Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Vypĺňanie rovinných útvarov uzavretými 2D tvarmi

Bakalárska práca

2020
Tomáš Farský

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Vypĺňanie rovinných útvarov uzavretými 2D tvarmi

Bakalárska práca

Študijný program:	informatika
Študijný odbor:	informatika
Školiace pracovisko:	Katedra informatiky
Školiteľ:	doc. Mgr. Tomáš Plachetka, Dr.

Bratislava, 2020
Tomáš Farský

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Tomáš Farský
Študijný program: informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Vypĺňanie rovinných útvarov uzavretými 2D tvarmi
Filling Planar Figures with Closed 2D Shapes

Anotácia: V tejto práci sa študuje a spracúva trieda problémov ukladania uzavretých 2D tvarov do roviny a prístupy k ich riešeniu. Vybrané metódy sa aplikujú na konkrétny praktický problém a implementuje sa metóda na minimalizáciu odpadu pri vyrezávaní 2D tvarov z obdĺžnikovej formy.

Vedúci: doc. Mgr. Tomáš Plachetka, Dr.
Katedra: FMFI.KI - Katedra informatiky
Vedúci katedry: prof. RNDr. Martin Škoviera, PhD.

Spôsob sprístupnenia elektronickej verzie práce:
archív

Dátum zadania: 11.10.2019

Dátum schválenia: 30.10.2019

doc. RNDr. Daniel Olejár, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Poďakovanie: Ďakujem môjmu školiteľovi doc. T. Plachetkovi za vedenie, pomoc a cenné rady. Ďakujem mojej rodine a priateľom za podporu. Ďakujem môjmu otcovi a bratom Jánovi a Štefanovi M. za konzultácie a poskytnuté dáta.

Abstrakt

Práca podáva komplexný prehľad triedy Problémov vypĺňania s jej históriou a hierarchiou. Zameriava sa na konkrétny problém vypĺňania obdĺžnikov nepravidelnými tvarmi. Nepravidelné objekty sú obalené do obdĺžnikov, na umiestnenie obdĺžnikov sa používa známa skyline heuristika. Následne skyline heuristiku modifikujeme tak, aby zjemnila možnosť uloženia objektov obalených obdĺžnikmi. Navrhnuté riešenie implementujeme. Náš program pracuje so súbormi v grafickom formáte DXF a dokáže riešiť aj situácie, kedy sa všetky objekty nezmestia do jedného obdĺžnika. Prezentujeme výsledky experimentov, v ktorých porovnávame kvalitu dosiahnutých riešení.

Kľúčové slová: Problémy vypĺňania, 2D Problém vypĺňania nádoby, skyline heuristika, DXF

Abstract

This thesis provides a comprehensive overview of the class of Cutting and Packing Problems with its history and hierarchy. The focus is on the specific problem of filling rectangles with irregular shapes. Irregular objects are wrapped in rectangles and the well-known skyline heuristics is used to place the rectangles. Subsequently, we modify the skyline heuristics to refine the placing of objects wrapped in rectangles. We implement the proposed solution. Our program works with files in the DXF graphic format and can also deal with situations where not all objects fit into one rectangle. We present the results of experiments in which we compare the quality of the achieved solutions.

Keywords: Cutting and Packing Problems, 2D Bin Packing Problem, skyline heuristics, DXF

Obsah

Zoznam obrázkov

Zoznam tabuliek

Úvod

S problémami vypĺňania priestoru sa stretávame každodenne v rôznych situáciách. Niektoré z nich sa riešia ľahko, napr. „Zmestí sa táto škatuľa (box) na poličku?“ Niektoré si, naopak, vyžadujú komplexné riešenia, napr. „Ako vyplniť boxami rôznych cien tento prepravný kontajner tak, aby hodnota kontajnera bola čo najvyššia?“ Prvá známa vedecká práca na túto tému pochádza zo 17. storočia, v 20. storočí sa jej venuje veľa odborných publikácií. Veľký pokrok priemyselnej výroby si vyžadoval hľadanie čoraz efektívnejších riešení. Výskum v tomto smere priniesol množstvo algoritmických riešení, ako aj dôležité teoretické výsledky, ktoré nachádzajú uplatnenie v priemyselnej výrobe.

Problém vypĺňania priestoru, o ktorom pojednáva táto práca, má pôvod v strojovej výrobe, presnejšie v leteckom modelárstve. Časti modelu lietadla sa programovateľnou frézou vyrezávajú z dreva. Predtým je potrebné súčiastky rozložiť na drevené pláty, z ktorých budú vyrezané. Rozhodli sme sa vytvoriť aplikáciu na automatizáciu procesu vypĺňania drevených plátov súčiastkami. Takáto aplikácia môže pomôcť modelárom, ktorí si radi sami vyrezávajú modely lietadiel.

Cieľom tejto práce je pomôcť čitateľovi zorientovať sa v rozsiahlej problematike vypĺňania. Táto trieda obsahuje veľa známych aj menej známych problémov a vyžaduje si presnú hierarchiu a jednoznačné názvoslovie problémov. Ďalším cieľom práce je vytvoriť jednoduché riešenie spomínaného problému a implementovať ho.

Prvá kapitola práce obsahuje všeobecnú štruktúru Problémov vypĺňania (názvy problémov budeme písať s veľkým prvým písmenom) a historický prehľad vývoja uvedenej triedy problémov. V tejto časti nadväzujeme sa prácu Haralda Dyckhoffa a uvádzame novú tabuľku s prehľadovými prácami z oblasti Problémov vypĺňania. Následne prezentujeme hierarchiu navrhnutú Gerhardom Wäscherom a kol., ktorá sa momentálne používa na charakteristiku problémov v danej oblasti. V ďalšej časti popisujeme a vysvetľujeme konkrétne problémy, ktoré vyplynuli zo spomínanej hierarchie, a pridávame k problémom referencie na novšie práce, ktoré ich riešia.

Druhá kapitola prezentuje problém vypĺňania dreveného plátu súčiastkami. V tejto kapitole tiež uvádzame náš postup pri hľadaní riešenia. Naším cieľom je minimalizovať počet vyplnených plátov a zanechať čo najväčšiu súvislú časť obdĺžnikového tvaru z posledného plátu.

V tretej kapitole detailnejšie opisujeme niektoré prvky programu, ktorý implementuje riešenie navrhnuté v predchádzajúcej kapitole. Uvádzame informácie o obsahu vstupných a výstupných súborov a pseudokód riešenia.

V poslednej, štvrtej kapitole prezentujeme a analyzujeme výsledky experimentov.

Kapitola 1

Problémy vypĺňania

S problémami vypĺňania, ukladania alebo balenia sa ľudia stretávajú každý deň. Ráno pred odchodom do práce si balia všetky potrebné veci do tašky, po nákupe väčšinou treba dať niektoré potraviny do chladničky a keď sa človek chystá na výlet, zbalí sa do kufra alebo do ruksaka. V takýchto prípadoch je výsledok závislý od ľudských schopností. Na druhej strane, podobné problémy, ktoré sa vyskytujú v priemysle, požadujú čo najefektívnejšie riešenia vzhľadom na náklady akéhokoľvek druhu.

Veľkú pomoc priniesla v tomto smere počítačová automatizácia a od 50-tych rokov sa ľudia na akademickej pôde, niekedy v spolupráci s priemyselnými odvetviami, zaoberajú hľadaním algoritmických riešení pre dané problémy. V matematike, špeciálne napríklad v geometrii, sa tiež stretneme so známymi problémami, ktoré sa týkajú vypĺňania priestoru, a aj v umení nájdeme autorov zaoberajúcich sa podobným motívom.

V tejto kapitole sa pozrieme na štruktúru problémov, históriu a hierarchiu tejto triedy problémov, prejdeme konkrétne typy a zdôrazníme podtriedy dôležité pre túto prácu.

