeci, Pomorski fakultet u Rijeci, Rijeka.
5. Kemme, N., 2013. Design and Operation of Automated Container Storage Systems, Springer. ed., London.
6. Klaws, J., Stahlbock, R., Voß, S., 2011. Container Terminal Yard Operations – Simulation of a Side-Loaded Container Block Served by Triple Rail Mounted Gantry Cranes, in: Böse, J.W., Hu, H., Jahn, C., Shi, X., Stahlbock, R., Voß, S. (Eds.), Computational Logistics. Springer- Verlag Berlin Heidelberg, pp. 243–255.
7. Michalewicz, Z., Fogel, D.B., 2004. How To Solve It: Modern Heuristics. Springer, London.
8. Nazari, D., 2005. Evaluating container yard layout: A simulation approach. Erasmus University Rotterdam, Rotterdam.
9. Thoresen, C.A., 2010. Port Designer's Handbook. Thomas Telford, London.
10. Vidović, M.B., 2007. Kvantitativna analiza sistema rukovanja materijalom. Univerzitet u Beogradu, Saobraćajni fakultet, Beograd.
11. Živković, D., 2007. Osnove dizajna i analize algoritma. Univerzitet u Beogradu, Računarski fakultet, Beograd.

Članci, studije, elaborati i priručnici

12. Barukčić, M., Hederić, Ž., Jović, F., 2008. Prilagodba genetskog algoritma učinkovitijoj minimizaciji djelatnih gubitaka elektroenergetske mreže. Tehnički vjesnik 15, 11–19.
13. Bazzazi, M., Safaei, N., Javadian, N., 2009. A genetic algorithm to solve the storage space allocation problem in container terminal. Computers and Industrial Engineering 56, 44–52.
14. Borgman, B., van Asperen, E., Dekker, R., 2010. Online rules for container stacking. OR Spectrum 32, 687–716.

15. Caserta, M., Schwarze, S., 2009. A New Binary Description of the Blocks Relocation Problem and Benefits in a Look Ahead Heuristic. Evolutionary Computation in Combinatorial Optimization 5482, 37–48.

16. Caserta, M., Voß, S., 2009. A Corridor Method-based Algorithm for the Premarshalling Problem. Applications of Evolutionary Computing 5484, 788–797.

17. Dekker, R., Voogd, P., n.d. Advanced methods for container stacking. OR Spectrum 28, 563–586.

18. Dundović, Č., Hess, S., 2005. Exploitability of the Port Container Terminal Stacking Area Capacity in the Circumstances of Increased Turnover. Zbornik Referatov - Intelligent Transport Systems - ITS.

19. Dundović, Č., Hess, S., Šantić, L., 2006. Proračun opterećenja i kapaciteta kontejnerskog terminala Ploče. Pomorstvo 20, 79–95.

20. Dundović, Č., Zenzerović, Z., 2000. An Optimal Capacity Planning Model for General Cargo Seaport. Promet-Traffic-Traffico, 217–221.

21. Froyland, G., Koch, T., Megow, N., Duane, E., Wren, H., 2008. Optimizing the landside operation of container terminal. OR Spectrum 30, 53–75.

22. Georgijević, M., Bojanić, V., Bojanić, G., Bojić, S., 2012. Simulation as the optimization tool- application to the complex logistic systems. Proceedings of Small Systems Simulation Symposium, 37-42.

23. Golbabaie, F., Seyedalizadeh Ganji, S.R., Arabshahi, N., 2012. Multi-criteria evaluation of stacking yard configuration. Journal of King Saud University - Science 24, 39–46.

24. Guldogan, E.U., 2010. Simulation-based analysis for hierarchical storage assignment policies in container terminal. Simulation 87, 523–537.

25. Gunther, H.O., Kim, K.H., 2006. Container terminals and terminal operations. OR Spectrum 28, 437–445.

26. Gupta, N., 1992. On the complexity of blocks-world planning. Artificial Intelligence 56, 223–254.

27. Kang, J., Oh, M.S., Ahn, E.Y., Ryu, K.R., Kim, K.H., 2006. Planning for Intra- block Remarshalling in a Container Terminal. Artificial Intelligence 60, 302–318.

28. Kim, K.H., 1997. Evaluation of the number of rehandles in container yards. Computers and Industrial Engineering 32, 701–711.

29. Kim, K.H., Bae, J.W., 1998. Re-marshalling export containers in port container terminals. Computers and Industrial Engineering 35, 655–658.

30. Kim, K.H., Hong, G.P., 2006. A heuristic rule for relocating blocks. Computers&OR 33, 940–954.

31. Kim, K.H., Park, K.T., 2003. A note on a dynamic space-allocation method for outbound containers. European Journal of Operational Research 148, 92–101.

32. Kim, K.H., Park, Y.M., Ryu, K.R., 2000. Deriving decision rules for export containers in container yards. European Journal of Operational Research 124, 89–101.

33. Kim, K.Y., Kim, K.H., 2003. Heuristic algorithms for routing yard-side equipment for minimizing loading times in container terminals. Naval Research Logistics 50, 498–514.

34. Kozan, E., Preston, P., 1999. Genetic algorithms to schedule container transfers at multimodal terminals. International Transactions in Operational Research 6, 311–329.

35. Lee, B.K., Kim, K.H., 2010. Optimizing the block size in container yards. Transportation Research Part E 46, 120–135.

36. Lee, Y., Chao, S.L., 2009. A neighborhood search heuristics for pre-marshalling export containers. European Journal of Operational Research 196, 468–475.

37. Lee, Y., Hsu, N.Y., 2007. An optimization model for the container pre-marshalling problem. Computers&OR 34, 3295–3313.

38. Li, W., Wu, Z., Petering, M.E.H., Goh, M., de Souza, R., 2009. Discrete time model and algorithms for container yard scheduling. European Journal of Operational Research 198, 165–172.

39. Meersmans, P.J.M., Dekker, R., 2001. Operations research supports container handling. Technical Report EI 22, 1–49.

40. Park, K., Dragović, B., 2009. A study of Container Terminal Planning. Faculty of Mechanic Engineering Transactions 37, 203–209.

41. Park, K., Park, T., Ryu, K.R., 2009. Planning for Remarshalling in an Automated Container Terminal using Cooperative Coevolutionary Alorithms. Proceedings of the 2009 ACM Symposium on Applied Computing, 1098–1105.

42. Sriphrabu, P., Sethanan, K., Arnonkijpanich, B., 2013. A solution of the Container Stacking Problem by Genetic Algorithm. International Journal of Engineering Technology 5, 45–48.

43. Steenken, D., Voß, S., Stahlbock, R., 2004. Container terminal operations and operations research- a classification and literature review. OR Spectrum 26, 3–49.

44. Taleb-Ibrahimi, M., De Castilho, B., Daganzo, C.F., 1993. Storage space vs handling work in container terminals. Transportation Research Part B 27, 13–32.

45. Tijan, E., Agatić, A., Hlača, B., 2010. Evolucija informacijsko-komunikacijskih tehnologija na kontejnerskim terminalima. Pomorstvo 24, 27–40.

46. Vis, I.F.A., de Koster, R., 2003. Transshipment of containers at a container terminal: an overview. European Journal of Operational Research 147, 1–16.

47. Wan, Y., Liu, J., Tsai, P.C., 2009. The assignment of storage locations to containers for container stack. Naval Research Logistics 56, 699–713.

48. Woldesenbet, Y.G., Yen, G.G., 2009. Dynamic Evolutionary Algorithm with Variable Relocation. Evolutionary Computation, IEEE Transaction 13, 500–513.

49. Wu, K.C., Ting, C.J., 2010. A beam search algorithm for minimizing reshuffle opertionc at container yards. Proceedings of the International Conference on Logistics and Maritime Systems.

50. Yang, J.H., Kim, K.H., 2006. Agrouped storage method for minimizing relocations in block stacking systems. Journal of Intelligent Manufacturing 17, 453–463.

51. Zhang, G.Z., 2010. An Investigation of IDA* Algorithms for the Container Relocation Problem. IEA/AIE, Part I, 31–40.

Doktorski radovi

52. Alcalde, E.M., 2014. Strategies for improving import yard performance at container marine terminals. Universitat Politecnica de Catalunya, Barcelona.

53. Ayub, Y., 2009. Container Terminal Operations Modeling through Multi-Agent based Simulation. School of Computing, Blekinge Institut of Technology, Ronneby.

54. Bojanić, G., 2012. Formulacija matematičkih modela za pretovarne sisteme i razvoj optimizacione metode za planiranje kontejnerskih pretovara kod vozova. Fakultet tehničkih nauka, Novi Sad.

55. Bojanić, V., 2012. Metodologija planiranja rada obalskih kontejnerskih dizalica pri pretovaru bodova i barži. Fakultet tehničkih nauka, Novi Sad.

56. Casey, B., 2011. Optimising Container Processes at Multimodal Seaport Container Terminals. An Integrated Approach and Application. Queensland University of Technology, Faculty of Science and Technology, Discipline of Mathematical Sciences, Queensland.

57. Gadeyne, B., 2011. Optimizing Maritime Container Terminal Operations. Ghent University, Faculty of Economics and Business Administration, Ghent.

58. Guan, C.Q., 2009. Analysis of Marine Container Terminal Gate Congestion, Truck Waiting Cost and System Optimization. New Jersey Institute of Technology, New Jersey.

59. Henesey, L.E., 2006. Multi-Agent Systems for Container Terminal Management. Department of Systems and Software Engineering, School of Engineering, Blekinge Institute of Technology, Ronneby.

60. Hussein, M.I., 2012. Container handling algorithms and outbound heavy truck movement modeling for seaport container transshipment terminals. University of Wisconsin- Milwaukee, Milwaukee.

61. Huynh, N.N., 2005. Methodologies for Reducing Truck Turn Time at Marine Container Terminals. Center of Transportation Research, University of Texas, Austin.

62. Jiangang, J., 2012. Storage Yard Management for Container Transshipment Terminals. National University of Singapore, Singapore.

63. Kun, L.M., 2010. Efficient algorithms for designing yard storage templates for export containers. University og Hong Kong, Hong Kong.

64. Mohseni, N.S., 2011. Developing a Tool for Designing a Container Terminal Yard. Delft University of Technology, Netherland.

65. Radić, J., 2013. Compatibility of means and equipment of intermodal transport. Fakultet prometnih znanosti, Sveučilište u Zagrebu, Zagreb.

66. Salminen, J.B., 2013. Measuring the Capacity of a Port SYstem: A Case Study on a Southeast Asian Port. Massachusetts Institute of Technology, Massachusetts.

67. Sibbesen, L.K., 2008. Mathematical models and heuristic solution for container positioning problems in port terminals. Techinal University of Denmark, Department of Management Engineering ,Lyngby.

68. Solomenikovs, A., 2006. Simulation Modelling and Research of Marine Container Terminal Logistics Chains. Transport and Telecommunication Institute, Riga.

69. Tus, A., 2014. Heuristic Solution Approaches for the Two Dimensional Pre-marshalling Problem. Vienna University of Technology, Faculty of Informatics, Vienna.

70. Wiese, J., 2012. Quantitative Decision Support for the Layout Design of Container Terminal . Universitat Paderborn, Paderborn.

71. Zehendner, E., 2013. Operations managemet at container terminals using advanced information technologies. Ecole Nationale Supérieure des Mines de Saint-Étienne, Gardanne, France.

