Дводимензиони проблем паковања у складишта са ограничењем гиљотине

Јелена Граовчевић
Математички факултет
Универзитет у Београду

Мастер рад

Београд, 2021.

Универзитет у Београду, Математички факултет, Мастер рад

Аутор:	Јелена Граовчевић
Наслов:	Дводимензиони проблем паковања у складишта са ограничењем гиљотине
Ментор:	проф. др Александар Савић Математички факултет, Универзитет у Београду
Чланови комисије:	проф. др Зоран Станић Математички факултет, Универзитет у Београду доц. др Зорица Дражић Математички факултет, Универзитет у Београду
Датум:	30.9.2021.

Наслов: Дводимензиони проблем паковања у складишта са ограничењем гиљотине

Резиме: Дводимензиони проблем паковања у складишта (енг. Two-dimensional bin paking problem - 2BPP) је проблем паковања датог скупа малих правоугаоних предмета, који се називају *пакети*, у минималан број великих правоугаоних објеката, који се називају *складишта*, тако да се пакети не преклапају и да су им ивице паралелне ивицама складишта. Примену налази у разним областима, од индустрије до компјутерских система и мрежа. Спада у групу NP тешких проблема. Варијанта овог проблема, где се пакети добијају низом резова од ивице до ивице и не могу бити ротирани, назива се Дводимензиони проблем паковања у складишта са ограничењем гиљотине (енг. Two-Dimensional Bin Packing Problem with Guillotine Restriction) и означава са 2BP|O|G. Њену појаву проузроковао је технолошки развој, јер се у новије време за сечење пакета користе аутоматске машине, али доводи до мање искоришћености складишта. Ограничење на гиљотинске резове је нарочито значајно у индустрији дрвета, стакла и металургији.

Циљ овог рада биће представљање дводимензионог проблема паковања у складишта, математичких формулација, хеуристика и метахеуристика за његово решавање, и горњих и доњих граница. Посебна пажња ће бити посвећена варијанти овог проблема са гиљотинским резовима, уз додатно ограничење да се пакети не могу ротирати, а која се, у складу са нотацијом коју су дали Lodi, Martello и Vigo, означава са 2BP|O|G. Биће дата класификација негиљотинабилних образаца, представљени начини за њихово откривање и услови под којима они постају гиљотинабилни, са илустрацијом на примерима. Уз све то, детаљно ће бити представљене две хеуристике: алгоритам парцијалне енумерације, за решавање 2BP|O|G проблема, и генетски алгоритам са Crow претрагом, за решавање 2BP|O|F проблема, тј. дводимензионог проблема паковања у складишта без ограничења гиљотине, заједно са објављеним експерименталним резултатима. Ти резултати, заједно са илустрацијом једноставног примера паковања помоћу наведених хеуристика, биће искоришћени за њихово поређење и извођење закључка о паковању уз ограничење гиљотине.

Кључне речи: дводимензионо паковање, гиљотински рез, хеуристике

Садржај

Глава 1

Дводимензиони проблем паковања у складишта

1.1 Увод

Дводимензиони проблем паковања у складишта (енг. Two-dimensional bin packing problem - 2BPP) се може представити на следећи начин. Дат је скуп од n правоугаоних предмета $i \in I = \{1, ..., n\}$, који се називају *пакети*, и неограничен број идентичних правоугаоних објеката, који се називају *складишта*. Сваки пакет i има ширину w_i и висину h_i, док складишта имају ширину W и висину H. Без губљења општости, може се претпоставити да су сви улазни подаци целобројни, и $0 < w_i \leq W$ и $0 < h_i \leq H$ за све $i \in I$. Проблем се састоји у распоређивању свих пакета, тако да они заузимају минималан број складишта, и да се не преклапају. Овај проблем представља дводимензиони наставак класичног једнодимензионог проблема паковања у складишта (енг. One-dimensional bin packing problem - 1BPP) и један је од највише изучаваних проблема у категорији паковања и сечења.

2BPP проблем се јавља у важним свакодневним применама као што су сечење папира, стакла, и челика, али и код компјутерских система и мрежа. У литератури се може наћи са различитим условима, као што су оријентација, гиљотински резови и сл. У складу са тим јављају се различите његове варијанте.

У одељцима 1.3 и 1.4 су описани проблеми паковања правоугаоних пакета и њихове примене. Математичке формулације паковања у складишта дате су у одељку 1.5. Коефицијенти перформанси апсолутно и асимптотски најнеповољнијег случаја дати су у 1.6. Горње границе, метахеуристике и алгоритми апроксимације дати су у одељку 1.7. Доње границе и егзактни алгоритми описани су у одељцима 1.8 и 1.9.

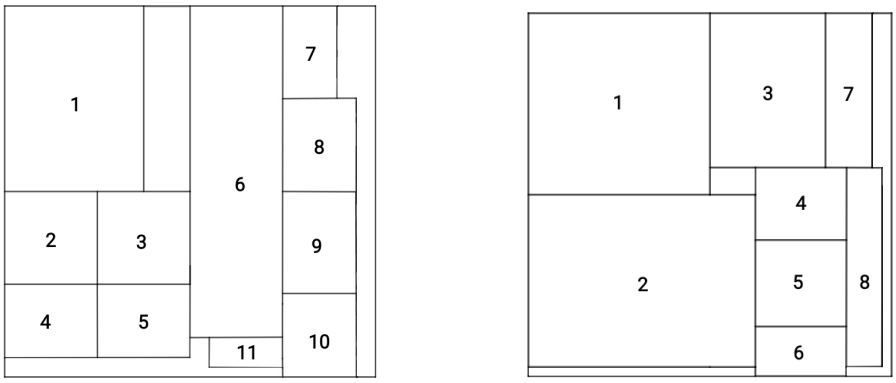
1.2 Сечење и паковање

Дводимензиони проблем паковања у складишта припада групи проблема сечења и паковања. Проблеми сечења и паковања обухватају смештање датог скупа (малих) предмета у један или више (великих) објеката, тако да се ти предмети не преклапају и да се оствари максимум или минимум дате функције циља. Ови проблеми су значајни у индустрији, нарочито код сечења (папир, дрво, стакло, челик...) и паковања (складиштење, транспорт...). Разликују се по димензији (најчешће се разматрају једнодимензиони, дводимензиони и тродимензиони проблеми) и облику пакета и складишта (неправилни и правилни). У овом раду ће бити представљен дводимензиони проблем паковања у складишта са пакетима правоугаоног (правилног) облика.

1.3 Проблем паковања правоугаоних пакета

Дат је скуп од n правоугаоних пакета $i \in I = \{1, ..., n\}$, ширине w_i и висине h_i, и један или више правоугаоних објеката, у које те пакете треба сместити. Пакете је потребно поставити ортогонално, тако да се не преклапају и да су им ивице паралелне ивицама објекта, а да се при томе оствари максимум (минимум) функције циља. Функција циља је најчешће површина објекта или број употребљених објеката. Овај проблем карактеришу два битна услова:

- **оријентација:** сваки пакет има оријентацију која је фиксирана, тј. није дозвољена ротација пакета за 90°.
- **гиљотински рез:** пакети морају бити добијени низом резова од ивице до ивице, паралелних ивици великог правоугаоног објекта. Гиљотински рез је последица техничких ограничења машина за сечење или материјала. Пример гиљотинског и негиљотинског реза дат је на слици 1.1.



Слика 1.1: Пример гиљотинског и негиљотинског реза.

Активно се проучава шест врста проблема паковања правоугаоних пакета. Ради једноставности, претпоставља се да је оријентација пакета фиксирана и да важи услов гиљотине, ако другачије није назначено.

- **Дводимензиони проблем паковања на траци** (енг. Strip packing problem - 2SPP): дат је скуп од n правоугаоних пакета $i \in I = \{1, ..., n\}$, ширине w_i и висине h_i, и један велики објекат, који називамо *трака*, фиксиране ширине W и променљиве висине H. Циљ је да се минимизира висина траке тако да се сви пакети могу спаковати на њу.
- **Проблем минимизације површине** (енг. Area minimization problem - 2AP): дат је скуп од n правоугаоних пакета $i \in I = \{1, ..., n\}$, ширине w_i и висине h_i, и један велики објекат, где су и ширина W и висина H променљиве. Циљ је да се минимизира површина WH објекта, тако да се сви пакети могу спаковати у њега.
- **Дводимензиони проблем паковања у складишта** (енг. Two-dimensional bin packing problem - 2BPP): дат је скуп од n правоугаоних пакета $i \in I = \{1, ..., n\}$, ширине w_i и висине h_i, и неограничен број коначних идентичних објеката, које називамо *складишта*, ширине W и висине H. Проблем се састоји у распоређивању пакета у минималан број складишта. Специјалан случај, где је $w_i = W$, $i \in I = \{1, ..., n\}$, је један од најпознатијих оптимизационих проблема, *једнодимензиони проблем паковања у складишта* (енг. One-dimensional bin packing problem - 1BPP): n пакета, величине h_i, спаковати у минималан број идентичких складишта, капацитета H, тако да величина пакета у било ком складишту не прелази капацитет H. Како је 1BPP проблем познат као NP тежак, исто закључак се изводи и за 2BPP.
- **Дводимензиони проблем ранца** (енг. Two-dimensional knapsack problem - 2KP): дат је скуп од n правоугаоних пакета $i \in I = \{1, ..., n\}$, чија је ширина w_i, висина h_i, и добит p_i, и правоугаони *ранац* ширине W и висине H. Циљ је да се нађе подскуп

$I^{'} \subseteq I$ пакета са максималном укупном вредношћу $\sum_{i \in I^{'}} p_i$, тако да сви пакети из $I^{'}$ могу бити спаковани у ранац.

- **Дводимензиони проблем смањења залиха** (Two-dimensional cutting stock problem - 2CP): дат је скуп од n правоугаоних пакета $i \in I = \{1, ..., n\}$ који имају ширину w_i, висину h_i, и потражњу d_i, и неограничен број коначних идентичних правоугаоних објеката, које називамо *складишта*, која имају ширину W и висину H. Проблем се састоји у распоређивању пакета у складишта, тако да број складишта буде минималан (тј. за свако i, потребно је спаковати d_i копија пакета i у складишта).
- **Проблем утовара палета** (енг. Pallet loading problem - 2PLP): дат је довољно велики број пакета идентичне величине (w, h), и један велики правоугаони објекат величине (W, H). Циљ је сместити максималан број пакета у објекат. Пакете је могуће ротирати за 90°.

Сви проблеми, осим 2PLP, су NP тешки. За 2PLP није познато да ли припада NP тешким проблемима.

1.4 Примене

Класичан проблем паковања у складишта има многе примене, од индустрије (сечење папира, дрвета, каблова...) до компјутерских система (алокација меморије) и мрежа (рутирање пакета у комуникационим мрежама) [33]. Јавља се и као потпроблем у другим поставкама. Најважнија примена је у подешавању производње правоугаоних комада, које треба исећи од залиха сировог материјала, тако да губитак буде минималан. Ограничења на ортогонално паковање и паковање са ограниченом ротацијом или чак без ротације у овим поставкама имају смисла ако се за добијање пакета користе фабрички обрасци и када је потребно задржати поравнање. Распоређивање независних задатака на групи процесора, од којих сваки захтева одређени број суседних процесора или алокацију меморије током одређеног временског периода, може бити моделиран као проблем паковања траке [46]. У овој примени ширина траке представља укупан број процесора или доступне меморије, а висина представља максимално време извршења. Даља примена се може наћи у VLSI (енг. Very large scale integration) дизајну (проблем паковања минималног правоугаоника [41]) и проблему постављања огласа [23], где је потребно поставити све огласе на минималан број страна новина или интернет страница.

1.5 Математичке формулације проблема

1.5.1 Једнодимензиони проблем паковања у складишта

Garey и Johnson су у [39] 1BPP дефинисали на следећи начин. Нека је дато n пакета величине h_i, $i \in I = \{1, ..., n\}$, и n складишта, капацитета H. Задатак је спаковати све пакете у минималан број складишта, тако да укупна величина не прелази капацитет складишта.

Математичка формулација проблема је

$$(1BPP) \quad \min \quad \sum_{j=1}^{n} y_j \tag{1.1а}$$

$$\text{п. о.} \quad \sum_{i=1}^{n} h_i x_{ji} \leq H y_j, \qquad j = 1, ..., n \tag{1.1б}$$

$$\sum_{j=1}^{n} x_{ji} = 1, \qquad i = 1, ..., n \tag{1.1ц}$$

$$y_j \in \{0, 1\}, \qquad j = 1, ..., n \tag{1.1д}$$

$$x_{ji} \in \{0, 1\}, \qquad i, j = 1, ..., n \tag{1.1е}$$

где је

$$y_j = \begin{cases} 1, & \text{ако се користи складиште } j \\ 0, & \text{иначе,} \end{cases} \tag{1.2}$$

$$x_{ji} = \begin{cases} 1, & \text{ако се пакет } i \text{ пакује у складиште } j \\ 0, & \text{иначе.} \end{cases} \tag{1.3}$$

Без губљења општости, може се претпоставити да су h_i и H позитивни цели бројеви и да је $h_i \leq H$, $i \in I = \{1, ..., n\}$. Ако пакет не задовољава последњу претпоставку, онда је инстанца тривијално недопустива.

1.5.2 Дводимензиони проблем паковања у складишта

Први покушај моделирања 2BPP проблема начинили су Gilmore и Gomory у [24], кроз наставак приступа 1BPP проблему. Они су предложили приступ генерисања колона базиран на енумерацији свих подскупова пакета (*образаца*) који могу бити спаковани у једно складиште. Нека је A_j бинарни вектор колоне, који чини n елемената a_{ij} $(i = 1, ..., n,\ j = 1, ..., m)$, чија је вредност 1, ако i припада обрасцу j, а 0 иначе. Скуп свих допустивих образаца је представљен као матрица A, састављена од свих могућих колона A_j, а одговарајући модел је

$$(2BPP) \quad \min \quad \sum_{j=1}^{m} x_j \tag{1.4а}$$

$$\text{п. о.} \quad \sum_{j=1}^{m} a_{ij} x_j \geq 1, \qquad i = 1, ..., n \tag{1.4б}$$

$$x_j \in \{0, 1\}, \qquad j = 1, ..., m \tag{1.4ц}$$

где x_j узима вредност 1 ако образац припада решењу, а 0 иначе.

Због великог броја колона које се могу појавити у A, једини начин за управљање моделом да се оне динамички формирају када је потребно. Gilmore и Gomory су у [25] и [26] за решавање 1BPP проблема приказали приступ динамичког програмирања за генерисање колона, решавањем, као *подређеног* проблема, придруженог $0-1$ проблема ранца (више о овом проблему се може наћи у [69]). За 2BPP проблем су посматрали тешкоће дводимензионог придруженог проблема. Зато су прешли на лакши случај, где пакети морају бити спаковани у редове, формирајући нивое (детаљи у следећем одељку), за које је подређени проблем решен кроз двофазни алгоритам динамичког програмирања.

Beasley је у [5] разматрао дводимензиони проблем сечења, где је сваком пакету придружена добит, са циљем да се у једно складиште спакује подскуп пакета са

максималном добити (дводимензиони проблем ранца). Дао је ILP формулацију базирану на дискретној репрезентацији геометријског простора и употреби координата којима су придружени пакети. Наиме,

$$x_{ipq} = \begin{cases} 1, & \text{ако је пакет } i \text{ смештен својим доњим левим углом у тачку } (p,q) \\ 0, & \text{иначе} \end{cases} \tag{1.5}$$

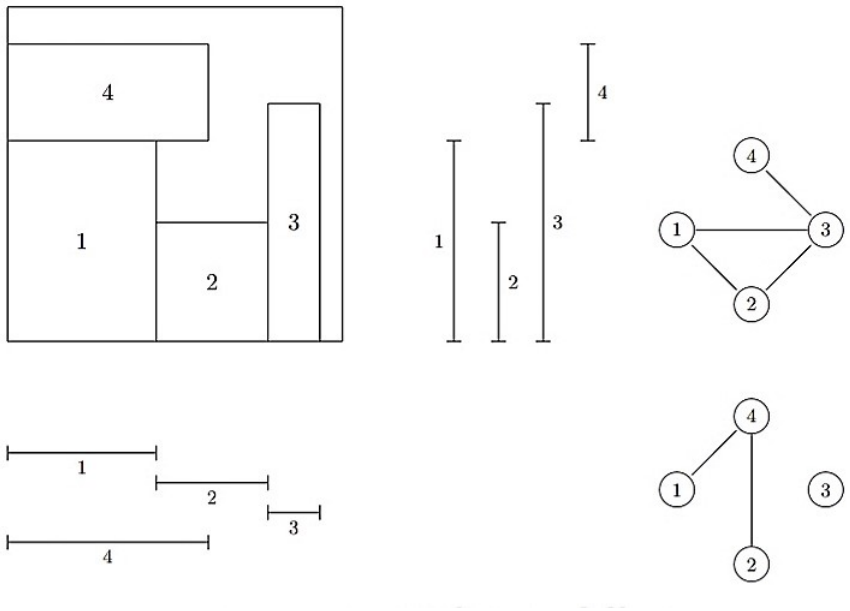
за $i = 1, ..., n,\ p = 1, ..., W - w_i$ и $q = 1, ..., H - h_i$. Сличан модел, у коме су p и q координате обрађене кроз различите промењиве, представили су Hadjiconstantinou и Christofides у [32]. Оба модела се користе за обезбеђивање горње границе путем Лагранжове релаксације и субградијентне оптимизације.

Потпуно другачији приступ моделирању предложили су Fekete и Schepers у [19], карактеризујући паковање скупа пакета у једно складиште теоријом графова. Нека је $G_w = (V, E_w)$ ($G_h = (V, E_h)$) интервални граф са чвором v_i, који представља пакет i који се пакује, и граном (v_i, v_j) између чворова v_i и v_j, при чему се та грана формира ако и само ако се пројекције пакета i и j на x-осу (y-осу) преклапају (слика 1.2). У [19] је доказано да, ако је паковање допустиво, онда

а) за сваки стабилан скуп $S \subset G_w$ (G_h), $\sum_{v_i \in S} w_i \leq W (\sum_{v_i \in S} h_i \leq H)$;

б) $E_w \cap E_h = \emptyset$.

Ова карактеризација се лако преноси и на више димензије.



Слика 1.2: Приступ моделирању који су представили Fekete и Schepers.

1.5.3 ILP модел за паковање по нивоима

ILP модели имају полиномијалан број промењливих, а ограничења за специјалан случај, где пакети треба да буду спаковани ,,по нивоима", представили су Lodi, Martelo и Vigo у [54].

Као што се може видети у наредном одељку, највећи број апроксимационих алгоритама за 2BPP и 2SPP проблем пакује пакете у редове, формирајући *нивое*. Први ниво је дно складишта, и пакети се на њега пакују својом основом. Следећи ниво је одређен хоризонталном линијом повученом на врху највишег пакета, спакованог на претходном нивоу итд. Пакети не морају имати исту висину. 2BPP проблем са оваквим начином паковања се означава са 2LBPP.

Без губљења општости, у наставку су разматрана само *нормализована паковања*, тј. паковања таква да:

1) на сваком нивоу, највиши пакет се налази на крајњој левој страни;

2) пакети су набројани по нерастућим висинама h_i.

Пакет који се налази са крајње леве стране (односно ниво на дну складишта) *иницијализује* ниво (односно складиште).

2LBPP проблем може бити ефикасно моделиран под претпоставком да постоји n потенцијалних нивоа (i-ти ниво је повезан са i-тим пакетом који га иницијализује), и n потенцијалних складишта (k-то складиште је повезано са k-тим нивоом који га иницијализује). Стога, са y_i, $i \in I$ (q_k, $k \in I$) је означена бинарна променљива која узима вредност 1 када пакет i иницијализује ниво i (ниво k иницијализује складиште k), а 0 иначе. Проблем може бити моделиран на следећи начин:

$$2LBPP \quad \min \quad \sum_{k=1}^{n} q_k \tag{1.6а}$$

$$\text{п. о.} \quad \sum_{i=1}^{j-1} x_{ij} + y_j = 1, \qquad j = 1, ..., n \tag{1.6б}$$

$$\sum_{j=i+1}^{n} w_j x_{ij} \leq (W - w_i) y_i, \qquad i = 1, ..., n-1 \tag{1.6ц}$$

$$\sum_{k=1}^{i-1} z_{ki} + q_i = y_i, \qquad i = 1, ..., n \tag{1.6д}$$

$$\sum_{i=k+1}^{n} h_i z_{ki} \leq (H - h_k) q_k, \qquad k = 1, ..., n-1 \tag{1.6е}$$

$$y_i, x_{ij}, q_k, z_{ki} \in \{0, 1\}, \qquad \forall\ i, j, k \tag{1.6ф}$$

где x_{ij}, $i \in I \setminus \{n\}$ и $j > i$ (односно z_{ki}, $k \in I \setminus \{n\}$ и $i > k$) узима вредност 1 ако је пакет j спакован на ниво i (односно ниво i смештен у складиште k), а 0 иначе. Ограничења $j > i$ и $i > k$ произилазе из претпоставки 1) и 2), које су наведене изнад. Једначине 1.6б и 1.6д намећу, респективно, да се сваки пакет спакује тачно једном, и да је сваки ниво смештен тачно једном у складиште. Једначине (1.6ц) и (1.6е) намећу, респективно, ограничење ширине нивоа и ограничење висине складишта.

Експериментални резултати су показали да је претходно приказан модел прилично користан у пракси. Његова директна употреба у комерцијалном ILP решавачу релативно брзо даје оптимална решења на инстанцама реалне величине.

1.6 Коефицијенти перформанси асимптотски и апсолутно најнеповољнијег случаја

Због тешког налажења оптималног решења проблема паковања, хеуристички алгоритми су од великог интереса. Особине алгоритма могу бити мерене учинком у најнеповољнијем и просечном случају, као што су учинили Simchi-Levi, Chen и Bramel у [64]. Анализа најнеповољнијег случаја показује колико је слаб хеуристички алгоритам када даје најгоре решење у поређењу са најбољим. Одступање најгорег решења од оптималног решења даје особине хеуристике. Међутим, хеуристика која даје слабо решење у најнеповољнијем случају, не мора да даје слабо решење у општем случају. Стога, анализа просечног случаја представља алтернативну методу која одређује особине хеуристичких метода. На пример, када је позната дистрибуција величина пакета, постоје два начина за одређивање особина просечног случаја. Први, аналитичком

методом, која је често врло тешка, и други, емпиријским тестовима који се изводе на неколико инстанци и копија.

Што се тиче коефицијента најнеповољнијег случаја, постоје два типа. У литератури се може наћи неколико дефиниција коефицијента асимптотски и апсолутно најнеповољнијег случаја. Оба мере раскорак између вредности решења нађеног алгоритмом и оптимума у најнеповољнијем случају.

Формално говорећи, дати су проблем *минимизације* P, инстанца проблема E, алгоритам $\tilde{A}$, вредност решења добијеног алгоритмом $\tilde{A}(E)$ и оптимална вредност $OPT(E)$. *Коефицијент перформанси асимптотски најнеповољнијег случаја* је најмањи позитиван број R^{∞} такав да за сваку инстанцу проблема важи:

$$\tilde{A}(E) \leq R^{\infty} \cdot OPT(E) + O(1),\ \forall\ E \in P. \tag{1.7}$$

Коефицијент перформанси апсолутно најнеповољнијег случаја је најмањи позитиван број ρ такав да за сваку инстанцу проблема важи:

$$\tilde{A}(E) \leq \rho \cdot OPT(E),\ \forall\ E \in P. \tag{1.8}$$

1.7 Горње границе

Хеуристички алгоритми за дводимензиони проблем паковања у складишта су подељени на *online* и *offline* алгоритме. У *online* верзији пакети се са листе у алгоритам додају један по један, а следећи пакети се додају чим се ови пакети неопозиво сместе. Дакле, алгоритам серијски обрађује улаз, при чему читав улаз није доступан од почетка. У *offline* верзији се претпоставља да алгоритам има потпуно знање о целокупном улазу.

Најпопуларнији *online* алгоритми су *first fit*, *best fit* и *next fit*, који су изведени од алгоритма за једнодимензиони случај. Пошто ће бити презентоване стратегије *offline* алгортитама засноване на *first fit*, *best fit* и *next fit* алгоритмима, они неће бити посебно представљани. Детаљан преглед *online* алгоритама дали су Csirik и Woeginger у [15].

Најпознатији *offline* хеуристички алгоритми у литератури су похлепног типа, и могу бити класификовани у две категорије:

- *једнофазни алгоритми*, који директно пакују пакете у коначна складишта;
- *двофазни алгоритми*, који започињу паковање пакета у једну траку ширине W и висине H, а затим у другој фази користе решења траке за паковање у коначна $W \times H$ складишта.

Пре него што буду приказани једнофазни и двофазни алгоритми, потребно је укратко представити алгоритме за паковање пакета на траку.

1.7.1 Паковање на траци

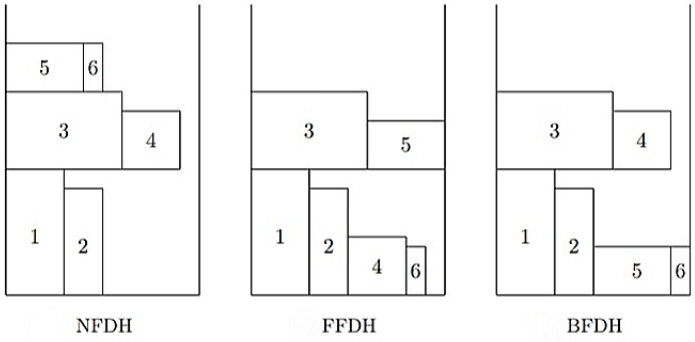
У дводимензионом проблему паковања на траци, потребно је све пакете спаковати на траку минималне висине. Три класичне методе, засноване на нивоима паковања, изведене су из добро познатих алгоритама за једнодимензиони случај. У сваком случају, пакети се иницијално сортирају по нерастућим висинама и пакују у одговарајуће низове.

- Frst-Fit Decreasing Height (FFDH) метода: трака је подељена на нивое, где сваки ниво користи целу ширину траке и ослања се на врх претходног нивоа (први ниво се ослања на дно траке). Пакети се у нивое пакују линеарно, сваки се доњом ивицом наслања на дно нивоа. Висина нивоа је висина највишег пакета који он садржи. Пакети се прво поређају нерастуће по висини, а затим се поступа на следећи начин: први пакет се смешта у први ниво, тако да буде лево поравнат.

После тога, пакети се додају редом, сваки пакет се поставља улево колико је могуће, у ниво са најнижим индексом у којем за њега има места, дуж његове доње ивице. Нови ниво се додаје на изнад текућег нивоа, који се налази на врху, када год пакет не може да стане ни у један од постојећих нивоа [13].