1.1 Cudzojazyčná terminológia

V anglickej terminológii sa táto trieda problémov nazýva `Cutting and Packing Problems`, často sa používa skratka `C & P`. V dnešnej dobe tento termín zahŕňa veľké množstvo problémov. V anglickom jazyku sa najčastejšie stretneme s termínmi Cutting Problems, Knapsack Problems, Container and Vehicle Loading Problems, Pallet Loading, Bin Packing, atď.

Na vysvetlenie názvu `Cutting and Packing` používa Harald Dyckhoff vo svojej knihe [33] nasledujúci príklad. Pri „rezaní“ musí byť väčší objekt rozdelený do menších objektov a pri „balení“ musia byť malé objekty zložené do väčšieho objektu. Takýmto spôsobom sa na problém balenia, napríklad napĺňanie kontajnera, môžeme pozrieť ako na problém rezania: Prázdny priestor kontajnera musí byť rozdelený (rozrezaný) na

menšie časti, dané rozmermi balíkov, ktoré treba naložiť. Rovnakým spôsobom sa dá pozrieť na rezanie pevného materiálu ako balenie prázdneho piestoru. V textilnom priemysle sa používajú papierové makety kusu oblečenia na nájdenie optimálneho využitia materiálu. Úlohou sa teda stane zaplniť kus látky takýmito maketami. Teda či sa problém považuje za problém balenia alebo problém rezania, zavisí od uhla pohľadu.

V našej práci túto triedu problémov nazývame Problémy vypĺňania.

1.2 Štruktúra problémov vypĺňania

Problémy vypĺňania majú jednotnú štruktúru, ktorú možno popísať nasledovne. Máme dve množiny elementov, množinu kontajnerov a množinu objektov, ktoré môžu byť definované v n dimenziách. Úlohou je vybrať niektoré alebo všetky objekty a spojiť ich do jednej alebo viacerých podmnožín a každú takúto podmnožinu priradiť jednému kontajneru tak, aby boli zachované špecifické podmienky, napr. geometrické: objekty z každej podmnožiny musia byť uložené do daného kontajnera tak, aby bol každý objekt celou svojou plochou v kontajneri a aby sa objekty navzájom neprekrývali. Cieľom je optimalizovať funkciu na kontajneroch alebo objektoch. Od typu problému závisí, či sa v riešení použijú všetky alebo len niektoré kontajnery a či sa použijú všetky alebo len niektoré objekty.

Kedže Problémy vypĺňania sú definované ako geometricko-kombinatorické problémy, dôraz sa kladie skôr na logickú štruktúru problému. Vďaka tomu je možné zdanlivo odlišné problémy spojiť na abstraktnej úrovni. Problémy sa často líšia typom objektov, s ktorými sa v nich narába. Objekty majú špecifické merateľné charakteristiky, na základe ktorých sa dajú rozlišovať a porovnávať. Pri hmotných veciach a geometrických objektoch to je väčšinou veľkosť a tvar, napríklad balenie auta alebo rezanie z dreva. Pri abstraktných objektoch môžu byť takými charakteristikami napríklad cena, čas alebo pamäť, napríklad problém batoha, multiprocesorové plánovanie alebo alokovanie pamäte. Každý problém má potom svoje špecifické podmienky, od ktorých závisí jeho riešenie.

V práci používame výraz „kontajner“ hlavne v zmysle tejto štruktúry problémov, ale vyskytne sa aj v zmysle prepravného (lodného) kontajnera.

1.3 História

S Problémami vypĺňania sa ľudstvo stretáva už od počiatku. Do oblasti vedy prišli spolu so snahou pochopiť štruktúru rôznych hmôt. Prvý známy problém vyplynul v roku 1611 z eseje Johannesa Keplera [67], ktorá sa považuje za prvú prácu v oblasti štruktúry kryštálov. Z tejto práce vyšiel problém známy ako Keplerova hypotéza, ktorá

sa týka vypĺňania 3D Euklidovho priestoru sférami. Túto hypotézu možeme nájsť aj medzi známymi matematickými problémami, ktoré v roku 1900 zverejnil David Hilbert [53] a bola dokázaná v roku 2005 Thomasom C. Halesom [52]. Oblasťou štruktúry kryštálov sa zaoberá aj práca Roberta Hookeho [55] z roku 1665, ktorá bola prvou veľkou publikáciou Kráľovskej spoločnosti a položila základy vedeckej oblasti mikroskopie.

Rýchly vývoj nabralo toto odvetvie v 20. storočí. V roku 1903 sa nemecký matematik Max W. Dehn vo svojej práci [27] začal zaoberať otázkou, či je možné väčší obdĺžnik rozdeliť na menšie obdĺžniky tak, aby sa z tých menších obdĺžnikov dal poskladať iný väčší obdĺžnik ako ten, ktorý bol na začiatku rozdelený. Japonec Michio Abe v roku 1932 vo svojom článku [1] rozoberá problém, ako vyplniť štvorec konečným počtom po dvoch rôznych štvorcoch a bez medzier medzi nimi.

Iný ako geometrický pohľad na Problémy vypĺňania ponúkol sovietsky matematik Leonid V. Kantorovich v roku 1939, keď sa vo svojej práci [65] začal zaoberať optimálnym využitím strojov, minimalizáciou odpadu, efektívnym využitím paliva alebo optimálnym splnením plánu konštrukcie s daným materiálom. V úvode tejto práce píše, že nesmierna náročnosť úlohy vyplývajúcej z tretieho Päťročného plánu si vyžaduje najvyššiu možnú produkciu na báze optimálneho využitia existujúcich zdrojov v priemysle: materiálu, práce a náradia. Teda v tomto období už existuje prepojenie medzi matematikou, priemyslom a politikou. Táto práca tiež položila základ metódy lineárneho programovania. V nasledúcom roku vyšiel článok [11], v ktorom sa štvorica autorov R. L. Brooks, C. A. B. Smith, A. H. Stone a W. T. Tutte zaoberá rozdeľovaním obdĺžnikov na po dvoch rôzne štvorce. Práve tieto články sa považujú za prvé matematické články v odbore Problémov vypĺňania.

V nasledujúcich desaťročiach sa Problémami vypĺňania začali na vedeckej úrovni zaoberať viaceré vedné odbory, ako napríklad manažment, strojárstvo, informatika a taktiež interdisciplinárny odbor operačný výskum. Rýchlo začali pribúdať vedecké články a konferencie o rôznych formách týchto problémov, ktoré prezentovali algoritmy a riešenia konkrétneho problému alebo niektorej podtriedy.

Dvojica vedcov z výskumného tímu IBM Paul C. Gilmore a Ralph E. Gomory medzi rokmi 1961–1966 vo svojich článkoch [44], [45] a [46] po prvýkrát prezentovala techniky, ktoré sa dali použiť na riešenie problémov nie veľmi veľkých rozmerov. Okrem prezentovania riešenia pomocou lineárneho programovania tiež zvýraznili dôležitosť Problému batoha.

V rokoch 1970–1990 vznikali takmer každoročne prehľadové práce, ktoré spracovávali dianie a štruktúru daného odboru alebo konkrétneho typu Problémov vypĺňania.

S cieľom podporiť medzinárodný výskum v oblasti Problémov vypĺňania vznikla v roku 1988 neformálna skupina *Special Interest Group on Cutting and Packing (SICUP)*. Založili ju Harald Dyckhoff a Gerhard Wäscher. Približne dvakrát do roka publikovali časopis a pomáhali posúvať články na medzinárodné konferencie.