Internetski i drugi izvori

72. AGCT- Operativni priručnik za korisnike usluga AGCT-a., Adriatic Gate Container Terminal, Rijeka, 2014.

73. G. Belamarić, Tehnologija prijevoza kontejnera., Prezentacija, Pomorski fakultet u Splitu, 2014.

74. Sinha, D., 2011. Container Yard Capacity Planning: A causal Approach, Indian Institut of Foreign Trade, New Delhi.

75. http://www.alphaliner.com/top100/
76. http://www.containerhandbuch.de/
77. http://www.worldshipping.org/
78. http://www.prometna-zona.com/

Popis slika

Popis grafikona

Popis tablica

Popis simbola

d_k	–	Duljina kromosoma
DRK_d	–	Dinamički stvarni kapacitet slagališnih dizalica
DRK_S	–	Dinamički stvarni kapacitet slagališta
DTK_d	–	Dinamički teorijski kapacitet dizalica
DTK_S	–	Dinamički teorijski kapacitet slagališta
H	–	Ukupan broj redova u odjeljku
i	–	Broj redova (visina) odjeljka
j	–	Broj stupaca (širina)odjeljka
k	–	Kontejner
Kp	–	Kontejnerska pozicija
k_{ps}	–	Koeficijent popunjenosti slagališta
N	–	Veličina populacije
N_k	–	Ukupan broj kontejnera složenih unutar odjeljka
n_d	–	Broj slagališnih dizalica
ne	–	Broj evolucija
n_{kp}	–	Broj kontejnerskih pozicija za smještaj 20' kontejnera
n_{rv}	–	Prosječan broj redova po visini za smještaj 20' kontejnera
pc	–	Vjerojatnost križanja
p_d	–	Proizvodnost dizalice
pm	–	Vjerojatnost mutacije
SRK_S	–	Statički stvarni kapacitet slagališta
STK_S	–	Statički teorijski kapacitet slagališta
t_{dt}	–	Dnevno radno vrijeme terminala
t_r	–	Broj radnih dana u godini
t_z	–	Prosječno vrijeme zadržavanja TEU kontejnera
W	–	Ukupan broj stupaca u odjeljku

Popis kratica

AGV	–	Automated guided vehicles
ASC	–	Automatic stacking cranes
B&B	–	Branch and bound
BAP	–	Berth allocation problem
BRP	–	Block relocation problem
BWP	–	Blocks-world planning
CRP	–	Container relocation problem
DGPS	–	Differential global positioning system
DH	–	Difference heuristics
DIN	–	Deutsches institut für normung
EDI	–	Electronic data interchange
EILU	–	European intermodal loading unit
GA	–	Genetic algorithm
GPS	–	Global positioning system
IT	–	Information technology
JGAP	–	Java genetic algorithm and programming
KB	–	Kontejnerski blokovi
LA	–	Look ahead
LIFO	–	Last-in, first-out
LKT	–	Lučki kontejnerski terminal
MTS	–	Multi trailer system
NP hard	–	Non- deterministic polynomial hard problem
OCR	–	Optical character recognition
PMP	–	Premarshalling problem
QC	–	Quay container crane
QCAP	–	Quay crane allocation problem
RFID	–	Radio frequency identification
RI	–	Reshuffle index
RMG	–	Rail mounted gantry crane
RMP	–	Remarshalling problem
RS	–	Reachstacker
RTG	–	Rubber tyred gantry crane
SAP	–	Storage space allocation
SC	–	Straddle carrier

TEU – Twenty-foot equivalent unit
TLP – Tier lowest position
TOS – Terminal operating system
TRMG – Triple rail mounted gantry crane
TTU – Truck trailer unit
TWRMG – Twin rail mounted gantry crane
XT – External truck