- Best-Fit Decreasing Height (BFDH) метода: поступак је сличан FFDH алгоритму, али се међу свим постојећим нивоима бира онај са најбољом попуњеношћу, тј. онај где би се смештањем пакета добила најмања неискоришћена ширина нивоа.
- Next-Fit Decreasing Height (NFDH) метода: поступак је, такође, сличан као код FFDH алгоритма, само што се за смештање пакета разматра само текући ниво. Уколико пакет не може да стане у њега, тај ниво се затвара и отвара се нови, изнад њега, који постаје текући ниво.

Наведене методе се користе као први корак у двофазним алгоритмима за дводимензиони проблем паковања у складишта. Приказане су на слици 1.3.



Слика 1.3: Три класичне методе за паковање нивоа.

Coffman, Garey, Johnson и Tarjan су у [14] анализирали NFDH и FFDH и одредили њихово понашање у асимптотски најнеповољнијем случају. Доказали су да, ако су висине нормализоване, тако да је $max_i\{h_i\} = 1$ онда важи

$$FFDH(E) \leq \frac{17}{10} \cdot OPT(E) + 1 \tag{1.9}$$

и

$$NFDH(E) \leq 2 \cdot OPT(E) + 1. \tag{1.10}$$

FFDH и NFDH се могу имплементирати тако да захтевају $O(n \log n)$ времена, коришћењем одговарајуће структуре података добијене из једнодимензионог случаја (Johnson [38]).

У наставку су дата кратка објашњења добро познатих алгоритама за паковање на траци.

- Split Fit (SF) алгоритам [14]: висине и ширине свих пакета су скалиране, тако да трака има јединичну ширину. Највећи цео број m је одређен тако да сви пакети у I имају ширину мању или једнаку $1/m$. Листа пакета, L, је подељена у две подлисте, L_{wide} и L_{narrow}, обе уређене по нерастућој висини, тако да се у L_{wide} налазе пакети чија је ширина већа од $1/(m+1)$, а у L_{narrow} пакети чија је ширина највише $1/(m+1)$. Пакети у листи L_{wide} се пакују уз помоћ алгоритма FFDH, и сви пакети постављени на истом нивоу се заједнички називају блок. Добијени блокови се затим преуређују тако да се блокови чија је укупна ширина већа од $(m+1)/(m+2)$ налазе на дну паковања, а затим следе блокови чија је укупна ширина највише $(m+1)/(m+2)$. Овај процес померања блокова формира правоугаону област R ширине $1/(m+2)$. Затим се пакују пакети у L_{narrow}. И поново се за паковање користи алгоритам FFDH. Паковање почиње у области R. Ако неки пакет не

стаје у R, паковање се наставља изнад L_{wide}. Доказано је да важи $SF(E) \leq (3/2) \cdot OPT(E) + 2$.

- Reverse-fit (RF) алгоритам [63]: нормализују се ширина траке и ширина пакета, тако да је трака јединичне ширине. RF прво слаже пакете ширине веће од 1/2. Преостали пакети се сортирају по нерастућој висини и пакују изнад висине H_0, коју достижу пакети са ширином већом од 1/2. RF затим понавља следећи процес. Пакети се пакују с лева надесно са доњом страном дуж линије висине H_0, све док више не буде простора. Затим се паковање пакета врши с десна налево и одозго на доле (што се још зове промена нивоа) све док укупна ширина не буде најмање 1/2. Затим се промењен ниво спушта доле, све док један од пакета не дотакне неки пакет испод. Спуштање се понавља. Доказано је да важи $RF(E) \leq 2 \cdot OPT(E)$;

- Стајнбергов алгоритам (енг. Steinberg's algorithm) [68]: процењује се горња граница висине H која је потребна да се сви улазни пакети спакују у правоугаоник ширине W и висине H. Затим се дефинише седам процедура са седам услова, тако да свака дели проблем на два мања и решава их рекурзивно. Показано је да сваки допустив проблем задовољава један од седам услова. Показано је, такође, да за Стајнбергов алгоритам важи: $Steinberg's\ algorithm(E) \leq 2 \cdot OPT(E)$.

- Слеаторов алгоритам (енг. Sleator's algorithm) [65]: нормализују се ширина траке и ширина пакета, тако да је трака јединичне ширине. Алгоритам се састоји из четири корака: (1) сви пакети ширине веће од 1/2 се пакују један на други на дну траке. Претпоставимо да је висина добијеног паковања h_0. Сва наредна паковања ће се десити изнад h_0. (2) Преостали пакети се уређују по нерастућој висини. Ниво од којег се пакети пакују (у редоследу по нерастућим висинама) с лева надесно је дуж линије на висини h_0. (3) Повлачи се вертикална линија која дели траку на два једнака дела (треба имати на уму да ова линија може пресећи пакет који је делимично смештен на десну страну). Извлаче се два хоризонтална линијска сегмента, дужине 1/2, један преко леве половине (назива се лева основна линија), и један преко десне половине (назива се десна основна линија) што је могуће ниже, тако да не прелазе преко било којег пакета. (4) Бира се лева или десна основна линија, у зависности која је нижа, и започиње се нови ниво паковања у одговарајућој половини, све док се не појави следећи пакет који је преширок. Затим се формира нова основна линија, па поново бира нижа и корак (4) се понавља док се не спакују сви пакети. Показано је да Слеаторов алгоритам има асимптотску границу једнаку 5/2.

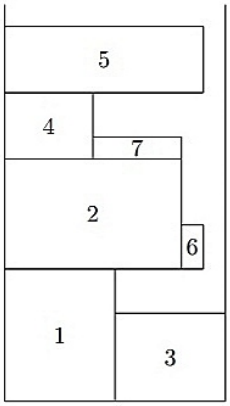
- Bottom-Left (BL) алгоритам: пакети се сортирају по нерастућој ширини, а текући пакет се пакује на најнижу могућу позицију, тако да буде лево поравнат. Baker, Coffman и Rivest су у [2] анализирали перформансе најнеповољнијег случаја BL алгоритма и показали следеће: (1) ако пакети нису уређени, BL може бити произвољно лош; (2) ако су пакети уређени по нерастућим ширинама тада је $BL(E) \leq 3 \cdot OPT(E)$.

1.7.2 Паковање у складишта: двофазне хеуристике

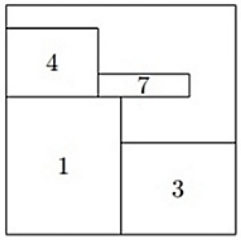
Двофазни алгоритам за коначан проблем паковања у складишта, који се назива Hybrid First Fit (HFF), представили су Chung и сарадници у [13]. У првој фази, паковање на траци се добија помоћу FFDH методе. Означимо са $H_1, H_2, ...$ висине нивоа, и нека важи $H_1 \geq H_2 \geq ....$ Решење се затим добија хеуристичким решавањем једнодимензионог проблема паковања у складиште (са величином пакета H_j и капацитетом складишта H) помоћу FFD алгоритма: иницијализује се складиште 1 да би се спаковао ниво 1, а затим се, за растуће $j = 2, 3, ...$, текући ниво j пакује у складиште са најнижим индексом, у које може да стане, ако оно постоји; ако ниједно складиште не може да прими j, иницијализује се ново. Пример је приказан на слици 1.4. Штавише, у истом раду, под претпоставком да су висине пакета нормализоване, представљен је следећи резултат:

$$HFF(E) \leq \frac{17}{8} OPT(E) + 5. \tag{1.11}$$

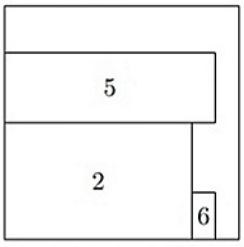
У најнеповољнијем случају је $HFF(E) = \frac{91}{45} \cdot (OPT(E) - 1)$. Обе фазе могу бити имплементиране тако да за извршење захтевају $O(n \log n)$ времена.



(а) HFF: фаза 1



(б) HFF: фаза 2, складиште број 1



(ц) HFF: фаза 2, складиште број 2

Слика 1.4: HFF алгоритам.

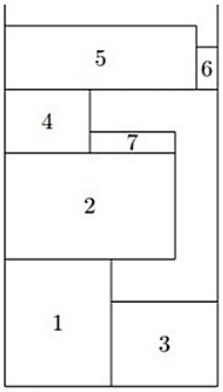
Berkey и Wang су у [6] представили двофазни алгоритам Finite Best-Strip (FBS), који представља варијанту HFF алгоритма. Прва фаза се изводи помоћу BFDH методе. У другој фази се једнодимензиони проблем паковања решава помоћу Best-Fit Decreasing алгоритма: текући ниво се пакује у складиште, у којем је, након паковања нивоа, неискоришћени вертикални простор минималан; ако ниједно складиште не може да прими тај ниво, отвара се ново. Пример је дат на слици 1.5.

Frenk и Galambos су у [22] размотрали варијанту HFF алгоритма где се у првој фази примењује NFDH метода, и једнодимензиони проблем паковања у складиште решава помоћу Next-Fit Decreasing алгоритма: текући ниво се пакује у у текуће складиште, ако може да стане, или се отвара ново складиште. Због next-fit стратегије, овај алгоритам је еквивалентан једнофазном алгоритму, где се пакет пакује на текући ниво текућег складишта, ако је могуће; иначе, иницијализује се нови ниво у оквиру текућег складишта (ако има довољно слободног вертикалног простора), или у новом складишту. Резултујући алгоритам назива се Hybrid Next-Fit (HNF). Перформансе асимптотски најнеповољнијег случаја посматране су као функција од $\max_i\{w_i\}$ и $\max_i\{h_i\}$. Претпостављајући да су ширине и висине нормализоване на 1, најлошије перформансе се добијају за $\max_i\{w_i\} > \frac{1}{2}$ и $\max_i\{h_i\} > \frac{1}{2}$ и важи:

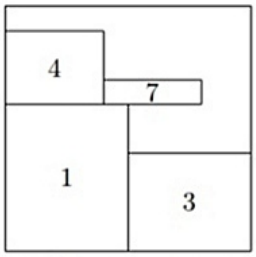
$$HNF(E) \leq 3,382... \cdot OPT(E) + 9 \tag{1.12}$$

где је 3,382... апроксимација за добру, али ирационалну границу. Претходно наведена три алгоритма се могу имплементирати тако да за извршење захтевају $O(n \log n)$ времена.

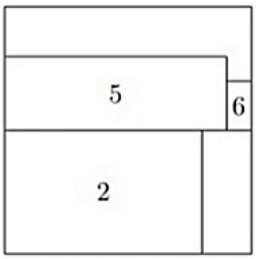
Наредна два алгоритма захтевају више времена у најнеповољнијем случају, али су



(а) FBS: фаза 1



(б) FBS: фаза 2, складиште број 1

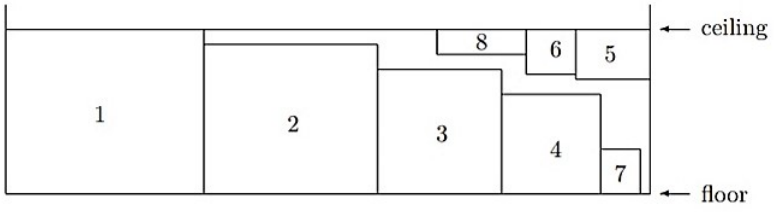


(ц) FBS: фаза 2, складиште број 2

Слика 1.5: FBS алгоритам.

у пракси веома брзи и ефикасни.

Lodi, Martello и Vigo су у [48] и [53] представили Floor-Ceiling (FC) приступ, који проширује начин паковања пакета на нивое. Хоризонтална линија, дефинисана горњом (односно доњом) ивицом највишег пакета спакованог на нивоу, назива се *плафон*, енг. ceiling (односно *под*, енг. floor) нивоа. Претходни алгоритми пакују пакете, с лева надесно, са њиховом доњом ивицом на поду нивоа. FC их додатно може паковати с десна налево, са горњом ивицом на плафону нивоа. Први пакет на плафону може бити само онај који не може бити спакован на поду испод. Пример оваквог паковања је дат на слици 1.6. У првој фази, текући пакет се пакује у складу са приоритетима: (1) на плафон (ако је горенаведени услов испуњен), у складу са best-fit стратегијом; (2) на под, у складу са best-fit стратегијом; (3) на под новог нивоа. У другој фази, нивои су спаковани у коначна складишта помоћу Best-Fit Decreasing алгоритма или помоћу егзактног алгоритма за решавање једнодимензионог проблема паковања у складишта заустављеног после унапред задатог броја итерација. Имплементација прве фазе, дата у [53], захтева $O(n^3)$ времена за извршење, док комплексност друге фазе зависи од изабраног алгоритма.



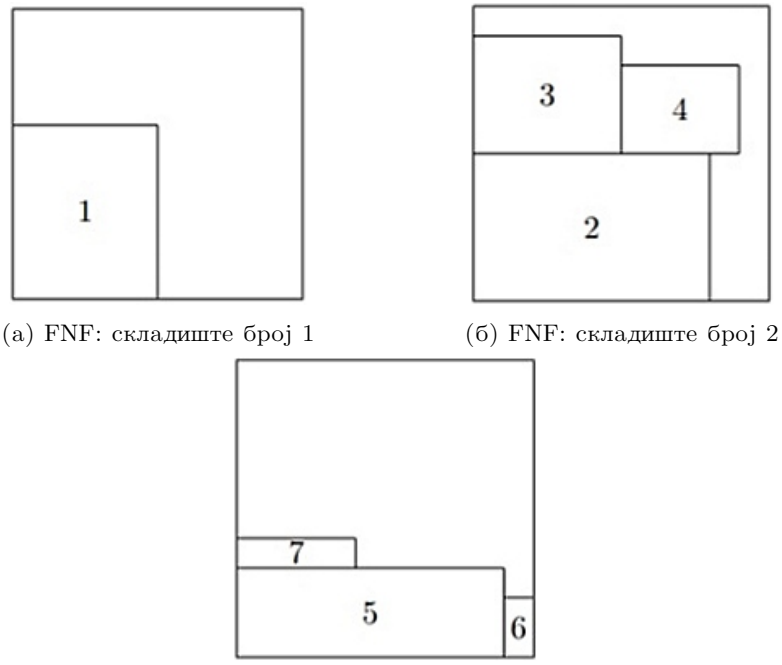
Слика 1.6: FC алгоритам.

Друга стратегија паковања по нивоима, заснована на тачном решењу индукованих потпроблема, усвојена је у проблему паковања ранца (енг. Knapsack Packing - KP), који су представили Lodi, Martello и Vigo у [48]. Прва фаза пакује један по један ниво на следећи начин. Први (највиши) неспакован пакет (означен са i^*) иницијализује ниво који се затим допуњује решавањем инстанце проблема паковања ранца над свим нераспоређеним пакетима, где је: (1) капацитет ранца $W - w_{i^*}$; (2) w_i ширина пакета i; (3) добит пакета i његова површина $w_i h_i$. Паковање у коначна складишта се врши као и код FC алгоритма. KP алгоритам (као и FC алгоритам изнад) може захтевати решавање NP тешких потпроблема, производећи неполиномијалну временску сложеност.

1.7.3 Паковање у складишта: једнофазне хеуристике са нивоима

Два једнофазна алгоритма представили су Berkey и Wang у [6].

Finite Next-Fit (FNF) алгоритам директно пакује пакете у коначна складишта на исти начин као и HNF алгоритам, представљен у претходном одељку, што се може видети у [6] и [22]. Пример је дат на слици 1.7.



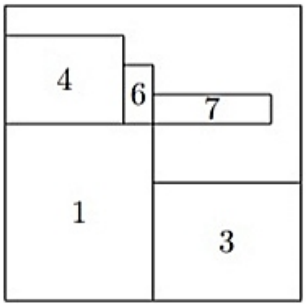
(а) FNF: складиште број 1

(б) FNF: складиште број 2

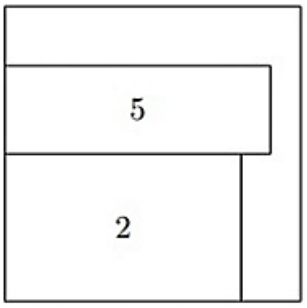
(ц) FNF: складиште број 3

Слика 1.7: FNF алгоритам.

Finite First-Fit (FFF) се користи уместо FFDH методе. Текући пакет се пакује на ниво са најнижим индексом, где може да стане; ако не постоји ниво на који се пакет може спаковати, формира се нови ниво у текућем складишту или новом складишту (ако у текућем нема довољно вертикалног простора). Пример је дат на слици 1.8.



(а) FFF: складиште број 1



(б) FFF: складиште број 2

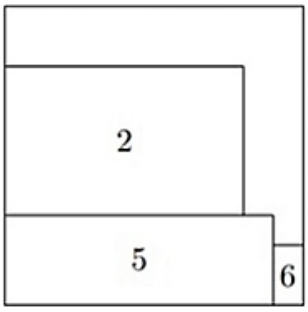
Слика 1.8: FFF алгоритам.

Оба алгоритма захтевају $O(n \log n)$ времена за извршење.

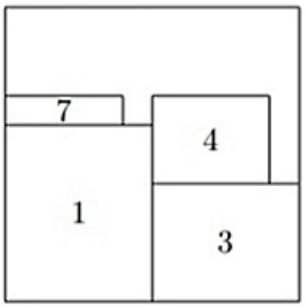
1.7.4 Паковање у складишта: једнофазне хеуристике без нивоа

Главна метода без нивоа позната је као Bottom-Left (BL) и састоји се у паковању тренутног пакета на најнижу могућу позицију, тако да он буде лево поравнат.

Berkey и Wang су у [6] представили BL приступ за случај коначног складишта. Њихов Finite Bottom-Left (FBL) алгоритам иницијално сортира пакете по нерастућим ширинама. Текући пакет се пакује на најнижу позицију било којег иницијализованог складишта, тако да буде лево поравнат; ако не постоји складиште које може примити пакет, иницијализује се ново. Пример FBL алгоритма је дат на слици 1.9.



(а) FBL: складиште број 1



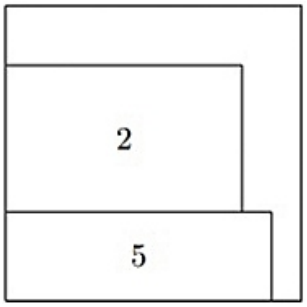
(б) FBL: складиште број 2

Слика 1.9: FBL алгоритам.

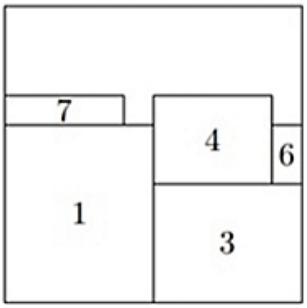
Рачунарску имплементацију BL алгоритма проучавао је Chazelle у [12], који је дао метод чије извођење захтева $O(n^2)$ времена. Исти приступ усвојили су Berkey и Wang у [6].

Berkey и Wang су у [6] представили и Next Bottom-left (NBL) алгоритам, који је сличан FBL алгоритму, али генерисање новог складишта значи да је сав слободан простор у претходним складиштима занемарен. Тако је у датом тренутку активно само једно складиште. Пример NBL алгоритма дат је на слици 1.10.

Lodi, Martello и Vigo су у [48] представили другачији приступ без нивоа, који се зове Alternate Directions (AD) алгоритам. Он иницијализује L складишта (L је доња граница оптималне вредности, о чему ће бити реч у одељку 1.8) паковањем подскупа пакета на њихово дно, у складу са best-fit decreasing стратегијом (пакети 1, 2, 3, 7 и 9 на слици 1.11, за $L = 2$). Преостали пакети се у складиште пакују један по један, у *траке*, алтернативно с лева надесно и с десна налево. Када пакет ни у једном смеру не може бити спакован у тренутно складиште, прелази се на следеће иницијализовано или ново, празно складиште (треће складиште на слици 1.11, у случају разматрања пакета 11). Сложеност алгоритма је $O(n^3)$.

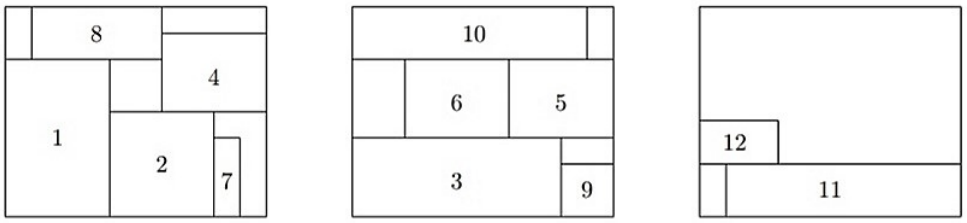


(а) NBL: складиште број 1



(б) NBL: складиште број 2

Слика 1.10: NBL алгоритам.



Слика 1.11: AD алгоритам.

1.7.5 Метахеуристике

Хеуристички приступ је користан код проблема који имају велику сложеност и за које методе које имају детерминистички приступ, као што је метода гранања и ограничавања, не могу да нађу решење у разумном времену. Иако су хеуристике брзе у генерисању решења за паковање, решење у великој мери зависи од улазног низа пакета.

Метахеуристички приступи се често користе за апроксимацију решења тешких проблема комбинаторне оптимизације. Метахеуристике, као што су еволутивни алгоритми (нпр. генетски алгоритам), алгоритам симулираног каљења и tabu search алгоритам се, такође, користе и за решавање проблема паковања у складиште.

Пре него што представимо специфичне алгоритме, у наставку се укратко бити представљени генетски алгоритам, алгоритам симулираног каљења и tabu search алгоритам.

Генетски алгоритам (GA)

Прве идеје о генетским алгоритмима изложене су у раду Holland-a, 1975. године [36], и јавиле су се у оквиру тзв. *теорије адаптивних система*, која проучава моделе ефикасног адаптивног понашања неких биолошких, специјално генетских, система. Овакви алгоритми су првобитно креирани да симулирају процес генетске еволуције једне популације јединки под дејством окружења и генетских оператора. У овом процесу је свака јединка окарактерисана хромозомом који представља њен генетски кôд. Оне јединке из популације које су у већој мери прилагођене окружењу, међусобно се даље репродукују (применом генетских оператора на њихове хромозоме) и тако се ствара нова генерација јединки, прилагођенија од претходне. Овај процес се понавља, при чему се из генерације у генерацију просечна прилагођеност чланова популације повећава.

GA формира почетну популацију јединки. Свака јединка представља могуће решење у претраживачком простору за дати проблем (простору свих решења). Најчешће

се користи бинарно кодирање, где се сваки хромозом (кôд), који карактерише јединку (решење), састоји од нула и јединица. Погодност хромозома сваке јединке популације рачуна се помоћу функције погодности. Функција погодности се дефинише за сваки проблем посебно. Случајним деловањем генетских оператора на случајно изабране јединке из текуће популације које имају већу погодност, добија се нова популација. Проверава се критеријум заустављања, и ако је испуњен, текуће решење се узима као апроксимација оптималног решења. Текуће решење се рачуна у свакој итерацији, односно за све јединке из новоформиране популације, и представља вредност за коју се остварује минимум или максимум функције циља датог проблема. Критеријуми заустављања GA алгоритма могу бити различити. Основна верзија GA алгоритма обично стаје када се досегне унапред задат, погодно изабран, број итерација.

Генетски оператори који се користе у оквиру GA алгоритма су оператори *селекције, укрштања и мутације.*

- *селекција:* бирају се оне јединке из тренутне популације које су прилагођеније окружењу, тј. чија је вредност функције погодности већа, да би се њиховом репродукцијом добиле нове јединке.
- *укрштање:* случајно се узајамно размењују делови хромозома две јединке и на тај начин добијају две нове јединке.
- *мутација:* мења се садржај хромозома неке јединке.

Симулирано каљење (SK)

SK алгоритам, који је прво развио Kirkpatrick у [43], заснован је на аналогији између налажења најбољег решења проблема комбинаторне оптимизације и процеса каљења неког растопљеног материјала до постизања кристализованог чврстог стања.

Процес претраге започиње од произвољног почетног решења и довољно велике почетне температуре. Температура је позитиван контролни параметар, чије су вредности задате низом $t_1, t_2, ...$ таквим да је $t_1 \geq t_2 \geq ...$ и $\lim_{n\to\infty} t_n = 0$. Овај низ се назива *схема хлађења* и он дефинише начин и брзину смањивања температуре током итерација. Затим се на случајан начин бира једно решење из околине текућег решења. Ако је то решење боље, онда оно постаје ново текуће решење. Ако је ново решење лошије од текућег, оно ипак може да постане текуће решење, али са одређеном вероватноћом. Лошије решење се прихвата са вероватноћом једнаком $e^{-\Delta C/t}$, где је ΔC разлика између вредности функције циља у тренутном решењу и вредности функције циља у решењу из околине тренутног решења, а t тренутна температура. Вероватноћа временом опада како се алгоритам извршава. Овакав приступ обезбеђује излазак из локалног оптимума. Пред крај извршавања алгоритма вероватноћа прихватања лошијег решења је јако мала, јер се сматра да је оптимум достигнут или се налази близу најбољег посећеног решења, па се избегава погоршање текућег решења.

Критеријум заустављања је унапред одређен. SK се може прекинути када температура падне испод неке минималне вредности, или када известан број итерација протекне без значајног смањења функције циља или без прихватања нових тачака претраживања. Понекад се унапред зна тачна или процењена минимална вредност функције циља, па се претраживање зауставља када се ова вредност достигне.

Табу претрага (TS)

TS претрага, коју је 1986. предложио Glover у [27], а затим и разрадио у [28], [29] и [30], представља метахеуристику засновану на локалном претраживању, која као своју најважнију компоненту користи меморију за памћење података о претходним фазама претраживања, како би превазишла заглављивање у локалном оптимуму.

Реч ,,табу" означава нешто што није друштвено прихватљиво, односно нешто што је забрањено и доноси ризик. Управо је то идеја TS претраге. Решења означена као

,,табу" се занемарују са циљем да се достигне што боље решење и изађе из локалног оптимума. Главну улогу за додељивање и мењање табу статуса има меморија. Меморија у којој се чувају информације о забрањеним потезима назива се табу листа.

Метода креће од једног решења, а затим користи методу локалне претраге све док не достигне локални оптимум. Када се достигне локални оптимум, решење које га представља се памти у табу листи и искључује из околине за претраживање у неколико наредних итерација. Потом се поново почиње са локалним претраживањем, али у тако модификованој околини. Решење се враћа у околину након истека његовог табу статуса, а нека друга решења постају забрањена.