Autor(i)	Rok	Kľúčové pojmy	Časopis/Kniha
Brown	1971	vypĺňanie	Optimum Packing and Depletion
Salkin, de Kluyver	1975	batoh	Naval Research Logistics Quarterly
Golden	1976	rezný materiál	AIIE Transactions
Hinxman	1980	redukcia straty	European Journal of Operational Research
Garey, Johnson	1981	vypĺňanie nádoby	Analysis and Design of Algorithms in Combinatorial Optimization
Israni, Sanders	1982	rezný materiál	Journal of Manufacturing Systems
Rayward–Smith, Shing	1983	vypĺňanie nádoby	Bulletin of the IMA
Coffman a kol.	1984	vypĺňanie nádoby	Algorithm Design for Computer System Design
Dowsland	1985	vypĺňanie	Computers & Operations Research
Dyckhoff a kol.	1985	redukcia straty	Omega
Israni, Sanders	1985	ukladanie častí	International Journal of Production Research
Berkey, Wang	1987	vypĺňanie nádoby	Journal of the Operational Research Society
Dudzinski, Walukiewicz	1987	batoh	European Journal of Operational Research
Martello, Toth	1987	batoh	Surveys in Combinatorial Optimization
Rode, Rosenberg	1987	redukcia straty	Engineering Costs and Production Economics
Dyckhoff a kol.	1988	rezný materiál	Essays on Production Theory and Planning

Tabuľka 1.1: Prehľadové práce v oblasti Problémov vypĺňania v období 1970–1990

V roku 1990 Harald Dyckhoff uverejnil prácu [32], v ktorej vytvoril logickú štruktúru pre všetky Problémy vypĺňania. Vo svojej práci našiel spoločné črty problémov, ktoré sa môžu zdať na prvý pohľad nesúvisiace. Na druhej strane problémy, ktoré sa zdali na prvý pohľad podobné, dokázal vďaka analýze ich charakteristík lepšie rozlíšiť. Je to prvá práca svojho druhu a zhruba na nasledujúce dve desaťročia bola jedným z hlavných zdrojov v tomto odbore. Tabuľka 1.1 pochádza z tejto práce a prezentuje spomínaný zoznam prehľadových prác. O dva roky neskôr, v roku 1992, vydal spolu s Ute Finkem knihu [33], ktorá obsahuje systematický prehľad existujúcej literatúry a snaží sa zhrnúť všetky dovtedy získané poznatky o Problémoch vypĺňania. V roku 1997 spolu s Guntramom Scheithauerom a Johannesom Ternom zhrnul napredovanie v práci [34].

Po roku 1990 vzniklo mnoho článkov, ktoré sa zaoberali prehľadom buď celej triedy Problémov vypĺňania, alebo špecifickej oblasti. V tabuľke 1.2 uvádzame nový zoznam na štýl tabuľky z Dyckhoffovej typológie.

Rýchly vývoj inšpiroval v roku 2007 Gerharda Wäschera, Heike Haußnera a Holgera Schumanna napísať novú typológiu Problémov vypĺňania. Vo svojej práci [108] taktiež vysvetľujú, že v Dyckhoffovej typológii sa časom ukázali niektoré nedostatky, napríklad čiastočná inkonzistencia, kvôli ktorej môže mať jej aplikácia mätúce výsledky, alebo nemožnosť priradiť každý Problém vypĺňania k práve jednému typu. Zatiaľ čo Dyckhoff použil na kategorizáciu štyri kritériá, Wäsher použil päť. Tento článok je v dnešnej dobe základným nástrojom na klasifikáciu Problémov vypĺňania. Aj v tejto práci sa držíme tejto klasifikácie.

Od 60-tych rokov 20. storočia sa čoraz viac ľudí venuje hľadaniu čo najefektívnejších algoritmov na riešenie Problémov vypĺňania. Články s algoritmickými riešeniami problémov sú pravidelne zverejňované vo vedeckých časopisoch a zborníkoch.

1.4 Hierarchia

Gerhard Wäscher, Heike Haußner a Holger Schumann používajú vo svojej práci [108] týchto päť kritérií na definovanie typov Problémov vypĺňania: počet rozmerov (dimenzií), typ priraďovania, typ objektov, typ kontajnerov a tvar malých objektov.

Pri počte rozmerov rozlišujeme problémy v jednom, dvoch alebo troch rozmeroch. V literatúre sa občas vyskytuje ešte ďalší variant pre $n > 3$ rozmerov.

Pri type priraďovania sa pozeráme na to, či je naším cieľom maximalizovať funkciu na objektoch, alebo minimalizovať funkciu na kontajneroch. Keď maximalizujeme funkciu na objektoch, množina kontajnerov nemôže pokryť celú množinu objektov a naším cieľom je vybrať podmnožinu objektov s najvyššou hodnotou tak, aby sme pokryli celú množinu kontajnerov. Na druhej strane, pri minimalizácii funkcie na kontajneroch je

Autor(i)	Rok	Kľúčové pojmy	Časopis/Kniha/Konferencia
Dyckhoff	1990	vypĺňanie	European Journal of Operational Research
Dyckhoff, Finke	1992	vypĺňanie	Cutting and Packing in Production and Distribution: A Typology and Bibliography
Ram	1992	nakladanie palety	International Journal of Production Economics
Sweeney, Paternoster	1992	bibliografia	Journal of the Operational Research Society
Cheng a kol.	1994	rezný metriál	International Journal of Production Economics
Galambos, Woeginger	1995	vypĺňanie nádoby	Zeitschrift für Operations Research
Coffman a kol.	1996	vypĺňanie nádoby	Approximation Algorithms for NP-Hard Problems
Dyckhoff a kol.	1997	vypĺňanie nádoby	Annotated Bibliographies in Combinatorial Optimization
Vemuganti	1998	vypĺňanie	Handbook of Combinatorial Optimization
Takayuki	1998	3D vypĺňanie nádoby	IBM Research Laboratory, Tokyo
Lodi a kol.	2002	2D vypĺňanie	European Journal of Operational Research
Puchinger, Raidl	2005	vypĺňanie	International Work-conference on the Interplay between Natural and Artificial Computation
Mukhacheva, Mukhacheva	2006	vypĺňanie	Journal of Mathematical Sciences
Wäscher a kol.	2007	vypĺňanie	European Journal of Operational Research
Ntene, van Vuuren	2009	vypĺňanie pásu	Discrete Optimization
Genova, Guliashki	2011	lineárne program.	Journal of Cybernetics and Information Technologies
Cherri a kol.	2014	rezný materiál	European Journal of Operational Research
Parmar a kol.	2014	rezný materiál	2014 International Conference on Green Computing Communication and Electrical Engineering (ICGCCEE)
Oliveira a kol.	2016	vypĺňanie pásu	Pesquisa Operacional
Delorme a kol.	2016	rezný materiál	European Journal of Operational Research
Christensen a kol.	2017	vypĺňanie nádoby	Computer Science Review

Tabuľka 1.2: Prehľadové práce v oblasti Problémov vypĺňania v období 1990–2020

množina kontajnerov schopná pokryť celú množinu objektov a našou snahou je vybrať podmnožinu kontajnerov s najmenšou hodnotou tak, aby sme vedeli prideliť všetky objekty vybranej podmnožine kontajnerov. Dôležitou poznámkou je, že pojem hodnota je v tomto kontexte veľmi abstraktný pojem a pri špecifických problémoch je potrebné presnejšie definovať, čo sa má pod pojmom hodnota chápať.

Pri type objektov rozlišujeme tri typy: identické objekty, podobné objekty a odlišné objekty. Pri každom probléme sú dôležité relevantné rozmery (napr. dĺžka, výška, hĺbka) a hodnota. Keď narábame s identickými objektmi a naším cieľom je maximalizovať funkciu na objektoch, môžeme sa na to pozrieť tak, že po jednom objekte je nelimitovaný dopyt. Pri podobných objektoch sa môžu objekty pospájať do skupín a dopyt po objektoch jedného typu môže byť limitovaný. Podľa definície sú rovnaké objekty, ktoré sa líšia iba orientáciou, považované za objekty rôznych typov. Po odlišných objektoch je dopyt po každom objekte rovný jedna, pretože každý je chápaný ako odlišný typ.

Pri type kontajnerov máme dva prípady: jeden kontajner alebo viacero kontajnerov. Jeden kontajner môže mať všetky (pre konkrétny problém relevantné) rozmery pevne dané, alebo jeden, prípadne viac rozmerov, može byť voľných. Pri viacerých kontajneroch sú všetky rozmery pevne dané. Podobne ako pri type objektov potom rozlišujeme medzi identickými, podobnými a odlišnými kontajnermi. Pri definícii základných typov Problémov vypĺňania budeme používať identické kontajnery obdĺžniky a kvádre.