Privitak 1. - Izvadak iz programskog koda u JGAP-u

```
package genetskialgoritam;
import java.util.ArrayList;
import java.util.List;
import org.jgap.Chromosome;
import org.jgap.Configuration;
import org.jgap.DefaultFitnessEvaluator;
import org.jgap.FitnessFunction;
import org.jgap.Gene;
import org.jgap.Genotype;
import org.jgap.IChromosome;
import org.jgap.event.EventManager;
import org.jgap.gp.impl.GPConfiguration;
import org.jgap.impl.BestChromosomesSelector;
import org.jgap.impl.ChromosomePool;
import org.jgap.impl.CrossoverOperator;
import org.jgap.impl.DefaultConfiguration;
import org.jgap.impl.GABreeder;
import org.jgap.impl.IntegerGene;
import org.jgap.impl.MutationOperator;
import org.jgap.impl.StockRandomGenerator;
import org.jgap.impl.TournamentSelector;
import org.jgap.impl.WeightedRouletteSelector;
/**
**
@author Lee
*/
public class GenetskiAlgoritam {
private static int MAX_ALLOWED_EVOLUTIONS = 20;
/**
* @param args the command line arguments
*/
public static void main(String[] args) throws Exception {
//int rows = 3;//
//int rows = 4;//
//int rows = 5;//
int rows = 6;//
int columns = 7;
//int brojKontejnera = 11;//3*7 (55%)
//int brojKontejnera = 15;//3*7 (75%)
//int brojKontejnera = 19;//3*7 (90%)
//int brojKontejnera = 15;//4*7 (55%)
//int brojKontejnera = 21;//4*7 (75%)
//int brojKontejnera = 25;//4*7 (90%)
//int brojKontejnera = 19;//5*7 (55%)
//int brojKontejnera = 26;//5*7 (75%)
//int brojKontejnera = 31;//5*7 (90%)
//int brojKontejnera = 23;//6*7 (55%)
//int brojKontejnera = 31;//6*7 (75%)
int brojKontejnera = 37;//6*7 (90%)
//int[][] pozicijeKontejnera = {{0, 0, 11, 9, 0, 8, 0}, {6, 0, 10, 7, 0, 5, 0}, {2,
0, 1, 3, 0, 4, 0}};// zapis po redovima 3*7 za 11
kontejnera(55%)
//int[][] pozicijeKontejnera = {{0, 12, 0, 11, 10, 14, 7}, {0, 8, 0, 15, 5, 9, 13},
{0, 6, 0, 1, 3, 4, 2}};// zapis po redovima 3*7
za 15 kontejnera(75%)
//int[][] pozicijeKontejnera = {{0, 12, 0, 11, 10, 14, 7}, {18, 8, 19, 15, 5, 9,
13}, {16, 6, 17, 1, 3, 4, 2}};// zapis po redovima
3*7 za 19 kontejnera(90%)
// za testiranje razliÄŤitih poÄŤetnih rasporeda za odjeljka 3x7(generirano u R-
u)//
//int[][] pozicijeKontejnera = {{6, 3, 2, 9, 7, 0, 0}, {14, 19, 11, 15, 12, 17,
10}, {4, 13, 5, 1, 8, 16, 18}};// zapis po redovima
3*7 za 19 kontejnera(90%)1.varijanta
```

```
//int[][] pozicijeKontejnera = {{0, 13, 0, 9, 6, 11, 4}, {17, 5, 8, 7, 16, 12, 19},
{1, 10, 2, 3, 14, 15, 18}};// zapis po redovima
3*7 za 19 kontejnera(90%)2.varijanta
//int[][] pozicijeKontejnera = {{0, 5, 4, 15, 14, 17, 2}, {0, 9, 11, 12, 18, 19,
13}, {16, 10, 8, 3, 6, 7, 1}};// zapis po redovima
3*7 za 19 kontejnera(90%)3.varijanta
//int[][] pozicijeKontejnera = {{15, 4, 14, 1, 0, 0, 19}, {16, 2, 12, 17, 11, 7,
8}, {18, 5, 9, 13, 10, 6, 3}};// zapis po redovima
3*7 za 19 kontejnera(90%)4.varijanta
//int[][] pozicijeKontejnera = {{2, 0, 8, 0, 10, 3, 1}, {11, 5, 14, 7, 18, 13, 4},
{6, 12, 19, 9, 17, 16, 15}};// zapis po redovima
3*7 za 19 kontejnera(90%)5.varijanta
//int[][] pozicijeKontejnera = {{19, 4, 0, 1, 0, 7, 8}, {2, 14, 12, 15, 6, 3, 18},
{17, 16, 11, 9, 5, 10, 13}};// zapis po redovima
3*7 za 19 kontejnera(90%)6.varijanta
//int[][] pozicijeKontejnera = {{8, 14, 0, 1, 0, 6, 17}, {13, 12, 9, 11, 18, 3, 5},
{19, 4, 15, 10, 7, 16, 2}};// zapis po redovima
3*7 za 19 kontejnera(90%)7.varijanta
//int[][] pozicijeKontejnera = {{0, 1, 17, 12, 0, 7, 19}, {6, 11, 2, 9, 8, 15, 14},
{16, 18, 10, 4, 13, 5, 3}};// zapis po redovima
3*7 za 19 kontejnera(90%)8.varijanta
//int[][] pozicijeKontejnera = {{0, 18, 0, 3, 12, 7, 14}, {10, 17, 13, 6, 4, 16,
8}, {1, 5, 11, 2, 9, 15, 19}};// zapis po redovima
3*7 za 19 kontejnera(90%)9.varijanta
//int[][] pozicijeKontejnera = {{10, 12, 14, 0, 15, 0, 7}, {5, 4, 3, 2, 1, 9, 17},
{6, 16, 8, 19, 13, 18, 11}};// zapis po redovima
3*7 za 19 kontejnera(90%)10.varijanta
//int[][] pozicijeKontejnera = {{0, 18, 0, 7, 4, 2, 1}, {6, 8, 11, 19, 16, 9, 3},
{17, 12, 15, 5, 10, 13, 14}};// zapis po redovima
3*7 za 19 kontejnera(90%)11.varijanta
//int[][] pozicijeKontejnera = {{18, 0, 16, 0, 10, 12, 14}, {11, 1, 15, 7, 5, 19,
8}, {2, 4, 9, 13, 3, 17, 6}};// zapis po redovima
3*7 za 19 kontejnera(90%)12.varijanta
//int[][] pozicijeKontejnera = {{17, 8, 7, 0, 14, 1, 0}, {10, 4, 9, 3, 6, 16, 13},
{5, 15, 19, 11, 2, 18, 12}};// zapis po redovima
3*7 za 19 kontejnera(90%)13.varijanta
//int[][] pozicijeKontejnera = {{2, 7, 17, 0, 14, 19, 0}, {10, 4, 16, 5, 18, 6,
13}, {8, 12, 1, 11, 3, 15, 9}};// zapis po redovima
3*7 za 19 kontejnera(90%)14.varijanta
//int[][] pozicijeKontejnera = {{0, 19, 0, 11, 2, 6, 7}, {5, 18, 17, 10, 9, 14, 4},
{3, 8, 1, 12, 16, 15, 13}};// zapis po redovima
3*7 za 19 kontejnera(90%)15.varijanta
//int[][] pozicijeKontejnera = {{13, 2, 18, 6, 0, 7, 0}, {14, 16, 8, 3, 10, 11, 5},
{19, 17, 15, 4, 9, 12, 1}};// zapis po redovima
3*7 za 19 kontejnera(90%)16.varijanta
//int[][] pozicijeKontejnera = {{8, 11, 0, 12, 3, 4, 0}, {16, 1, 13, 9, 15, 19,
17}, {6, 14, 7, 5, 10, 2, 18}};// zapis po redovima
3*7 za 19 kontejnera(90%)17.varijanta
//int[][] pozicijeKontejnera = {{1, 0, 0, 12, 7, 11, 9}, {8, 2, 18, 15, 19, 16,
13}, {3, 5, 17, 14, 4, 10, 6}};// zapis po redovima
3*7 za 19 kontejnera(90%)18.varijanta
//int[][] pozicijeKontejnera = {{18, 6, 16, 0, 19, 4, 13}, {11, 15, 9, 0, 5, 1,
12}, {17, 10, 14, 7, 3, 2, 8}};// zapis po redovima
3*7 za 19 kontejnera(90%)19.varijanta
//int[][] pozicijeKontejnera = {{0, 0, 18, 7, 15, 4, 17}, {5, 2, 6, 9, 14, 11, 13},
{10, 12, 3, 1, 8, 19, 16}};// zapis po redovima
3*7 za 19 kontejnera(90%)20.varijanta
//int[][] pozicijeKontejnera = {{6, 13, 18, 17, 0, 0, 1}, {2, 15, 5, 14, 9, 7, 12},
{11, 10, 16, 4, 19, 8, 3}};// zapis po redovima
3*7 za 19 kontejnera(90%)21.varijanta
//int[][] pozicijeKontejnera = {{0, 6, 2, 10, 0, 19, 8}, {13, 4, 16, 3, 14, 7, 18},
{17, 5, 15, 11, 12, 9, 1}};// zapis po redovima
3*7 za 19 kontejnera(90%)22.varijanta
//int[][] pozicijeKontejnera = {{9, 13, 0, 3, 18, 2, 0}, {11, 15, 6, 8, 19, 12, 4},
{16, 14, 7, 10, 17, 1, 5}};// zapis po redovima
3*7 za 19 kontejnera(90%)23.varijanta
//int[][] pozicijeKontejnera = {{9, 7, 10, 19, 11, 3, 0}, {13, 14, 12, 2, 1, 6, 0},
{18, 4, 15, 5, 16, 17, 8}};// zapis po redovima
```

```
3*7 za 19 kontejnera(90%)24.varijanta
//int[][] pozicijeKontejnera = {{17, 8, 11, 6, 0, 13, 1}, {18, 14, 12, 7, 0, 15,
5}, {10, 3, 2, 4, 9, 16, 19}};// zapis po redovima
3*7 za 19 kontejnera(90%)25.varijanta
//int[][] pozicijeKontejnera = {{15, 3, 14, 0, 5, 0, 13}, {6, 8, 7, 11, 16, 18,
10}, {2, 1, 19, 12, 9, 4, 17}};// zapis po redovima
3*7 za 19 kontejnera(90%)26.varijanta
//int[][] pozicijeKontejnera = {{0, 0, 17, 9, 7, 19, 13}, {2, 1, 16, 12, 8, 10, 6},
{3, 15, 5, 14, 4, 18, 11}};// zapis po redovima
3*7 za 19 kontejnera(90%)27.varijanta
//int[][] pozicijeKontejnera = {{14, 0, 5, 17, 4, 8, 15}, {7, 0, 12, 3, 19, 10, 2},
{9, 6, 1, 13, 18, 16, 11}};// zapis po redovima
3*7 za 19 kontejnera(90%)28.varijanta
//int[][] pozicijeKontejnera = {{7, 0, 13, 0, 4, 17, 10}, {1, 2, 6, 15, 9, 19, 12},
{11, 8, 5, 3, 14, 16, 18}};// zapis po redovima
3*7 za 19 kontejnera(90%)29.varijanta
//int[][] pozicijeKontejnera = {{11, 6, 0, 14, 9, 17, 12}, {19, 16, 0, 7, 18, 4,
1}, {10, 15, 3, 5, 2, 8, 13}};// zapis po redovima
3*7 za 19 kontejnera(90%)30.varijanta
//int[][] pozicijeKontejnera = {{7, 17, 3, 14, 12, 0, 19}, {1, 2, 5, 10, 8, 0, 6},
{4, 15, 18, 13, 16, 11, 9}};// zapis po redovima
3*7 za 19 kontejnera(90%)31.varijanta
//int[][] pozicijeKontejnera = {{18, 6, 2, 14, 0, 11, 0}, {7, 4, 13, 3, 12, 5, 17},
{9, 16, 10, 15, 8, 19, 1}};// zapis po redovima
3*7 za 19 kontejnera(90%)32.varijanta
//int[][] pozicijeKontejnera = {{12, 10, 17, 11, 8, 4, 0}, {2, 18, 6, 3, 14, 15,
0}, {1, 7, 13, 9, 19, 16, 5}};// zapis po redovima
3*7 za 19 kontejnera(90%)33.varijanta
//int[][] pozicijeKontejnera = {{0, 13, 2, 12, 0, 1, 15}, {9, 16, 10, 14, 4, 18,
19}, {7, 17, 5, 8, 11, 3, 6}};// zapis po redovima
3*7 za 19 kontejnera(90%)34.varijanta
//int[][] pozicijeKontejnera = {{0, 0, 16, 1, 2, 19, 6}, {11, 13, 10, 15, 5, 9, 8},
{17, 4, 12, 3, 7, 14, 18}};// zapis po redovima
3*7 za 19 kontejnera(90%)35.varijanta
//int[][] pozicijeKontejnera = {{1, 10, 15, 14, 16, 3, 0}, {8, 19, 4, 11, 12, 7,
0}, {6, 2, 5, 9, 17, 13, 18}};// zapis po redovima
3*7 za 19 kontejnera(90%)36.varijanta
//int[][] pozicijeKontejnera = {{0, 15, 13, 2, 8, 11, 4}, {0, 17, 14, 3, 18, 7,
10}, {5, 19, 16, 12, 9, 1, 6}};// zapis po redovima
3*7 za 19 kontejnera(90%)37.varijanta
//int[][] pozicijeKontejnera = {{4, 15, 5, 19, 0, 13, 8}, {2, 9, 14, 6, 0, 1, 11},
{10, 18, 3, 7, 12, 17, 16}};// zapis po redovima
3*7 za 19 kontejnera(90%)38.varijanta
//int[][] pozicijeKontejnera = {{0, 12, 6, 9, 7, 4, 14}, {0, 13, 11, 18, 8, 2, 3},
{19, 10, 5, 15, 16, 1, 17}};// zapis po redovima