Да би се смањила меморија потребна за смештање решења у табу листу често се памте само нека његова својства, тако да се у табу листи могу чувати само забрањени потези, тј. потези који доводе до забрањених решења. Могуће је користити и неколико табу листа, па се у том случају потез сматра забрањеним ако се налази у свим листама, а иначе је дозвољен. У случају да је дужина табу листе превелика, изражена је диверзификација претраживања, тј. ,,расејавања" у нове области. У случају да је дужина табу листе сувише мала, повећава се могућност кружења у околини неког решења. Да би се превазишли наведени недостаци ове методе, могуће је увести укидање табу статуса неком потезу. Један од најједноставнијих начина је коришћење табу листе променљиве дужине. Други, често коришћени приступ је тзв. *критеријум аспирације*, којим се укида табу статус потезу који води у довољно добру тачку. Метода табу претраге може се унапредити увођењем краткорочне, средњорочне и дугорочне меморије. Краткорочна меморија се користи за формирање табу листе и има улогу у спречавању кружења у околини неког решења. Средњорочна меморија обезбеђује интензификацију, тј. интензивно претраживање у неком региону, у коме се налази тренутно најбоље решење, док је сврха дугорочне меморије диверзификација, тј. преношење претраге у делове простора претраживања који нису истражени. И у средњорочној и у дугорочној меморији се памте својства решења, а затим се у процесу интензификације бирају решења која имају заједничка својства са тренутно најбољим решењем, док се у процесу диверзификације бирају решења код којих су та својства различита од тренутно најбољег решења.

Критеријум заустављања може да има разне облике. На пример, претрага се може зауставити ако је број узастопних итерација, у којима није било побољшања тренутно најбоље вредности функције циља, већи од задате вредности. Ако се унапред зна минимална вредност функције циља, тада се стаје чим се ова вредност достигне (детаљније о овим алгоритмима може се наћи у [16]).

Lodi, Martello и Vigo су у [52], [48] и [53] развили ефикасан TS алгоритам за 2BPP проблем и његове варијанте, укључујући могућност ротације пакета за 90° или додатан услов да се пакети могу добити из образаца насталих гиљотинским резовима. Главна карактеристика овог TS алгоритма је бирање схеме претраге и околине који су независни од проблема паковања који треба решити. Алгоритам се практично може користити за било коју варијанту 2BPP проблема променом специфичног детерминистичког алгоритма који се користи за процену потеза у претрази околине.

У односу на дато решење потези се модификују мењањем паковања подскупа пакета S, у покушају да се испразни специфично *циљно складиште*, које је изабрано међу онима која имају малу површину за паковање и релативно велики број пакета које треба спаковати. Подскуп S је дефинисан тако да укључује један пакет, j, из циљног складишта, и тренутни садржај k других складишта, а ново паковање се добија применом одговарајућег хеуристичког алгоритма над S. Ако потез пакује пакете из S у k (или мање) складишта, тј. пакет j бива уклоњен из циљног складишта, бира се нови пакет, у складу са тим се дефинише нови скуп S и изводи се нови потез. Иначе, S се мења бирањем другог скупа од k различитих складишта или избором другог пакета j из циљног складишта.

TS алгоритам, који су представили Lodi, Martello и Vigo у [52], [48] и [53], Iori, Martello и Monaci су у [37] искомбиновали са генетским алгоритмом и добили хибридни алгоритам за решавање 2SPP проблема, који се лако може прилагодити другим проблемима

паковања у две или више димензија.

Færø, Pisinger и Zachariasen су у [17] дали другачији метахеуристички приступ за 2BPP проблем. Њихов алгоритам локалне претраге почиње од допустивог решења и произвољно уклања нека складишта, додељујући њихове пакете другим складиштима. Ново решење је генерално недопустиво и доводи до оптимизационог проблема у којем је потребно минимизирати функцију циља која мери два пута преклопљену површину. Придружена околина се претражује померањем објеката, док се не нађе допустиво решење.

Boschetti и Mingozzi су у [7] и [8] представили нове доње границе и ефикасну хеуристику за 2BPP која:

1) додељује цену сваком пакету;

2) пакује пакете по нерастућим вредностима одговарајућих цена;

3) ажурира резултате коришћењем специфичних критеријума;

4) понавља кораке 2) и 3) док не пронађе оптимално решење или не достигне максималан број итерација.

Извршавање алгоритма се понавља за дати скуп различитих критеријума који су коришћени за ажурирање цена пакета.

Soke и Bingul су у [66] представили хибридни генетски алгоритам и алгоритам симулираног каљења за дводимензиони негиљотински правоугаони проблем паковања. У њиховом раду, GA и SK се користе одвојено, како би се добила пермутација за смештање малих пакета. Побољшани BL алгоритам се користи за смештање правоугаоних пакета. Приступи решавању, где су GA и SK комбиновани са унапређеним BL алгоритмом, су познати као хибридни GA и хибридни SK.

Досадашња истраживања о проблему паковања у складишта су углавном фокусирана на минимизацију непопуњеног простора. Међутим, у већини проблема паковања потребно је да буде постигнута минимизација непопуњеног простора и баланс потребних складишта. Liu, Tan, Huang, Goh и Ho [47] су дефинисали дводимензиони проблем паковања у складишта са више функција циља (енг. multiobjective two-dimensional bin packing model - MOBPP-2D). Једна функција циља је минимизација непопуњеног простора складишта, а друга балансирање пуњења складишта.

1.7.6 Апроксимативни алгоритми

Дуго постављано питање о могућности апроксимације 2BPP и 2SPP проблема је нашло одговор последњих година. Потпуну полиномијалну апроксимацију развили су Kenyon и Rémila у [42], која лако даје $2+\epsilon$ гаранцију за 2BPP проблем. Caprara, Lodi и Monaci су у [10] дали асимптотски потпуно полиномијалну апроксимациону схему (енг. Asymptotic Fully Polynomial Time Approximation Scheme - AFPTAS) за 2BPP проблем са рестрикцијом на нивое. Касније је Caprara у [9] представио алгоритам за генерални 2BPP проблем са $T_\infty+\epsilon$ гаранцијом за асимптотски најнеповољнији случај, где је $T_\infty = 1,691...$ добро позната гаранција за хармонијски 1BPP проблем ([45]). Овај проблем су даље унапредили Bansal, Caprara и Sviridenko [3], који су представили основу за унапређење претходног апроксимативног алгоритма и добили гаранцију асимптотске апроксимације произвољно блиску вредности $1,525...$ за паковање са и без ротације. Ово је тренутно најбољи асимптотски резултат. Коначно, Bansal и Sviridenko су у [4] показали да без могућности апроксимације APTAS не може постојати за 2BPP проблем.

Сви претходни резултати се баве могућношћу асимптотске апроксимације, тј. коефицијент апроксимације је близак наведеним вредностима за инстанце које укључују велики број пакета. О коефицијенту апсолутне апроксимације може се наћи у радовима [70], [67] и [35]. Harren и van Stee су у [34] извели апроксимативни алгоритам за 2BPP са коефицијентом апсолутне апроксимације једнаким 2, што је најбоље полиномијално време апроксимације за овај проблем.

1.8 Доње границе

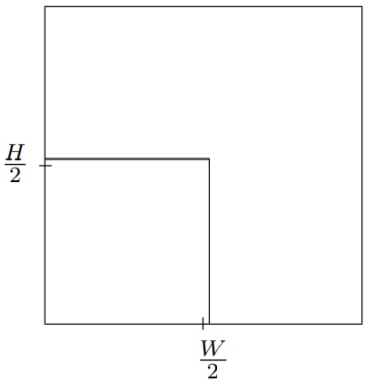
Добре доње границе оптималног решења су врло битне и за имплементацију егзактних приступа енумерације и за емпиријску процену апроксимације решења. Најједноставнија граница 2BPP проблема је *површинска граница*

$$L_0 = \left\lceil \frac{\sum_{i=1}^{n} w_i h_i}{WH} \right\rceil$$

која има линеарну временску сложеност. Martello и Vigo су [57] одредили апсолутно најнеповољнији случај понашања L_0:

$$L_0(E) \geq \frac{1}{4} \cdot OPT(E)$$

где $L_0(E)$ и $OPT(E)$ представљају вредност доње границе и оптималну вредност решења, респективно, за инстанцу E датог проблема. Граница је добра, што се може видети на слици 1.12. Резултат важи чак и ако је дозвољена ротација (за било који угао).



Слика 1.12: Најнеповољнији случај површинске границе.

Боља доња граница, у неполиномијалном времену, се може добити решавањем једнодимензионог проблема паковања, где су величине елемената $w_i h_i$ $(i = 1, ..., n)$, а капацитет складишта WH. Caprara и Monaci су у [3] показали да оптимално решење за такву 1BPP инстанцу даје валидну доњу границу за 2BPP проблем L_1, такву да важи $L_1(E) \geq \frac{1}{3} \cdot OPT(E)$.

У многим случајевима, апроксимација коју обезбеђују ове две границе може бити слаба или време рачунања може бити велико за ефикасну употребу у оквиру ефикасног алгоритма. Боље границе предложили су Martello и Vigo у [57]. За дати цео број q, такав да је $1 \leq q \leq W/2$, дефинисани су следећи скупови:

$$K_1 = \{i \in I : w_i > W - q\} \tag{1.13}$$

$$K_2 = \{i \in I : W - q \geq w_i > W/2\} \tag{1.14}$$

$$K_3 = \{i \in I : W/2 \geq w_i \geq q\} \tag{1.15}$$

Може се приметити да пакети из $K_1 \cup K_2$ не могу бити спаковани један до другог у складиште. Стога, доња граница L_1^W за подинстанцу одређену пакетима из $K_1 \cup K_2$ може бити добијена коришћењем било које доње границе за 1BPP инстанцу дефинисану елементима величине h_i $(i \in K_1 \cup K_2)$ и капацитетом складишта H.

Доња граница за целокупну инстанцу се сада добија разматрањем пакета из K_3, јер

се ниједан од њих не може спаковати до пакета из K_1:

$$L_2^W(q) = L_1^W + \max\left\{0, \left\lceil \frac{\sum_{i\in K_2\cup K_3} w_i h_i - (HL_1^W - \sum_{i\in K_1} h_i)W}{WH} \right\rceil\right\}. \tag{1.16}$$

Симетрична граница $L_2^H(q)$ се добија заменом ширина и висина. Уочавањем да обе границе важе за било које q, укупна доња граница је:

$$L_2 = \max\left(\max_{1\le q\le W/2}\{L_2^W(q)\}, \max_{1\le q\le H/2}\{L_2^H(q)\}\right). \tag{1.17}$$

У [57] је показано да за било коју инстанцу 2BPP проблема вредност добијена са границом L_2 није мања од оне која је добијена са L_0, и да је за рачунање L_2 потребно $O(n^2)$ времена. У истом раду, Martello и Vigo су представили и рачунски сложенију границу, која у неким случајевима надмашује L_2. За дате целе бројеве p и q, такве да је $1 \le p \le H/2$ и $1 \le q \le W/2$, дефинисани су следећи скупови:

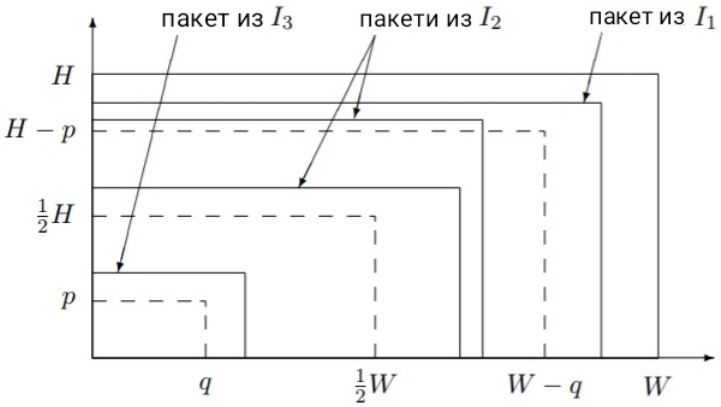
$$I_1 = \{i \in I : w_i > W - q \wedge h_i > H - p\} \tag{1.18}$$

$$I_2 = \{i \in I \setminus I_1 : w_i > W/2 \wedge h_i > H/2\} \tag{1.19}$$

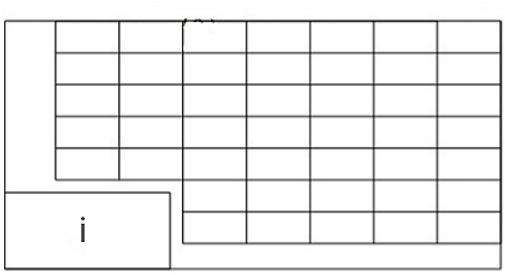
$$I_3 = \{i \in I : W/2 \ge w_i \ge q \wedge H/2 \ge h_i \ge p\} \tag{1.20}$$

(видети слику 1.13). Може се приметити следеће: (1) $I_1 \cup I_2$ не зависи од p и q; (2) не постоје два пакета из $I_1 \cup I_2$ која могу бити спакована један поред другог у исто складиште; (3) не постоји пакет из I_3 који стаје у исто складиште које садржи пакет из I_1. Добра доња граница се може добити додавањем минималног броја складишта, која су потребна за оне пакете из I_3 који не могу бити спаковани у складишта коришћена за пакете из I_2, на број $|I_1 \cup I_2|$. Таква граница може бити одређена разматрањем релаксиране инстанце где сваки пакет $j \in I_3$ има минималну величину, тј. $h_j = p$ и $w_j = q$. За дато складиште које садржи пакет i, максималан број $p \times q$ пакета који могу бити спаковани у то складиште је (видети слику 1.14):

$$m(i,p,q) = \left\lfloor \frac{H}{p} \right\rfloor \left\lfloor \frac{W - w_i}{q} \right\rfloor + \left\lfloor \frac{W}{q} \right\rfloor \left\lfloor \frac{H - h_i}{p} \right\rfloor - \left\lfloor \frac{H - h_i}{p} \right\rfloor \left\lfloor \frac{W - w_i}{q} \right\rfloor. \tag{1.21}$$



Слика 1.13: Пакети у I_1, I_2 и I_3.



Слика 1.14: Релаксирана инстанца са смањеним пакетима.

Стога, за свако p и q добра доња граница је

$$L_3(p,q) = |I_1 \cup I_2| + \max\left\{0, \left\lceil \frac{|I_3| - \sum_{i \in I_2} m(i,p,q)}{\left\lfloor \frac{H}{p} \right\rfloor \left\lfloor \frac{W}{q} \right\rfloor} \right\rceil \right\} \tag{1.22}$$

тако да је свеукупна граница

$$L_3 = \max_{1 \le q \le W/2, 1 \le p \le H/2} \{L_3(p,q)\}. \tag{1.23}$$

За рачунање доње границе L_3 је потребно $O(n^3)$ времена. Ниједна од граница L_2 и L_3 не доминира над другом.

Наведене границе су побољшали Boschetti и Mingozzi у [7] и [8], који су такође представили и неке доње границе за 2BPP проблем где је дозвољена ротација пакета за 90°.

Fekete и Schepers су у [18] и [20] представили општу технику ограничавања за проблем паковања у складиште и проблем паковања на траци, за једну и више димензија, засновану на *дуално допустивој функцији*. Функција $u : [0,1] \to [0,1]$ се назива дуално допустивом [56] ако за сваки коначан скуп S ненегативних реалних бројева важи:

$$\sum_{x \in S} x \le 1 \Rightarrow \sum_{x \in S} u(x) \le 1. \tag{1.24}$$

Инстанца 1BPP проблема се нормализује стављањем $h_i = h_i/H$, $(i = 1, ..., n)$, а затим и $H = 1$. За сваку допустиву функцију u, свака доња граница за трансформисану инстанцу, код које су величине пакета $u(h_1), ..., u(h_n)$, је добра доња граница за оригиналну инстанцу. У [18] је представљена класа дуално допустивих функција за 1BPP проблем, док је у [20] приступ проширен на паковање у две и више димензија. За d-димнзиони проблем паковања у складиште, скуп од d допустивих функција $\{u_1, ..., u_d\}$ се назива *конзервативна скала*. Тако је, за било коју конзервативну скалу $\mathcal{C} = \{u_1, u_2\}$, добра доња граница за 2BPP проблем

$$L(\mathcal{C}) = \sum_{i=1}^{n} u_1(w_i) u_2(h_i) \tag{1.25}$$

где су w_i и h_i по претпоставци нормализоване вредности, на начин који је приказан горе. За дати скуп $\mathcal{V}$ конзервативних скала, добра доња граница је

$$L^b = \max_{\mathcal{C} \in \mathcal{V}} L(\mathcal{C}). \tag{1.26}$$

Приступ који су дали Fekete и Schepers даље су истраживали Caprara и Monaci у [11]. Основна идеја је да било који пар дуално допустивих функција, са придруженим ширинама и висинама пакета, респективно, доводи до добре доње границе за 2BPP инстанцу. Проблем одређивања пара дуално допустивих функција који доводи до најбоље (највише) доње границе је дефинисан као раздвојен билинеарни програм. Рачунарски експерименти у [10] су показали да је за највећи број инстанци у литератури добијена вредност доње границе еквивалентна оној која је добијена непрекидном релаксацијом формулације (1.4а)-(1.4ц) покривања скупа, док су захтевана времена рачунања за ред величине мања. Детаљније се може наћи у [51].

1.9 Егзактно решавање

Martello и Vigo су у [57] за егзактно решавање 2BPP проблема представили алгоритам гранања и ограничавања који користи доње границе, такође разматране у истом раду, а наведене у претходном одељку.

Martello, Monaci и Vigo су у [58] за егзактно решавање 2SPP проблема представили алгоритам гранања и ограничавања, где су доње границе добијене кроз релаксацију која сваки $w_i \times h_i$ пакет мења са једнодимензионим пакетом јединичне висине h_i и ширине w_i, чиме се добија инстанца 1BPP проблема.

Fekete, Schepers и van der Veen су у [21] развили енумеративни приступ езактном решавању проблема паковања скупа пакета у једно складиште. Такав приступ је заснован на моделу који је представљен у [19], а о којем је било речи у одељку 1.5, и који може бити коришћен као алтернативни приступ у решавању 2BPP и 2SPP проблема.

У новије време Pisinger и Sigurd су у [61] за егзактно решавање (1.4а)-(1.4ц) имплементирали алгоритам гранања и цена (енг. branch-and-price).

Глава 2

Дводимензиони проблем паковања у складишта: 2BP|O|G случај

2.1 Увод

У наставку је представљен проблем паковања скупа дводимензионих пакета I у минималан број идентичних дводимензионих складишта, преузет из [55]. Са w_i и h_i су обележени ширина и висина сваког пакета, респективно. Слично, са W и H су обележени ширина и висина сваког складишта. Без губљења општости, претпостављено је да је $W = H = 1$. Захтева се да се спаковани пакети не преклапају, да се не могу ротирати и да њихове ивице морају бити паралелне ивицама складишта. Ова варијанта проблема означава се са 2BP|O|F. Додатно, разматрана је варијанта проблема у којој се сваки пакет добија низом резова од ивице до ивице, тј. свако складиште треба да одговара *гиљотинабилном обрасцу* (енг. guillotinable pattern). Ова варијанта проблема означава се са 2BP|O|G. Њен развој подстакнут је технолошким ограничењима, јер се у новије време за сечење пакета користе аутоматске машине, али може довести до мање искоришћености складишта.

2.1.1 Циљеви

Ова глава има три главна доприноса:

P_0 Пружити (једноставну) математичку карактеризацију негиљотинабилног обрасца.

Решење овог проблема је класификација различитих образаца који одговарају негиљотинабилним решењима, и њихово представљање преко математичког алата.

P_1 За дати негиљотинабилан образац, P, помоћу којег се дати скуп пакета пакује у јединствено складиште, одредити минималну површину пакета, $MA(P)$, коју треба уклонити како би се добио гиљотинабилан образац. Формално,

$$MA(P) = \sup \min \left\{ \sum_{i \in S} w_i h_i : P \setminus S \text{ је гиљотинабилан образац} \right\}. \tag{2.1}$$

P_2 За дату инстанцу N (тј. за дати скуп пакета), нека $opt_{2BP|O|F}(N)$ и $opt_{2BP|O|G}(N)$ означавају оптималну вредност решења проблема 2BP|O|F и 2BP|O|G, респективно. Одредити вредност асимптотске *цене гиљотинабилности* дефинисане као

$$PoG = \lim_{z \to \infty} \sup \left\{ \frac{opt_{2BP|O|G}(N)}{opt_{2BP|O|F}(N)} : opt_{2BP|O|F}(N) \geq z \right\}. \tag{2.2}$$

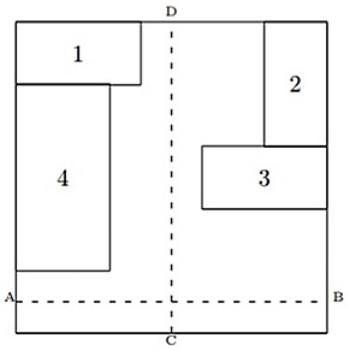
У наредном одељку су дате дефиниције важних појмова који ће бити коришћени у наставку. У току анализирања се разматра једно по једно складиште. У одељку 2.1.3 је представљен алгоритам конвексификације, за који се претпоставља да се примењује на скуп пакета спакованих у тренутном складишту. Након његове примене, примењује се алгоритам за добијање сепарабилног обрасца, описан у одељку 2.1.4. Затим је описан најмањи несепарабилни образац, у одељку 2.2, и образац под називом блокиран прстен (енг. Blocked Ring), у одељку 2.3. Ово доводи до комплетне карактеризације негиљотинабилних образаца, што је описано у одељку 2.4, и анализе проблема P_1 и P_2, у одељку 2.5.

2.1.2 Дефиниције

Нека P означава скуп пакета спакованих у тренутно складиште. Затим, нека су x_i и y_i координате доњег левог угла у који се пакује пакет i. Уз злоупотребу записа, са P ће бити означен и образац паковања у складиште. Разматра се једно по једно складиште.

Дефиниција 1 (Гиљотински рез)**.** *Гиљотински рез за образац P је рез од ивице до ивице који не пресеца ниједан пакет у P.*

Гиљотински рез може бити вертикалан или хоризонталан. Пример је дат на слици 2.1 (AB линија одређује хоризонталан рез, а CD линија одређује вертикалан рез). Сваки гиљотински рез који се не поклапа са ивицама складишта дели образац на два дела, тј. на два подобрасца. У обзир се узимају само *правилни гиљотински резови*, тј. они гиљотински резови код којих оба подобрасца укључују најмање један пакет. Гиљотински резови који ниједан пакет не одвајају од осталих, попут линије AB на слици 2.1, се занемарују.



Слика 2.1: Пример гиљотинског реза.

Дефиниција 2 (Сепарабилан образац)**.** *За образац P кажемо да је сепарабилан ако за њега постоји правилан гиљотински рез.*

За образац P који не садржи правилан гиљотински рез кажемо да је несепарабилан. Несепарабилан образац P се назива *минималним* ако уклањање било ког пакета $u \in P$ доводи до сепарабилног обрасца.

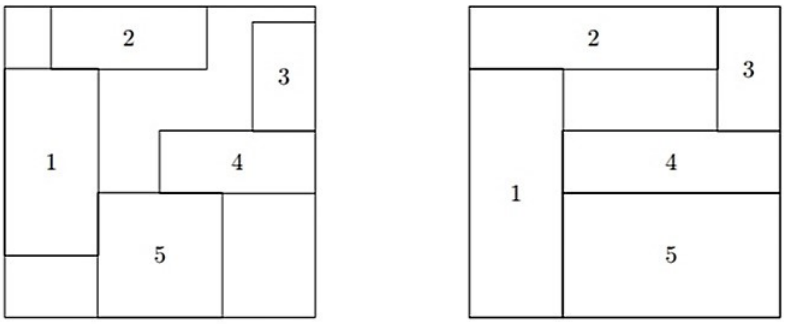
Дефиниција 3 (Гиљотинабилан образац)**.** *За образац P кажемо да је гиљотинабилан ако су сви индуковани подобрасци P', где је $|P'| \geq 2$, сепарабилни.*

2.1.3 Алгоритам конвексификације

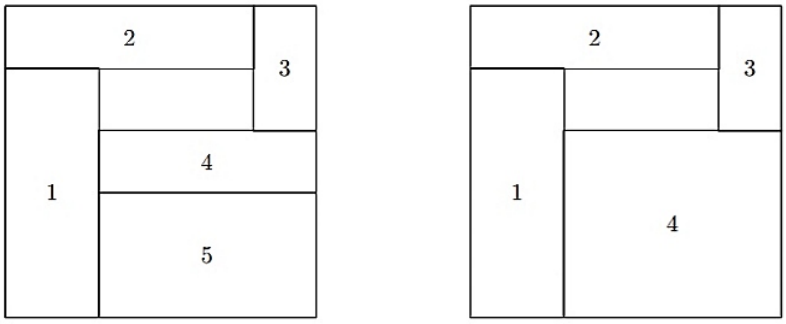
Алгоритам конвексификације састоји се у следећем: сваком пакету је додељена ознака (1. корак), и они се разматрају по редоследу по којем су означени, при чему се тренутни пакет продужава хоризонтално и вертикално док не додирне неки други пакет (2. корак, слика 2.2). Затим се спајају пакети који се могу спојити у већи правоугаоник (4. и 5. корак, слика 2.3). На крају се, у 7. кораку, дефинишу *вештачки пакети* за сваки слободан простор у обрасцу (слика 2.4).

Алгоритам конвексификације

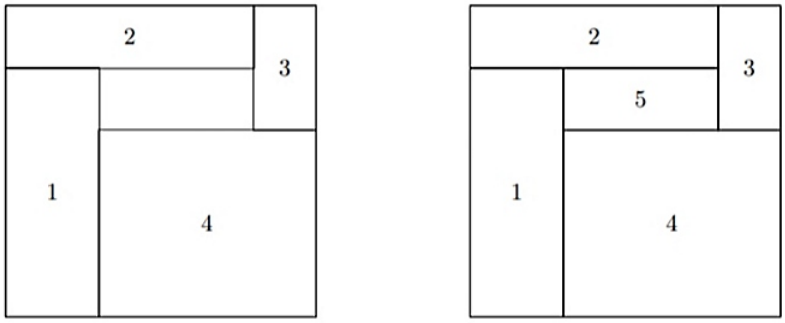
1: доделити ознаку сваком пакету;
2: продужити сваки пакет хоризонтално и вертикално све док не додирне неки други пакет;
3: **repeat**
4: спојити два предмета који се додирују вертикално, имају исту висину и постављени су на истој y-координати;
5: спојити два предмета који се додирују хоризонтално, имају исту ширину и по стављени су на истој x-координати;
6: **until** док је спајање могуће.
7: дефинисати *вештачке* пакете за сваки слободан простор у обрасцу



Слика 2.2: Алгоритам конвексификације: 2. корак.



Слика 2.3: Алгоритам конвексификације: 5. корак.



Слика 2.4: Алгоритам конвексификације: 7. корак.