Pri tvare malých objektov v prípade dvoj– a trojrozmerných problémov rozlišujeme dva typy: pravidelné (obdĺžniky, kruhy, kvádre, gule, atď.) a nepravidelné objekty.

Základné typy problémov v hierarchii vypĺňania vznikli podľa dvoch kritérií: typ priraďovania (cieľ) a typ objektov.

Typy problémov, pri ktorých je naším cieľom maximalizovať funkciu na objektoch, majú spoločnú črtu – počet kontajnerov je limitovaný, teda nemusia poňať všetky objekty. Cieľom je vyplniť všetky kontajnery tak, aby hodnota uložených objektov bola čo najvyššia. Rozlišujeme tieto základné typy problémov, pri ktorých maximalizujeme funkciu na objektoch:

- Problém vypĺňania identickými objektmi – `Identical Item Packing Problem`

  Pri problémoch tohoto základného typu sa snažíme limitovaný počet kontajnerov vyplniť čo najväčším počtom objektov. Všetky ukladané objekty sú identické, čiže pri hľadaní riešenia neprebieha žiaden výber objektov ani spájanie objektov do skupín.

- Problém rozmiestnenia – `Placement Problem`

  V tomto prípade vypĺňame limitovaný počet kontajnerov podmnožinou množiny navzájom podobných objektov tak, aby hodnota uložených objektov bola čo najvyššia. Tento základný typ problémov má v literatúre viacero názvov.

- Problém batoha – `Knapsack Problem`

  Problém batoha reprezentuje základný typ vypĺňania, v ktorom opäť vypĺňame limitovaný počet kontajnerov, ale podmnožinou množiny objektov, ktoré sú navzájom odlišné, s cieľom maximalizovať hodnotu uložených objektov.

Pre typy problémov s cieľom minimalizovať funkciu na kontajneroch je charakteristické, že kontajnerov je dostatočné množstvo nato, aby poňali všetky objekty. Cieľom je kontajnery vyplniť množinou objektov tak, aby hodnota použitých kontajnerov bola čo najnižšia. Rozlišujeme tieto základné typy problémov, pri ktorých minimalizujeme funkciu na kontajneroch:

- Problém otvorenej dimenzie – `Open Dimension Problem`

  Pri tomto základnom type Problémov vypĺňania ukladáme množinu objektov do jedného kontajnera (výnimočne viacerých). Kontajner, ktorý máme v zadaní problému, má aspoň jednu dimenziu otvorenú. Iba tá časť takéhoto otvoreného kontajnera, ktorá je potrebná na poňatie celej množiny objektov, sa považuje za kontajner, ktorý je v zadaní všeobecnej štruktúry Problémov vypĺňania. Cieľom pri vypĺňaní je minimalizovať funkciu na kontajneri, napríklad dĺžku, obsah alebo objem. Pri definícii tohoto základného typu sa najčastejšie používajú kontajnery, ktoré po zafixovaní otvorenej dimenzie (dimenzií) predstavujú obdĺžniky (2–dimenzionálne problémy), alebo hranoly (3–dimenzionálne problémy). V tejto kategórii sú však aj problémy, pri ktorých sa vypĺna kontajner, ktorý nie je obdĺžnikového tvaru, napríklad problém, v ktorom dané objekty – kružnice musíme uložiť do kontajnera – kružnice s čo najmenším polomerom.

- Problém rezania materiálu – `Cutting Stock Problem`

  Problém rezania materiálu zastrešuje problémy, pri ktorých sa množina objektov skladá z navzájom podobných objektov a celá táto množina musí byť uložená do podmnožiny množiny kontajnerov tak, aby hodnota, počet alebo totálna veľkosť kontajnerov bola čo najnižšia. Rozmery kontajnerov sú všetky pevne dané. Množina kontajnerov pri tomto type problémov môže obsahovať identické, podobné aj odlišné kontajnery.

- Problém vypĺňania nádoby – `Bin Packing Problem`

  Na rozdiel od predchádzajúceho základneho typu, pri Problémoch vypĺňania nádoby sú v množine objektov navzájom odlišné objekty. Rozmery kontajnerov sú tiež pevne dané. V množine kontajnerov môžu byť identické, navzájom podobné alebo odlišné kontajnery. Cieľom je vyplniť podmnožinu množiny kontajnerov tak, aby hodnota príslušnej podmnožiny kontajnerov bola čo najnižšia.

charakteristika kontajnerov \ typ objektov		identické	podobné	odlišné
všetky rozmery pevné	jeden kontajner	Problém vypĺňania identickými objektmi	Problém vypĺňania jedného kontajnera	Problém jedného batoha
	identické kontajnery		Problém vypĺňania viacerých identických kontajnerov	Problém viacerých rovnakých batohov
	odlišné kontajnery		Problém vypĺňania viacerých odlišných kontajnerov	Problém viacerých odlišných batohov

Tabuľka 1.3: Odvodené typy problémov: maximalizovanie funkcie na objektoch

charakteristika kontajnerov \ typ objektov		podobné	odlišné
všetky rozmery pevné	identické kontajnery	Problém rezania z materiálu rovnakého tvaru	Problém vypĺňania nádoby jedného tvaru
	podobné kontajnery	Problém rezania z materiálu rôznych tvarov	Problém vypĺňania nádoby rôznych tvarov
	odlišné kontajnery	Problém rezania zo zvyškového materiálu	Problém vypĺňania zvyškovej nádoby
jeden kontajner s otvorenou dimenziou/dimenziami		Problém otvorenej dimenzie	

Tabuľka 1.4: Odvodené typy problémov: minimalizovanie funkcie na kontajneroch

1.5 Konkrétne odvodené problémy

V tejto časti sa zameráme na konkrétne problémy, ktoré spadajú do odvodených typov z tabuliek 1.3 a 1.4, ktoré pochádzajú z práce [108]. Snažíme sa k jednotlivým problémom spomenúť práce, ktoré problém uviedli pod príslušným názvom, a taktiež práce, ktoré nazývame novšie–publikované po roku 2005, zaoberajúce sa daným problémom. Pri vypĺňaní pravidelnými objektmi sa spravidla uvažuje o pravouhlom ukladaní obdĺžnikov alebo kvádrov.

Problém vypĺňania identickými objektmi

Medzi známe problémy tohoto typu vypĺňania patrí Výrobcov problém s nakladaním palety – `Manufacturer's Pallet Loading Problem`. V tomto probléme ide o to, ako naložiť maximálny počet identických kvádrov na jednu paletu. Pri riešení sa paleta nakladá po vrstvách, v ktorých majú všetky kvádre pevnú vertikálnu orientáciu. Takýmto postupom sa problém zredukuje na 2–rozmerný, pričom paletu reprezentuje veľký obdĺžnik a kvádre reprezentujú obdĺžniky. Tento problém sa spomína v práci [29]. Z novších prác sú to napríklad [91], [93] a [101].

Ďalším známym problémom tohoto typu je Problém vypĺňania valcom – `Cylinder Packing Problem`, pri ktorom sa snažíme vyplniť daný obdĺžnik čo najväčším počtom rovnakých kruhov tak, aby sa kruhy neprekrývali. S takýmto problémom sa môžeme stretnúť pri vyrezávaní dna a viečka plechovky alebo pri nakladaní plechoviek na paletu. Tento problém je rozoberaný v práci [21] a tiež v novšej práci [13].

Do tohoto typu problémov sa tiež radí Problém vypĺňania kontajnera kvádrami rovnakého typu – `Single-Box-Type Container Packing Problem`, ktorý je 3–rozmerný problém a rieši, ako vyplniť veľký kváder čo najväčším počtom malých kvádrov rovnakej veľkosti. Na rozdiel od Výrobcovho problému s nakladaním palety kvádre nemajú fixnú vertikálnu orientáciu. Tento problém v spomínanej podobe bol rozoberaný v práci [42]. Venujú sa mu aj novšie práce [82] a [62].