3*7 za 19 kontejnera(90%)39.varijanta
//int[][] pozicijeKontejnera = {{0, 0, 0, 12, 9, 0, 15}, {11, 0, 0, 7, 8, 0, 13},
{14, 0, 0, 3, 6, 0, 5},{1, 0, 0, 10, 4, 0, 2}};// zapis
po redovima 4*7 za 15 kontejnera(55%)
//int[][] pozicijeKontejnera = {{12, 0, 14, 20, 0, 7, 0}, {15, 0, 19, 2, 0, 18,
13}, {10, 4, 16, 8, 0, 3, 21},{1, 11, 6, 17, 0, 9,
5}};// zapis po redovima 4*7 za 21 kontejner(75%)
//int[][] pozicijeKontejnera = {{12, 0, 0, 20, 25, 7, 0}, {15, 23, 19, 2, 24, 18,
13}, {10, 4, 16, 8, 14, 3, 21},{1, 11, 6, 17, 22,
9, 5}};// zapis po redovima 4*7 za 25 kontejnera(90%)
// za testiranje razliÄŠitih poÄŠetnih rasporeda za odjeljka 4x7(generirano u R-
u)//
//int[][] pozicijeKontejnera = {{17, 0, 0, 20, 9, 16, 1}, {3, 0, 4, 12, 2, 19, 25},
{21, 10, 18, 13, 6, 15, 24},{11, 14, 7, 5, 23,
22, 8}};// zapis po redovima 4*7 za 25 kontejnera(90%)1.varijanta
//int[][] pozicijeKontejnera = {{1, 0, 16, 0, 4, 7, 18}, {10, 6, 9, 0, 13, 2, 22},
{19, 12, 15, 25, 17, 14, 11},{23, 24, 20, 21, 8,
3, 5}};// zapis po redovima 4*7 za 25 kontejnera(90%)2.varijanta
//int[][] pozicijeKontejnera = {{15, 19, 24, 0, 1, 0, 0}, {25, 4, 3, 18, 10, 20,
22},{14, 12, 16, 5, 9, 21, 7},{6, 11, 2, 23, 8, 17,
13}};// zapis po redovima 4*7 za 25 kontejnera(90%)3.varijanta
//int[][] pozicijeKontejnera = {{0, 0, 24, 25, 5, 21, 0}, {14, 6, 18, 10, 13, 1,
23}, {16, 3, 4, 9, 8, 15, 20},{2, 22, 12, 17, 19,
```

```
11, 7}};// zapis po redovima 4*7 za 25 kontejnera(90%)4.varijanta
//int[][] pozicijeKontejnera = {{0, 24, 6, 8, 0, 3, 9}, {0, 15, 14, 10, 18, 5, 11},
{25, 12, 16, 17, 4, 20, 2},{13, 19, 22, 23, 21,
7, 1}};// zapis po redovima 4*7 za 25 kontejnera(90%)5.varijanta
//int[][] pozicijeKontejnera = {{15, 6, 9, 24, 0, 8, 0}, {4, 14, 23, 3, 0, 25, 13},
{7, 18, 17, 19, 16, 12, 2},{10, 22, 1, 20, 21,
11, 5}};// zapis po redovima 4*7 za 25 kontejnera(90%)6.varijanta
//int[][] pozicijeKontejnera = {{6, 0, 20, 17, 0, 22, 21}, {16, 10, 9, 5, 0, 18,
11}, {8, 4, 1, 24, 23, 14, 25},{15, 13, 2, 7, 3, 12,
19}};// zapis po redovima 4*7 za 25 kontejnera(90%)7.varijanta
//int[][] pozicijeKontejnera = {{0, 7, 11, 0, 21, 8, 6}, {0, 5, 3, 22, 19, 1, 4},
{20, 24, 15, 23, 14, 16, 13},{17, 10, 9, 12, 2, 18,
25}};// zapis po redovima 4*7 za 25 kontejnera(90%)8.varijanta
//int[][] pozicijeKontejnera = {{13, 2, 9, 20, 6, 0, 0}, {15, 1, 5, 17, 8, 0, 21},
{10, 11, 18, 25, 19, 12, 4},{22, 3, 16, 7, 24, 14,
23}};// zapis po redovima 4*7 za 25 kontejnera(90%)9.varijanta
//int[][] pozicijeKontejnera = {{19, 2, 1, 0, 9, 21, 24}, {15, 12, 13, 0, 14, 16,
23}, {10, 20, 5, 0, 4, 25, 22},{18, 6, 8, 3, 7, 11,
17}};// zapis po redovima 4*7 za 25 kontejnera(90%)10.varijanta
//int[][] pozicijeKontejnera = {{16, 0, 19, 22, 1, 0, 2}, {9, 0, 23, 11, 12, 14,
5}, {3, 13, 24, 25, 7, 15, 8},{4, 21, 20, 10, 18,
17, 6}};// zapis po redovima 4*7 za 25 kontejnera(90%)11.varijanta
//int[][] pozicijeKontejnera = {{10, 1, 12, 0, 0, 14, 21}, {11, 25, 17, 23, 0, 2,
3}, {19, 18, 22, 7, 15, 16, 8},{24, 9, 4, 6, 13, 5,
20}};// zapis po redovima 4*7 za 25 kontejnera(90%)12.varijanta
//int[][] pozicijeKontejnera = {{19, 8, 3, 25, 0, 0, 20}, {23, 7, 1, 24, 14, 0, 4},
{17, 9, 18, 6, 11, 15, 10},{12, 13, 16, 2, 21, 5,
22}};// zapis po redovima 4*7 za 25 kontejnera(90%)13.varijanta
//int[][] pozicijeKontejnera = {{18, 0, 4, 3, 0, 13, 15}, {16, 17, 24, 19, 0, 12,
14}, {7, 25, 10, 8, 23, 9, 20},{2, 21, 22, 11, 5,
1, 6}};// zapis po redovima 4*7 za 25 kontejnera(90%)14.varijanta
//int[][] pozicijeKontejnera = {{8, 14, 0, 11, 0, 18, 0}, {21, 5, 13, 22, 25, 17,
3}, {12, 1, 10, 20, 19, 2, 16},{4, 6, 23, 15, 7, 9,
24}};// zapis po redovima 4*7 za 25 kontejnera(90%)15.varijanta
//int[][] pozicijeKontejnera = {{5, 0, 11, 13, 0, 12, 0}, {7, 22, 4, 14, 1, 19,
17}, {6, 16, 25, 10, 15, 9, 23},{21, 3, 18, 2, 20,
24, 8}};// zapis po redovima 4*7 za 25 kontejnera(90%)16.varijanta
//int[][] pozicijeKontejnera = {{13, 0, 14, 23, 19, 0, 7}, {2, 0, 6, 15, 1, 3, 25},
{24, 5, 22, 10, 21, 20, 18},{12, 4, 16, 11, 9, 8,
17}};// zapis po redovima 4*7 za 25 kontejnera(90%)17.varijanta
//int[][] pozicijeKontejnera = {{0, 0, 0, 16, 4, 15, 14}, {8, 2, 6, 7, 25, 19, 17},
{11, 5, 3, 1, 13, 21, 24},{10, 18, 12, 23, 9, 20,
22}};// zapis po redovima 4*7 za 25 kontejnera(90%)18.varijanta
//int[][] pozicijeKontejnera = {{15, 5, 6, 0, 11, 0, 0}, {14, 18, 7, 12, 3, 2, 20},
{24, 10, 4, 1, 9, 17, 21},{23, 8, 25, 13, 16, 22,
19}};// zapis po redovima 4*7 za 25 kontejnera(90%)19.varijanta
//int[][] pozicijeKontejnera = {{10, 13, 21, 24, 0, 4, 0}, {9, 12, 18, 23, 1, 6,
0}, {5, 2, 7, 19, 16, 3, 25},{22, 14, 15, 11, 17,
20, 8}};// zapis po redovima 4*7 za 25 kontejnera(90%)20.varijanta
//int[][] pozicijeKontejnera = {{0, 11, 13, 12, 0, 20, 24}, {0, 3, 21, 1, 25, 2,
6}, {10, 4, 14, 16, 19, 17, 15},{22, 18, 5, 9, 8, 7,
23}};// zapis po redovima 4*7 za 25 kontejnera(90%)21.varijanta
//int[][] pozicijeKontejnera = {{0, 0, 6, 0, 2, 25, 21}, {7, 5, 20, 19, 11, 1, 18},
{23, 24, 4, 16, 8, 14, 9},{10, 13, 3, 12, 22, 17,
15}};// zapis po redovima 4*7 za 25 kontejnera(90%)22.varijanta
//int[][] pozicijeKontejnera = {{0, 0, 10, 17, 9, 25, 5}, {1, 0, 14, 4, 19, 18, 6},
{15, 12, 23, 11, 21, 3, 2},{8, 7, 22, 16, 13, 20,
24}};// zapis po redovima 4*7 za 25 kontejnera(90%)23.varijanta
//int[][] pozicijeKontejnera = {{21, 6, 23, 0, 7, 10, 0}, {9, 13, 16, 0, 20, 22,
1}, {19, 2, 8, 18, 15, 5, 12},{14, 4, 11, 17, 24, 3,
25}};// zapis po redovima 4*7 za 25 kontejnera(90%)24.varijanta
//int[][] pozicijeKontejnera = {{25, 0, 1, 0, 20, 14, 22}, {8, 0, 13, 9, 16, 6, 4},
{23, 21, 5, 11, 19, 18, 2},{17, 3, 10, 24, 7, 15,
12}};// zapis po redovima 4*7 za 25 kontejnera(90%)25.varijanta
//int[][] pozicijeKontejnera = {{0, 0, 8, 16, 13, 24, 10}, {0, 4, 3, 5, 7, 11, 12},
{23, 19, 22, 25, 17, 15, 2},{14, 1, 18, 20, 6,
21, 9}};// zapis po redovima 4*7 za 25 kontejnera(90%)26.varijanta
```

```
//int[][] pozicijeKontejnera = {{11, 0, 13, 22, 1, 0, 0}, {16, 25, 6, 21, 2, 19,
3}, {10, 5, 7, 24, 4, 23, 9},{8, 20, 12, 14, 15, 18,
17}};// zapis po redovima 4*7 za 25 kontejnera(90%)27.varijanta
//int[][] pozicijeKontejnera = {{1, 20, 0, 0, 19, 23, 5}, {17, 24, 16, 0, 6, 12,
11}, {3, 21, 7, 8, 22, 2, 10},{13, 25, 15, 18, 9, 4,
14}};// zapis po redovima 4*7 za 25 kontejnera(90%)28.varijanta
//int[][] pozicijeKontejnera = {{25, 0, 16, 9, 3, 0, 22}, {12, 23, 21, 17, 4, 0,
18}, {7, 6, 20, 13, 2, 24, 11},{15, 10, 1, 8, 5, 19,
14}};// zapis po redovima 4*7 za 25 kontejnera(90%)29.varijanta
//int[][] pozicijeKontejnera = {{17, 20, 15, 9, 0, 0, 0}, {11, 5, 14, 12, 4, 19,
25}, {23, 8, 7, 6, 22, 10, 3},{16, 24, 2, 21, 18, 1,
13}};// zapis po redovima 4*7 za 25 kontejnera(90%)30.varijanta
//int[][] pozicijeKontejnera = {{25, 12, 2, 24, 0, 21, 0}, {3, 10, 9, 7, 18, 14,
0}, {5, 6, 4, 8, 13, 20, 23},{11, 15, 19, 16, 17, 1,
22}};// zapis po redovima 4*7 za 25 kontejnera(90%)31.varijanta
//int[][] pozicijeKontejnera = {{7, 5, 22, 21, 23, 0, 0}, {4, 24, 16, 11, 6, 0,
18}, {2, 20, 8, 1, 15, 13, 17},{14, 12, 3, 10, 25,
19, 9}};// zapis po redovima 4*7 za 25 kontejnera(90%)32.varijanta
//int[][] pozicijeKontejnera = {{23, 0, 5, 0, 0, 15, 20}, {14, 3, 6, 24, 17, 10,
16}, {2, 18, 8, 4, 21, 19, 12},{25, 1, 11, 13, 7, 9,
22}};// zapis po redovima 4*7 za 25 kontejnera(90%)33.varijanta
//int[][] pozicijeKontejnera = {{1, 0, 25, 8, 12, 0, 18}, {20, 0, 15, 21, 22, 16,
2}, {7, 5, 13, 6, 9, 14, 10},{17, 23, 24, 3, 19,
11, 4}};// zapis po redovima 4*7 za 25 kontejnera(90%)34.varijanta
//int[][] pozicijeKontejnera = {{14, 9, 5, 7, 0, 0, 22}, {21, 20, 10, 12, 0, 15,
13}, {18, 25, 3, 24, 6, 17, 11},{1, 8, 2, 19, 23,
16, 4}};// zapis po redovima 4*7 za 25 kontejnera(90%)35.varijanta
//int[][] pozicijeKontejnera = {{17, 4, 15, 21, 0, 0, 0}, {8, 3, 9, 19, 14, 12,
11}, {22, 13, 16, 18, 1, 20, 5},{24, 2, 10, 7, 25,
23, 6}};// zapis po redovima 4*7 za 25 kontejnera(90%)36.varijanta
//int[][] pozicijeKontejnera = {{20, 0, 0, 14, 25, 24, 10}, {5, 0, 13, 6, 22, 4,
12}, {21, 2, 16, 8, 17, 18, 23},{15, 1, 11, 7, 19,
9, 3}};// zapis po redovima 4*7 za 25 kontejnera(90%)37.varijanta
//int[][] pozicijeKontejnera = {{0, 7, 0, 0, 18, 14, 3}, {4, 21, 23, 1, 11, 17,
25}, {22, 24, 6, 2, 15, 12, 16},{9, 8, 19, 20, 10, 5,
13}};// zapis po redovima 4*7 za 25 kontejnera(90%)38.varijanta
//int[][] pozicijeKontejnera = {{0, 24, 13, 23, 0, 25, 0}, {9, 16, 8, 21, 12, 17,
18}, {1, 11, 14, 15, 5, 6, 10},{22, 7, 19, 3, 4, 2,
20}};// zapis po redovima 4*7 za 25 kontejnera(90%)39.varijanta
//int[][] pozicijeKontejnera = {{0, 19, 0, 0, 0, 0, 0}, {0, 13, 16, 0, 15, 0, 0},
{7, 10, 12, 0, 14, 18, 0},{5, 9, 4, 0, 8, 17, 0},{3,
1, 2, 0, 6, 11, 0}};// zapis po redovima 5*7 za 19 kontejnera(55%)
//int[][] pozicijeKontejnera = {{19, 0, 16, 0, 10, 0, 22}, {21, 0, 13, 0, 5, 0,