Након извршења алгоритма на датом обрасцу P, може се дефинисати нови образац $\overline{P}$, такав да је укупна површина у $\overline{P}$ једнака површини складишта. Треба напоменути да сваком пакету $u \in P$ одговара неки пакет $\overline{u} \in \overline{P}$, док сваки пакет $\overline{u} \in \overline{P}$ може водити порекло од више различитих пакета из P.

Наредна теорема говори о томе да алгоритам конвексификације не може дати сепарабилан образац $\overline{P}$, ако P није сепарабилан.

Теорема 1. *[60] Нека је P несепарабилан образац и $\overline{P}$ образац добијен извршавањем алгоритма конвексификације на обрасцу P. Тада је $\overline{P}$ несепарабилан.*

Доказ. Означимо са (w_u, h_u) и (x_u, y_u) димензије и координате паковања за сваки пакет $u \in P$, респективно. Слично, нека $(\overline{w}_{\overline{u}}, \overline{h}_{\overline{u}})$ и $(\overline{x}_{\overline{u}}, \overline{y}_{\overline{u}})$ означавају димензије и координате паковања за сваки пакет $\overline{u} \in \overline{P}$.

Посматрајмо пакет $\overline{u} \in \overline{P}$ и нека је u било који пакет из P од којег пакет $\overline{u}$ потиче. Како алгоритам конвексификације не може смањити величину пакета, имамо

$$\overline{x}_{\overline{u}} \leq x_u,\ x_u + w_u \leq \overline{x}_{\overline{u}} + \overline{w}_{\overline{u}},\ \overline{y}_{\overline{u}} \leq y_u,\ y_u + h_u \leq \overline{y}_{\overline{u}} + \overline{h}_{\overline{u}}. \tag{2.3}$$

Претпоставимо супротно, да је образац $\overline{P}$ сепарабилан, тј. да постоје два пакета $\overline{u}$ и $\overline{v}$ која могу бити раздвојена гиљотинским резом. Без губљења општости, претпоставимо да је тај рез хоризонталан, на висини $\overline{y}$, и да је пакет $\overline{u}$ спакован испод $\overline{v}$, тј.

$$\overline{y}_{\overline{u}} + \overline{h}_{\overline{u}} \leq \overline{y},\ \text{и}\ \overline{y} \leq \overline{y}_{\overline{v}}. \tag{2.4}$$

Означимо са u (односно v) било који пакет у P који одговара пакету $\overline{u}$ (односно $\overline{v}$). Из (2.3) следи да хоризонтални рез $\overline{y}$ раздваја u и v у оригиналном обрасцу P; међутим, како P није сепарабилан, овај рез пресеца неки други пакет k, тј. $\exists k \in I$ такав да је

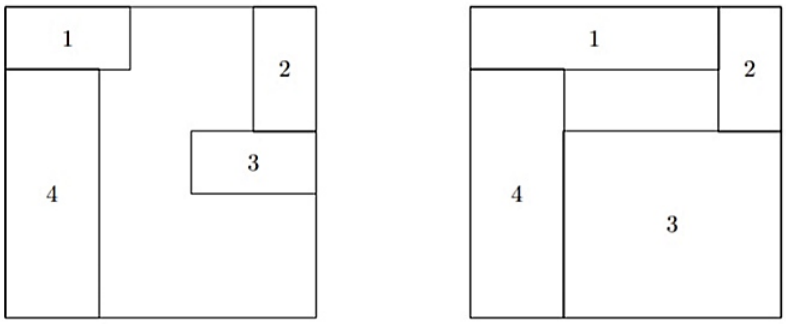
$$y_k < \overline{y} < y_k + h_k. \tag{2.5}$$

Означимо са $\overline{k}$ пакет из $\overline{P}$ који одговара пакету k из P. Комбинујући (2.3) и (2.5) имамо

$$\overline{y}_{\overline{k}} < \overline{y} < \overline{y}_{\overline{k}} + \overline{h}_{\overline{k}} \tag{2.6}$$

што значи да хоризонтални рез $\overline{y}$ пресеца пакет $\overline{k}$, што је у контрадикцији са претпоставком да је образац $\overline{P}$ сепарабилан. Дакле, $\overline{P}$ је несепарабилан. □

Треба напоменути да се применом алгоритма конвексификације на сепарабилан образац P може добити несепарабилан образац $\overline{P}$. Пример је приказан на слици 2.5.

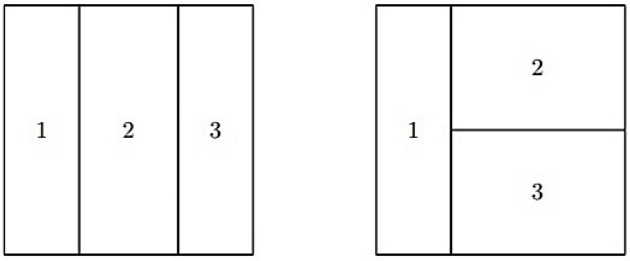


Слика 2.5: Примена алгоритма конвексификације на сепарабилан образац која доводи до несепарабилног обрасца.

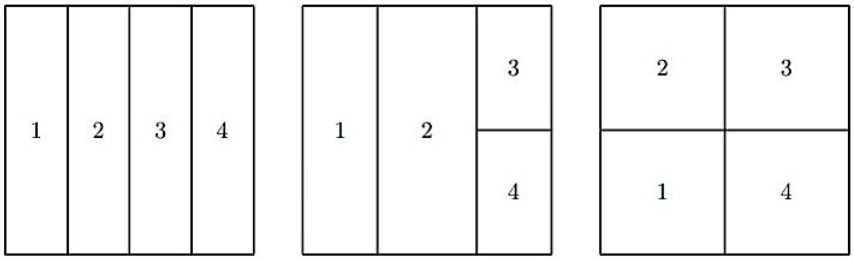
Напомена 1: Извршавање алгоритма конвексификације на обрасцу P не производи негиљотинабилност ако су редови обрасца P несепарабилни (видети дефиницију 7).

Теорема 2. *[60] Нека је $\overline{P}$ образац добијен алгоритмом конвексификације. Тада је $|\overline{P}| = 1$ или $|\overline{P}| \geq 5$.*

Доказ. Може се приметити да алгоритам конвексификације даје образац $\overline{P}$ који у потпуности пуни складиште. Јасно је да ако $\overline{P}$ садржи два пакета, онда ће они бити спојени у један пакет (4. и 5. корак алгоритма). Слично, сви обрасци са три пакета итеративним поступком могу бити сведени на један пакет. Слике 2.6 и 2.7 показују све могуће обрасце са 3 и 4 пакета, респективно, без узимања у обзир могућност ротације за 90°. □



Слика 2.6: Обрасци са 3 пакета.



Слика 2.7: Обрасци са 4 пакета.

Последица 1. *Нека је $\overline{P}$ несепарабилан образац који настаје као резултат примене алгоритма конвексификације. Тада је $|\overline{P}| \geq 5$.*

2.1.4 Алгоритам за добијање сепарабилног обрасца

Алгоритам као улаз узима допустиво решење за 2BP|O|F и дефинише допустиво решење за 2BP|O|G, разматрајући једно по једно складиште. За сваки такав образац складишта, извршава се алгоритам конвексификације из одељка 2.1.3, чиме се производи нови образац, рецимо $\overline{P}$. Затим се извршава унутрашњи алгоритам паковања, како би се добио сепарабилан образац. То је или оригиналан образац $\overline{P}$ (у случају да он садржи само један пакет или је сепарабилан) или се добија од $\overline{P}$, уклањањем неких пакета, као што је описано у 10. кораку.

Алгоритам паковања (P)

1: **if** $|\overline{P}| = 1$ **then**
2: **return** $\overline{P}$
3: **end if**
4: **if** $\overline{P}$ је сепарабилан **then**
5: дефинисати два подобрасца, P_1 и P_2, коришћењем одговарајућег гиљотинског реза;
6: $\overline{P}_1$ =Алгоритам паковања(P_1);
7: $\overline{P}_2$ =Алгоритам паковања(P_2);
8: **return** $\overline{P}_1 \cup \overline{P}_2$
9: **else**
10: уклонити неке пакете из $\overline{P}$ и дефинисати сепарабилни образац $\overline{Q}$;
11: **return** $\overline{Q}$;
12: **end if**

Свакако, теорема 1 осигурава да је било који скуп пакета $Q \subset P$, који производи сепарабилан образац $\overline{Q}$ применом алгоритма конвексификације, сепарабилан.

2.2 Најмањи несепарабилан образац

У овом одељку проучавамо структуру несепарабилног обрасца минималне величине.

2.2.1 Редови и пресеци

Дефиниција 4 (Хоризонталан ред)**.** *Хоризонталан ред је ланац хоризонталних ивица (различитих од ивица складишта) који образују пакети поређани један поред другог на истој висини.*

Дефиниција 5 (Вертикалан ред)**.** *Вертикалан ред је ланац вертикалних ивица (различитих од ивица складишта) који образују пакети поређани један до другог на истој ширини.*

Дефиниција 6 (Пресек)**.** *Пресек је тачка која се добија од једног хоризонталног и једног вертикалног реда који се укрштају.*

Дефиниција 7 (Сепарабилни редови)**.** *Редови су сепарабилни ако постоји рез од ивице до ивице који их раздваја без пресецања ниједног од њих, осим ако овај рез није дуж постојећег пресека.*

У наставку је са x и y означен број редова и пресека, респективно; додатно, са x_h и x_v је означен број хоризонталних и вертикалних редова, респективно ($x = x_h + x_v$).

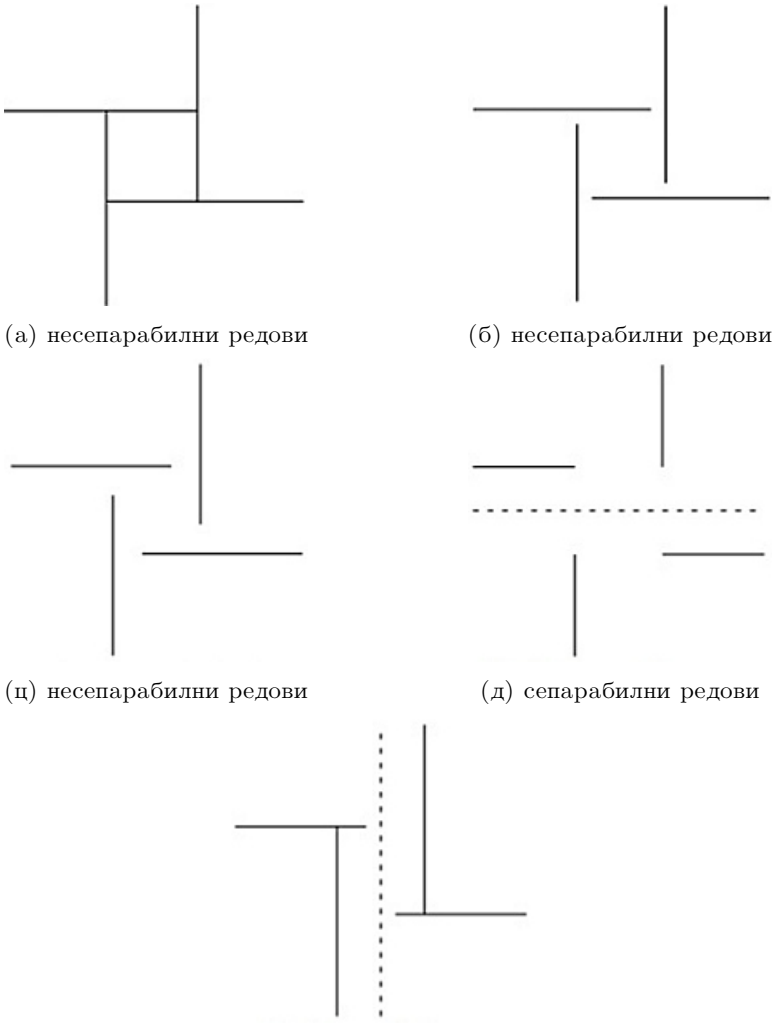
Лема 1. *[60] За сваки образац P је $y \leq x^2/4$.*

Доказ. Сваки пресек укључује један хоризонталан и један вертикалан рез, па је $y \leq x_v x_h$. Да би се доказала ова тврдња, довољно је уочити да се оптимална вредност следећег проблема

$$\max\{y : y = x_v x_h;\ x_v + x_h = x, x_h, x_v \geq 0,\ \text{целобројни}\}$$

постиже за $x_h = x_v = x/2$ (под претпоставком да је x парно) и има вредност $y = x^2/4$. □

Лема 2. *[60] За сваки образац P је $|P| = (x + 1) + y$.*



(а) несепарабилни редови (б) несепарабилни редови

(ц) несепарабилни редови (д) сепарабилни редови

(е) сепарабилни редови

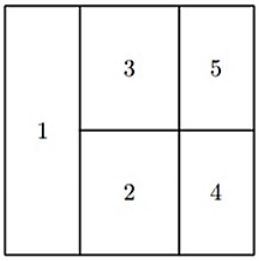
Слика 2.8: Сепарабилни и несепарабилни редови.

Доказ. Свако повећање x за један, при чему се не повећава y, додаје један пакет. Очигледно, свако повећање броја редова x може да произведе k нових пресека и сваки пресек је повезан са 4 пакета, два су већ ту, трећи је додат повећањем x или претходним пресеком, и четврти је нов пакет. □

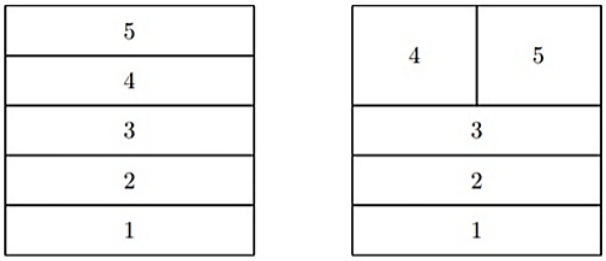
Нека је образац P добијен кроз алгоритам конвексификације. По последици 1 за сваки несепарабилан образац P мора да важи $|P| \geq 5$. Наредна теорема показује да је минимална величина несепарабилног обрасца једнака 5.

Теорема 3. *[60] Нека је P образац генерисан алгоритмом конвексификације. Ако је $|P| = 5$ онда је P несепарабилан.*

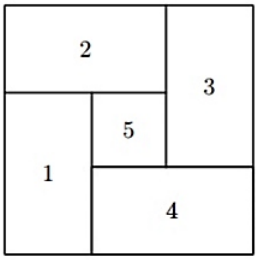
Доказ. Из лема 1 и 2, $|P| = 5$ произилази за $x = 3$ и $y = 1$ или $x = 4$ и $y = 0$. Први случај је приказан на слици 2.9 и одговара сепарабилном обрасцу који се своди на један пакет. Што се тиче другог, можемо имати (1) $x_h = 4$ и $x_v = 0$ или (2) $x_h = 3$ и $x_v = 1$ или (3) $x_h = x_v = 2$, где су прва два случаја (приказана на слици 2.10) сепарабилни обрасци који се своде на један пакет. Једини преостао случај је образац са $x_h = x_v = 2$ и $y = 0$ (приказан на слици 2.11) који одговара несепарабилном обрасцу. □



Слика 2.9: Образац са 5 пакета, $x = 3$ и $y = 1$.



Слика 2.10: Обрасци са 5 пакета: $x_h = 4$, $x_v = 0$ и $y = 0$; $x_h = 3$, $x_v = 1$ и $y = 0$.



Слика 2.11: Једноставан блокирани прстен.

Из теореме 3 произилази следећа дефиниција.

Дефиниција 8 (Једноставан блокиран прстен)**.** *Несепарабилан образац P се назива једноставним блокираним прстеном ако је $|P| = 5$ и $x_h = x_v = 2$ и $y = 0$.*

Из претходне дискусије јасно је да једноставан блокиран прстен садржи 5 пакета, где је један пакет централно позициониран и додирује ивице осталих пакета, али не додирује ивице складишта. Уклањање тог пакета не производи сепарабилан образац, док уклањање било којег од преостала четири пакета доводи до сепарабилног обрасца. Дакле, једноставан блокиран прстен је најједноставнија структура која може довести до несепарабилности у датом обрасцу P (под претпоставком да је P генерисан коришћењем алгоритма конвексификације). Тако да уклањање било којег пакета из ове конфигурације доводи или до истог обрасца (ако је уклоњен централни пакет и поново примењен алгоритам конвексификације) или до сепарабилног обрасца (у случају да је уклоњен један од преостала 4 пакета).

2.3 Блокиран прстен

Дефиниција 9. *Несепарабилан образац је блокиран прстен (енг.* Blocked Ring - BR*) ако*

- $|P| \geq 5$, *и*
- *постоји комбинација два хоризонтална и два вертикална несепарабилна реда таква да уклањање свих пакета који нису изабрани овим редовима и примена алгоритма конвексификације доводи до једноставног блокираног прстена.*

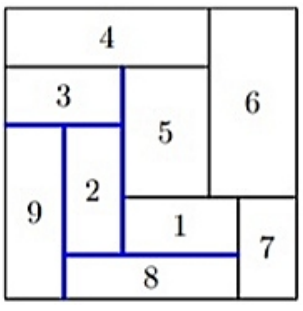
2.3.1 Откривање блокираног прстена

Наивни алгоритам за откривање да ли је образац блокиран прстен или не је $O(n^4)$ енумерација свих четворки пакета, која проверава (1) да ли они одређују два хоризонтална и два вертикална несепарабилна реда, и (2) уклања све преостале пакете и примењује алгоритам конвексификације који доводи до једноставног блокираног прстена.

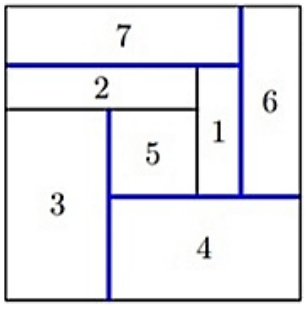
Ефикаснији алгоритам за проверу да ли је образац блокирани прстен, преузет из [55], састоји се у следећем: Уз напомену да је број комбинација које треба проверити највише $(x_h!)/[(x_h-2)!\cdot 2!]\cdot(x_v!)/[(x_v-2)!\cdot 2!]$, тј. $O(n^4)$, разматра се по једна комбинација несепарабилних редова $x_h = x_v = 2$. За сваку комбинацију примењују се следећи кораци:

1. Идентификовати пакете обухваћене изабраним редовима. Ти пакети су дефинисани на следећи начин. Почевши од леве ивице складишта померати се у десно. Када се дође до првог изабраног вертикалног реда, бирају се сви пакете смештени са његове леве стране, чија је ивица комплетно дефинисана тим редом. Ако таквих пакета има више од једног, задржава се онај који је пресечен продужетком једног од два изабрана хоризонтална реда. Затим се померање врши десно до другог вертикалног реда. Поступак је на исти, само што се сада гледају пакети са његове десне стране. За хоризонталне редове примењује се иста процедура, али у овом случају се креће са дна складишта. За први хоризонталан ред задржава се један пакет испод њега, а за други пакет изнад њега.
2. Уклонити из складишта све пакете који нису обухваћени хоризонталним и вертикалним редовима.
3. Применити алгоритам конвексификације.
4. Ако је добијен једноставан блокиран прстен, полазни образац је блокиран прстен.

Кораци за откривање блокираног прстена приказани су на примерима датим на сликама 2.12-2.20.

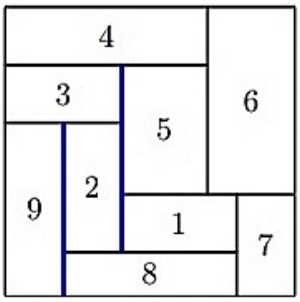


(а) Пр. 1: изабрани редови

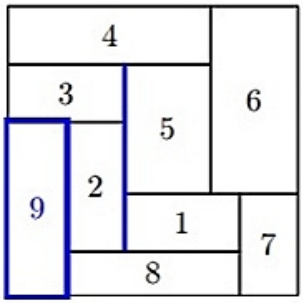


(б) Пр. 2: изабрани редови

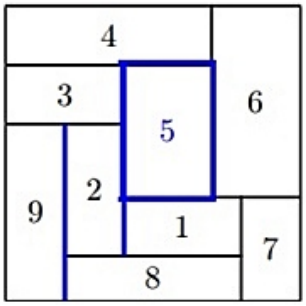
Слика 2.12: Два примера на које примењујемо алгоритам откривања блокираног прстена.



(а) Изабрани вертикални редови

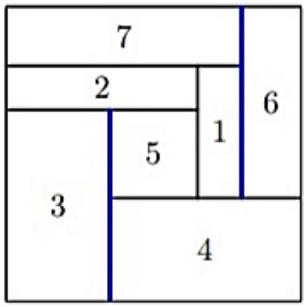


(б) Задржан пакет 9

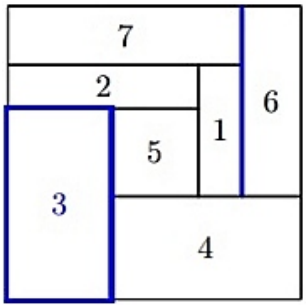


(ц) Задржан пакет 5

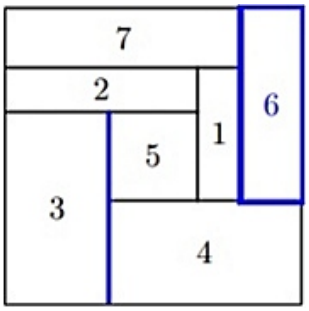
Слика 2.13: Корак 1: Пакети обухваћени изабраним вертикалним редовима (пр. 1)



(а) Изабрани вертикални редови

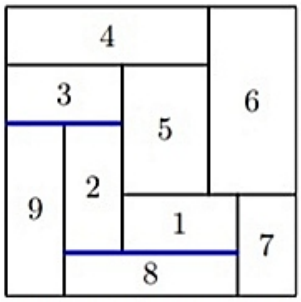


(б) Задржан пакет 3

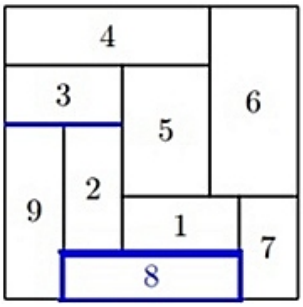


(ц) Задржан пакет 6

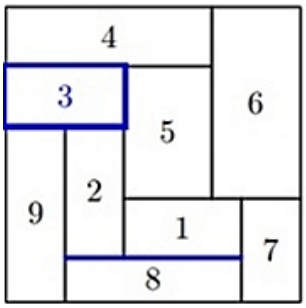
Слика 2.14: Корак 1: Пакети обухваћени изабраним вертикалним редовима (пр. 2).



(а) Изабрани хоризонтални редови

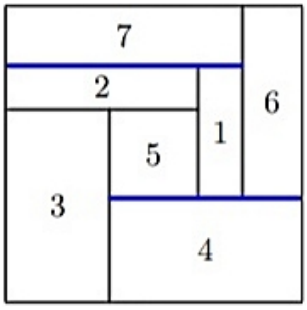


(б) Задржан пакет 8

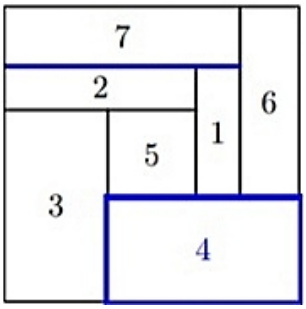


(ц) Задржан пакет 3

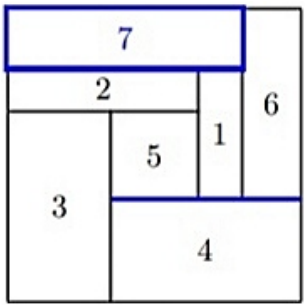
Слика 2.15: Корак 1: Пакети обухваћени изабраним хоризонталним редовима (пр. 1)



(а) Изабрани хоризонтални редови

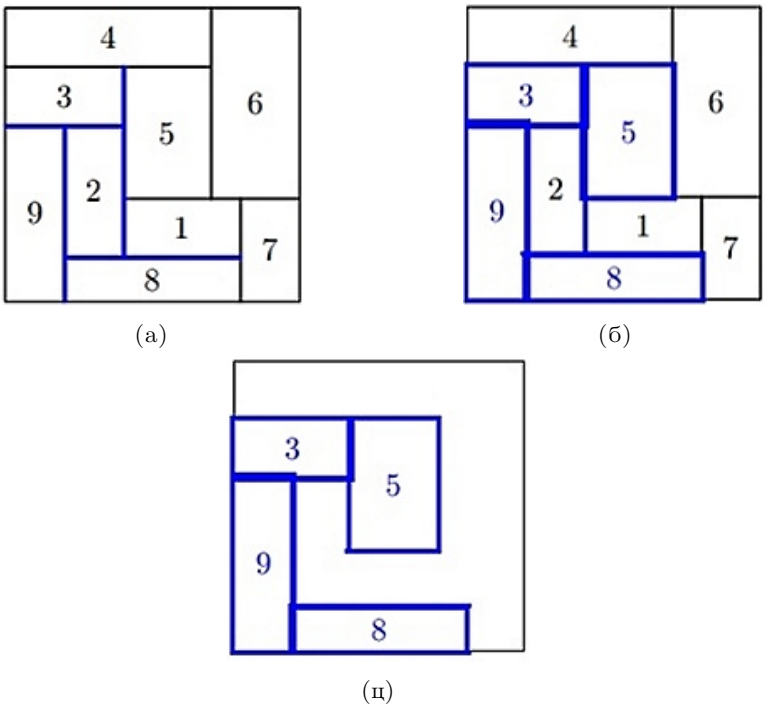


(б) Задржан пакет 4

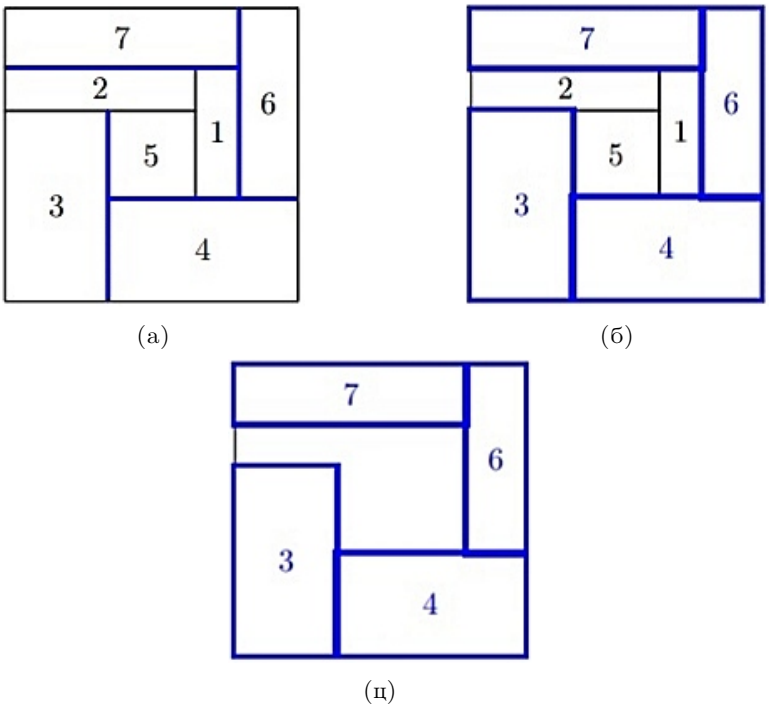


(ц) Задржан пакет 7

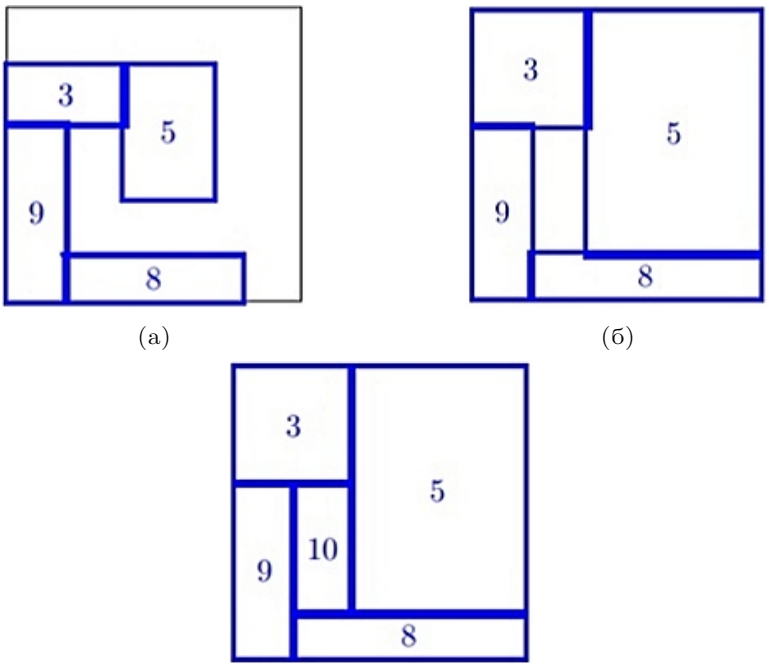
Слика 2.16: Корак 1: Пакети обухваћени изабраним хоризонталним редовима (пр. 2).