Problém vypĺňania jedného kontajnera

Známe 1–rozmerné problémy Ohraničený problém batoha a Neohraničený problém batoha – `Bounded/Unbounded Knapsack Problem` sú zaradené do tohoto typu. Princíp týchto problémov spočína v tom, že máme jeden kontajner–batoh s obmedzenou nosnosťou a musíme ho vyplniť podmnožinou objektov, pričom každý objekt má svoju cenu a váhu, tak, aby cena objektov v batohu bola čo najväčšia a aby súčet objektov v batohu neprekročil jeho nosnosť. Pri type objektov hovoríme o podobných objektoch. Pri vypĺňaní batoha môže byť počet objektov konkrétneho typu obmedzený (ohraničený problém) alebo neobmedzený (neohraničený problém). Tento problém bol opísaný

v knihe [105]. Novšie práce, ktoré sa venujú ohraničenému problému, sú napríklad [81], [66], [19] a neohraničenému problému [103], [88] a [58].

Ďalším problémom, ktorý spadá do tohoto typu, je Problém rozloženia šablóny – `Template-Layout Problem`. Je to 2–rozmerný problém, pri ktorom musí byť množina podobných pravidelných alebo nepravidlených tvarov (foriem) vyrezaná z jedného veľkého obdĺžnika tak, aby hodnota vyrezaných tvarov bola čo najväčšia. Tento problém má veľa modifikácií, v anglickej literatúre ich môžeme nájsť ako `Two-Dimensional Cutting Problem` (objekty sú obdĺžniky), `(Constrained) Circular Cutting Problem` (objekty sú kruhy s rôznymi polomermi). Problém rozloženia šablóny bol uvedený v práci [51] a zaoberajú sa ním aj novšie práce [69] a [70].

Medzi 3–rozmerné problémy tohoto typu patrí Problém nakladania jedného kontajnera – `Single Container Loading Problem`, ktorého cieľom je uloženie pomerne veľkej množiny podobných kvádrov (balíkov) do veľkého kvádra (prepravný kontajner) tak, aby objem alebo cena naložených objektov bola čo najväčšia. O tomto probléme sa pojednáva v práci [43] a z novších prác sú to [56], [48] a [64].

Taktiež 3–rozmerným problémom tohto typu je Distribútorov problém s nakladaním palety – `Distributor's Pallet Loading Problem`. Na rozdiel od Výrobcovho problému s nakladaním palety je Distribútorov problém naozaj 3–rozmerný, lebo kvôli rôznym veľkostiam kvádrov nemusí byť najlepšie koncentrovať sa na vrstvy. Tento problém je rozoberaný v práci [54] a tiež v novších prácach [5], [102] a [2].

Problém vypĺňania viacerých identických kontajnerov

Problémy tohoto typu sú prirodzeným rozšírením Problému vypĺňania jedného kontajnera. Napríklad 2–rozmerný Obdĺžnikový problém vypĺňania viacerých identických kontajnerov, ktorý sa spomína v práci [24]. Problémy tohoto typu sa spomínajú aj v novších prácach [10] a [100].

Problém vypĺňania viacerých odlišných kontajnerov

Problémy tohoto typu sú ďalším rozšírením Problému vypĺňania jedného kontajnera, pri ktorých je v množine kontajnerov viacero odlišných kontajnerov. Napríklad 3–rozmerný Problém nakladania kontajnera z práce [35], ktorý sa týka vypĺňania kontajnerov rôznych veľkostí podmnožinou podobných objektov (kvádrov). Cieľom je maximalizovať objem uložených objektov. Problémy tohoto typu sa spomínajú aj v novších prácach [37] a [26].

Problém jedného batoha

Najznámejším problémom tohoto typu je Klasický problém batoha – `Classic (0-1) Knapsack Problem`, ktorý rieši, ako do batoha s obmedzenou nosnosťou uložiť podmnožinu odlišných objektov (každý so svojou váhou a cenou) tak, aby súčet cien zbalených objektov bol čo najväčší a aby súčet váh neprekročil nosnosť batoha. Tento problém má tiež veľa variácií, v anglickej literatúre napríklad `Subset-Sum Knapsack Problem` (cena je rovná váhe), `Multiconstraint Knapsack Problem` (obmedzení môže byť viac ako len nosnosť). Klasický problém batoha je 1–rozmerný, ale viacero modifikácií tohoto problému sa zaoberá rošírením do viacerých rozmerov: `Two-Dimensional Knapsack Problem` (objekty aj kontajner sú obdĺžniky), `Three-Dimensional Knapsack Problem` (objekty aj kontajner sú kvádre), a nakoniec aj zovšeobecnenie na n–rozmerov. Zaujímavou modifikáciou je 2–rozmerný Kruhový problém jedného batoha – `Circular Single Knapsack Problem`, pri ktorom je potrebné batoh (spodok–kruh) naplniť podmnožinou rúr rôzneho tvaru tak, aby hodnota rúr bola čo najväčšia. Klasický problém batoha je spomínaný v prácach [95], [105], [80] a v novších prácach [112], [110] a [38].

Problém viacerých rovnakých batohov

Medzi problémy tohoto typu patrí napríklad Problém vypĺňania nádoby s maximálnou kardinalitou – `Maximum Cardinality Bin Packing Problem`, ktorý bol uvedený v práci [71]. Názov tohoto problému je zavádzajúci, lebo cieľom je maximalizovať počet uložených objektov, ktoré ukladáme do určeného počtu kontajnerov s rovnakou kapacitou. Objekty sú navzájom odlišné. Pri Probléme vypĺňania nádoby je cieľom minimalizovať počet kontajnerov, čiže je možné vidieť nesúlad v názve tohoto problému a problém ako taký vskutku patrí do tohoto typu odvodených problémov. Riešeniu tohoto problému sa venujú aj novšie práce [85] a [77].

Taktiež 3–rozmerný Problém vypĺňania viacerých kontajnerov – `Multiple Container Packing Problem` z práce [92] má štruktúru problému tohoto typu. Daný počet rovnakých kontajnerov je potrebné naplniť podmnožinou navzájom odlišných objektov, ktoré majú svoju váhu a svoju cenu. Cieľom je maximalizovať počet objektov za zvyčajných podmienok. Týmto problémom sa zaoberá aj novšia práca [111].

Problém viacerých odlišných batohov

Známym rozšírením Klasického problému batoha je 0–1 Problém viacerých batohov – `0-1 Multiple Knapsack Problem`, ktorý je tiež 1–rozmerný. Cieľom je naplniť viacero batohov s odlišnými nosnosťami podmnožinou navzájom odlišných objektov tak, aby hodnota uložených objektov bola čo najvyššia, samozrejme, za zvyčajných podmienok. Týmto problémom sa zaoberajú práce [105], [86] a novšie práce [16] a [79].

Problém otvorenej dimenzie

Problém otvorenej dimenzie sa v literatúre spomína v súvislosti s jedným kontajnerom a problémy tohto typu majú zmysel, ak sa bavíme o dvoch alebo viacerých rozmeroch.

Známy 2–rozmerný problém tohoto typu je Problém vypĺňania pásu – `Strip Packing Problem`, pri ktorom je cieľom množinu objektov uložiť do obdĺžnikového kontajnera s pevnou šírkou tak, aby výška uložených objektov bola čo najnižšia. Ak sú ukladané objekty obdĺžniky, odvodený problém sa nazýva Problém vypĺňania pásu obdĺžnikmi – `Rectangular Strip Packing Problem`. Problém vypĺňania pásu sa spomína v práci [15] a z novších prác sú to napríklad [73], [22] a [17].

Ak majú objekty, ktoré ukladáme, nepravidelný tvar, tak takýto typ problémov sa nazýva Nepravidelný problém vypĺňania pásu – `Irregular Strip Packing Problem` alebo `Nesting Problem`. Stretneme sa s ním v práci [30] a v novších prácach napríklad [36] a [18].