14}, {17, 23, 11, 0, 1, 0, 7},{4, 15, 2, 18, 20, 0,
25},{6, 12, 8, 26, 3, 24, 9}};// zapis po redovima 5*7 za 26 kontejnera(75%)
//int[][] pozicijeKontejnera = {{19, 0, 16, 0, 10, 0, 22}, {21, 30, 13, 31, 5, 0,
14}, {17, 23, 11, 29, 1, 28, 7},{4, 15, 2, 18, 20,
27, 25},{6, 12, 8, 26, 3, 24, 9}};// zapis po redovima 5*7 za 31 kontejnera(90%)
//za testiranje razlicitih pocetnih rasporeda za odjeljak velicine 5x7//
//int[][] pozicijeKontejnera = {{21, 30, 0, 0, 20, 0, 24}, {28, 19, 31, 3, 14, 0,
25}, {17, 1, 11, 27, 8, 16, 6},{18, 9, 12, 7, 13,
2, 23},{10, 29, 4, 22, 26, 15, 5}};// zapis po redovima 5*7 za 31
kontejnera(90%)1.varijanta
//int[][] pozicijeKontejnera = {{0, 0, 7, 0, 15, 0, 5}, {10, 12, 8, 31, 2, 22, 21},
{27, 11, 19, 9, 29, 1, 13},{28, 25, 3, 18, 17,
23, 14},{4, 26, 20, 16, 24, 6, 30}};// zapis po redovima 5*7 za 31
kontejnera(90%)2.varijanta
//int[][] pozicijeKontejnera = {{0, 17, 31, 0, 0, 0, 6}, {2, 19, 28, 12, 5, 21,
22}, {1, 25, 3, 11, 13, 8, 10},{16, 15, 4, 20, 7, 24,
23},{18, 26, 30, 14, 29, 27, 9}};// zapis po redovima 5*7 za 31
kontejnera(90%)3.varijanta
//int[][] pozicijeKontejnera = {{0, 0, 8, 3, 31, 0, 28}, {0, 18, 5, 30, 15, 7, 29},
{11, 4, 6, 2, 23, 13, 21},{26, 16, 17, 24, 12,
27, 25},{10, 19, 9, 1, 14, 20, 22}};// zapis po redovima 5*7 za 31
kontejnera(90%)4.varijanta
//int[][] pozicijeKontejnera = {{0, 0, 2, 0, 29, 30, 31}, {0, 22, 1, 15, 11, 7,
13}, {19, 9, 3, 4, 12, 21, 24},{17, 26, 28, 16, 25,
```

```
5, 6},{27, 10, 18, 20, 8, 14, 23}};// zapis po redovima 5*7 za 31
kontejnera(90%)5.varijanta
//int[][] pozicijeKontejnera = {{0, 0, 1, 0, 13, 29, 0}, {10, 24, 31, 30, 14, 20,
26}, {6, 7, 8, 12, 4, 19, 5},{9, 17, 27, 2, 22, 15,
28},{23, 3, 11, 18, 25, 16, 21}};// zapis po redovima 5*7 za 31
kontejnera(90%)6.varijanta
//int[][] pozicijeKontejnera = {{0, 0, 25, 26, 0, 13, 0}, {8, 6, 22, 12, 2, 14,
11}, {7, 24, 21, 1, 3, 4, 19},{28, 5, 29, 27, 23, 10,
15},{9, 31, 30, 17, 18, 16, 20}};// zapis po redovima 5*7 za 31
kontejnera(90%)7.varijanta
//int[][] pozicijeKontejnera = {{0, 0, 24, 0, 29, 30, 0}, {23, 25, 4, 28, 3, 13,
15}, {31, 12, 20, 6, 8, 5, 14},{22, 27, 18, 7, 2,
17, 16},{9, 21, 1, 11, 10, 19, 26}};// zapis po redovima 5*7 za 31
kontejnera(90%)8.varijanta
//int[][] pozicijeKontejnera = {{0, 0, 0, 1, 2, 22, 0}, {29, 30, 23, 19, 9, 16,
18}, {28, 12, 25, 3, 8, 17, 7},{11, 10, 24, 4, 20, 5,
31},{27, 13, 26, 21, 6, 14, 15}};// zapis po redovima 5*7 za 31
kontejnera(90%)9.varijanta
//int[][] pozicijeKontejnera = {{1, 0, 0, 0, 7, 29, 0}, {18, 24, 6, 14, 27, 13,
12}, {23, 2, 15, 17, 28, 5, 10},{19, 21, 16, 3, 4,
30, 11},{20, 22, 25, 8, 26, 9, 31}};// zapis po redovima 5*7 za 31
kontejnera(90%)10.varijanta
//int[][] pozicijeKontejnera = {{0, 0, 0, 31, 29, 30, 15}, {0, 26, 1, 22, 16, 3,
4}, {23, 17, 18, 6, 2, 10, 11},{25, 24, 19, 7, 9, 5,
12},{28, 27, 20, 8, 21, 14, 13}};// zapis po redovima 5*7 za 31
kontejnera(90%)11.varijanta
//int[][] pozicijeKontejnera = {{0, 0, 0, 14, 0, 3, 28}, {30, 22, 5, 6, 10, 2, 31},
{18, 17, 12, 7, 11, 1, 16},{29, 19, 13, 8, 25, 4,
26},{23, 21, 20, 9, 15, 24, 27}};// zapis po redovima 5*7 za 31
kontejnera(90%)12.varijanta
//int[][] pozicijeKontejnera = {{0, 0, 0, 1, 0, 3, 7}, {17, 15, 28, 10, 11, 4, 21},
{22, 16, 5, 12, 2, 14, 8},{23, 24, 26, 13, 18,
29, 19},{31, 25, 27, 9, 30, 6, 20}};// zapis po redovima 5*7 za 31
kontejnera(90%)13.varijanta
//int[][] pozicijeKontejnera = {{0, 30, 0, 0, 1, 29, 0}, {23, 7, 6, 18, 2, 10, 4},
{24, 21, 19, 8, 5, 14, 13},{27, 26, 20, 16, 9, 3,
12},{25, 31, 28, 22, 17, 15, 11}};// zapis po redovima 5*7 za 31
kontejnera(90%)14.varijanta
//int[][] pozicijeKontejnera = {{31, 0, 3, 28, 29, 0, 0}, {16, 1, 2, 6, 12, 7, 0},
{25, 21, 4, 15, 10, 8, 11},{27, 23, 17, 5, 13, 9,
14},{30, 24, 22, 18, 19, 20, 26}};// zapis po redovima 5*7 za 31
kontejnera(90%)15.varijanta
//int[][] pozicijeKontejnera = {{0, 20, 21, 0, 0, 1, 0}, {31, 26, 29, 4, 18, 5, 2},
{14, 28, 22, 25, 9, 12, 7},{16, 27, 15, 3, 10, 6,
11},{13, 30, 17, 8, 19, 23, 24}};// zapis po redovima 5*7 za 31
kontejnera(90%)16.varijanta
//int[][] pozicijeKontejnera = {{0, 0, 0, 7, 6, 0, 3}, {16, 2, 4, 5, 12, 30, 29},
{9, 17, 20, 24, 21, 31, 28},{15, 10, 22, 18, 23,
26, 25},{8, 1, 14, 13, 19, 27, 11}};// zapis po redovima 5*7 za 31
kontejnera(90%)17.varijanta
//int[][] pozicijeKontejnera = {{0, 28, 24, 27, 15, 21, 22}, {0, 26, 7, 1, 2, 4,
9}, {0, 13, 14, 16, 3, 12, 6},{0, 25, 18, 8, 23, 10,
11},{19, 29, 31, 17, 20, 5, 30}};// zapis po redovima 5*7 za 31
kontejnera(90%)18.varijanta
//int[][] pozicijeKontejnera = {{0, 0, 0, 0, 30, 3, 31}, {26, 21, 8, 9, 4, 19, 15},
{7, 22, 20, 16, 25, 1, 14},{29, 5, 23, 10, 11,
12, 6},{27, 2, 24, 28, 17, 13, 18}};// zapis po redovima 5*7 za 31
kontejnera(90%)19.varijanta
//int[][] pozicijeKontejnera = {{1, 0, 0, 29, 30, 0, 0}, {27, 16, 23, 19, 22, 21,
5}, {8, 26, 7, 6, 11, 4, 10},{28, 2, 15, 14, 3, 25,
31},{18, 17, 24, 20, 13, 9, 12}};// zapis po redovima 5*7 za 31
kontejnera(90%)20.varijanta
//int[][] pozicijeKontejnera = {{0, 0, 0, 28, 0, 30, 31}, {12, 11, 10, 9, 2, 3,
18}, {13, 14, 4, 1, 15, 17, 8},{25, 23, 5, 16, 6, 21,
22},{29, 27, 24, 7, 20, 19, 26}};// zapis po redovima 5*7 za 31
kontejnera(90%)21.varijanta
//int[][] pozicijeKontejnera = {{0, 9, 7, 5, 0, 0, 20}, {0, 1, 2, 3, 10, 19, 13},
{12, 28, 4, 18, 23, 16, 21},{27, 25, 30, 6, 14, 11,
```

```
15},{29, 26, 8, 31, 24, 17, 22}};// zapis po redovima 5*7 za 31
kontejnera(90%)22.varijanta
//int[][] pozicijeKontejnera = {{0, 0, 5, 7, 12, 10, 9}, {0, 13, 14, 29, 2, 3, 28},
{0, 24, 31, 6, 27, 17, 11},{22, 21, 4, 15, 1, 8,
19},{23, 30, 25, 26, 16, 20, 18}};// zapis po redovima 5*7 za 31
kontejnera(90%)23.varijanta
//int[][] pozicijeKontejnera = {{1, 28, 25, 26, 17, 31, 0}, {22, 2, 6, 10, 5, 27,
0}, {21, 13, 3, 11, 14, 8, 0},{18, 7, 19, 4, 9, 15,
0},{20, 24, 12, 23, 16, 29, 30}};// zapis po redovima 5*7 za 31
kontejnera(90%)24.varijanta
//int[][] pozicijeKontejnera = {{8, 7, 20, 0, 0, 0, 0}, {26, 31, 21, 1, 2, 27, 28},
{9, 25, 14, 30, 4, 24, 5},{12, 11, 15, 3, 16, 18,
19},{10, 6, 22, 13, 17, 23, 29}};// zapis po redovima 5*7 za 31
kontejnera(90%)25.varijanta
//int[][] pozicijeKontejnera = {{31, 30, 28, 9, 0, 0, 29}, {25, 15, 23, 24, 0, 21,
4}, {6, 26, 5, 8, 0, 1, 22},{18, 7, 14, 13, 27, 2,
19},{17, 16, 11, 10, 12, 3, 20}};// zapis po redovima 5*7 za 31
kontejnera(90%)26.varijanta
//int[][] pozicijeKontejnera = {{0, 0, 1, 23, 3, 8, 12}, {0, 27, 22, 4, 24, 13,
11}, {0, 26, 2, 21, 10, 7, 14},{6, 5, 25, 15, 16, 18,
20},{31, 28, 29, 30, 17, 9, 19}};// zapis po redovima 5*7 za 31
kontejnera(90%)27.varijanta
//int[][] pozicijeKontejnera = {{0, 0, 30, 17, 6, 3, 4}, {0, 19, 14, 9, 1, 2, 12},
{0, 22, 13, 8, 5, 10, 11},{20, 23, 15, 16, 7, 18,
25},{21, 27, 24, 26, 28, 29, 31}};// zapis po redovima 5*7 za 31
kontejnera(90%)28.varijanta
//int[][] pozicijeKontejnera = {{0, 0, 28, 0, 27, 2, 0}, {16, 14, 24, 3, 8, 29,
13}, {23, 22, 4, 7, 1, 11, 12},{17, 5, 19, 6, 18, 30,
10},{15, 26, 21, 25, 20, 9, 31}};// zapis po redovima 5*7 za 31
kontejnera(90%)29.varijanta
//int[][] pozicijeKontejnera = {{30, 31, 27, 0, 0, 0, 0}, {29, 28, 5, 20, 19, 8,
11}, {23, 22, 1, 6, 2, 9, 12},{24, 4, 18, 15, 14, 3,
13},{26, 25, 21, 17, 7, 10, 16}};// zapis po redovima 5*7 za 31
kontejnera(90%)30.varijanta
//int[][] pozicijeKontejnera = {{0, 31, 22, 27, 28, 30, 0}, {0, 1, 17, 3, 5, 8, 9},
{0, 2, 15, 20, 23, 12, 10},{25, 14, 21, 16, 19,
26, 29},{18, 13, 7, 11, 4, 24, 6}};// zapis po redovima 5*7 za 31
kontejnera(90%)31.varijanta
//int[][] pozicijeKontejnera = {{0, 28, 15, 3, 13, 22, 0}, {16, 19, 21, 4, 2, 5,
0}, {17, 7, 6, 1, 12, 20, 0},{18, 8, 9, 10, 11, 27,
26},{30, 29, 25, 24, 14, 23, 31}};// zapis po redovima 5*7 za 31
kontejnera(90%)32.varijanta
//int[][] pozicijeKontejnera = {{11, 27, 7, 0, 0, 31, 29}, {12, 2, 25, 0, 30, 15,
23}, {6, 5, 3, 0, 26, 1, 16},{24, 9, 4, 28, 21, 14,
20},{13, 10, 8, 22, 18, 17, 19}};// zapis po redovima 5*7 za 31
kontejnera(90%)33.varijanta
//int[][] pozicijeKontejnera = {{0, 0, 0, 29, 30, 31, 20}, {0, 8, 28, 6, 7, 12,
11}, {26, 15, 23, 17, 19, 10, 21},{27, 14, 5, 18,
16, 2, 3},{25, 24, 13, 9, 1, 4, 22}};// zapis po redovima 5*7 za 31
kontejnera(90%)34.varijanta
//int[][] pozicijeKontejnera = {{0, 0, 0, 29, 6, 9, 0}, {27, 1, 2, 20, 8, 7, 13},
{28, 26, 19, 3, 4, 10, 24},{21, 18, 5, 16, 14, 11,
30},{25, 22, 17, 15, 23, 12, 31}};// zapis po redovima 5*7 za 31
kontejnera(90%)35.varijanta
//int[][] pozicijeKontejnera = {{0, 0, 4, 0, 29, 0, 31}, {28, 18, 5, 17, 2, 11,
22}, {7, 8, 15, 3, 10, 1, 30},{25, 24, 16, 9, 13, 12,
6},{27, 19, 26, 20, 21, 14, 23}};// zapis po redovima 5*7 za 31
kontejnera(90%)36.varijanta
//int[][] pozicijeKontejnera = {{0, 0, 31, 10, 1, 20, 22}, {0, 0, 30, 24, 2, 16,
15}, {6, 7, 28, 21, 19, 3, 14},{29, 27, 8, 9, 11,