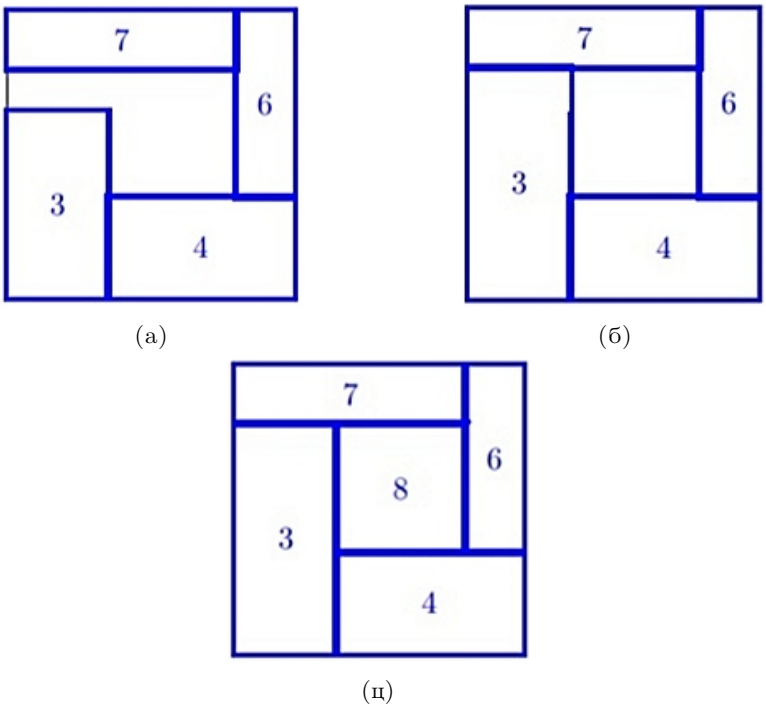
Слика 2.17: Корак 2: Задржани пакети (пр. 1).



Слика 2.18: Корак 2: Задржани пакети (пр. 2).



Слика 2.19: Корак 3: Пакети након алгоритма конвексификације (пр. 1).



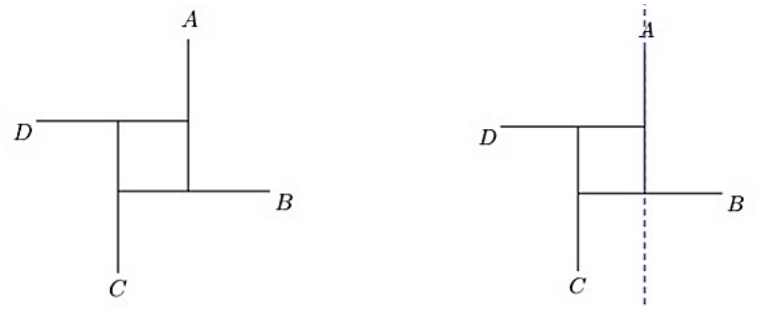
Слика 2.20: Корак 3: Пакети након алгоритма конвексификације (пр. 2).

Теорема 4. *[60] Несепарабилан образац P увек садржи блокиран прстен.*

Доказ. Нека је P образац добијен коришћењем алгоритма конвексификације. Јасно је да ако P садржи блокиран прстен, онда је P несепарабилан.

Претпоставимо супротно, да је P несепарабилан, али да не садржи блокиран прстен. У овом случају не постоји ниједна комбинација несепарабилних редова $x_h = x_v = 2$.

Та комбинација свакако доводи до једноставног блокираног прстена, помоћу корака који откривају блокиран прстен. Са теоремом 3 је доказано да када је $x \leq 4$ само комбинација $x_h = x_v = 2$ доводи до несепарабилног обрасца. Стога, P је несепарабилан ако постоји комбинација несепарабилних редова S, где је $|S| \geq 5$, која да не садржи било који подскуп несепарабилних редова таквих да је $x_h = x_v = 2$. У несепарабилном обрасцу од 4 реда, ако се изведе рез од ивице до ивице на реду који треба одвојити, долази до пресецања другог реда. На слици 2.21 се може видети да се при одвајању реда A пресеца ред B.



(а) Несепарабилни редови (б) Рез на реду A

Слика 2.21: Рез од ивице до ивице на обрасцу несепарабилних редова.

Стога, ако треба одвојити ред A, долази до пресецања реда B, ако треба одвојити ред B, долази до пресецања реда C, ако треба одвојити ред C, долази до пресецања реда D, и на крају ако је потребно одвојити ред D, долази до пресецања реда A. Такво својство се може обележити на следећи начин:

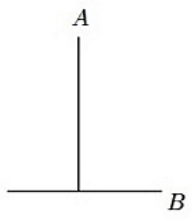
$$A \to B \to C \to D \to A \tag{2.7}$$

Даље два суседна реда имају супротну оријентацију. Наиме, у складу са сликом 2.21, A је вертикалан, B је хоризонталан, C је вертикалан и D је хоризонталан. Враћајући се на образац S, где је $|S| \geq 5$ и несепарабилан, добија се следеће својство:

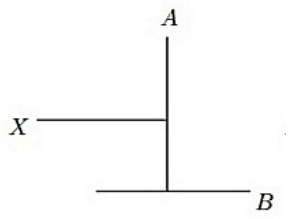
$$A \to B \to C \to ... \to X - 1 \to X \to A \tag{2.8}$$

Без губљења општости, може се претпоставити да је ред A вертикалан ред, па ће ред B бити хоризонталан и може се поставити, без губљења општости изнад или испод реда A, слика 2.22а. Уколико је потребно одвојити ред X, долази до пресецања реда A, па ред X, без губљења општости, може бити постављен или лево или десно од реда A, слика 2.22б. За постављање реда C јављају се три могућности:

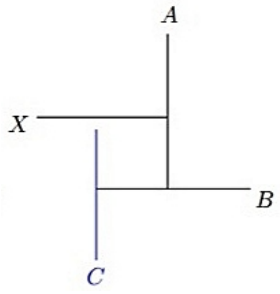
- Позиција 1: на леву страну од реда B, слика 2.22ц. У овом случају се добија подскуп од четири несепарабилна реда. Ова позиција је недопустива.
- Позиција 2: на десну страну од реда B, слика 2.22д. У овом случају долази до мимоилажења, јер последњи постављени ред, или његови продужеци, не пресецају ниједан од претходно постављених редова.
- Позиција 3: Продужава се ред B тако да пролази поред реда X и поставља се ред C лево од реда B, слика 2.22е. И у овом случају долази до мимоилажења.



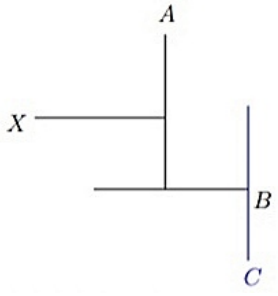
(а) Редови A и B



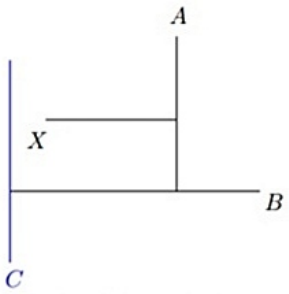
(б) Редови A, B и X



(ц) Редови A, B, C и X



(д) Редови A, B, C и X



(е) Редови A, B, C и X

Слика 2.22: Како поставити ред C.

За постављање реда D. Јављају се две почетне могућности: слике 2.22д и 2.22е.

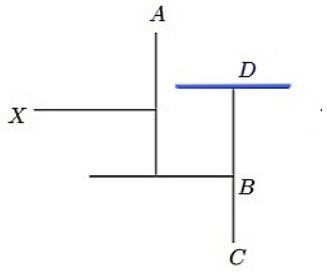
Ако је ред C постављен као на слици 2.22д, за постављање реда D на располагању су три позиције.

- Позиција 1: изнад реда C, слика 2.23а. У овом случају се добија подскуп од четири несепарабилна реда. Ова позиција није допустива.
- Позиција 2: испод реда C, слика 2.23б. У овом случају долази до мимоилажења, јер последње постављен ред, или његови продужеци, не пресецају ниједан од претходно постављених редова.
- Позиција 3: продужава се ред C тако да пролази поред реда A и ред D поставља изнад реда C, слика 2.23ц. У овом случају долази до мимоилажења, такође.

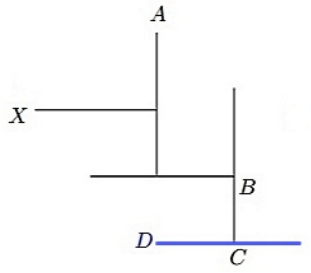
Ако је ред C постављен као на слици 2.22е, за постављање реда D на располагању су поново три позиције.

- Позиција 1: изнад реда C, слика 2.24а. У овом случају се добија подскуп од четири несепарабилна реда. Ова позиција је недопустива.
- Позиција 2: испод реда C, слика 2.24б. У овом случају долази до мимоилажења, јер последње постављен ред, или његови продужеци, не пресецају ниједан од претходно постављених редова.

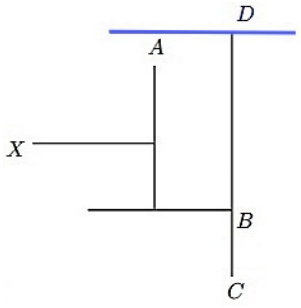
- Позиција 3: продужава се ред C тако да пролази поред реда A и ред D поставља изнад реда C, слика 2.24ц. У овом случају долази до мимоилажења, такође.



(а) Редови A, B, C, D, X

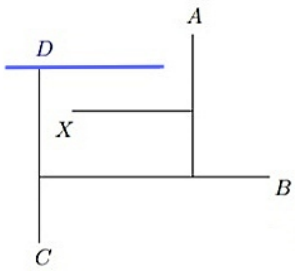


(б) Редови A, B, C, D, X

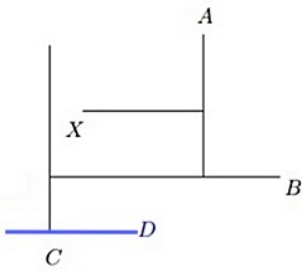


(ц) Редови A, B, C, D, X

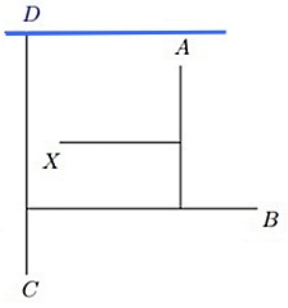
Слика 2.23: Како поставити ред D (1.случај).



(а) Редови A, B, C, D, X



(б) Редови A, B, C, D, X



(ц) Редови A, B, C, D, X

Слика 2.24: Како поставити ред D (2.случај).

Може се закључити да се сваки пакет може поставити на три начина, и очигледно позиције горе, доле, лево и десно се морају процењивати од случаја до случаја. Образац S је несепарабилан ако важи својство 2.8. Ово својство има своје кружење. И заиста,

како важи $A \to B ... X \to A$, редови се не могу увек мимоилазити. То значи да не могу користити Позиција 2 или Позиција 3 за постављање реда X, јер његови продужеци неће пресећи ред A. Стога, користи се Позиција 1. Међутим, овај избор доводи до несепарабилне комбинације $x_h = x_v = 2$, у супротности са претпоставком. То значи да је немогуће имати образац S, где је $|S| \geq 5$ и несепарабилан, који не садржи било који подскуп несепарабилних редова таквих да је $x_h = x_v = 2$. □

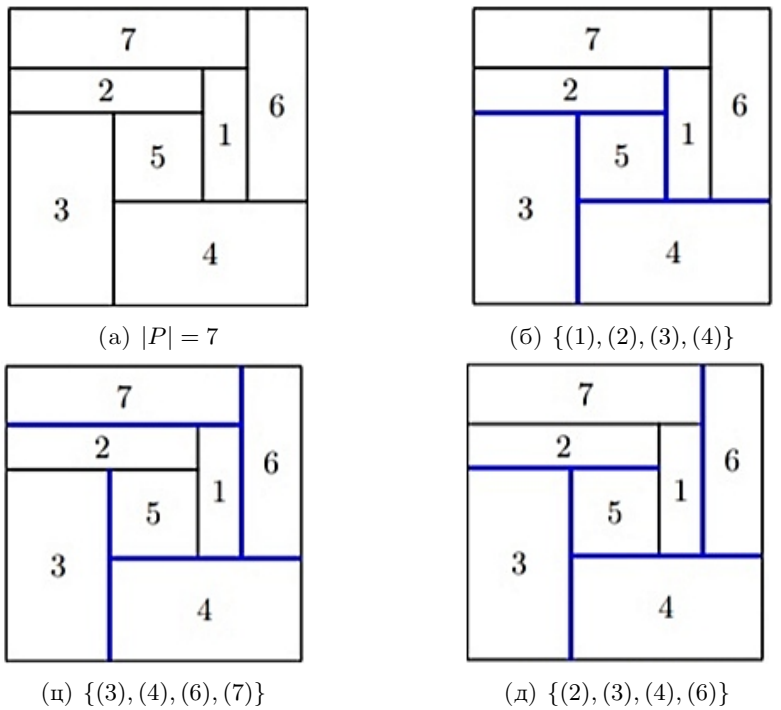
2.4 Карактеризација блокираног прстена

У овом одељку је дата карактеризација несепарабилних образаца у погледу редова и пресека.

2.4.1 Једноструко блокиран прстен

Дефиниција 10 (Једноструко блокиран прстен)**.** *Блокиран прстен је једноструко блокиран прстен ако сви једноставни блокирани прстенови, добијени помоћу алгоритма за откривање блокираног прстена, деле бар један пакет и уклањањем тог пакета даје сепарабилан образац.*

Може се уочити да је једноставан блокиран прстен описан у одељку 2.2 једноструко блокиран прстен. Међутим фамилија једноструко блокираних прстенова укључује комплексније обрасце. Један је приказан на слици 2.25а.



(а) $|P| = 7$ (б) $\{(1), (2), (3), (4)\}$

(ц) $\{(3), (4), (6), (7)\}$ (д) $\{(2), (3), (4), (6)\}$

Слика 2.25: Једноструко блокиран прстен.

Разматрањем несепарабилних редова таквих да је $x_h = x_v = 2$ на слици 2.25, могу се уочити три могуће комбинације, и на тај начин три групе пакета, које дају једноставан блокиран прстен након примене алгоритма конвексификације.

- $\{(1), (2), (3), (4)\}$
- $\{(3), (4), (6), (7)\}$
- $\{(2), (3), (4), (6)\}$

Једноставни блокирани прстенови деле (3) и (4), што значи да се одстрањивањем (3) и (4) добија гиљотинабилан образац.

2.4.2 Вишеструко блокиран прстен

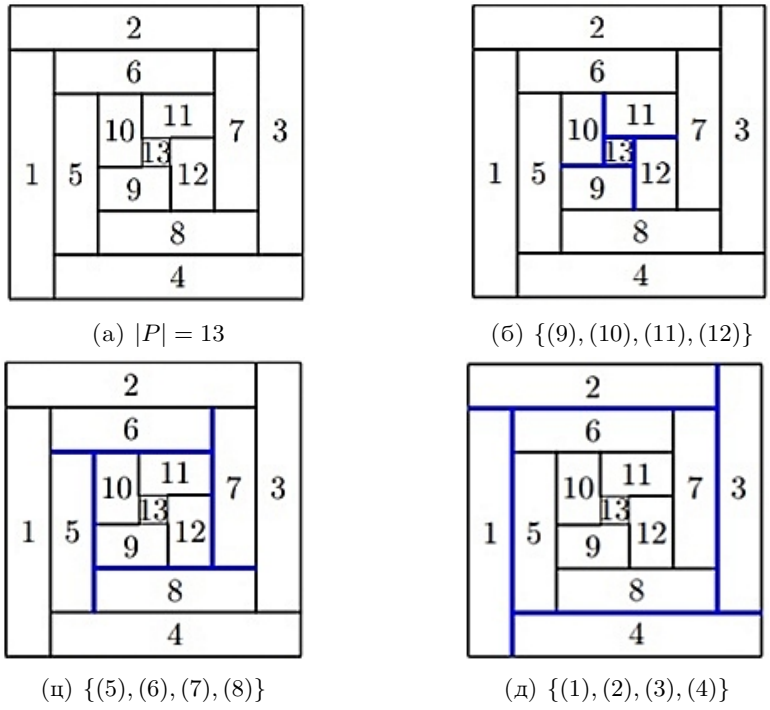
Дефиниција 11 (Вишеструко блокиран прстен)**.** *Блокиран прстен који није једноструко блокиран прстен називамо вишеструко блокиран прстен.*

По дефиницији за вишеструко блокиран прстен, не постоји пакет који припада свим једноставним блокираним прстеновима добијен алгоритмом за откривање блокираног прстена.

Вишеструко блокиран прстен може бити:

- угњежден блокиран прстен
- повезан блокиран прстен
- сложен блокиран прстен. Комбинује особине угњежденог и повезаног блокираног прстена.

Дефиниција 12 (Угњежден блокиран прстен)**.** *Вишеструко блокиран прстен је угњежден блокиран прстен ако сви једноставни блокирани прстенови, добијени алгоритмом за откривање блокираног прстена, не деле ниједан пакет.*



(а) $|P| = 13$ (б) $\{(9), (10), (11), (12)\}$

(ц) $\{(5), (6), (7), (8)\}$ (д) $\{(1), (2), (3), (4)\}$

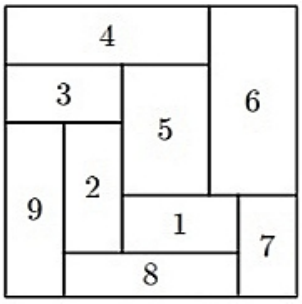
Слика 2.26: Угњежден блокиран прстен.

Разматрањем несепарабилних редова таквих да је $x_h = x_v = 2$ на слици 2.26, могу се уочити три могуће комбинације, и на тај начин три групе пакета, које дају једноставан блокиран прстен након примене алгоритма конвексификације.

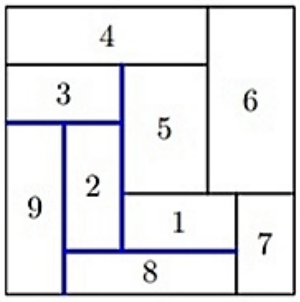
- $\{(9), (10), (11), (12)\}$
- $\{(5), (6), (7), (8)\}$
- $\{(1), (2), (3), (4)\}$

Ови једноставни блокирани прстенови не деле ниједан пакет, па је у питању угњежден блокиран прстен. Да би се добио гиљотинабилан образац, сви нађени једноставни прстенови морају бити гиљотинабилни, па мора бити уклоњен по један пакет из сваког од њих.

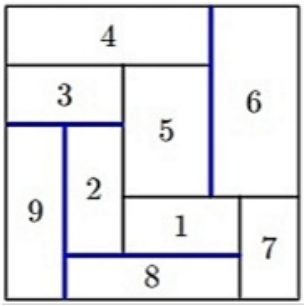
Дефиниција 13 (Повезан блокиран прстен)**.** *Вишеструко блокиран прстен је повезан блокиран прстен ако сваки једноставан блокиран прстен, добијен алгоритмом за откривање блокираног прстена, дели бар један пакет са другим једноставним блокираним прстеном.*



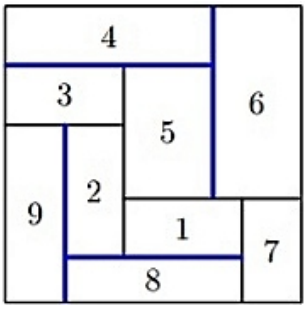
(а) $|P| = 9$



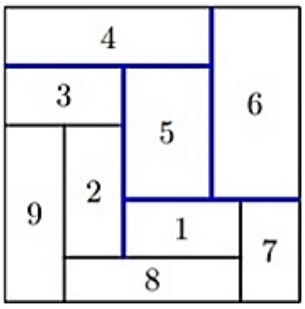
(б) $\{(3), (5), (8), (9)\}$



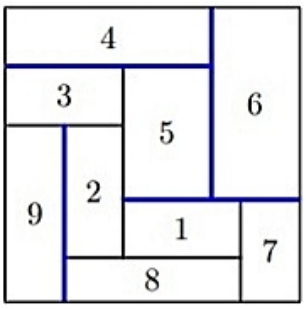
(ц) $\{(3), (6), (8), (9)\}$



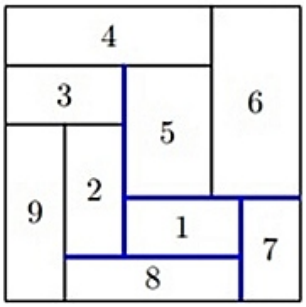
(д) $\{(4), (6), (8), (9)\}$



(е) $\{(1), (2), (4), (6)\}$



(ф) $\{(1), (4), (6), (9)\}$



(г) $\{(2), (6), (7), (8)\}$

Слика 2.27: Повезан блокиран прстен.

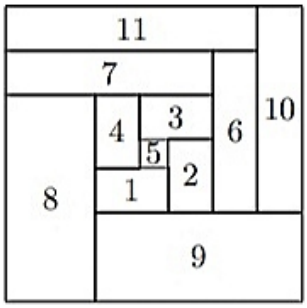
Разматрањем несепарабилних редова таквих да је $x_h = x_v = 2$ на слици 2.27, може се уочити шест могућих комбинација, и на тај начин шест група пакета, које дају једноставан блокиран прстен након примене алгоритма конвексификације.

- $\{(3), (5), (8), (9)\}$

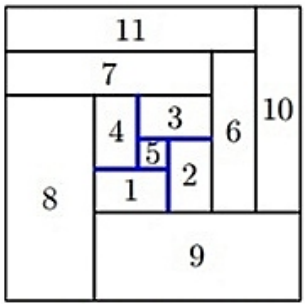
- $\{(3), (6), (8), (9)\}$
- $\{(4), (6), (8), (9)\}$
- $\{(1), (2), (4), (6)\}$
- $\{(1), (4), (6), (9)\}$
- $\{(2), (6), (7), (8)\}$

Не постоји пакет који деле сви једноставни блокирани прстенови, али сваки једноставан блокиран прстен дели бар један пакет са другим једноставним блокираним прстеном, па је у питању један повезан блокиран прстен. Да би се добио гиљотинабилан образац сви нађени једноставни блокирани прстенови морају бити гиљотинабилни, па, на пример, морају бити уклоњени пакети (4) и (8).

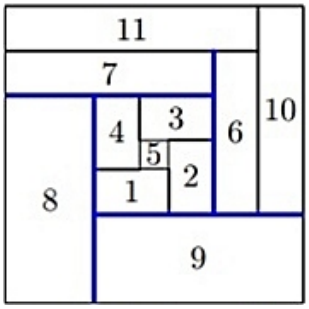
Дефиниција 14 (Сложен блокиран прстен)**.** *Вишеструко блокиран прстен је сложен блокиран прстен ако постоје два или више једноставних блокираних прстенова, добијених алгоритмом за откривање блокираног прстена, где сваки од њих дели бар један пакет са другим једноставним блокираним прстеном и један или више једноставних блокираних прстенова, добијених такође алгоритмом откривања блокираног прстена, који не деле ниједан пакет.*



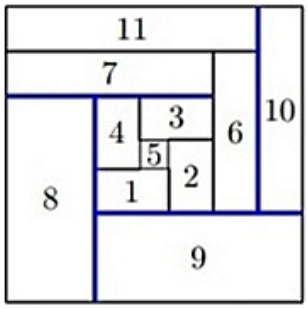
(а) $|P| = 11$



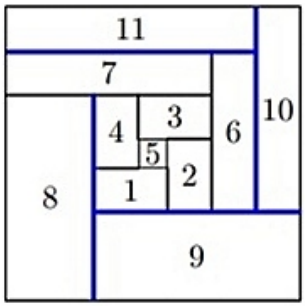
(б) $\{(1), (2), (3), (4)\}$



(ц) $\{(6), (7), (8), (9)\}$



(д) $\{(7), (8), (9), (10)\}$



(е) $\{(8), (9), (10), (11)\}$

Слика 2.28: Сложен блокиран прстен.

Слика 2.28 приказује сложен блокиран прстен. На слици 2.28б је приказан једнос-

таван блокиран прстен који не дели ниједан пакет са осталим, док су на сликама 2.28ц, 2.28д и 2.28е дати једноставни блокирани прстенови који деле пакете. Да би се добио гиљотинабилан образац сви нађени једноставни блокирани прстенови морају бити гиљотинабилни, па на пример морају бити уклоњени пакети (1) и (8).