Zaujímavý 2–rozmerný problém je Problém minimálneho uzáveru – `Minimal Enclosure Problem`, ktorý rieši problém, ako objekty uložiť do tvaru obdĺžnika tak, aby obsah tohoto obdĺžnika bol čo najmenší. V tomto prípade sú aj šírka aj výška kontajnera otvorené. Nájdeme ho v práci [50], aj v novšej práci [59].

Medzi 3–rozmerné problémy tohoto typu patrí napríklad problém v anglickej literatúre nazývaný `Three-Dimensional Open Dimension Rectangular Packing Problem`, ktorý sa zaoberá tým, ako množinu objektov–kvádrov poskladať do kontajnera–kvádra s minimálnym objemom. Tomuto problému sa venujú novšie práce [106] a [61].

Podľa práce [108] sa Problém otvorenej dimenzie, pri ktorom by sa spomínal viac ako jeden kontajner, v problematike Problémov vypĺňania nevyskytuje pravidelne, ale medzi takéto problémy by sa mohol zaradiť Problém plánovania úloh v distribuovaných systémoch – `Multiprocessor Scheduling Problem`. Množinu objektov tvoria nedeliteľné úlohy s daným časom na ich vykonanie a množinu kontajnerov tvoria rovnaké procesory. Cieľom je priradiť procesorom úlohy tak, aby maximálny čas vykonávania úloh bol čo najmenší. Čas vykonávania úloh jedného procesora je čas potrebný na to, aby procesor spracoval všetky úlohy, ktoré mu boli priradené. Tento problém je 1–rozmerný. Spomína sa v práci [104] a z novších prác sú to napríklad [74] a [7].

Problém rezania z materiálu rovnakého tvaru

Medzi problémy tohoto typu patrí 1–rozmerný Klasický problém rezania materiálu – `Classic One-Dimensional Cutting Stock Problem`, ktorý sa zaoberá tým, ako rozrezať materiál jednotnej dĺžky (v zmysle štruktúry Problémov vypĺňania – kontajnery) na objekty podobných rozmerov–dĺžok tak, aby počet alebo hodnota kontajnerov bola čo najnižšia. Napríklad, keď treba rozrezať vodárenské trubky štandardných rozmerov na kratšie kusy použiteľné v nejakom systéme. Tento problém bol uvedený v práci [44]

a venujú sa mu novšie práce [8] a [63].

V 2–rozmernom Klasickom probléme rezania materiálu sú objekty obdĺžniky podobných rozmerov a tie sa rozmiestňujú na pláty materiálu rovnakej veľkosti. Problém je rozoberaný v práci [45] a zaujímajú sa oň aj novšie práce [78] a [99].

Medzi 3–rozmerné problémy tohoto typu patria napríklad Problém nakladania viacerých paliet – `Multi-Pallet Loading Problem` a Problém nakladania viacerých kontajnerov – `Multi-Container Loading Problem`. Pri oboch problémoch množina objektov obsahuje kvádre podobných rozmerov a cieľom je uložiť celú množinu objektov na čo najmenší počet paliet alebo do čo najmenšieho počtu prepravných kontajnerov. Rozdiel je v tom, že pri Probléme nakladania viacerých paliet je vertikálna orientácia pevne daná. Paletami sa zaoberajú práce [98], z novších [72], a kontajnermi [97], z novších [14] a [3].

Problém rezania z materiálu rôznych tvarov

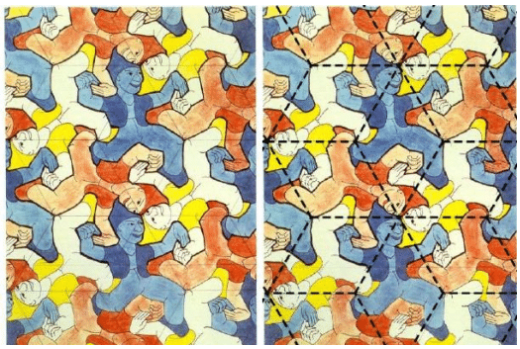
Tieto problémy sú rozšírením predchádzajúceho typu v zmysle množiny kontajnerov, ktorá môže obsahovať materiál podobných rozmerov. V anglickej terminológii sa tieto problémy nazývajú `Multiple Stock-Size Cutting Stock Problem`. Takýmito problémami sa zaoberá napríklad práca [109] a novšie práce [89], [41] a [40].

Problém rezania zo zvyškového materiálu

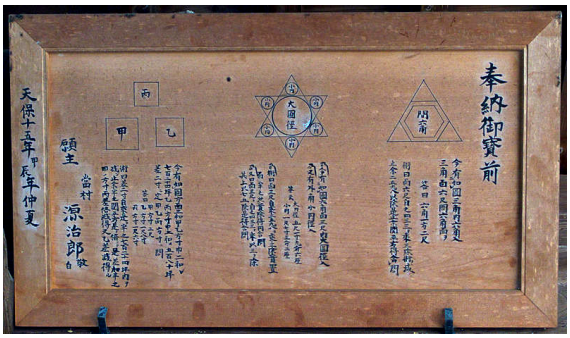
Problémy tohoto typu sú zaujímavé, pretože v praxi sa stáva, že z predchádzajúcich procesov rezania materiálu a vypĺňania zostane materiál odlišných rozmerov, ktorý sa v niektorých prípadoch dá opäť použiť. V anglickej terminológii sa stretneme s názvom `Residual Cutting Stock Problem`. Takýmto problémom sa venujú práce [31], [96] aj novšia práca [25].

Problém vypĺňania nádoby jedného tvaru

Klasický problém vypĺňania nádoby – `Classic Bin Packing Problem` je 1–rozmerný problém tohoto typu. Množinu objektov tvoria navzájom odlišné objekty a každý má svoju váhu. Kontajnery–nádoby sú identické a majú rovnakú nosnosť. Cieľom je povypĺňať nádoby objektmi tak, aby sme na to použili čo najmenej nádob a aby nosnosť každej nádoby nebola prekročená. V anglickej literatúre má tento problém tiež názvy `Vehicle Loading Problem` a `Binary Cutting Stock Problem`. Jednou z modifikácií tohoto problému je Problém vypĺňania nádoby k–objektmi – `k-Item Bin Packing Problem`, pri ktorom je maximálny počet objektov na jednu nádobu obmedzený na k. S Klasickým problémom vypĺňania nádoby sa môžeme stretnúť v práci [60] aj v novších prácach [76] a [28].



(a) Časti M. C. Escherových obrázkov a ich teselácia



(b) Sangaku z chrámu Enmanji v meste Nara, Japonsko

Obr. 1.1: Vypĺňanie v umení

Ďalším problémom tohoto typu je 2–rozmerný Problém vypĺňania nádoby – `Two-Dimensional Bin Packing Problem`, ktorý sa zaoberá tým, ako vyplniť kontajnery–obdĺžniky množinou navzájom odlišných objektov tak, aby sme minimalizovali počet kontajnerov. Podľa tvaru objektov môže mať tento problém rozvinutý názov, napríklad Obdĺžnikový problém vypĺňania nádoby alebo Kruhový problém vypĺňania nádoby. Takéto problémy rozoberá práca [68] a z novších prác sú to napríklad [90], [9] a [47].

Pri 3–rozmernom Probléme vypĺňania nádoby sú kontajnery rovnaké kvádre a objekty sú navzájom odlišné. Cieľom je uložiť všetky objekty do čo najmenšieho počtu kontajnerov. Ak sú objekty kocky, tak sa môžeme stretnúť s názvom Problém vypĺňania kockami – `Cube Packing Problem`. Spomedzi novších prác sa 3–rozmerným Problémom vypĺňania zaoberajú napríklad [23] a [49].

Problém vypĺňania nádoby rôznych tvarov

Problémy tohoto typu sú rozšírením predchádzajúceho typu a konkrétny problém je napríklad v anglickej literatúre spomínaný ako `Variable-Size Bin Packing Problem`. Tento problém nájdeme napríklad v novších prácach [87] a [107].

Problém vypĺňania zvyškovej nádoby

Problémy tohoto typu sú podobné ako Problémy rezania zo zvyškového materiálu, rozdiel je v tom, že množinu objektov tvoria navzájom odlišné objekty. V anglickej literatúre sa označujú ako `Residual Bin Packing Problem`. Stretneme sa s nimi v novších prácach [84] a [6].