17, 13},{4, 5, 25, 26, 18, 12, 23}};// zapis po redovima 5*7 za 31
kontejnera(90%)37.varijanta
//int[][] pozicijeKontejnera = {{0, 0, 0, 31, 17, 16, 14}, {28, 30, 0, 22, 2, 10,
13}, {29, 8, 4, 7, 6, 1, 11},{24, 25, 23, 5, 15, 3,
12},{26, 9, 27, 21, 20, 18, 19}};// zapis po redovima 5*7 za 31
kontejnera(90%)38.varijanta
//int[][] pozicijeKontejnera = {{0, 0, 1, 22, 0, 0, 9}, {21, 7, 3, 2, 4, 5, 29},
{8, 19, 24, 30, 10, 11, 31},{20, 17, 16, 27, 12, 13,
```

```
26},{28, 18, 23, 6, 14, 15, 25}};// zapis po redovima 5*7 za 31
kontejnera(90%)39.varijanta
//int[][] pozicijeKontejnera = {{0, 0, 21, 0, 22, 0, 0}, {0, 0, 19, 0, 12, 0, 0},
{0, 23, 15, 0, 10, 16, 0},{0, 8, 11, 0, 7, 14, 20},
{0, 5, 6, 0, 4, 13, 18},{0, 2, 3, 0, 1, 9, 17}};// zapis po redovima 6*7 za 23
kontejnera(55%)
//int[][] pozicijeKontejnera = {{0, 29, 27, 0, 31, 0, 10}, {0, 16, 22, 28, 26, 0,
7}, {0, 9, 8, 23, 18, 0, 30},{0, 24, 5, 17, 11, 0,
21},{0, 2, 19, 13, 4, 25, 1},{0, 14, 15, 6, 12, 20, 3}};// zapis po redovima 6*7 za
31 kontejnera(75%)
//int[][] pozicijeKontejnera = {{0, 29, 27, 0, 31, 0, 10}, {0, 16, 22, 28, 26, 37,
7}, {0, 9, 8, 23, 18, 35, 30}, {36, 24, 5, 17, 11,
33, 21}, {34, 2, 19, 13, 4, 25, 1}, {32, 14, 15, 6, 12, 20, 3}};// zapis po
redovima 6*7 za 38 kontejnera(90%)
//za testiranje razlicitih pocetnih rasporeda za odjeljak velicine 6x7//
//int[][] pozicijeKontejnera = {{0, 0, 16, 0, 37, 31, 12}, {24, 13, 3, 0, 30, 19,
15},{20, 36, 29, 0, 21, 34, 11},{6, 17, 9, 7, 5,
23, 1},{8, 25, 14, 32, 27, 4, 2},{22, 10, 18, 28, 26, 33, 35}};// zapis po redovima
6*7 za 37 kontejnera(90%)1.varijanta
//int[][] pozicijeKontejnera = {{3, 16, 0, 0, 0, 19, 24}, {12, 6, 0, 23, 0, 33,
26}, {7, 14, 20, 4, 30, 34, 37},{27, 32, 11, 17, 28,
25, 22},{5, 29, 2, 9, 1, 21, 35},{15, 36, 31, 8, 13, 18, 10}};// zapis po redovima
6*7 za 37 kontejnera(90%)2.varijanta
//int[][] pozicijeKontejnera = {{0, 0, 0, 24, 1, 3, 10}, {0, 0, 25, 36, 13, 29,
17}, {11, 31, 15, 19, 23, 9, 30},{22, 32, 16, 21, 6,
2, 28},{33, 4, 20, 35, 26, 18, 14},{5, 37, 12, 34, 27, 8, 7}};// zapis po redovima
6*7 za 37 kontejnera(90%)3.varijanta
//int[][] pozicijeKontejnera = {{0, 26, 36, 24, 0, 0, 25}, {22, 18, 27, 20, 0, 0,
15}, {8, 14, 37, 32, 6, 11, 35},{28, 33, 13, 29,
12, 5, 1},{31, 23, 9, 2, 16, 7, 19},{17, 10, 3, 34, 21, 30, 4}};// zapis po
redovima 6*7 za 37 kontejnera(90%)4.varijanta
//int[][] pozicijeKontejnera = {{6, 0, 8, 0, 16, 30, 0}, {20, 0, 1, 4, 12, 21, 0},
{9, 28, 22, 27, 25, 5, 7},{19, 10, 37, 32, 29, 31,
26},{36, 18, 13, 24, 17, 14, 23},{33, 34, 35, 2, 3, 15, 11}};// zapis po redovima
6*7 za 37 kontejnera(90%)5.varijanta
//int[][] pozicijeKontejnera = {{24, 0, 5, 0, 6, 0, 0}, {21, 7, 23, 19, 8, 12, 0},
{35, 9, 14, 25, 28, 10, 3},{2, 20, 15, 31, 17, 36,
26},{37, 18, 30, 13, 11, 33, 1},{29, 27, 32, 34, 22, 16, 4}};// zapis po redovima
6*7 za 37 kontejnera(90%)6.varijanta
//int[][] pozicijeKontejnera = {{16, 0, 0, 33, 3, 8, 0}, {35, 0, 7, 23, 2, 31, 0},
{37, 13, 34, 9, 4, 24, 12},{22, 27, 10, 18, 19,
17, 14},{25, 28, 6, 5, 29, 15, 21},{20, 30, 1, 32, 11, 36, 26}};// zapis po
redovima 6*7 za 37 kontejnera(90%)7.varijanta
//int[][] pozicijeKontejnera = {{24, 2, 19, 0, 14, 0, 34}, {8, 25, 4, 0, 27, 3,
29}, {28, 30, 26, 0, 21, 33, 5},{23, 20, 16, 0, 13,
35, 1},{22, 31, 18, 32, 6, 9, 17},{15, 37, 7, 12, 10, 36, 11}};// zapis po redovima
6*7 za 37 kontejnera(90%)8.varijanta
//int[][] pozicijeKontejnera = {{0, 0, 18, 16, 8, 19, 0}, {0, 3, 25, 37, 26, 24,
29}, {0, 32, 4, 30, 6, 9, 14},{7, 34, 10, 1, 13, 5,
28},{2, 27, 15, 22, 17, 20, 36},{31, 21, 33, 23, 12, 11, 35}};// zapis po redovima
6*7 za 37 kontejnera(90%)9.varijanta
//int[][] pozicijeKontejnera = {{0, 6, 23, 7, 30, 1, 35}, {0, 12, 10, 16, 15, 31,
21}, {0, 20, 34, 8, 4, 33, 26},{0, 9, 17, 32, 2,
27, 3},{0, 24, 11, 25, 19, 14, 22},{18, 37, 29, 28, 5, 13, 36}};// zapis po
redovima 6*7 za 37 kontejnera(90%)10.varijanta
//int[][] pozicijeKontejnera = {{0, 34, 1, 0, 10, 18, 0}, {8, 7, 24, 0, 6, 5, 26},
{14, 9, 27, 0, 19, 36, 32},{30, 37, 13, 4, 22, 33,
17},{3, 11, 31, 20, 35, 28, 21},{29, 23, 25, 15, 16, 12, 2}};// zapis po redovima
6*7 za 37 kontejnera(90%)11.varijanta
//int[][] pozicijeKontejnera = {{0, 0, 20, 0, 11, 22, 35}, {31, 0, 19, 0, 25, 7,
2}, {34, 26, 33, 6, 1, 14, 30},{17, 12, 24, 37, 29,
15, 18},{3, 5, 10, 16, 27, 13, 36},{8, 23, 28, 21, 9, 4, 32}};// zapis po redovima
6*7 za 37 kontejnera(90%)12.varijanta
//int[][] pozicijeKontejnera = {{0, 0, 0, 0, 25, 11, 16}, {0, 15, 26, 5, 34, 36,
21}, {27, 12, 18, 28, 30, 23, 29},{14, 8, 7, 19,
22, 3, 17},{9, 32, 24, 1, 35, 4, 33},{13, 20, 10, 31, 2, 37, 6}};// zapis po
redovima 6*7 za 37 kontejnera(90%)13.varijanta
```

```
//int[][] pozicijeKontejnera = {{0, 0, 10, 21, 32, 0, 0}, {0, 33, 4, 6, 11, 24, 1},
{14, 8, 36, 18, 27, 30, 22},{19, 3, 34, 29, 35,
23, 9},{5, 17, 26, 31, 28, 16, 7},{2, 13, 25, 12, 20, 37, 15}};// zapis po redovima
6*7 za 37 kontejnera(90%)14.varijanta
//int[][] pozicijeKontejnera = {{16, 14, 6, 10, 26, 0, 0}, {20, 37, 33, 19, 24, 0,
0}, {11, 25, 3, 27, 1, 17, 0},{9, 35, 23, 28, 2,
12, 18},{4, 22, 21, 34, 7, 15, 31},{8, 29, 32, 5, 36, 30, 13}};// zapis po redovima
6*7 za 37 kontejnera(90%)15.varijanta
//int[][] pozicijeKontejnera = {{31, 4, 24, 0, 16, 0, 32}, {26, 27, 30, 0, 29, 20,
1}, {8, 17, 3, 0, 9, 12, 33},{15, 11, 23, 0, 21,
37, 35},{10, 13, 18, 36, 22, 25, 7},{2, 14, 6, 5, 19, 28, 34}};// zapis po redovima
6*7 za 37 kontejnera(90%)16.varijanta
//int[][] pozicijeKontejnera = {{0, 13, 0, 0, 20, 5, 29}, {0, 35, 0, 37, 23, 30,
24}, {1, 31, 10, 6, 3, 2, 28},{34, 4, 21, 27, 22, 7,
15},{33, 8, 18, 26, 14, 16, 12},{19, 25, 9, 36, 32, 17, 11}};// zapis po redovima
6*7 za 37 kontejnera(90%)17.varijanta
//int[][] pozicijeKontejnera = {{0, 0, 0, 18, 33, 32, 3}, {0, 0, 26, 12, 19, 15,
22}, {20, 30, 35, 34, 8, 25, 27},{6, 9, 31, 1, 4,
17, 13},{16, 23, 7, 14, 28, 10, 21},{11, 37, 29, 5, 24, 36, 2}};// zapis po
redovima 6*7 za 37 kontejnera(90%)18.varijanta
//int[][] pozicijeKontejnera = {{3, 13, 0, 0, 31, 0, 0}, {5, 21, 0, 17, 4, 26, 24},
{9, 27, 18, 22, 1, 19, 2},{15, 35, 28, 14, 23,
36, 20},{12, 16, 7, 33, 32, 29, 37},{8, 10, 25, 11, 30, 6, 34}};// zapis po
redovima 6*7 za 37 kontejnera(90%)19.varijanta
//int[][] pozicijeKontejnera = {{35, 0, 0, 0, 17, 7, 0}, {2, 0, 13, 27, 11, 10, 8},
{19, 31, 37, 25, 18, 22, 1},{3, 30, 5, 29, 12, 6,
9},{28, 26, 14, 36, 24, 21, 20},{15, 32, 33, 16, 4, 34, 23}};// zapis po redovima
6*7 za 37 kontejnera(90%)20.varijanta
//int[][] pozicijeKontejnera = {{0, 33, 5, 36, 0, 0, 0}, {9, 12, 31, 11, 18, 0,
32}, {8, 13, 19, 2, 23, 35, 14},{21, 1, 20, 17, 10,
3, 25},{27, 4, 28, 24, 34, 7, 29},{26, 37, 30, 15, 6, 22, 16}};// zapis po redovima
6*7 za 37 kontejnera(90%)21.varijanta
//int[][] pozicijeKontejnera = {{0, 19, 0, 0, 11, 33, 13}, {0, 24, 3, 1, 37, 31,
5}, {0, 29, 25, 15, 16, 10, 9},{6, 27, 20, 22, 12,
26, 32},{36, 35, 21, 4, 7, 8, 23},{34, 28, 18, 17, 14, 30, 2}};// zapis po redovima
6*7 za 37 kontejnera(90%)22.varijanta
//int[][] pozicijeKontejnera = {{1, 0, 0, 21, 0, 27, 0}, {7, 0, 34, 12, 24, 9, 6},
{14, 4, 28, 18, 23, 11, 22},{15, 36, 37, 17, 2,
26, 13},{35, 5, 32, 33, 16, 29, 30},{31, 3, 8, 10, 19, 20, 25}};// zapis po
redovima 6*7 za 37 kontejnera(90%)23.varijanta
//int[][] pozicijeKontejnera = {{23, 0, 12, 0, 0, 0, 0}, {18, 34, 11, 24, 4, 2,
15}, {9, 17, 33, 28, 30, 1, 21},{31, 29, 5, 7, 26,
10, 22},{19, 14, 16, 27, 25, 32, 36},{6, 3, 13, 20, 35, 37, 8}};// zapis po
redovima 6*7 za 37 kontejnera(90%)24.varijanta
//int[][] pozicijeKontejnera = {{10, 7, 12, 27, 0, 15, 0}, {6, 8, 23, 34, 0, 9, 0},
{30, 19, 37, 36, 16, 25, 0},{17, 35, 22, 24, 4,
14, 29},{1, 31, 3, 33, 11, 18, 5},{32, 28, 20, 2, 21, 13, 26}};// zapis po redovima
6*7 za 37 kontejnera(90%)25.varijanta
//int[][] pozicijeKontejnera = {{15, 14, 0, 22, 0, 36, 0}, {33, 27, 0, 6, 1, 37,
0}, {25, 9, 7, 3, 35, 11, 24},{2, 20, 21, 5, 16, 10,
28},{12, 8, 4, 26, 17, 32, 30},{23, 18, 29, 19, 31, 13, 34}};// zapis po redovima
6*7 za 37 kontejnera(90%)26.varijanta
//int[][] pozicijeKontejnera = {{0, 31, 0, 0, 37, 30, 22}, {9, 25, 0, 14, 5, 1,
13}, {18, 17, 0, 21, 20, 35, 28},{11, 6, 23, 15, 16,
27, 2},{33, 26, 10, 19, 32, 3, 34},{29, 24, 7, 8, 4, 36, 12}};// zapis po redovima
6*7 za 37 kontejnera(90%)27.varijanta
//int[][] pozicijeKontejnera = {{0, 21, 17, 18, 0, 16, 0}, {0, 7, 29, 8, 10, 22,
15}, {0, 33, 14, 28, 3, 11, 13},{19, 36, 26, 27,
31, 1, 35},{4, 9, 23, 12, 32, 24, 30},{6, 5, 37, 25, 20, 34, 2}};// zapis po
redovima 6*7 za 37 kontejnera(90%)28.varijanta
//int[][] pozicijeKontejnera = {{8, 0, 27, 16, 0, 0, 30}, {24, 0, 5, 18, 35, 0,
28}, {2, 37, 7, 26, 33, 29, 6},{36, 20, 4, 22, 11, 9,
15},{31, 19, 10, 12, 17, 3, 23},{14, 13, 21, 25, 32, 1, 34}};// zapis po redovima
6*7 za 37 kontejnera(90%)29.varijanta