2.5 Анализа најнеповољнијег случаја

У овом одељку су разматрани проблеми P_1 и P_2, представљени у одељку 2.1, и дати одговори за ове проблеме у неким специјалним случајевима. Како се разматра један по један образац, случајеви се разликују по структури тренутног обрасца. У свим случајевима образац P је добијен извршавањем алгоритма конвексификације, који је описан у одељку 2.1.3. Додатно, претпоставља се да је P несепарабилан, ако другачије није наведено.

У одељку 2.5.1 су проблеми P_1 и P_2 решавани за најједноставнији случај у којем је P једноставан блокиран прстен. У одељку 2.5.2 је разматран случај у којем је P једноструко блокиран прстен. На крају је решавање проблема P_1 и P_2, у случају да је P вишеструко блокиран прстен, остављено отворено.

Што се тиче проблема P_2, разматра се једно по једно складиште док се рачунање доње границе PoG-а заснива на следећем:

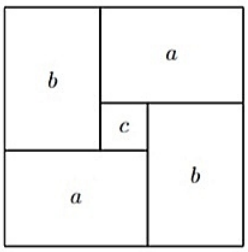
Теорема 5. *[60]*

$$PoG \geq 4/3.$$

Доказ. Дата је инстанца која се састоји од

- $2k$ пакета типа $a : w_i = 3/5$ и $h_i = 2/5$,
- $2k$ пакета типа $b : w_i = 2/5$ и $h_i = 3/5$ и
- k пакета типа $c : w_i = h_i = 1/5$

за неко целобројно k. Сви пакети треба да буду спаковани у 1×1 складишта. Оптимална вредност решења 2BP|O|F проблема се састоји од k идентичних складишта приказаних на слици 2.29.



Слика 2.29: Негиљотинабилан образац.

Лако се види да гиљотинабилан образац може садржати највише два пакета типа a, и један пакет типа b (или обрнуто), и један пакет типа c. То значи да оптимално решење проблема 2BP|O|G захтева $4k$ складишта за паковање $6k$ пакета типа a и b, и $3k$ пакета типа c, док би оптимално решење проблема 2BP|O|F спаковало ове пакете у $3k$ складишта. Што доказује теорему. □

2.5.1 1. случај: P је једноставан блокиран прстен

У овом одељку је разматран случај када је P једноставан блокиран прстен.

Теорема 6. *[60] Ако је P једноставан блокиран прстен онда је*

$$MA(P) = 1/4.$$

Доказ. Како је P једноставан блокиран прстен, он има структуру која као што је описано у одељку 2.2.1, тј. $|P| = 5$ и садржи централни пакет који додирује сва четири преостала пакета. Очигледно је да постоји један пакет међу прва четири, чија је површина једнака највише $1/4$ површине складишта. Одстрањивање овог пакета доводи до сепарабилног обрасца Q. По теореми 1 било који скуп пакета који производи Q, у смислу алгоритма конвексификације, је сепарабилан.

Да би се показало да је резултат добар, у [55] је размотрана инстанца у којој пакети 1, 2, 3 и 4 имају исту површину, а пакет 5 је произвољно мали. Тада је површина пакета који треба уклонити да би се добио сепарабилан образац блиска $1/4$ површине складишта. □

Теорема 7. *[60] Ако се P састоји од једног или више једноставних блокираних прстенова, тада је*

$$PoG = 4/3.$$

Доказ. Из сваке тројке складишта у решењу, које деле исти образац P, дефинишу су четири складишта на следећи начин: за $i = 1, 2, 3$ складиште i складишти све пакете осим пакета i, а четврто складиште складишти пакете $1, 2, 3$ на исте координате као у P. Даље, количник оптималног решења проблема 2BP|O|G и датог решења 2BP|O|F проблема не може бити већи од $4/3$. Комбинујући ово са теоремом 5 добија се тачна вредност $PoG = 4/3$ у овом специјалном случају. □

2.5.2 2. случај: P је једноструко блокиран прстен

Из одељка 2.4.1 произилази да, ако је P једноставан блокиран прстен, онда постоји скуп четворки пакета, Q, које одговарају једноставним блокираним прстеновима, и које деле бар један пакет.

Ово доводи до следећег резултата

Теорема 8. *[60] Ако је P једноструко блокиран прстен, онда је*

$$MA(P) = 1/3.$$

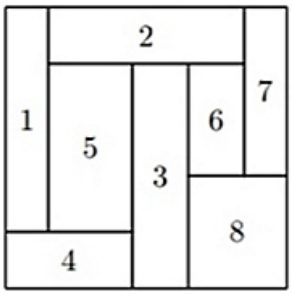
Доказ. Нека (w_u, h_u) и (x_u, y_u) представљају димензије и координате паковања за сваки пакет $u \in P$. Сваком пакету u додељује се једна од боја: жута, црвена или плава. Ове три боје представљају поделу пакета у три подскупа. Циљ је показати да се уклањањем подскупа било које боје добија сепарабилан образац. Познато је да постоји пакет i, заједнички за све четворке. У специјалном случају, када су два пакета (i, k) заједничка свим четворкама, бојење је лако: i је жуто, k је црвено, а сви остали пакети су плави. Када је тачно један пакет i заједнички свим четворкама, прво бојење, жутом бојом, је лако, садржи само један пакет i. Уклањањем пакета i добија се сепарабилан образац. Црвени пакети су ти који не дозвољавају да пакет i буде одвојен вертикалним резовима. Црвеном бојом се боје пакети пресечени вертикалним резовима $x = x_i$ и $x = x_i + w_i$. Образац код којег су одстрањени само црвени пакети је сепарабилан. Заиста, одстрањивањем црвених пакета, по дефиницији, пакет i може бити изрезан помоћу два вертикална реза. Једном када се пакет i одстрани, остатак је очигледно сепарабилан.

Слично, плавом бојом се боје пакети пресечени хоризонталним резовима $y = y_i$ и $y = y_i + h_i$. Образац без плавих пакета је такође сепарабилан. Очигледно је да постоји подскуп између жутих, црвених и плавих пакета чија је површина највише $1/3$ површине складишта. Уклањањем овог подскупа добија се сепарабилан образац Q.

Да би се доказало да је резултат добар, у [55] је разматрана инстанца приказана на слици 2.30, која одговара обрасцу са 8 пакета. У овом случају могуће је поделити пакете у три подскупа: $\{2\}$, $\{1,7\}$ и $\{4,8\}$ таква да

- сваки подскуп пакета је сепарабилан
- одстрањивање било којег подскупа пакета доводи до сепарабилног обрасца.

У најнеповољнијем случају, сва три подскупа имају исту површину, док остали пакети заузимају занемарљиву површину. Минимална површина MA је она коју треба одстранити да би се добио сепарабилан образац је блиска 1/3. □



Слика 2.30: Најнеповољнији случај за проблем P_1 када је P једноструки блокиран прстен.

Уопштеније, рачунање $MA(P)$ за образац P се може извршити на следећи начин.

Проблем одстрањивања: За дати скуп пакета P и скуп од q четворки Q_j $(j = 1, ...q)$, одредити поделу пакета у t подскупова $S_1, ..., S_t$, таквих да сваки подскуп S_k укључује (1) барем један пакет и (2) највише три пакета из сваке четворке Q_j.

Услов (1) осигурава да уклањање свих пакета из било којег подскупа S_k даје гиљотинабилан образац, док услов (2) осигурава да скуп пакета који су уклоњени даје, сам по себи, гиљотинабилан образац. Коришћењем истог резона као у доказу теореме 6, ако је нађена t партиција скупа P, одстрањивање највише $1/t$ површине складишта може дати сепарабилан образац $\overline{P}$, тј. $MA(P) \leq 1/t$. Да би се минимизирала површина коју треба уклонити и дефинисали $\overline{P}$, природно је максимизирати број подскупова партиције.

Уз напомену да је $t \leq |P|$ и коришћењем следећих бинарних променљивих

$$y_k = \begin{cases} 1, & \text{ако је генерисан подскуп } k \\ 0, & \text{иначе} \end{cases} \qquad (k = 1, ..., |P|) \tag{2.9}$$

$$x_{ik} = \begin{cases} 1, & \text{ако је пакет } i \text{ садржан у подскупу } k \\ 0, & \text{иначе.} \end{cases} \qquad (i \in P;\ k = 1, ..., |P|) \tag{2.10}$$

Може се извести следећи математички модел за посматрани проблем.

$$\max t = \sum_{k} y_k \tag{2.11а}$$

$$\sum_{i \in Q_j} x_{ik} \geq y_k, \ \forall k, \ \forall j \tag{2.11б}$$

$$\sum_{i \in Q_j} x_{ik} \leq 3y_k, \ \forall k, \ \forall j \tag{2.11ц}$$

$$\sum_{k} x_{ik} \leq 1, \ \forall i \tag{2.11д}$$

$$y_k, x_{ik} \in \{0, 1\}, \ \forall i, \ \forall k \tag{2.11е}$$

Функција циља (2.11а) максимизује број генерисаних подскупова. Услови (2.11б)-(2.11ц) намећу да сваки изабрани подскуп има најмање један, а највише три пакета из сваке четворке. На крају, неједнакости (2.11д) осигуравају да сваки пакет припада највише једном подскупу партиције, док услов (2.11е) намеће да су све променљиве бинарне.

Ако се модел (2.11а)-(2.11е) примени на образац приказан на слици 2.30, добијају се три подскупа: $\{2\}$, $\{1, 8\}$ и $\{4, 7\}$. Овај модел и они који ће бити описани у наставку могу бити примењени и на вишеструки блокиран прстен.

На сличан начин се разматра проблем рачунања вредности цене гиљотинабилности за дати скуп пакета P. Ова вредност се означава са $PoG(P)$ и важи да је $PoG = \sum_P PoG(P)$.

Може се приметити да свако допустиво решење вредности t модела (2.11а)-(2.11е) производи гиљотинабилно решење за дати скуп пакета P. Вредност овог решења 2BP|O|G проблема имплицитно даје горњу границу за $PoG(P)$. Заиста, као и у доказу теореме 7, може се дефинисати t гиљотинабилних образаца који пакују све пакете спаковане у $t - 1$ складишта у решењу 2BP|O|F проблема: свако складиште j ($j = 1, ..., t$) садржи све пакете из оригиналног обрасца, али не и оне који припадају j-тој партицији скупа пакета. Како оптимална вредност решења 2BP|O|G проблема не може бити лошија од овог решења, закључује се да је $PoG(P) \leq \frac{t}{t-1}$.

Како се $PoG(P)$ тиче асимптотског односа, може се искористити доступност вишеструких копија пакета. У том погледу, може се поделити садржај $t - D$ складишта решења 2BP|O|F проблема у t подскупова пакета, где сваки од њих производи гиљотинабилан образац. То значи да треба узети у обзир да се сваки пакет може додати у највише D подскупова, тј. да се може уклонити из највише D складишта из почетног

решења. Према томе, модел за рачунање $PoG(P)$ је следећи

$$\min \frac{t}{t-D} \tag{2.12а}$$

$$t = \sum_k y_k \tag{2.12б}$$

$$\sum_{i \in Q_j} x_{ik} \geq y_k, \ \forall k, \ \forall j \tag{2.12ц}$$

$$\sum_{i \in Q_j} x_{ik} \leq 3y_k, \ \forall k, \ \forall j \tag{2.12д}$$

$$\sum_k x_{ik} y_k \leq D, \ \forall i \tag{2.12е}$$

$$x_{ik} \in \{0,1\}, \ \forall i, \ \forall k \tag{2.12ф}$$

$$y_k \geq 0 \text{ целобројно}, \ \forall k \tag{2.12г}$$

$$D \geq 0 \text{ целобројно} \tag{2.12х}$$

где променљиве y_k показују колико се пута је изабран сваки образац k, а D је нова променљива која показује максималан број копија сваког пакета који се користи, тј. максималан број копија сваког пакета који може да остане неспакован. Нова функција циља 2.12а минимизира $PoG(P)$, који је дат као количник између броја изведених гиљотинабилних образаца и броја складишта који је разматран у решењу 2BP|O|F проблема.

Модел (2.12а)-(2.12х) је крајње нелинеаран

- функција циља је нелинеарна по променљивим t и D; и
- услови 2.12е укључују производ променљивих x и y.

Да би се поново прешло на линеарну функцију циља, модел се може решити неколико пута, за различите (и фиксиране) вредности D, који постаје параметар у овим подешавањима. Даље, потребно је минимизирати нерастућу конвексну функцију по t, што је еквивалентно максимизацији променљиве t. Што се тиче осталих нелинеарности, директан начин за линеаризацију производа који укључују бинарну променљиву је увођење додатних променљивих α_{ik} које означавају број копија пакета i које су стварно додате у подскуп k.

За дату вредност D модел изгледа

$$\max t \tag{2.13а}$$

$$\sum_k y_k = t \tag{2.13б}$$

$$\sum_{i \in Q_j} x_{ik} \geq y_k, \ \forall k, \ \forall j \tag{2.13ц}$$

$$\sum_{i \in Q_j} x_{ik} \leq 3y_k, \ \forall k, \ \forall j \tag{2.13д}$$

$$\sum_k \alpha_{ik} \leq D, \ \forall i \tag{2.13е}$$

$$x_{ik} \in \{0,1\}, \ \forall i, \ \forall k \tag{2.13ф}$$

$$y_k \geq 0 \text{ целобројно}, \ \forall k \tag{2.13г}$$

$$\alpha_{ik} \geq y_k - M(1 - x_{ik}), \ \forall i, \ \forall k \tag{2.13х}$$

$$\alpha_{ik} \geq 0 \text{ целобројно}, \ \forall i, \ \forall k \tag{2.13и}$$

где услови (2.13х)-(2.13и) служе за повезивање α променљивих са x и y променљивим,

а M је велики коефицијент.

Потребно је израчунати $PoG = \sum_P PoG(P)$, тј. циљ је да се утврди вредност цене гиљотинабилности за најнеповољнији случај P. Важи следећи резултат

Теорема 9. *[60] Ако се P састоји од једног или више једноструко блокираних прстенова, тада је*

$$\frac{7}{5} \leq PoG \leq \frac{3}{2}.$$

Доказ. Што се тиче горње границе, треба напоменути да теорема 8 имплицитно даје решење за проблем (2.12a)-(2.12x) где је $t = 3$ и $D = 1$. То доводи до решења 2BP|O|G проблема које користи број складишта једнак 3/2 броја складишта добијених било којим решењем 2BP|O|F проблема.

Размотрањем инстанце приказане на слици 2.30, оптимална вредност решења модела (2.12a)-(2.12x) даје $t = 7$ и $D = 2$, и представљена је следећим скупом: $\{2\}$, $\{4, 8\}$ и $\{1, 7\}$, $\{3, 8\}$, $\{1, 3, 5\}$ и $\{4, 7\}$ где је први скуп пакета узет два пута. Ово показује да се за дато решење од 5 складишта решења 2BP|O|F проблема може конструисати решења са 7 складишта 2BP|O|G проблема, тј. $PoG(P) \leq \frac{7}{5}$ што даје добру доњу границу PoG. □

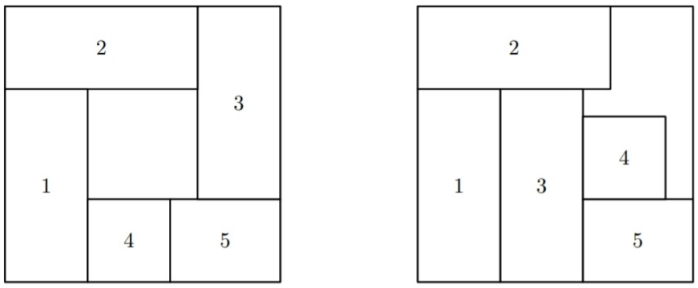
Глава 3

Хеуристичко решавање дводимензионог проблема паковања са и без ограничења гиљотине

3.1 Увод

У дводимензионом проблему паковања у складишта (2BPP) захтева се паковање датог скупа малих правоугаоних *пакета* у минималан број великих правоугаоних *складишта*. Пакети морају бити постављени ортогонално, са ивицама паралелним ивицама складишта, и без преклапања. Како паковање пакета у складишта такође моделује сечење мањих комада од већих плоча залиха, овај проблем налази примену у сечењу стакла и дрвета, утовару кутија у возила, складиштењу робе, телекомуникацији и сл.

У зависности од конкретне примене и врсте образаца који се могу добити, до сада су у литератури проучене многе варијанте 2BPP проблема. Релевантан случај настаје када се дозволи ротирање пакета за 90°, да би се постигло боље коришћење простора складишта. Треба имати на уму да ротација може бити дозвољена за подскуп пакета, док би је требало избегавати у случају да складиште има специфичне карактеристике (нпр. садржи стаклене плоче). Други релевантан специјалан случај је извођење k-фазног обрасца, тј. решења у којем сваки пакет може бити добијен низом од (највише) k резова од ивице до ивице, паралелних ивицама складишта. Ова врста проблема, представљена у [24], користи се у применама у којима се за сечење користе аутоматске машине, где при сваком резу настају неки трошкови. Многи радови у литератури се баве случајем када је $k = 2$, тј. где пакети треба да буду упаковани у *нивое*. Модели целобројног линеарног програмирања (енг. Integer Linear Programming - ILP), за паковање у нивое са полиномијалним бројем променљивих и ограничењима, дати су у [52] и [54], и проширени су за случај $k = 3$ у [62]. Проблем у којем није наметнута граница на број резова познат је под називом *Дводимензиони проблем паковања у складишта са ограничењем гиљотине*. Слика 3.1 показује пример негиљотинског обрасца за дат скуп пакета (лево), као и паковање истог скупа пакета тако да буде испуњен услов гиљотине (десно).



Слика 3.1: Пример негиљотинабилног и гиљотинабилног обрасца

У одељцима 3.2 и 3.3 су детаљно представљене две хеуристике за решавање дводимензионог проблема паковања у складишта. Прва је хеуристика базирана на парцијалној енумерацији, коју су у [55] представили Lodi, Monaci и Pietrobuoni, и која користи ограничење гиљотине, а друга генетски алгоритам са Crow претрагом, коју су у [44] представили Laabadi, Naimi, Amri и Achchab, и која не користи ограничење гиљотине. У оба случаја ротација пакета није дозвољена. На крају је, у одељку 3.4, дато њихово поређење и изведен закључак.

3.2 Алгоритам парцијалне енумерације за решавање 2BP|O|G проблема

3.2.1 Основни хеуристички алгоритам

Алгоритам се заснива на стаблу енумерације, које на сваком нивоу p дефинише садржај p-тог складишта у решењу. Хеуристичке је природе, тако да нису узети у обзир сви допустиви начини паковања. Чворови листови одговарају комплетним решењима, у којима су сви пакети спаковани. Средишњи чворови имају бројне чворове наследнике, повезане са различитим начинима паковања следећих складишта на гиљотински начин. Конкретно, свако складиште се пуни применом *стратегије паковања* која пакује један по један пакет у складу са датим *правилом избора* и *правилом гиљотинског дељења*. Правило избора одређује следећи пакет који треба спаковати (и његову позицију у складишту), док се правило гиљотинског дељења користи како би се обезбедила гиљотабилност изведеног обрасца. Користе се скуп правила избора, $\mathcal{S}$, и скуп правила гиљотинског дељења, $\mathcal{G}$, и генерише чвор наследник за сваки пар (s, g), где је $s \in \mathcal{S}$ и $g \in \mathcal{G}$. Стабло претраге се претражује претрагом у дубину, разматрањем чворова наследника по правилу избора, а затим по правилу гиљотинског дељења. Ако се нађе допустиво решење са вредношћу (рецимо) z, на нивоу $z-1$ нема генерисаних чворова наследника, јер ови чворови не могу допринети побољшању решења.

Паковање у тренутно складиште

У наставку је представљена стратегија паковања која се користи на сваком чвору, за паковање у тренутно складиште. Са $N = \{1, ..., n\}$ је означен укупан број пакета које треба распоредити, а са $I \subseteq N$ скуп пакета који нису распоређени у претходна складишта (тј. на претходним новима стабла). На сваки чвор се примењује устаљено правило избора и устаљено правило гиљотинског дељења.

Стратегија паковања пакује један по један пакет у слободан простор складишта, што гарантује да ће бити добијен гиљотинабилан образац. Заправо, прави се листа $\mathcal{F} = \{F_1, ..., F_m\}$ парова раздвојених слободних правоугаоника где пакети из I могу бити распоређени. На почетку је $\mathcal{F} = \{F_1\}$, тј. постоји само један правоугаоник који одговара унутрашњости складишта.

У свакој итерацији се одређује скуп $\mathcal{P}$ парова (i, F_j) који асоцира на паковање допустивог пакета $i \in I$ у слободан правоугаоник $F_j \in \mathcal{F}$, и за сваки пар се рачуна *резултат убацивања*. Резултат убацивања може бити неискоришћена површина складишта,

дужина краће ивице складишта, преостале након смештања пакета, или дужина дуже ивице складишта, преостале након смештања пакета, и зависи од примењеног правила избора. Правила избора су представљена у наредном одељку.

Нека је изабран пар (i^*, F_{j^*}), који даје минимум. Доњи леви угао пакета i^* се поставља у доњи леви угао правоугаоника F_{j^*}, и елиминише из I. Изабрани правоугаоник се затим елиминише из $\mathcal{F}$, и дели помоћу правила гиљотинског дељења да би се евентуално произвела два мања правоугаоника која су убачена у $\mathcal{F}$. На крају се преиспитују слободни правоугаоници и проверава се да ли постоје парови правоугаоника који се могу спојити у један већи. Тренутно складиште се затвара ако је $I = \emptyset$, или $\mathcal{F} = \emptyset$ или нема пакета $i \in I$ који може бити убачен у било који слободан правоугаоник $F_j \in \mathcal{F}$.

У наставку је дат псеудокод имплементације алгоритма, преузет из [55], где се претпоставља да алгоритам на улазу прима

- скуп I нераспоређених пакета,
- димензије W и H складишта,
- функцију $s(i, F_j)$ која враћа резултат убацивања за пакет $i \in I$ у слободан правоугаоник F_j и
- функцију $g(i, F_j)$ која враћа (могуће празан) скуп слободних правоугаоника добијених изрезивањем пакета i од правоугаоника F_j.

Алгоритам гиљотинског складишта

1: дефинисати полазни $W \times H$ правоугаоник F_1 на позицији $(0, 0)$ и скуп $\mathcal{F} = \{F_1\}$;
2: **repeat**
3: нека је $\mathcal{P}$ скуп парова (i, F_j) таквих да $i \in I$ може бити убачен у правоугаоник $F_j \in \mathcal{F}$;
4: **if** $\mathcal{P} \neq \emptyset$ **then**
5: нека је $(i^*, F_{j^*}) = \arg\min\{s(i, F_j) : (i, F_j) \in \mathcal{P}\}$;
6: спаковати пакет i^* у правоугаоник F_{j^*} и поставити $I := I \setminus \{i^*\}$;
7: поставити $\mathcal{F} := \mathcal{F} \setminus \{F_{j^*}\} \cup \{g(i^*, F_{j^*})\}$;
8: **end if**
9: **until** $\mathcal{P} = 0$

Правило избора

Правила избора се користе да се истовремено изабере следећи пакет који ће бити распоређен и придружени слободан правоугаоник. Доњи леви угао пакета ставља у доњи леви угао изабраног правоугаоника. Сва правила рачунају резултат за сваки пар (i, F_j), који показује ,,квалитет" паковања. Када се упореде резултати два пара, мањи резултат је бољи. У случају да код пара, на пример (i, F_j), пакет i савршено стаје у тренутно слободни правоугаоник F_j, ставља се да је $s(i, F_j) = -\infty$, тј. пакет i се пакује у правоугаоник F_j без рачунања осталих резултата. Слично, $s(i, F_j) = \infty$ ако пакет i не може да стане у складиште F_j.

Нека W_j и H_j означавају ширину и висину, респективно, слободног правоугаоника F_j. Да бисмо направили избор користимо следећа три правила.

1. **Најбоља површина** (енг. Best Area): $s_{BA}(i, F_j) = W_j H_j - w_i h_i$
2. **Најбоља краћа страна** (енг. Best Short Side): $s_{BSS}(i, F_j) = \min(W_j - w_i, H_j - h_i)$
3. **Најбоља дужа страна** (енг. Best Long Side): $s_{BLS}(i, F_j) = \max(W_j - w_i, H_j - h_i)$

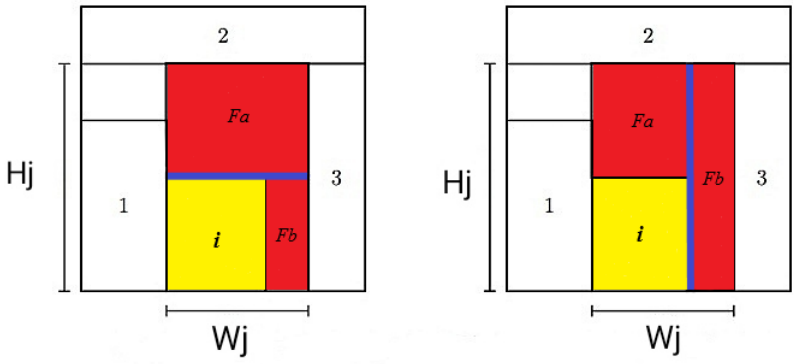
Ова правила минимизирају неискоришћену површину у слободном правоугаонику, дужину краће преостале стране и дужину дуже преостале стране, респективно. Слична правила коришћена су у [40], заједно са правилима која максимизирају ове бројеве, како би се одредио најбољи слободан правоугаоник за смештање датог пакета.

Правило гиљотинског дељења

Нека i и F_j означавају пакет и слободан правоугаоник, респективно, које треба одредити коришћењем неког правила избора. Алгоритам пакује доњи леви угао пакета i у доњи леви угао правоугаоника F_j, чиме се ствара слободан простор у облику слова L. Овај слободан простор се може поделити на два правоугаоника, хоризонталним или вертикалним резом, као што је приказано на слици 3.2. Пакет i је обојен жутом бојом, резови си обојени плавом, а правоугаоници настали резовима, црвеном бојом. У случају хоризонталног реза, генеришу се два нова правоугаоника: F_a, димензија $W_j, H_j - h_i$, и F_b, димензија $W_j - w_i, h_i$. Ако је рез вертикалан, као резултат настају правоугаоници F_a, димензија $w_i, H_j - h_i$, и F_b, димензија $W_j - w_i, H_j$. У оба случаја, ако је $w_i = W_j$ или $h_i = H_j$, добија се само један правоугаоник, док у случају да важе обе једнакости не настаје ниједан правоугаоник. За одређивање типа реза који ће бити изведен примењен је низ различитих стратегија, представљених у [40], и на основу прелиминарних експерименталних тестова се дошло до тога да се у овом алгоритму, преузетом из [55], користе следеће три стратегије:

- **Дужи остатак** (енг. Longer Leftover)**:**
  рез је хоризонталан ако је $W_j - w_i \geq H_j - h_i$, иначе је вертикалан;
- **Краћи остатак** (енг. Shorter Leftover)**:**
  рез је хоризонталан ако је $W_j - w_i < H_j - h_i$, иначе је вертикалан;
- **Минимална површина** (енг. Min Area)**:**
  рез је хоризонталан ако је $h_i(W_j - w_i) < w_i(H_j - h_i)$, иначе је вертикалан;

Прва два правила дељења одређују смер реза по дужој и краћој преосталој страни, респективно, док је треће намењено формирању два нова слободна правоугаоника сличних површина.