Príbuzné problémy a využitia vypĺňania

Teselácia (`Tessellation`) v 2–rozmernom priestore je vypĺňanie roviny jedným alebo viacerými geometrickými tvarmi tak, že sa tvary neprekrývajú a nie sú medzi nimi medzery. Môžeme sa s ňou stretnúť napríklad pri dláždení. V prírode je snáď najznámejším príkladom včelí plást. V matematike je teselácia zovšeobecnená na viacero rozmerov a s týmto pojmom sa stretneme aj v počítačovej grafike. Teselácia sa považuje aj za umelecký prvok a jedným zo známych umelcov, ktorý sa ňou zaoberal, bol dánsky umelec Maurits Cornelis Escher (1898–1972), ktorého motív možeme vidieť na obrázku 1.1a z [83].

Sangaku sú japonské geometrické problémy, ktoré boli maľované na drevené dosky a prinášané do chrámov v období 17.–19. storočia. Viaceré z nich sa zaoberajú vypĺňaním geometrických tvarov, ako môžeme vidieť na obrázku 1.1b z [39]. Viac informácií môžeme nájsť v práci [20].

Zhrnutie

Vo všeobecnosti odvodzovanie názvu problému prebieha zhruba nasledovne: prvú časť tvorí počet dimenzií, druhú tvar objektov a tretiu konkrétny názov alebo názov odvodeného typu Problémov vypĺňania (tabuľka 1.4). Napríklad v tejto práci sa venujeme 2D obdĺžnikovému Problému vypĺňania nádoby (jedného tvaru) – vypĺňame rovnaké obdĺžniky množinou navzájom rôznych obdĺžnikov, 2D obdĺžnikovému Problému vypĺňania pásu — vypĺňame pás s fixnou šírkou množinou navzájom rôznych obdĺžnikov — a 2D obdĺžnikovému Problému vypĺňania zvyškovej nádoby – vypĺňame zvyšok obdĺžnika po predchádzajúcom rezaní množinou navzájom rôznych obdĺžnikov. Neskôr sa venujeme týmto trom problémom aj vo verzii, keď sú objekty nepravidelného (iného ako obdĺžnikového) tvaru.

V anglickej terminológii sa najčastejšie používajú skratky, ktoré vyplynuli z typológie zavedenej spomínanou prácou [108], ktorú tu spracovávame. Teda stretneme sa s `2D rectangular/irregular SBSBPP` (Single–Bin–Size Bin Packing Problem), `2D rectangular/irregular SPP` (Strip Packing Problem) a `2D rectangular/irregular RBPP` (Residual Bin Packing Problem).

Kapitola 2

Problém vypĺňania obdĺžnika polygónmi bez rotácie

V kapitole 1 sme predstavili triedu Problémov vypĺňania. V tejto kapitole popíšeme konkrétny problém a náš prístup k jeho riešeniu.

2.1 Zadanie

Problém, ktorý riešime, je 2–rozmerný nepravidelný Problém vypĺňania nádoby (jedného tvaru). Množina kontajnerov sú rovnaké obdĺžniky a množina objektov sú navzájom odlišné uzavreté polygóny. Cieľom je vyplniť minimálny počet obdĺžnikov všetkými objektami tak, aby z posledného obdĺžnika zostala nevyplnená čo najväčšia súvislá časť obdĺžnikového tvaru. Tento problém je NP–ťažký [57], čiže nie je možné nájsť optimálne riešenie v polynomiálnom čase ak P≠NP. Preto sa na riešenie tohto a podobných problémov používajú heuristické algoritmy.

2.2 Motivácia

Takýmto problémom sme sa začali zaoberať na podnet z oblasti leteckého modelárstva. Modely lietadiel sa dajú kúpiť v modelárskom obchode už vyrezané a zabalené, pripravené na lepenie. Skúsení modelári si však rozkresľujú vlastné plány modelov lietadiel v grafickom softvéri. Jednotlivé súčiastky tohto plánu je potom potrebné v počítači uložiť do obdĺžnika, ktorý predstavuje plát materiálu, z ktorého budú vyrezané. Následne sa súčiastky vyrežú programovateľnou CNC frézou z obdĺžnikových plátov balzy alebo preglejky. Súčiastky sa očistia, obrúsia a môže sa začať stavba modelu.

Z tejto oblasti leteckého modelárstva a procesu, akým sa súčiastky vyrezávajú, vyplývajú isté špecifikácie vyššie uvedeného problému. Prvou dôležitou špecifikáciou je, že objekty–súčiastky nemôžu byť pri ukladaní rotované kvôli štruktúre dreva, z

ktorého budú vyrezávané. Je to dôležité z hľadiska pevnosti modelu lietadla. Druhou dôležitou špecifikáciou je, že medzi uloženými súčiastkami v obdĺžniku musia byť určité medzery kvôli tomu, že vrták frézy reže súčiastky po vonkajšom obvode a mohlo by sa stať, že pri rezaní zasiahne susediacu súčiastku a obe by sa mohli poškodiť.

2.3 Riešenie

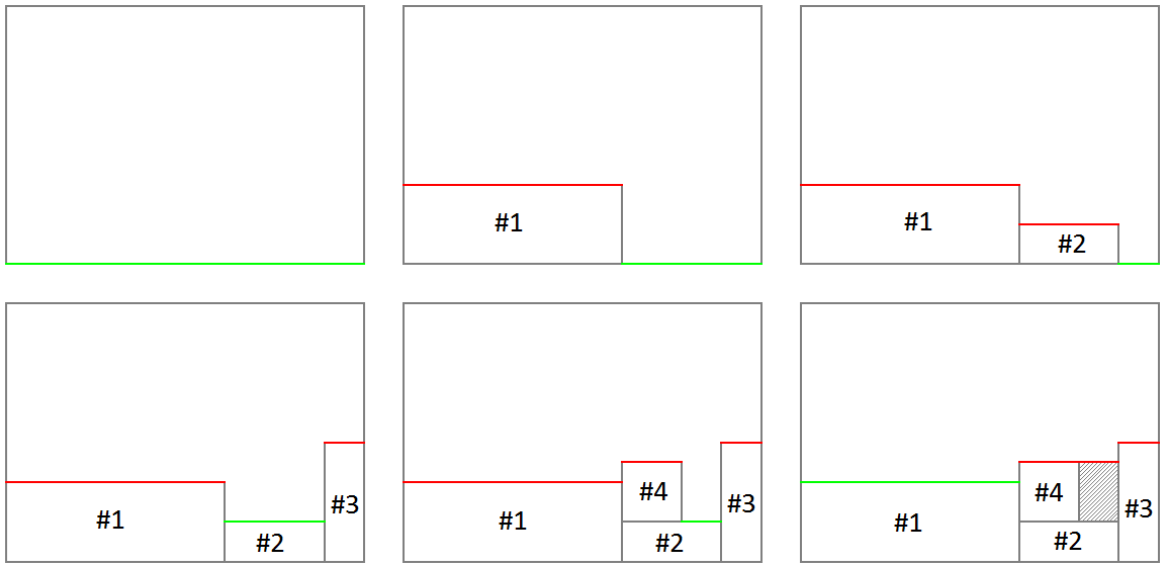
Naše riešenie tohto problému vznikalo v dvoch fázach. V prvej fáze sme sa rozhodli, že súčiastky „obalíme“ do obdĺžnikov. Týmto rozhodnutím sme 2D nepravidelný Problém vypĺňania nádoby zredukovali na 2D obdĺžnikový Problém vypĺňania nádoby. Obdĺžnik volíme tak, aby jeho strany boli paralelné s nádobou (obdĺžnikovým plátom) a aby do seba vpísanú súčiastku obaľoval tesne. Tento postup sme si zvolili preto, lebo 2D obdĺžnikový Problém vypĺňania nádoby je jednoduchší ako jeho nepravidelný variant. V ďalšom kroku sme si vybrali best–fit heuristický algoritmus z práce [57] od Shinji Imahoriho a Mutsunori Yagiura.