//int[][] pozicijeKontejnera = {{26, 0, 22, 25, 20, 0, 0}, {19, 0, 29, 31, 2, 35,
0}, {1, 11, 9, 32, 30, 15, 36},{12, 16, 23, 6, 3,
5, 14},{34, 18, 37, 33, 17, 4, 27},{24, 28, 21, 7, 8, 10, 13}};// zapis po redovima
6*7 za 37 kontejnera(90%)30.varijanta
```

```
//int[][] pozicijeKontejnera = {{20, 3, 0, 0, 35, 1, 0}, {33, 23, 12, 0, 30, 24,
0}, {10, 11, 26, 13, 18, 7, 31},{15, 21, 22, 27,
29, 25, 9},{32, 37, 34, 4, 36, 2, 14},{28, 17, 5, 16, 19, 6, 8}};// zapis po
redovima 6*7 za 37 kontejnera(90%)31.varijanta
//int[][] pozicijeKontejnera = {{0, 29, 14, 0, 5, 28, 0}, {0, 21, 19, 0, 11, 31,
7}, {33, 18, 1, 3, 9, 17, 20},{30, 15, 37, 35, 32,
16,34},{27, 22, 10, 12, 26, 13, 24},{8, 6, 23, 36, 25, 4, 2}};// zapis po redovima
6*7 za 37 kontejnera(90%)32.varijanta
//int[][] pozicijeKontejnera = {{0, 0, 8, 21, 35, 27, 0}, {0, 30, 1, 13, 7, 23,
24}, {0, 11, 2, 18, 31, 16, 4},{26, 10, 3, 17, 20,
22, 32},{9, 12, 6, 36, 29, 34, 19},{28, 37, 25, 14, 33, 5, 15}};// zapis po
redovima 6*7 za 37 kontejnera(90%)33.varijanta
//int[][] pozicijeKontejnera = {{27, 6, 26, 0, 19, 0, 5}, {Â¸1, 8, 12, 0, 17, 0,
14}, {34, 23, 2, 0, 33, 11, 4},{16, 3, 36, 21, 30,
7, 37},{25, 18, 24, 13, 22, 10, 9},{28, 35, 20, 15, 32, 31, 29}};// zapis po
redovima 6*7 za 37 kontejnera(90%)34.varijanta
//int[][] pozicijeKontejnera = {{0, 0, 12, 18, 0, 0, 34}, {0, 31, 27, 30, 4, 21,
20}, {25, 3, 19, 26, 24, 8, 28},{11, 23, 33, 2, 36,
13, 10},{15, 35, 14, 32, 16, 1, 7},{6, 17, 29, 9, 37, 5, 22}};// zapis po redovima
6*7 za 37 kontejnera(90%)35.varijanta
//int[][] pozicijeKontejnera = {{32, 0, 14, 0, 8, 26, 0}, {22, 0, 11, 25, 18, 4,
0}, {21, 31, 27, 35, 5, 33, 17},{1, 34, 28, 15, 9,
30, 3},{13, 19, 24, 20, 7, 36, 29},{23, 2, 37, 6, 12, 16, 10}};// zapis po redovima
6*7 za 37 kontejnera(90%)36.varijanta
int[][] pozicijeKontejnera = {{14, 0, 23, 0, 8, 0, 29}, {33, 2, 11, 0, 7, 28, 30},
{19, 18, 12, 0, 16, 9, 21},{4, 35, 36, 5, 27, 13,
24},{15, 32, 3, 6, 22, 10, 1},{37, 17, 26, 31, 25, 20, 34}};// zapis po redovima
6*7 za 37 kontejnera(90%)37.varijanta
//int[][] pozicijeKontejnera = {{17, 7, 27, 0, 0, 0, 0}, {20, 29, 32, 0, 21, 9,
22}, {35, 15, 5, 26, 13, 19, 8},{33, 30, 37, 34, 3,
14, 28},{25, 4, 10, 24, 12, 2, 16},{25, 31, 11, 36, 6, 1, 18}};// zapis po redovima
6*7 za 37 kontejnera(90%)38.varijanta
//int[][] pozicijeKontejnera = {{13, 10, 36, 0, 30, 0, 9}, {14, 33, 7, 26, 34, 0,
12}, {18, 37, 23, 24, 11, 0, 31},{16, 6, 5, 20,
29, 0, 35},{15, 32, 27, 19, 22, 2, 21},{3, 28, 4, 8, 25, 1, 17}};// zapis po
redovima 6*7 za 37 kontejnera(90%)39.varijanta
int brojGena = brojKontejnera * 10; //*broj pravila
Configuration conf = new MyConfiguration();
conf.setPreservFittestIndividual(true);
conf.setAlwaysCaculateFitness(true);
conf.setKeepPopulationSizeConstant(true);//da bi veliÄŤina populacije bila ista
conf.setSelectFromPrevGen(0.60);
FitnessFunction myFunc = new MyFitnessFunction(brojGena, brojKontejnera, rows,
columns);
conf.setFitnessFunction(myFunc);
Gene[] sampleGenes = new Gene[brojGena];
for (int i = 0; i < brojGena; i++) {
sampleGenes[i] = new IntegerGene(conf, 1, 4);
}
Chromosome sampleChromosome = new Chromosome(conf, sampleGenes);
conf.setSampleChromosome(sampleChromosome);
conf.setPopulationSize(8000);
//conf.
//Genotype.setStaticConfiguration(conf);
conf.verifyStateIsValid();
Genotype population = Genotype.randomInitialGenotype(conf);
population.applyGeneticOperators();
IChromosome bestSolution;
for (int t = 0; t < MAX_ALLOWED_EVOLUTIONS; t++) {
System.out.println("populacija: " + t);
population.evolve();
bestSolution = population.getFittestChromosome();
int fitnessUkupno = (int) (100000 - bestSolution.getFitnessValue());
if (t == MAX_ALLOWED_EVOLUTIONS - 1) {
System.out.println("rjesenje: " + fitnessUkupno);
int[] kromosom = new int[brojGena];
Gene[] genes = bestSolution.getGenes();
for (int i = 0; i < genes.length; i++) {
```

```
razlikaPremjestanjaVisina += 2 + (int) Math.abs(i - maxI);
} else {
razlikaPremjestanjaVisina += 2 * (Math.abs(maxVisinaPremjestaj -
maxVisinaDizanjaStupca) + 1) +
(int) Math.abs(i - maxI);
}
pozicijeKontejnera[maxI][minIndexJ] = trenutniKontejner;
pozicijeKontejnera[i][indexJ] = 0;
System.out.println("premjeĹˇtanje po visini: " + razlikaPremjestanjaVisina);
System.out.println("premjeĹˇtanje po Ĺˇirini: " + razlikaPremjestanjaSirina);
}
} else if (pravilo == 4) {//provjeravam da stupac nije prazan i do max visine
popunjen
List<Integer> moguciStupci = new ArrayList<>();
int minVisina = 100;
for (int j = 0; j < columns; j++) {
if (j != indexJ) {
if (pozicijeKontejnera[rows - 1][j] != 0) {//provjeravam da li je kontejner u
stupcu
if (pozicijeKontejnera[0][j] == 0) {//provjeravam da li ima pozicija u stupcu s
obzirom na
ograniÄŤenje visine slaganja
moguciStupci.add(j);
}
}
}
}
////System.out.println("indexJ = " +indexJ);
System.out.println("moguci stupci:");
int razlikaStupaca = 100;
int minIndexJ = 100;
for (Integer moguciJ : moguciStupci) {
System.out.println(moguciJ);
if (Math.abs(moguciJ - indexJ) < razlikaStupaca) {
razlikaStupaca = Math.abs(moguciJ - indexJ);
minIndexJ = moguciJ;
}
}
////System.out.println("min index: " + minIndexJ);
if (razlikaStupaca == 100) {
System.out.println("nismo uspjeli");
brojacKromosoma++;
} else {
kromosomRjesenje[brojacRjesenja] = kromosom[brojacKromosoma];
brojacKromosoma++;
brojacRjesenja++;
nasao = true;
brojPremestanja++;//brojim premjeĹˇtanja kontejnera
razlikaPremjestanjaSirina += razlikaStupaca;// suma ukupne udaljenosti
premjeĹˇtanja
//minIndexJ
int maxI = 0;
for (int ii = 0; ii < rows; ii++) {
if (pozicijeKontejnera[ii][minIndexJ] == 0) {
maxI = ii;
}
}
int maxVisinaPremjestaj = rows + 1;
if (minIndexJ < indexJ) {
for (int j = minIndexJ + 1; j < indexJ; j++) {
int minStupac = 0;
for (int ii = 0; ii < rows; ii++) {
if (pozicijeKontejnera[ii][j] == 0) {
minStupac = ii;
}
}
if (minStupac < maxVisinaPremjestaj) {
maxVisinaPremjestaj = minStupac;
```

```
}
}
} else {
for (int j = indexJ + 1; j < minIndexJ; j++) {
int minStupac = 0;
for (int ii = 0; ii < rows; ii++) {
if (pozicijeKontejnera[ii][j] == 0) {
minStupac = ii;
}
}
if (minStupac < maxVisinaPremjestaj) {
maxVisinaPremjestaj = minStupac;
}
}
}
int maxVisinaDizanjaStupca;
if (maxI < i) {
maxVisinaDizanjaStupca = maxI;
} else {
maxVisinaDizanjaStupca = i;
}
if ((maxVisinaPremjestaj - maxVisinaDizanjaStupca) > 0) {
razlikaPremjestanjaVisina += (int) Math.abs(i - maxI);
} else if ((maxVisinaPremjestaj - maxVisinaDizanjaStupca) == 0) {
razlikaPremjestanjaVisina += 2 * 1 + (int) Math.abs(i - maxI);
} else {
razlikaPremjestanjaVisina += 2 * (Math.abs(maxVisinaPremjestaj -
maxVisinaDizanjaStupca) + 1) +
(int) Math.abs(i - maxI);
}
pozicijeKontejnera[maxI][minIndexJ] = trenutniKontejner;
pozicijeKontejnera[i][indexJ] = 0;
System.out.println("premjeĹˇtanje po visini: " + razlikaPremjestanjaVisina);
System.out.println("premjeĹˇtanje po Ĺˇirini: " + razlikaPremjestanjaSirina);
}
}
}
System.out.println(" nakon pravila " + kromosomRjesenje[brojacRjesenja - 1]);
System.out.println("");
for (int iii = 0; iii < rows; iii++) {
for (int j = 0; j < columns; j++) {
System.out.print(pozicijeKontejnera[iii][j] + "\t");
}
System.out.println("");
}
}
}
pozicijeKontejnera[indexI][indexJ] = 0;//pozicija kontejnera nakon premjeĹˇtanja
System.out.println("");
System.out.println("Izgled nakon izbacivanja kontejnera " + k);
for (int iii = 0; iii < rows; iii++) {
for (int j = 0; j < columns; j++) {
System.out.print(pozicijeKontejnera[iii][j] + "\t");
}
System.out.println("");
}
////System.out.println("");
System.out.println("broj premjestanja nakon izbacivanja " + k + ". kontejnera: " +
brojPremestanja + ", a razlika
premjestanja po Ĺˇirini je " + razlikaPremjestanjaSirina + ", a razlika
premjestanja po visini je " +
razlikaPremjestanjaVisina);
}/
///System.out.println("\n\n\n\n\n\n");
System.out.print("Random rjeĹˇenje: ");
//Random r = new Random();
for (int i = 0; i < brojGena; i++) {
//kromosom[i] = r.nextInt(4) + 1;
```

```
System.out.print(kromosom[i] + " ");
//kromosom[i] = 4;//za provjeru 2.pravila stavi 2
}
System.out.println("");
System.out.print("Kromosom rjeĹˇenje: ");
//Random r = new Random();
for (int i = 0; i < brojGena; i++) {
//kromosomRjesenje[i] = r.nextInt(4) + 1;
System.out.print(kromosomRjesenje[i] + " ");
//kromosom[i] = 4;//za provjeru 2.pravila stavi 2
}
System.out.println("");
//System.out.println("ukupan broj premjestanja: " + brojPremestanja + ", a razlika
premjestanja sirina je " +
razlikaPremjestanjaSirina + ", a razlika premjestanja visina je " +
razlikaPremjestanjaVisina);
//fitness = brojPremestanja * 1000 + razlikaPremjestanjaSirina +
razlikaPremjestanjaVisina;
//fitness = 100000 - fitness;
//return fitness;
} else {
System.out.println("trenutno rjeĹˇenje: " + fitnessUkupno);
}
}
}
}
```