Слика 3.2: Пример хоризонталног (лево) и вертикалног (десно) гиљотинског реза.

3.2.2 Побољшани хеуристички алгоритам

Основна схема претраге описана у одељку 3.2.1 доводи до стабла претраге са великим бројем чворова, чак и за мале инстанце. Заиста, комбиновањем три правила избора са три правила гиљотинског дељења, производи девет потенцијалних различитих чворова наследника за сваки чвор. Ово значи да је број чворова на p-том нивоу стабла претраге једнак 9^p, и може бити екстремно велики, чак и за мале вредности p. Зато су у наставку представљена два начина за побољшање алгоритма, преузета из [55]. Први има за циљ избегавање вишеструког генерисања чворова одлуке који производе исти образац, док је други хеуристичко орезивање чворова.

Одстрањивање чворова дупликата

Посматра се чвор d, на датом нивоу p стабла претраге. Нека су (s_1, g_1) и (s_2, g_2) два могућа чвора наследника повезана са различитим правилима избора $s_1, s_2 \in \mathcal{S}$ и/или различитим правилима гиљотинског дељења $g_1, g_2 \in \mathcal{G}$. Иако су правила коришћена у два чвора различита, стратегија паковања може произвести два (могуће различита) обрасца која пакују исти подскуп пакета. У овом случају, подстабла која произилазе из ова два чвора би била потпуно идентична и њихово претраживање би било губљење времена.

Да би се ово заобишло, основна схема треба да буде модификована да генерише чворове наследнике само за оне обрасце који пакују различите подскупове пакета. У принципу, потребно је упоредити скуп нераспоређених пакета после обраде чвора d, рецимо $I(d)$, са истим скупом након обраде сваког чвора на нивоу p. Како ова провера може захтевати пуно времена, скуп $I(d)$ се упоређује са истим скупом добијеним од братских чворова, тј. чворова на нивоу p који су генерисани од истог чвора коришћењем различитих правила избора и/или правила гиљотинског дељења.

Хеуристичко орезивање

Друга стратегија је хеуристичко орезивање чворова на основу тренутног делимичног решења. Нека су $A = \sum_{i \in N} w_i h_i$ и z укупна површина пакета и вредност постојећег решења, респективно, где је $N = \{1, ..., n\}$ скуп пакета које треба распоредити (иницијално, $z = n$). Скуп $I(d)$ представља скуп нераспоређених пакета након испитивања датог чвора d на нивоу p.

Рачуна се просечно пуњење сваког складишта у парцијалном решењу на тренутном чвору d, које се означава са AVG, и ова вредност се пореди са мером просечног пуњења сваког складишта у решењу које користи једно складиште мање од постојећег. Конкретно, ако је услов

$$AVG := \frac{A - \sum_{i \in I(d)} w_i h_i}{pWH} \leq \alpha \frac{A}{(z-1)WH} \tag{3.1}$$

задовољен, тренутни чвор је већ досегнут, тј. тренутно решење има просечно пуњење већ коришћених складишта и неће довести до побољшања постојећег решења. Вредност параметра α из (3.1) се ажурира током извршења алгоритма: на почетку се ставља да је $\alpha = 0$, тј. на почетку ниједан чвор није орезан. Након што се истраже бројни чворови, нпр. њих N_1, без побољшања постојећег, повећава се параметар α за вредност δ, како би се заобишла комплетна енумерација. При расту параметра α (до 1) могуће је понављање N_2 чворова који не доводе до побољшања.

3.2.3 Експериментални резултати

Основни алгоритам описан у одељку, 3.2.1 (у наставку означен са BSC) и његова побољшана верзија (ENH) из одељка 3.2.2 кодирани су у програмском језику C и тестирани на рачунару са процесором Intel Xeon E3-1220V2, чија је брзина 3.10 GHz. Прелиминарни експериментални резултати сугерисали су постављање $N_1 = 500$, $N_2 = 500$ и $\delta = 0.1$ за алгоритам ENH. За тестирање је разматрано десет класа инстанци представљених у [6] и [57] за 2BPP проблем. Свака класа садржи 50 инстанци (по 10 инстанци за сваку вредност $n \in \{20, 40, 60, 80, 100\}$), тако да је разматрано 500 инстанци. Резултати су преузети из [55].

Првих шест класа које су представили Berkey и Wang у [6] окарактерисане су на следећи начин:

Класа 1: w_i и h_i равномерно распоређени случајни бројеви на $[1, 10]$, $W = H = 10$;

Класа 2: w_i и h_i равномерно распоређени случајни бројеви на $[1, 10]$, $W = H = 30$;

Класа 3: w_i и h_i равномерно распоређени случајни бројеви на $[1, 35]$, $W = H = 40$;

Класа 4: w_i и h_i равномерно распоређени случајни бројеви на $[1, 35]$, $W = H = 100$;

Класа 5: w_i и h_i равномерно распоређени случајни бројеви на на $[1, 100]$, $W = H = 100$;

Класа 6: w_i и h_i равномерно распоређени случајни бројеви на на $[1, 100]$, $W = H = 300$;

Додатне четири класе су представили Martello и Vigo у [57], а пакети су класификовани у четири типа:

Тип 1: w_i равномерно распоређен случајни број на $[\frac{2}{3}W, W]$, h_i равномерно распоређен случајни број на $[1, \frac{1}{2}H]$;

Тип 2: w_i равномерно распоређен случајни број на $[1, \frac{1}{2}W]$, h_i равномерно распоређен случајни број на $[\frac{2}{3}H, H]$;

Тип 3: w_i равномерно распоређен случајни број на $[\frac{1}{2}W, W]$, h_i равномерно распоређен случајни број на $[\frac{1}{2}H, H]$;

Тип 4: w_i равномерно распоређен случајни број на $[1, \frac{1}{2}W]$, h_i равномерно распоређен случајни број на $[1, \frac{1}{2}H]$;

Величине складишта за све класе су $W = H = 100$, док за пакете важи следеће:

Класа 7: тип 1 са вероватноћом 70%, тип 2, 3, 4 сваки са вероватноћом 10%;

Класа 8: тип 2 са вероватноћом 70%, тип 1, 3, 4 сваки са вероватноћом 10%;

Класа 9: тип 3 са вероватноћом 70%, тип 1, 2, 4 сваки са вероватноћом 10%;

Класа 10: тип 4 са вероватноћом 70%, тип 1, 2, 3 сваки са вероватноћом 10%;

Табеле 3.1 и 3.2 су преузете из [55], и показују резултате експеримената за алгоритме BSC и ENH, респективно. Сваки је тестиран коришћењем различитих временских ограничења: 60, 600 и 1800 CPU секунди. За свако временско ограничење, табеле извештавају о

- суми најбољег решења нађених датом хеуристиком,
- броју инстанци за које је пронађено решење доказиво оптимално,
- просечној процентуалној грешци-за сваку инстанцу процентуална грешка је рачуната као $(UB - LB)/LB$, где је UB вредност решења нађеног алгоритмом, а LB означава најпознатију вредност доње границе.

За сваку класу и вредност n дата је сума најпознатијих доњих граница, LB, за десет инстанци 2BP|O|F проблема, тј. када ограничење гиљотине није дато [1]. Имајући у виду да је 2BP|O|F релаксација проблема 2BP|O|G, овај број даје границу оптималне

[1] Најбоље доње и горње границе за 2BP|O|F проблем су узете из [59] и доступне су на сајту http://www.or.deis. unibo.it/research_pages/ORinstances/ORinstances.htm

вредности решења. На крају, последњи ред обе табеле даје исте сумиране вредности у односу на цео тестирани проблем.

Резултати рачунарских експеримената, преузети из [55], показују да је, чак и са најмањим временским ограничењем од 60 секунди, основни алгоритам у стању да нађе оптимално решење за 361 од 500 инстанци, а просечна грешка је испод $3,6\%$. Побољшања која се могу добити са већим временским ограничењем су ограничена, али регуларна: са 1800 секунди алгоритам штеди 24 складишта и доказује оптималност за још 20 инстанци, са просечним раскораком испод $3,4\%$. Имајући у виду да се добијена решења пореде са доњим границама (или оптималним вредностима) за релаксацију проблема, у којој ограничења гиљотине нису наметнута, ови резултати показују да је чак и основна верзија алгоритма прилично ефикасна. Алгоритам достиже временско ограничење од 1800 секунди у 193 инстанце; стога, могу се очекивати неки побољшани резултати у случају да је дозвољено веће временско ограничење. У преосталих 307 инстанци алгоритам је завршен када је испитао све конфигурације које су (хеуристички) генерисане; у просеку је за то било потребно 38 секунди и процена 24 милиона чворова.

Резултати су још бољи за побољшани алгоритам који је у стању да оптимално реши 386 инстанци у ограничењу од 60 секунди, а просечна грешка у процентима је приближно $3,3\%$. Може се приметити да су ови бројеви незнатно бољи од резултата добијених основним алгоритмом у временском ограничењу од 30 минута. Стога, није изненађујуће то да се са дужим временским ограничењима могу постићи само маргинална побољшања над овим решењима. Конкретно, повећање временског ограничења на 1800 CPU секунди доводи до уштеде од 8 складишта и додатних 7 оптималних решења. Ова верзија алгоритма је достигла временско ограничење у 124 случаја; за преостале инстанце време рачунања је слично оном у основном алгоритму (31 секунда), док је број чворова смањен на 16 милиона.

Да би се оценио квалитет добијених резултата, у табели 3.3, преузетој из [55], је дата информација о најбољим доњим и горњим границама за 2BP|O|F проблем (преузетих из [59]) и ове вредности су упоређене са резултатима из табеле 3.2 за алгоритам ENH. На крају, последње две колоне показују сличне информације за branch-and-price алгоритам за 2BP|O|G проблем, представљен у [61]. Међутим, како вредности горњих граница нису директно доступне у [61], за овај алгоритам дата је сума доњих граница које су рачунате на коренском чвору, и број доказаних оптималних решења[2](узетих из табела 4 и 7 из [61], респективно).

Табела 3.3 показује да је глобалан број складишта коришћених за 500 инстанци од стране ENH алгоритма за око $0,5\%$ већи од броја складишта коришћених за 2BP|O|F проблем (7281 наспрам 7241). Ово потврђује да је квалитет решења добијен представљеним алгоритмом упоредив са најбољим решењима проблема без ограничења гиљотине, и да мора доћи до маргиналног раста у вредности решења када је наметнуто ограничење гиљотине. Како су представљени алгоритми специјализовани за 2BP|O|G проблем, примећује се да, иако су доње границе израчунате у [61] боље, број доказаних оптималних решења за алгоритам ENH је већи него у [61]. Поређење добијених вредности хеуристичких решења са вредностима доњих граница између друге и седме колоне довело би до оптималности за 21 додатан случај, тј. за 414 од 500 инстанци.

[2]Број оптималних решења у [61] односи се на комплетан branch-and-price алгоритам.

			TL=60			TL=600			TL=1800		
Класа	n	LB	UB	$\#opt$	$\%gap$	UB	opt	$\%gap$	UB	$\#opt$	$\%gap$
	20	71	71	10	0.000	71	10	0.000	71	10	0.000
	40	134	134	10	0.000	134	10	0.000	134	10	0.000
1	60	197	200	7	0.017	200	7	0.017	200	7	0.017
	80	274	275	9	0.004	275	9	0.004	275	9	0.004
	100	317	318	9	0.003	317	10	0.000	317	10	0.000
	20	10	10	10	0.000	10	10	0.000	10	10	0.000
	40	19	20	9	0.100	20	9	0.100	20	9	0.100
2	60	25	25	10	0.000	25	10	0.000	25	10	0.000
	80	31	32	9	0.033	32	9	0.033	32	9	0.033
	100	39	39	10	0.000	39	10	0.000	39	10	0.000
	20	51	54	7	0.078	54	7	0.078	54	7	0.078
	40	92	96	6	0.058	96	6	0.058	96	6	0.058
3	60	136	141	5	0.038	140	6	0.032	140	6	0.032
	80	187	195	2	0.044	194	3	0.039	194	3	0.039
	100	221	230	2	0.045	228	4	0.035	228	4	0.035
	20	10	10	10	0.000	10	10	0.000	10	10	0.000
	40	19	19	10	0.000	19	10	0.000	19	10	0.000
4	60	23	25	8	0.100	25	8	0.100	25	8	0.100
	80	30	33	7	0.100	33	7	0.100	33	7	0.100
	100	37	39	8	0.067	39	8	0.067	39	8	0.067
	20	65	66	9	0.020	66	9	0.020	66	9	0.020
	40	119	119	10	0.000	119	10	0.000	119	10	0.000
5	60	179	182	7	0.020	182	7	0.020	181	8	0.013
	80	241	247	4	0.026	247	4	0.026	247	4	0.026
	100	279	288	2	0.035	288	2	0.035	288	2	0.035
	20	10	10	10	0.000	10	10	0.000	10	10	0.000
	40	15	19	6	0.400	19	6	0.400	19	6	0.400
6	60	21	22	9	0.050	22	9	0.050	22	9	0.050
	80	30	30	10	0.000	30	10	0.000	30	10	0.000
	100	32	35	7	0.100	35	7	0.100	35	7	0.100
	20	55	55	10	0.000	55	10	0.000	55	10	0.000
	40	109	113	6	0.038	113	6	0.038	113	6	0.038
7	60	156	162	5	0.037	161	5	0.032	159	7	0.019
	80	224	235	1	0.051	233	1	0.041	232	2	0.037
	100	269	279	1	0.037	277	3	0.030	277	3	0.030
	20	58	58	10	0.000	58	10	0.000	58	10	0.000
	40	112	114	8	0.017	113	9	0.009	113	9	0.009
8	60	159	163	6	0.025	162	7	0.018	162	7	0.018
	80	223	230	3	0.031	228	5	0.022	227	6	0.018
	100	274	282	3	0.029	281	3	0.025	280	4	0.022
	20	143	143	10	0.000	143	10	0.000	143	10	0.000
	40	278	278	10	0.000	278	10	0.000	278	10	0.000
9	60	437	437	10	0.000	437	10	0.000	437	10	0.000
	80	577	577	10	0.000	577	10	0.000	577	10	0.000
	100	695	695	10	0.000	695	10	0.000	695	10	0.000
	20	42	44	8	0,045	44	8	0.045	44	8	0.045
	40	74	74	10	0,000	74	10	0.000	74	10	0.000
10	60	98	103	5	0,053	102	6	0.045	102	6	0.045
	80	123	130	3	0,056	130	3	0.056	130	3	0.056
	100	153	163	0	0,066	162	1	0.059	161	2	0.052
Глобално		7173	7319	361	0.036	7302	374	0.035	7295	381	0.034

Табела 3.1: Резултати основног алгоритма за 2BPP проблем на инстанцама из литературе.

			TL=60			TL=600			TL=1800		
Класа	n	LB	UB	$\#opt$	$\%gap$	UB	$\#opt$	$\%gap$	UB	$\#opt$	$\%gap$
1	20	71	71	10	0,000	71	10	0,000	71	10	0,000
	40	134	134	10	0.000	134	10	0.000	134	10	0.000
	60	197	200	7	0.017	200	7	0.017	200	7	0.017
	80	274	275	9	0.004	275	9	0.004	275	9	0.004
	100	317	317	10	0.000	317	10	0.000	317	10	0.000
2	20	10	10	10	0.000	10	10	0.000	10	10	0.000
	40	19	20	9	0.100	20	9	0.100	20	9	0.100
	60	25	25	10	0.000	25	10	0.000	25	10	0.000
	80	31	32	9	0.033	32	9	0,033	32	9	0.033
	100	39	39	10	0.000	39	10	0.000	39	10	0.000
3	20	51	54	7	0.078	54	7	0.078	54	7	0.078
	40	92	96	6	0.058	96	6	0.058	96	6	0.058
	60	136	140	6	0.032	140	6	0.032	140	6	0.032
	80	187	190	7	0.017	190	7	0.017	190	7	0.017
	100	221	226	5	0.024	226	5	0.024	225	6	0.019
4	20	10	10	10	0.000	10	10	0.000	10	10	0.000
	40	19	19	10	0.000	19	10	0.000	19	10	0.000
	60	23	25	8	0.100	25	8	0.100	25	8	0.100
	80	30	33	7	0.100	33	7	0.100	33	7	0.100
	100	37	39	8	0.067	39	8	0.067	39	8	0.067
5	20	65	66	9	0.020	66	9	0.020	66	9	0.020
	40	119	119	10	0.000	119	10	0.000	119	10	0.000
	60	179	181	8	0.013	181	8	0.013	181	8	0.013
	80	241	247	4	0.026	247	4	0.026	247	4	0.026
	100	279	288	2	0.035	288	2	0.035	286	4	0.027
6	20	10	10	10	0.000	10	10	0.000	10	10	0.000
	40	15	19	6	0.400	19	6	0.400	19	6	0.400
	60	21	22	9	0.050	22	9	0.050	22	9	0.050
	80	30	30	10	0.000	30	10	0.000	30	10	0.000
	100	32	35	7	0.100	35	7	0.100	35	7	0.100
7	20	55	55	10	0.000	55	10	0.000	55	10	0.000
	40	109	113	6	0.038	113	6	0.038	113	6	0.038
	60	156	159	7	0.019	159	7	0.019	159	7	0.019
	80	224	232	2	0.037	232	4	0.037	232	2	0.037
	100	269	277	3	0.030	276	2	0.026	275	4	0.022
8	20	58	58	10	0.000	58	10	0.000	58	10	0.000
	40	112	113	9	0.009	113	9	0.009	113	9	0.009
	60	159	162	7	0.018	162	7	0.018	162	7	0.018
	80	223	227	6	0.018	227	6	0.018	226	7	0.014
	100	274	281	3	0.025	280	4	0.022	280	4	0.022
9	20	143	143	10	0.000	143	10	0.000	143	10	0.000
	40	278	278	10	0.000	278	10	0.000	278	10	0.000
	60	437	437	10	0.000	437	10	0.000	437	10	0.000
	80	577	577	10	0.000	577	10	0.000	577	10	0.000
	100	695	695	10	0.000	695	10	0.000	695	10	0.000
10	20	42	44	8	0.045	44	8	0.045	44	8	0.045
	40	74	74	10	0.000	74	10	0.000	74	10	0.000
	60	98	102	6	0.045	102	6	0.045	102	6	0.045
	80	123	130	3	0.056	130	3	0.056	130	3	0.056
	100	153	160	3	0.046	159	4	0.040	159	4	0.040
Глобално		7173	7289	386	0.033	7286	389	0.033	7281	393	0.033

Табела 3.2: Резултати побољшаног алгоритма за 2BPP проблем на инстанцама из литературе са $N_1 = 500$, $N_2 = 500$ и $\delta = 0.1$.

	2BP|O|F		ENH			2BP|O|G	
Класа	LB	UB	UB	$\#opt$	$\%gap$	LB	$\#opt$
1	993	997	997	46	0.004	997	47
2	124	124	126	48	0.027	125	30
3	687	696	705	32	0.041	694	44
4	119	124	126	43	0.053	119	27
5	883	892	899	35	0.017	889	46
6	108	112	116	42	0.110	110	30
7	813	827	834	29	0.023	812	34
8	826	835	839	37	0.013	826	41
9	2130	2130	2130	50	0.000	2130	50
10	490	504	509	31	0.037	491	28
Глобално	7173	7241	7281	393	0.033	7193	377

Табела 3.3: Поређење између гиљотинске и негиљотинске хеуристике. Временско ограничење = 1800 CPU секунди.

3.3 Генетски алгоритам са Crow претрагом за решавање 2BP|O|F проблема

Генетски алгоритам је описан у одељку 1.7.5, па је у наставку, пре представљања генетског алгоритма са Crow претрагом, укратко објашњен само алгоритам Crow претраге.

Алгоритам Crow претраге

Алгоритам Crow претраге (енг. Crow Search Algorithm), у наставку CSA, је заснован на понашању врана (енг. crow) приликом скривање хране и њеном каснијем проналажењу. CSA укључује јато од N врана. Свака врана је кодирана својом позицијом (скровиштем) у $d-$димензионом окружењу $x_i(G) = (x_i^1(G), x_i^2(G), ..., x_i^d(G))$, где је $i \in \{1, ..., N\}$, а G генерацијски број. Свака позиција вране одговара допустивом обрасцу у простору претраге. Свака врана i има меморијски простор $m_i(G)$ у генерацији G, где се чува податак о њеној најбољој позицији. Квалитет позиције се мери функцијом погодности, односно функцијом циља оптимизационог проблема. CSA започиње са почетном позицијом врана и постављањем меморије на њихове почетне позиције, а затим итеративно ажурира позиције и меморију врана по следећим формулама:

$$x_i(G) = \begin{cases} x_{i-1}(G) + r_i FL(m_j(G-1) - x_i(G-1)), & \text{ако } r_j \geq AP \\ \text{случајна позиција}, & \text{иначе} \end{cases} \tag{3.2}$$

$$m_i(G) = \begin{cases} x_i(G), & \text{ако } f(x_i(G)) \geq f(m_i(G-1)) \\ m_i(G-1), & \text{иначе.} \end{cases} \tag{3.3}$$

Заправо, у свакој генерацији, врана i случајно бира врану j из јата како би је пратила и украла храну. Ако врана j зна за намеру вране i, онда она заварава врану i крећући се према случајној позицији. Са FL је означена дужина лета вране, са AP вероватноћа свести вране j о намери вране i, са $f(\cdot)$ функцију погодности, док r_i и r_j представљају равномерно распоређене случајне бројеве на интервалу $[0, 1]$. Алгоритам се зауставља када се достигне максималан број генерација. Онда извлачи најбољу меморију из јата као глобални оптимум. Детаљније о CSA алгоритму може се наћи у [1].

3.3.1 Генетски алгоритам са Crow претрагом

У наставку је представљен хибридни алгоритам за решавање 2BP|O|F проблема, преузет из [44]. Овај алгоритам комбинује генетски алгоритам, GA, са CSA алгоритмом, и назива се генетски алгоритам са Crow претрагом (енг. Crow Search-based Genetic Algorithm - CSGA). Мотивација за њихово комбиновање је, прво, паметно понашање врана у CSA алгоритму, а затим и еволуцијско понашање јединке у GA. Осим тога, GA оператори се могу лако и директно применити у процесу бинарне претраге. У поређењу са GA, у CSGA свака јединка популације учествује са сопственим информацијама у операцији укрштања. Дакле, вероватноћа укрштања се не узима у обзир и алгоритму није потребна стратегија замене за креирање предстојеће генерације. У поређењу са CSA, у CSGA свака врана i бира једну врану у јату коју ће пратити коришћењем оператора селекције. Додатно, оператор мутације се примењује са вероватноћом мутације, како би се повећала разноврсност популације. У наставку следи детаљан опис CSGA алгоритма, преузет из [44].

Почетна популација

Позиције врана у $d-$димензионом окружењу се одређују распоређивањем или нераспоређивањем датог пакета у дато складиште. Тако се свака позиција разматра као потенцијално решење проблема. Она се кодира као $n \times d$ матрица, где је n број пакета које је потребно спаковати у складишта, а d број расположивих складишта. Решење i, у популацији која има N врана, се представља на следећи начин:

$$X^i = (x^i_{jk}) \in \{0,1\}^{n \times d};\ \text{за } i \in \{1,...,N\}. \tag{3.4}$$

За све i, променљива x^i_{jk} узима вредност 1 ако се $j-$ти пакет пакује у $k-$то складиште, а 0 иначе. Ради јасноће, један пакет се додељује једном и само једном складишту, а затим се уклања са листе пакета. Штавише, паковање мора задовољавати Bottom-Left стратегију, односно сваки пакет се пакује на најнижу позицију што је могуће више лево.

Матрица меморије M^i за сваку врану има исте димензије као матрица позиције X^i. У почетној фази, коефицијенти матрице позиције су генерисани случајно и она се поклапа са матрицом меморије, за сваку врану i.

Процена допустивости и квалитета решења

Са једне стране, случајни процес коришћен у првој фази може генерисати недопустива решења. Решење се сматра недопустивим ако крши ограничење капацитета (видети једначину 3.5) најмање једног употребљеног складишта:

$$\text{За свако } i \in \{1,...,N\} : \sum_{j=1}^{n} w_j h_j x^i_{jk} \leq WHy_k;\ \forall k \in \{1,...,d\} \tag{3.5}$$

где је

$$y_k = \begin{cases} 1 & \text{ако се користи складиште } k, \\ 0 & \text{иначе.} \end{cases} \tag{3.6}$$

Складиште k се сматра употребљеним ако је $\sum_{j=1}^{n} x^i_{jk} \geq 1$, тј. садржи најмање један пакет, а неупотребљеним ако је $\sum_{j=1}^{n} x^i_{jk} = 0$.

Оператор поправке се употребљава за одстрањивање пакета из складишта у којима је прекорачен капацитет, и смештање тих пакета у недовољно попуњена складишта.

Са друге стране, за процену квалитета допустивих решења користи се следећа

функцију погодности:

$$f = \frac{1}{\sum_{k=1}^{d} y_k}. \tag{3.7}$$

Што је мање складишта за паковање пакета коришћено, то је већа вредност функције погодности.

Ажурирање меморије и позиције врана

GA оператори и CSA алгоритам се примењују како би се ажурирала позиција врана. Од сада позиције врана разматрамо као хромозоме. Свака колона матрице позиције одговара гену хромозома, док је сваки ген кодиран као бинарни вектор од n алела (различитих облика истог гена). Слика 3.3 приказује структуру хромозома.

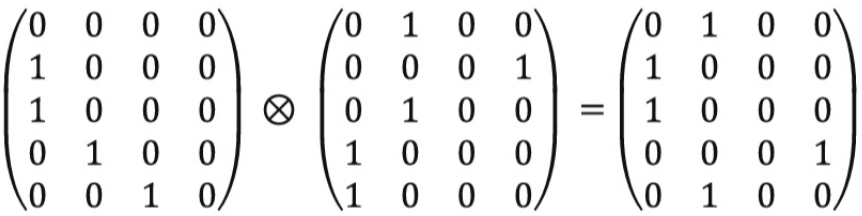
$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

Слика 3.3: Представљање хромозома са $(n = 5)$ и $(d = 4)$.