Privitak 2. – „R“ generator početnih rasporeda kontejnera unutar odjeljka

```
> ########################################################################
> # rm(list=ls())
> ############SKRIPTA ZA ODREDJIVANJE DOPUSTENIH POZICIJA  .... [TRUNCATED]

> A <- matrix(x, 6, 7, byrow=TRUE)####velicina odjeljka####

> dodjela.pozicija <- function(B) {
+   uspjeh <- FALSE
+   while(!uspjeh) {
+     B <- matrix(sample(A), 6, 7, byrow=TRUE)
+     D <- apply(B, 2, dif .... [TRUNCATED]

> n = 39#### broj generiranja početnih rasporeda####

> for(i in 1:n) {
+   poz <- dodjela.pozicija(B)
+   show(poz)
+ }
```

Privitak 3. - Generirani početni raspored kontejnera za odjeljak veličine 6x7 i udjela popunjenosti od približno 90%

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,] -100 -100   16 -100   37   31   12
[2,]   24   13    3 -100   30   19   15
[3,]   20   36   29 -100   21   34   11
[4,]    6   17    9    7    5   23    1
[5,]    8   25   14   32   27    4    2
[6,]   22   10   18   28   26   33   35
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,]    3   16 -100 -100 -100   19   24
[2,]   12    6 -100   23 -100   33   26
[3,]    7   14   20    4   30   34   37
[4,]   27   32   11   17   28   25   22
[5,]    5   29    2    9    1   21   35
[6,]   15   36   31    8   13   18   10
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,] -100 -100 -100   24    1    3   10
[2,] -100 -100   25   36   13   29   17
[3,]   11   31   15   19   23    9   30
[4,]   22   32   16   21    6    2   28
[5,]   33    4   20   35   26   18   14
[6,]    5   37   12   34   27    8    7
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,] -100   26   36   24 -100 -100   25
[2,]   22   18   27   20 -100 -100   15
[3,]    8   14   37   32    6   11   35
[4,]   28   33   13   29   12    5    1
[5,]   31   23    9    2   16    7   19
[6,]   17   10    3   34   21   30    4
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,]    6 -100    8 -100   16   30 -100
[2,]   20 -100    1    4   12   21 -100
[3,]    9   28   22   27   25    5    7
[4,]   19   10   37   32   29   31   26
[5,]   36   18   13   24   17   14   23
[6,]   33   34   35    2    3   15   11
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,]   24 -100    5 -100    6 -100 -100
[2,]   21    7   23   19    8   12 -100
[3,]   35    9   14   25   28   10    3
[4,]    2   20   15   31   17   36   26
[5,]   37   18   30   13   11   33    1
[6,]   29   27   32   34   22   16    4
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,]   16 -100 -100   33    3    8 -100
[2,]   35 -100    7   23    2   31 -100
[3,]   37   13   34    9    4   24   12
[4,]   22   27   10   18   19   17   14
[5,]   25   28    6    5   29   15   21
[6,]   20   30    1   32   11   36   26
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,]   24    2   19 -100   14 -100   34
[2,]    8   25    4 -100   27    3   29
[3,]   28   30   26 -100   21   33    5
[4,]   23   20   16 -100   13   35    1
[5,]   22   31   18   32    6    9   17
[6,]   15   37    7   12   10   36   11
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,] -100 -100   18   16    8   19 -100
[2,] -100    3   25   37   26   24   29
[3,] -100   32    4   30    6    9   14
[4,]    7   34   10    1   13    5   28
[5,]    2   27   15   22   17   20   36
[6,]   31   21   33   23   12   11   35
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,] -100    6   23    7   30    1   35
[2,] -100   12   10   16   15   31   21
[3,] -100   20   34    8    4   33   26
[4,] -100    9   17   32    2   27    3
[5,] -100   24   11   25   19   14   22
[6,]   18   37   29   28    5   13   36
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,] -100   34    1 -100   10   18 -100
[2,]    8    7   24 -100    6    5   26
[3,]   14    9   27 -100   19   36   32
[4,]   30   37   13    4   22   33   17
[5,]    3   11   31   20   35   28   21
[6,]   29   23   25   15   16   12    2
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,] -100 -100   20 -100   11   22   35
[2,]   31 -100   19 -100   25    7    2
[3,]   34   26   33    6    1   14   30
[4,]   17   12   24   37   29   15   18
[5,]    3    5   10   16   27   13   36
[6,]    8   23   28   21    9    4   32
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,] -100 -100 -100 -100   25   11   16
[2,] -100   15   26    5   34   36   21
[3,]   27   12   18   28   30   23   29
[4,]   14    8    7   19   22    3   17
[5,]    9   32   24    1   35    4   33
[6,]   13   20   10   31    2   37    6
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,] -100 -100   10   21   32 -100 -100
[2,] -100   33    4    6   11   24    1
[3,]   14    8   36   18   27   30   22
[4,]   19    3   34   29   35   23    9
[5,]    5   17   26   31   28   16    7
[6,]    2   13   25   12   20   37   15
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,]   16   14    6   10   26 -100 -100
[2,]   20   37   33   19   24 -100 -100
[3,]   11   25    3   27    1   17 -100
[4,]    9   35   23   28    2   12   18
[5,]    4   22   21   34    7   15   31
[6,]    8   29   32    5   36   30   13
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,]   31    4   24 -100   16 -100   32
[2,]   26   27   30 -100   29   20    1
[3,]    8   17    3 -100    9   12   33
[4,]   15   11   23 -100   21   37   35
[5,]   10   13   18   36   22   25    7
[6,]    2   14    6    5   19   28   34
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,] -100   13 -100 -100   20    5   29
[2,] -100   35 -100   37   23   30   24
[3,]    1   31   10    6    3    2   28
[4,]   34    4   21   27   22    7   15
[5,]   33    8   18   26   14   16   12
[6,]   19   25    9   36   32   17   11
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,] -100 -100 -100   18   33   32    3
[2,] -100 -100   26   12   19   15   22
[3,]   20   30   35   34    8   25   27
[4,]    6    9   31    1    4   17   13
[5,]   16   23    7   14   28   10   21
[6,]   11   37   29    5   24   36    2
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,]    3   13 -100 -100   31 -100 -100
[2,]    5   21 -100   17    4   26   24
[3,]    9   27   18   22    1   19    2
[4,]   15   35   28   14   23   36   20
[5,]   12   16    7   33   32   29   37
[6,]    8   10   25   11   30    6   34
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,]   35 -100 -100 -100   17    7 -100
[2,]    2 -100   13   27   11   10    8
[3,]   19   31   37   25   18   22    1
[4,]    3   30    5   29   12    6    9
[5,]   28   26   14   36   24   21   20
[6,]   15   32   33   16    4   34   23
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,] -100   33    5   36 -100 -100 -100
[2,]    9   12   31   11   18 -100   32
[3,]    8   13   19    2   23   35   14
[4,]   21    1   20   17   10    3   25
[5,]   27    4   28   24   34    7   29
[6,]   26   37   30   15    6   22   16
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,] -100   19 -100 -100   11   33   13
[2,] -100   24    3    1   37   31    5
[3,] -100   29   25   15   16   10    9
[4,]    6   27   20   22   12   26   32
[5,]   36   35   21    4    7    8   23
[6,]   34   28   18   17   14   30    2
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,]    1 -100 -100   21 -100   27 -100
[2,]    7 -100   34   12   24    9    6
[3,]   14    4   28   18   23   11   22
[4,]   15   36   37   17    2   26   13
[5,]   35    5   32   33   16   29   30
[6,]   31    3    8   10   19   20   25
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,]   23 -100   12 -100 -100 -100 -100
[2,]   18   34   11   24    4    2   15
[3,]    9   17   33   28   30    1   21
[4,]   31   29    5    7   26   10   22
[5,]   19   14   16   27   25   32   36
[6,]    6    3   13   20   35   37    8
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,]   10    7   12   27 -100   15 -100
[2,]    6    8   23   34 -100    9 -100
[3,]   30   19   37   36   16   25 -100
[4,]   17   35   22   24    4   14   29
[5,]    1   31    3   33   11   18    5
[6,]   32   28   20    2   21   13   26
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,]   15   14 -100   22 -100   36 -100
[2,]   33   27 -100    6    1   37 -100
[3,]   25    9    7    3   35   11   24
[4,]    2   20   21    5   16   10   28
[5,]   12    8    4   26   17   32   30
[6,]   23   18   29   19   31   13   34
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,] -100   31 -100 -100   37   30   22
[2,]    9   25 -100   14    5    1   13
[3,]   18   17 -100   21   20   35   28
[4,]   11    6   23   15   16   27    2
[5,]   33   26   10   19   32    3   34
[6,]   29   24    7    8    4   36   12
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,] -100   21   17   18 -100   16 -100
[2,] -100    7   29    8   10   22   15
[3,] -100   33   14   28    3   11   13
[4,]   19   36   26   27   31    1   35
[5,]    4    9   23   12   32   24   30
[6,]    6    5   37   25   20   34    2
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,]    8 -100   27   16 -100 -100   30
[2,]   24 -100    5   18   35 -100   28
[3,]    2   37    7   26   33   29    6
[4,]   36   20    4   22   11    9   15
[5,]   31   19   10   12   17    3   23
[6,]   14   13   21   25   32    1   34
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,]   26 -100   22   25   20 -100 -100
[2,]   19 -100   29   31    2   35 -100
[3,]    1   11    9   32   30   15   36
[4,]   12   16   23    6    3    5   14
[5,]   34   18   37   33   17    4   27
[6,]   24   28   21    7    8   10   13
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,]   20    3 -100 -100   35    1 -100
[2,]   33   23   12 -100   30   24 -100
[3,]   10   11   26   13   18    7   31
[4,]   15   21   22   27   29   25    9
[5,]   32   37   34    4   36    2   14
[6,]   28   17    5   16   19    6    8
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,] -100   29   14 -100    5   28 -100
[2,] -100   21   19 -100   11   31    7
[3,]   33   18    1    3    9   17   20
[4,]   30   15   37   35   32   16   34
[5,]   27   22   10   12   26   13   24
[6,]    8    6   23   36   25    4    2
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,] -100 -100    8   21   35   27 -100
[2,] -100   30    1   13    7   23   24
[3,] -100   11    2   18   31   16    4
[4,]   26   10    3   17   20   22   32
[5,]    9   12    6   36   29   34   19
[6,]   28   37   25   14   33    5   15
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,]   27    6   26 -100   19 -100    5
[2,]    1    8   12 -100   17 -100   14
[3,]   34   23    2 -100   33   11    4
[4,]   16    3   36   21   30    7   37
[5,]   25   18   24   13   22   10    9
[6,]   28   35   20   15   32   31   29
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,] -100 -100   12   18 -100 -100   34
[2,] -100   31   27   30    4   21   20
[3,]   25    3   19   26   24    8   28
[4,]   11   23   33    2   36   13   10
[5,]   15   35   14   32   16    1    7
[6,]    6   17   29    9   37    5   22
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,]   32 -100   14 -100    8   26 -100
[2,]   22 -100   11   25   18    4 -100
[3,]   21   31   27   35    5   33   17
[4,]    1   34   28   15    9   30    3
[5,]   13   19   24   20    7   36   29
[6,]   23    2   37    6   12   16   10
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,]   14 -100   23 -100    8 -100   29
[2,]   33    2   11 -100    7   28   30
[3,]   19   18   12 -100   16    9   21
[4,]    4   35   36    5   27   13   24
[5,]   15   32    3    6   22   10    1
[6,]   37   17   26   31   25   20   34
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,]   17    7   27 -100 -100 -100 -100
[2,]   20   29   32 -100   21    9   22
[3,]   35   15    5   26   13   19    8
[4,]   33   30   37   34    3   14   28
[5,]   25    4   10   24   12    2   16
[6,]   23   31   11   36    6    1   18
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,]   13   10   36 -100   30 -100    9
[2,]   14   33    7   26   34 -100   12
[3,]   18   37   23   24   11 -100   31
[4,]   16    6    5   20   29 -100   35
[5,]   15   32   27   19   22    2   21
[6,]    3   28    4    8   25    1   17
```

Privitak 4. - Grafički prikaz dobivenih rezultata za odjeljke 3x7, 4x7, 5x7 i 6x7 te udjele popunjenosti od približno 55% za odabrane primjere

3x7

(1)

		11	9		8	
6		10	7		5	
2		1	3		4	

(2)

			9		8	
6	10		7		5	
2	11		3		4	

(3)

	6		9		8	
	10		7		5	
	11		3		4	

(4)

	6				8	
	10	7			5	
	11	9			4	

(5)

	6					
	10	7		5		
	11	9		8		

(6)

	10					
	11	9		8		

4x7

(1)

			12	9		15
11			7	8		13
14			3	6		5
1			10	4		2

(2)

			12	9		15
			7	8		13
			3	6		5
	11	14	10	4		2

(3)

			12	9		
			7	8	5	
			3	6	13	
	11	14	10	4	15	

(4)

				9		
		7		8	5	
		12		6	13	
	11	14	10	4	15	

(5)

			6			
		7	8		5	
		12	9		13	
	11	14	10		15	

5x7

(1)

	19					
	13	16		15		
7	10	12		14	18	
5	9	4		8	17	
3	1	2		6	11	

(2)

		16	9	15		
7		12	10	14	18	
5		4	13	8	17	
3		2	19	6	11	

(3)

			9	15		
7	4		10	14	18	
5	12		13	8	17	
3	16		19	6	11	

(4)

			9	15		
	4		10	14	18	
	12	5	13	8	17	
	16	7	19	6	11	

(5)

			8			
			9			
			10		18	
	12		13		17	14
	16	7	19		11	15

(6)

	12		13	17		14
	16		19	18		15

6x7

(1)

		21		22		
		19		12		
	23	15		10	16	
	8	11		7	14	20
	5	6		4	13	18
	2	3		1	9	17

(2)

		21				
		19	4			
	23	15	7		16	
	8	11	10		14	20
	5	6	12		13	18
	2	3	22		9	17

(3)

		21				
		19	4			
		15	7		16	
5		11	10		14	20
8		6	12		13	18
23		3	22		9	17

(4)

	6		4			
	11		7		16	
5	15		10		14	20
8	19		12		13	18
23	21		22		9	17

(5)

	11					16
	15		10			20
	19		12	13		18
23	21		22	14		17

(6)

	19				18	
23	21		22		20	