Селекција. У овој фази свака врана i бира врану j из популације, уз помоћ бинарног такмичења. Бинарно такмичење је заправо бирање две вране из популације, на случајан начин, и њихово надметање. Победник је она врана чија је вредност функције погодности већа. Тако врана i бира победника као мету коју треба пратити. На тај начин се њена позиција ажурира.

Укрштање. Укрштање је процес размене гена између јединки, како би се добила нова јединка. Његов циљ је ширење простора претраге решења. У овој фази се комбинују CSA правила са стратегијом укрштања, како би се ажурирале позиције, и то на следећи начин. Ако је $r_j \geq AP_j$, укршта се позиција вране i са позицијом изабране вране j. Иначе, позиција вране i се укршта са њеном меморијом. PMX оператор укрштања, који су представили Goldberg и Lingle у [31], се користи како би се рекомбиновале две позиције, или чак позиција и меморија. Пример на слици 3.4 показује како се он користи у овом случају. Насумично се бира складиште од првог родитеља (нпр. складиште 1), и мења његов садржај (пакет 2 и пакет 3) са потомком. Остала складишта наслеђују садржај од другог родитеља. Дупликати су замењени пакетима спакованим у складиште 1 другог родитеља, поштујући исти ред. У приказаном примеру, пакети 2 и 3 су спаковани у складишта 4 и 2 другог родитеља. Стога, пакет 2 се замењује са пакетом 4 и пакет 3 са пакетом 5. Знак $\otimes$ означава оператор укрштања.

Када се заврши процес укрштања, проверава се допустивост позиција потомака. Ако је потребно, примењује се горепоменути оператор преправке, како би се вратила изводљивост решења.



Слика 3.4: Пример оператора укрштања са $(n = 5)$ и $(d = 4)$.

Мутација. Оператор мутације се, као и оператор укрштања, користи за ширење простора претраге, како би се нашла боља решења. Неки генерисани потомци мутирају коришћењем механизма поделе складишта, представљеног у [47]. Он укључује бирање једног складишта на случајан начин, и поделу његовог садржаја у два складишта. У нашем алгоритму, овај механизам се примењује на следећи начин. Први део пакета се чува у оригиналном складишту, а други се спаја са садржајем другог складишта, које може да прими те пакете, а да се не прекорачи његов капацитет. Ако се ти пакети не могу спаковати ни у једно коришћено складиште, отвара се ново.

Ажурирање меморије. Након мутације, квалитет сваке потомак позиције се проверава коришћењем функције погодности, и пореди са меморијом сваке вране. Ако је потомак позиција боља од до сада запамћеног положаја, онда се меморија вране ажурира са потомак позицијом. Иначе се задржава претходна меморија.

Конструкција CSGA **алгоритма**

Целокупни алгоритам се укратко може описати на следећи начин.

1. Иницијализовати почетну популацију са N врана, и поставити контролне параметре, као што је максималан генерацијски број, дужина лета вране и вероватноћа њихове свести о намери друге вране.

   За сваку врану i, поставити $M^i(G = 1) = X^i(G = 1)$.

2. Проценити допустивост решења и вратити допустивост недопустивих решења помоћу оператора поправке. Затим, израчунати њихову погодност.

3. За сваку врану i, применити бинарно такмичење како би се изабрала врана j.

4. Ажурирати позицију сваке вране i коришћењем следећих једнакости:

$$\text{Ако је } r_j \geq AP_j, \text{ тада је } X^i(G+1) = X^i(G) \otimes X^j(G). \tag{3.8}$$

$$\text{Иначе, } X^i(G+1) = X^i(G) \otimes M^i(G). \tag{3.9}$$

5. Поправити допустивост недопустивих потомак позиција.

6. Применити мутацију за генерисање потомак позиција са вероватноћом p_m.

7. Израчунати вредност функције погодности за сваку потомак позицију и ажурирати меморију сваке вране помоћу једначине (3.3).

8. Вратити се на 3. корак све док се не достигне максималан број генерација, а затим најбољу пронађену меморију прогласити за оптимално решење 2BP|O|F проблема.

3.3.2 Експериментални резултати

CSGA је кодиран у програмском језику JAVA на рачунару са Intel Core i5 процесором, чија је брзина 2.5 GHz, 4.0 GB RAM меморије и 64-битним Windows 7 оперативним системом. Тестови су спроведени на десет класа, описаних у одељку 3.2.3, с тим што су резултати приказани само за 4 најрепрезентативније, 1, 3, 5. и 9. Како би се видела ефикасност алгоритма, упоређен је са GA алгоритмом, јер користе исте генетске операторе. Контролни параметри су приказани у табели 3.4, преузетој из [44]. То су: $Popsize$, који означава величину популације; $Maxgen$, означава максималан број генерација; $BTS-size$, означава величину бинарног такмичења; FL, означава дужину лета вране; AP, означава вероватноћу свести; p_m, вероватноћа мутације; p_c, ниво поузданости.

Алгоритам	Вредност параметра
CSGA	$Popsize = 100; FL = 2; AP = 0.01; pm = 0.10; Maxgen = 100$
GA	$Popsize = 100; BTS - size = 2; pc = 0.95; pm = 0.01; Maxgen = 100$

Табела 3.4: подешавање параметара

Сви резултати тестирања су процењени након 30 покретања, и задржани су просечни резултати. Да би се проценили резултати, у смислу погодности, примењен је Wilcoxon тест, да би се утврдило да ли се вредности функције погодности CSGA и GA значајно разликују или не. Ниво поузданости је фиксиран на 0.95, а добијени резултати $p-$ вредности су приказани у табели 3.5, преузетој из [44]. За тестирање је коришћен SPSS statistics 22. Ако је $p-$ вредност мања од 0.05, одбацује се нулта хипотеза, која претпоставља да нема значајне разлике између два алгоритма. У супротном прихвата се алтернативна хипотеза, која претпоставља да постоји значајна разлика између њих. Вредност $R-$ (односно $R+$) у табели 3.5 означава суму рангова одговарајуће негативне (односно позитивне) разлике. Заправо, сума рангова је апсолутна вредност ралике између резултата два алгоритма. Вредност s у табели 3.5 показује статистички резултат поређења два пара: $s = 1$ показује да је први алгоритам значајно бољи од другог, док $s = -1$ показује да је први алгоритам значајно лошији од другог. Обе вредности, -1 и 1, показују значајну разлику. Уколико је $s = 0$, то значи да између алгоритама нема значајне разлике. Из табеле 3.5 се може видети да је CSGA бољи од GA алгоритма.

Табеле 3.6, 3.7 и 3.8, преузете из [44], показују емпиријске резултате у погледу просечно коришћених складишта, добијених након 30 покретања, и просечног времена извршења. Ови тестови су изведени над класом 9. Из ових табела се може видети да је број коришћених складишта од стране CSGA алгоритма у свакој инстанци близак или једнак доњој граници. Штавише, даје боље резултате, за краће време у односу на GA. Бољи резултати су подебљани. Време је изражено у секундама.

Класа	Број пакета	Поређени алгоритми	R+	R-	$p-$ вредност	s
Класа 1	20	CSGA - GA	55	0	0.005	1
	60	CSGA - GA	55	0	0.005	1
	100	CSGA - GA	55	0	0.005	1
Класа 3	20	CSGA - GA	55	0	0.004	1
	60	CSGA - GA	55	0	0.005	1
	100	CSGA - GA	55	0	0.005	1
Класа 5	20	CSGA - GA	55	0	0.005	1
	60	CSGA - GA	55	0	0.005	1
	100	CSGA - GA	55	0	0.005	1

Табела 3.5: Резултати Wilcoxon теста просечне вредности функције погодности CSGA наспрам GA

Тест инстанца	LB	CSGA		GA	
		Коришћена складишта	CPU време	Коришћена складишта	CPU време
Инстанца 1	**25**	29	**0.067**	37	2.013
Инстанца 2	32	**29**	**0.066**	36	1.916
Инстанца 3	**29**	**29**	**1.914**	35	1.914
Инстанца 4	31	**28**	**0.063**	35	2.085
Инстанца 5	**27**	**27**	**0.064**	33	1.993
Инстанца 6	**29**	**29**	**0.067**	36	2.030
Инстанца 7	**24**	26	**0.068**	32	1.920
Инстанца 8	**26**	28	**0.063**	35	2.006
Инстанца 9	**21**	25	**0.062**	33	1.995
Инстанца 10	34	**30**	**0.064**	38	2.028
Просек	**27,8**	28	**0.250**	35	1.990

Табела 3.6: Резултати примене алгоритама CSGA и GA на инстанце са 40 пакета

Тест инстанца	LB	CSGA		GA	
		Коришћена складишта	CPU време	Коришћена складишта	CPU време
Инстанца 1	**59**	60	**0.264**	70	7.816
Инстанца 2	**58**	59	**0.271**	69	7.718
Инстанца 3	**57**	59	**0.252**	70	8.001
Инстанца 4	**53**	60	**0.263**	69	7.753
Инстанца 5	62	**61**	**0.259**	74	7.772
Инстанца 6	62	**59**	**0.251**	71	7.576
Инстанца 7	**59**	**59**	**0.262**	70	7.833
Инстанца 8	**58**	**58**	**0.264**	71	7.902
Инстанца 9	**49**	54	**0.261**	59	40.232
Инстанца 10	**60**	61	**0.274**	74	7.575
Просек	**57.7**	59	**0.262**	69.7	11.018

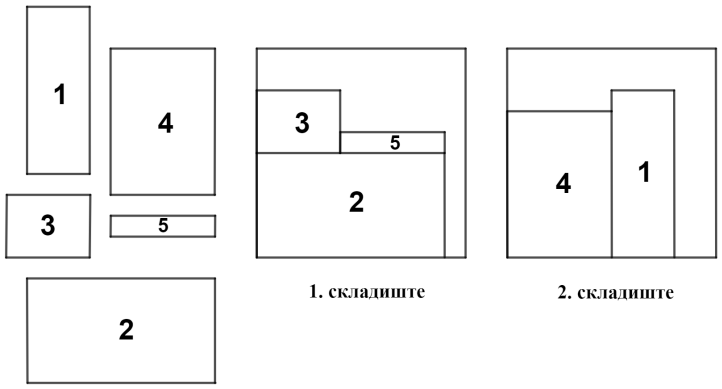
Табела 3.7: Резултати примене алгоритама CSGA и GA на инстанце са 80 пакета

Тест инстанца	LB	CSGA		GA	
		Коришћена складишта	CPU време	Коришћена складишта	CPU време
Инстанца 1	**71**	75	**0.377**	87	11.755
Инстанца 2	**64**	69	**0.371**	86	11.785
Инстанца 3	68	**61**	**0.384**	82	11.767
Инстанца 4	78	**76**	**0.372**	92	11.872
Инстанца 5	**65**	73	**0.388**	84	11.892
Инстанца 6	**71**	74	**0.381**	78	64.425
Инстанца 7	**66**	69	**0.380**	87	12.084
Инстанца 8	74	**73**	**0.375**	90	11.859
Инстанца 9	**66**	**66**	**0.379**	85	12.039
Инстанца 10	**72**	75	**0.384**	84	64.165
Просек	**69.5**	71.1	**0.379**	85.5	22.364

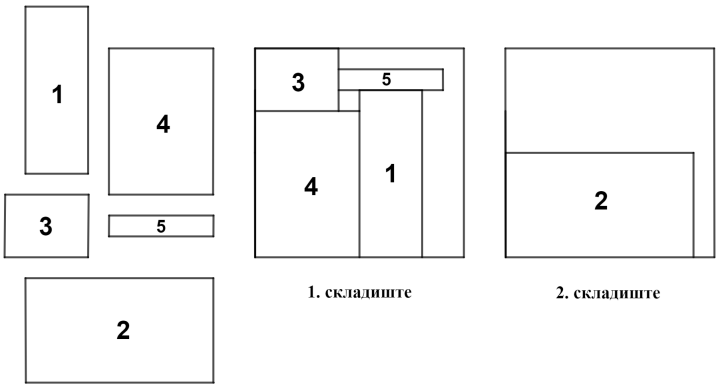
Табела 3.8: Резултати примене алгоритама CSGA и GA на инстанце са 100 пакета

3.4 Поређење представљених хеуристика

Обе представљене хеуристике тестиране су на на десет класа, наведених у 3.2.3. За алгоритам парцијалне енумерације су приказани резултати за свих десет класа, док су за генетски алгоритам са Crow претрагом дати резултати само за четири класе: 1, 3, 5. и 9. За разлику од прве хеуристике, где су експерименти вршени за временска ограничења од 60, 600 и 1800 секунди, код друге хеуристике је приказано време за које се добија оптимално решење. Узимајући у обзир експерименте спроведене на 9. класи, са 40, 80 и 100 пакета, и имајући у виду да је просечно време потребно за добијање оптималних решења код алгоритма парцијалне енумерације (за обе варијанте алгоритма) за највећи број инстанци 31 секунда, прво што се може закључити јесте да је генетски алгоритам са Crow претрагом бржи, с обзиром на то да је његово време извршења за све случајеве мањи од 4 секунде. Са друге стране, у погледу броја коришћених складишта, оба алгоритма дају решења блиска познатим доњим границама. Оно што прави разлику је коришћење простора складишта, што се може видети у примеру на сликама 3.5 и 3.6. Дата су два складишта димензија 5×5 и пакети: пакет 1 димензија 1.5×4, пакет 2 димензија 4.5×2.5, пакет 3 димензија 2×1.5, пакет 4 димензија 2.5×3.5 и пакет 5 димензија 2.5×0.5. У случају када је коришћена хеуристика са ограничењем гиљотине, неискоришћена површина складишта је 9.5, док је у случају без ограничења гиљотине та вредност 6. Дакле, боља искоришћеност простора се постиже уколико није наметнуто ограничење гиљотине.



Слика 3.5: Примена хеуристике са ограничењем гиљотине.



Слика 3.6: Примена хеуристике без ограничења гиљотине.

Глава 4

Закључак

У овом раду разматран је дводимензиони проблем паковања у складишта, у којем пакети не могу бити ротирани и где је наметнуто ограничење гиљотине. За овај проблем, означен као 2BP|O|G, је представљен хеуристички алгоритам заснован на парцијалној енумерацији, преузет из [54]. Опсежна рачунска анализа на великом броју инстанци из литературе, преузета из [44], показује да је алгоритам у стању да реши више од 78% проблема, са просечним раскораком од $3,3\%$. Поред алгоритма парцијалне енумерације, представљен је и генетски алгоритам са Crow претрагом, преузет из [44], који је намењен решавању 2BP|O|F проблема. Након упоређивања ова два алгоритма, закључује се да хеуристика без ограничења гиљотине ефикасније користи простор, па будућа истраживања треба да буду усмерена на проналажење алгоритма који ће брзо решавати проблем паковања у складишта са ограничењем гиљотине и боље користити простор складишта.

Литература

[1] A. Askarzadeh A novel metaheuristic method for solving constrained engineering optimization problems: crow search algorithm. Comput. Struct. 169, 1–12, 2016.

[2] B. S. Baker, E. G. Coffman, Jr, R. L. Rivest. Orthogonal packings in two dimensions. SIAM Journal on Computing, 9(4):846-855, 1980.

[3] N. Bansal, A. Caprara, M. Sviridenko. Improved approximation algorithms for multidimensional bin packing problems. In Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on, pages 697-708. IEEE, 2006.

[4] N. Bansal, M. Sviridenko. New approximability and inapproximability results for 2-dimensional bin packing. In Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms, pages 196-203. Society for Industrial and Applied Mathematics, 2004.

[5] J. E. Beasley. An exact two-dimensional non-guillotine cutting tree search procedure. Operations Research, 33(1):49-64, 1985.

[6] J. Berkey, P. Wang. Two-dimensional finite bin-packing algorithms. Journal of the Operational Research Society, 38:423-429, 1987.

[7] M. A. Boschetti, A. Mingozzi. The two-dimensional finite bin packing problem. part i: New lower bounds for the oriented case. Quarterly Journal of the Belgian, French and Italian Operations Research Societies, 1(1):27-42, 2003

[8] M. A. Boschetti, A. Mingozzi. The two-dimensional finite bin packing problem. part ii: New lower and upper bounds. Quarterly Journal of the Belgian, French and Italian Operations Research Societies, 1(2):135-147, 2003.

[9] A. Caprara. Packing 2-dimensional bins in harmony. In Foundations of Computer Science, 2002. Proceedings. The 43rd Annual IEEE Symposium on, pages 490-499. IEEE, 2002.

[10] A. Caprara, A. Lodi, and M. Monaci. An approximation scheme for the two-stage, two-dimensional bin packing problem. In Integer Programming and Combinatorial Optimization, pages 315-328. Springer Berlin Heidelberg, 2002.

[11] A. Caprara, M. Monaci. Bidimensional packing by bilinear programming. Mathematical Programming, 118(1):75-108, 2009

[12] B. Chazelle. The bottomn-left bin-packing heuristic: An efficient implementation. Computers, IEEE Transactions on, 100(8):697707, 1983.

[13] F. R. K. Chung, M. R. Garey, D. S. Johnson. On packing two-dimensional bins. Society for Industrial and Applied Mathematics, 1982.

[14] E. G. Coffman, Jr, M. R. Garey, D. S. Johnson, R. E. Tarjan. Performance bounds for level-oriented two-dimensional packing algorithms. SIAM Journal on Computing, 9(4):808-826, 1980.

[15] J. Csirik, G. Woeginger. On-line packing and covering problems. In Amos Fiat and Gerhard J. Woeginger, editors, Online Algorithms, volume 1442 of Lecture Notes in Computer Science, pages 147-177. Springer Berlin Heidelberg, 1998.

[16] Д. Цветковић, М. Чангаловић. Комбинаторна оптимизација - математичка теорија и алгоритми, ДОПИС, Београд, 1996.

[17] O. Faroe, D. Pisinger, M. Zachariasen. Guided local search for the threedimensional bin packing problem. Department of Computer Science, University of Copenhagen, 1999.

[18] S.P. Fekete, J. Schepers. New classes of lower bounds for bin packing problems. In Integer Programming and Combinatorial Optimization, pages 257-270. Springer, 1998.

[19] S.P. Fekete, J. Schepers. A combinatorial characterization of higherdimensional orthogonal packing. Mathematics of Operations Research, 29(2):353–368, 2004.

[20] S.P. Fekete, J. Schepers. A general framework for bounds for higherdimensional orthogonal packing problems. Mathematical Methods of Operations Research, 60(2):311-329, 2004.

[21] S.P. Fekete, J. Schepers, J. C. Van der Veen. An exact algorithm for higherdimensional orthogonal packing. Operations Research, 55(3):569-587, 2007.

[22] J.B. Frenk, G.G. Galambos. Hybrid next-fit algorithm for the two-dimensional rectangle binpacking problem. Computing, 39:201-217, 1987.

[23] A. Freund, J. S. Naor. Approximating the advertisement placement problem. Journal of Scheduling, 7(5):365–374, 2004.

[24] P.C. Gilmore, R.E. Gomory. Multistage cutting problems of two and more dimensions. Operations Research, 13:94–119, 1965.

[25] P. C. Gilmore, R. E. Gomory. A linear programming approach to the cutting stock problem. Operations Research, 9:849–859, 1961.

[26] P. C. Gilmore, R. E. Gomory. A linear programming approach to the cutting stock problem - part II. Operations Research, 11:863–888, 1963.

[27] F. Glover. Future paths for integer programming and links to artificial intelligence.Computers and Operations Research, 13(5):533–549, 1986.

[28] Fred Glover. Tabu search-part i. ORSA Journal on computing, 1(3):190–206, 1989.

[29] F. Glover. Tabu search—part ii. ORSA Journal on computing, 2(1):4–32, 1990.

[30] F. Glover, G. A. Kochenberger. Handbook of metaheuristics. Springer, 2003.

[31] D. Goldberg, R. Lingle: Alleles, loci, and the travelling salesman problem. In: First International Conference on Genetic Algorithms and their Applications, pp. 154–159. Lawrence Erlbaum Associates, Hillsdale, 1985.

[32] E. Hadjiconstantinou, N. Christofides. An exact algorithm for general, orthogonal, twodimensional knapsack problems. European Journal of Operational Research, 83(1):39–56, 1995.

[33] R. Harren. Two-dimensional packing problems. PhD thesis, Universität des Saarlandes, Germany, 2010.

[34] R. Harren, R. van Stee. An absolute 2-approximation algorithm for the twodimensional bin packing. submitted for publication.

[35] R. Harren, R. van Stee. Absolute approximation ratios for packing rectangles into bins. Journal of Scheduling, 15(1):63-75, 2012.

[36] J. Holland. H., 1975, adaptation in natural and artificial systems. Ann Arbor, MI: University of Michigan Press, 1975.

[37] M. Iori, S. Martello, M. Monaci. Metaheuristic algorithms for the strip packing problem. Applied Optimization, 78:159-180, 2003.

[38] D.S. Johnson. Near-optimal bin packing algorithms. PhD thesis, Massachusetts Institute of Technology, 1973.

[39] D. Johnson, A. Demers, J. Ulman, M. Garey, R. Graham. Worst-case performance bounds for simple one dimensional packing algorithms. SIAM Journal on computing, 1974, p. 229-325

[40] J. Jylánki. A thousand ways to pack the bin - a practical approach to two-dimensional rectangle bin packing. Technical report, 2010. http://clb.demon.fi/files/RectangleBinPack.pdf.

[41] C. Kenyon N. Bansal, J.R. Correa, M. Sviridenko. Bin packing in multiple dimensions: inapproximability results and approximation schemes. Mathematics of Operations Research, 31(1):31–49, 2006.

[42] C. Kenyon and E. Rémila. A near-optimal solution to a two-dimensional cutting stock problem. Math. Oper. Res., 25(4):645-656, 2000.

[43] S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, et al. Optimization by simmulated annealing. science, 220(4598):671-680, 1983.

[44] S. Laabadi, M. Naimi, H. E. Amri, B. Achchab. A Crow Search-Based Genetic Algorithm for Solving Two-Dimensional Bin Packing Problem. In: Benzmüller C., Stuckenschmidt H. (eds) KI 2019: Advances in Artificial Intelligence. KI 2019. Lecture Notes in Computer Science, vol 11793. Springer, 2019.

[45] C.C. Lee, D.T. Lee. A simple on-line bin-packing algorithm. Journal of the ACM (JACM), 32(3):562-572, 1985.

[46] K. Li, K. H. Cheng. Static job scheduling in partitionable mesh connected systems. Journal of Parallel and Distributed Computing, 10(2):152–159, 1990.

[47] D.S. Liu, K.C. Tan, S.Y. Huang, C.K. Goh, W.K. Ho. On solving multiobjective bin packing problems using evolutionary particle swarm optimization. European Journal of Operational Research, 190(2):357-382, 2008.

[48] A. Lodi, S. Martello, D. Vigo. Heuristics and metaheuristic approaches for a class of two-dimensional bin packing problems. INFORMS Journal on Computing, 11:345-357, 1999.

[49] A. Lodi. Algorithms for Two-Dimensional Bin Packing and Assignment Problems. PhD thesis, University of Bologna, Italy, 2000.

[50] A. Lodi, S. Martello, and M. Monaci. Two-dimensional packing problems: A survey. European Journal of Operational Research, 141(2):241-252, 2002.

[51] A. Lodi, S. Martello, M. Monaci, D. Vigo. Two-Dimensional Bin Packing Problems. Paradigms of Combinatorial Optimization: Problems and New Approaches: 2nd Edition, 2014.

[52] A. Lodi, S. Martello, D. Vigo. Approximation algorithms for the oriented two-dimensional bin packing problem. European Journal of Operational Research, 112:158-166, 1999.

[53] A. Lodi, S. Martello, D. Vigo. Neighborhood search algorithm for the guillotine non-oriented two-dimensional bin packing problem. In Meta-Heuristics, pages 125-139. Springer US, 1999.

[54] A. Lodi, S. Martello, D. Vigo. Models and bounds for two-dimensional level packing problems. Journal of Combinatorial Optimization, 8(3):363-379, 2004.

[55] A. Lodi, M. Monaci, E. Pietrobuoni. Partial enumeration algorithms for Two-Dimensional Bin Packing Problem with guillotine constraints. Discrete Applied Mathematics, 2015.

[56] G.S. Lueker. Bin packing with items uniformly distributed over intervals [a,b]. In Foundations of Computer Science, 1983, 24th Annual Symposium on, pages 289-297, Nov 1983.

[57] S. Martello, D. Vigo. Exact solution of the two-dimensional finite bin packing problem. Management Science, 44:388-399, 1998.

[58] S. Martello, M. Monaci, D. Vigo. An exact approach to the strip-packing problem. INFORMS Journal on Computing, 15(3):310-319, 2003.

[59] M. Monaci, P. Toth. A set-covering based heuristic approach for bin-packing problems. INFORMS Journal on Computing, 18:71-85, 2006.

[60] E. Pietrobuoni. Two-Dimensional Bin Packing Problem with Guillotine Restrictions, [Dissertation thesis], Alma Mater Studiorum Universita di Bologna. Dottorato di ricerca in Automatica e ricerca operativa, 26 Ciclo, 2015.

[61] D. Pisinger, M. Sigurd. Using decomposition techniques and constraint programming for solving the two-dimensional bin-packing problem. INFORMS Journal on Computing, 19:36-51, 2007.

[62] J. Puchinger, G.R. Raidl. Models and algorithms for three-stage twodimensional bin packing. European Journal of Operational Research, 183:1304- 1327, 2007.

[63] I. Schiermeyer. Reverse-fit: A 2-optimal algorithm for packing rectangles. In Proceedings of the Second Annual European Symposium on Algorithms, ESA '94, pages 290-299, London, UK, UK, 1994. Springer-Verlag.

[64] D. Simchi-Levi, X. Chen, J. Bramel. The logic of logistics: theory, algorithms, and applications for logistics and supply chain management. Springer, 2007.

[65] D.D. Sleator. A 2.5 times optimal algorithm for packing in two dimensions. Information Processing Letters, 10(1):37-40, 1980.

[66] A. Soke, Z. Bingul. Hybrid genetic algorithm and simulated annealing for two-dimensional non-guillotine rectangular packing problems. Engineering Applications of Artificial Intelligence, 19(5):557-567, 2006.

[67] R. van Stee. An approximation algorithm for square packing. Operations Research Letters, 32(6):535-539, 2004.

[68] A. Steinberg. A strip-packing algorithm with absolute performance bound 2. SIAM J. Comput., 26(2):401-409, 1997.

[69] P. Toth and S. Martello. Knapsack problems: Algorithms and computer implementations. Discrete Mathematics and Optimization. Wiley, 1990.

[70] G. Zhang. A 3-approximation algorithm for two-dimensional bin packing. Operations Research Letters, 33(2):121 - 126, 2005.