Best–fit heuristický algoritmus

Tento algoritmus pôvodne uviedli E. K. Burke, G. Kendall a G. Whitwell vo svojej práci [12]. Pôvodne je navrhnutý na riešenie 2D obdĺžnikového Problému vypĺňania pásu a je známy svojou jednoduchosťou a dobrou kvalitou výsledkov. Best–fit heuristika je pažravý algoritmus, ktorý sa zaoberá momentálne najnižším voľným miestom a snaží sa uložiť tam obdĺžnik, ktorý tam najlepšie pasuje (`best-fit`). Tento výraz zadefinujeme presne v definícii 1. Na rozdiel od metód, ktorých výsledok závisí od postupnosti obdĺžnikov na vstupe, best–fit heuristika dynamicky vyberá, ktorý obdĺžnik ide uložiť.

Charakteristickým prvkom best–fit algoritmu je *skyline*. Skladá sa z postupnosti segmentov–čiar, ktoré spĺňajú nasledujúce podmienky [57]: (1) každý segment je rovnobežný s osou x, (2) dva susedné segmenty majú rozdielne y súradnice a práve jednu rovnakú x súradnicu, (3) žiaden bod na segmente nie je zakrytý už uloženým obdĺžnikom a (4) každý segment sa dotýka hornej hrany už uloženého obdĺžnika alebo spodnej hrany pásu, resp. obdĺžnika, ktorý vypĺňame. Spomedzi všetkých segmentov je *najnižší segment* taký, ktorý ma najmenšiu y súradnicu.

Best–fit heuristika opakuje v cykle dve operácie, kým nie sú všetky obdĺžniky uložené. Toto platí pre 2D obdĺžnikový Problém vypĺňania pásu. Aby sa táto heuristika dala použiť na 2D obdĺžnikový Problém vypĺňania nádoby, upravíme podmienku nasledovne: ... kým nie sú všetky obdĺžniky, ktoré svojím uložením na *najnižší segment* neprekročia výšku obdĺžnika, ktorý vypĺňame, uložené. Túto podmienku podrobnejšie vysvetlíme neskôr. Dve operácie, ktoré sa opakujú, sú: (1) nájdi *najnižší segment* na aktuálnej *skyline* a (2) ulož na toto miesto obdĺžnik.



Obr. 2.1: Príklady skyline a ich najnižších segmentov

Na začiatku vypĺňania je obdĺžnik prázdny a *skyline* je jedna čiara—spodná hrana obdĺžnika. Na obrázku 2.1 sú červenou farbou zvýraznené segmenty *skyline* a zelenou farbou je vyznačený *najnižší segment*. Zakaždým, keď sa uloží obdĺžnik, časť *najnižšieho segmentu* sa presunie hore tak, že tá časť, ktorá je zakrytá spodnou hranou uloženého obdĺžnika, je nahradená hornou hranou toho obdĺžnika. Ak je viacero segmentov s rovnakou najmenšou y súradnicou, algoritmus vyberie ten segment, ktorý je najviac vľavo.

Definícia 1 (Najlepšie pasujúci objekt – obdĺžnikové objekty) *Pre daný najnižší segment je najlepšie pasujúci objekt najširší obdĺžnik, ktorý ešte nebol uložený, jeho šírka nie je väčšia ako šírka najnižšieho segmentu a jeho výška po uložení neprekročí výšku obdĺžnika, ktorý vypĺňame.*

Ak je šírka ukladaného obdĺžnika menšia ako šírka *najnižšieho segmentu*, obdĺžnik bude uložený čo najviac vľavo (ľavá stratégia). Existujú ešte ďalšie dve stratégie: obdĺžnik bude uložený k segmentu–susedovi, ktorý je vyšší (vysoká stratégia), a obdĺžnik bude uložený k segmentu–susedovi, ktorý je nižší (nízka stratégia). Pokiaľ pre *najnižší segment* nie je medzi neuloženými obdĺžnikmi pasujúci obdĺžnik, *najnižší segment* je zdvihutý na úroveň jeho nižšieho suseda a tieto dva segmenty sú spojené. Stručný pseudokód k tomuto algoritmu je spomenutý v kapitole 3 (algoritmus 1, algoritmus 2).

Rotácia o 90° kvôli špecifikácii nášho problému nie je dovolená. Vo všeobecnosti dáva pri hľadaní riešenia dodatočnú voľnosť, čiže môže viesť k lepším riešeniam. V tejto práci sa riešeniami s povolenou rotáciou nezaoberáme, ani nepoznáme heuristiku, ktorá pri použití rotácie ponúka garancie podobné tým, ktoré ponúka best–fit heuristika bez rotácie.

Aproximačná garancia best–fit heuristiky

Best–fit heuristiku sme si vybrali aj preto, lebo Shinji Imahori a Mutsunori Yagiura vo svojej práci [57] dokázali hornú hranicu odchýlky heuristického riešenia od optima v najhoršom prípade (`Worst-case approximation ratio`). V tejto časti v krátkosti odprezentujeme ich výsledky. Nech h_{OPT} je optimálna výška pásu pre danú inštanciu a h_{BF} je výška vypočítaná best–fit heuristikou. Prvá vec, ktorú autori ukázali, je negatívny aspekt tejto heuristiky: keď je rotácia obdĺžnikov o 90° povolená, heuristika nevie garantovať odchýlku aproximácie.

Veta 2 *Nech pri 2D obdĺžnikom Probléme vypĺňania pásu môže byť každý obdĺžnik rotovaný o 90°. Potom pre každú konštantu $M > 0$ existuje inštancia, pre ktorú platí $h_{BF}/h_{OPT} > M$.*

Potom sa pozreli na prípad bez rotácie obdĺžnikov. Nech n je počet obdĺžnikov, ktoré treba uložiť, a α je kladné číslo spĺňajúce $\alpha^\alpha = n$. Následne uviedli tieto dve lemy:

Lema 3 *Pre 2D obdĺžnikový Problém vypĺňania pásu s fixnou orientáciou obdĺžnikov existuje inštancia, pre ktorú platí $h_{BF}/h_{OPT} > \alpha/2$ - 1.*

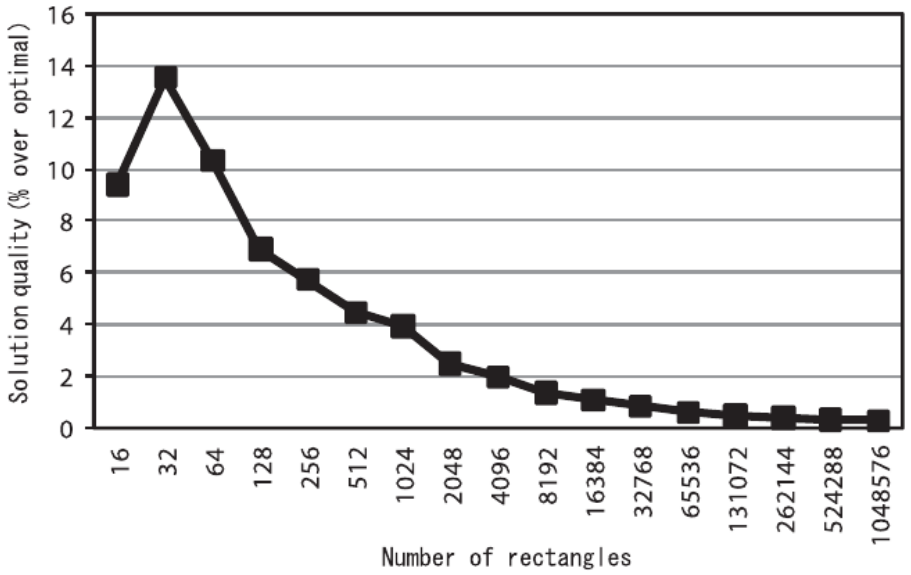
Lema 4 *Best–fit heuristika vždy nájde riešenie 2D obdĺžnikového Problému vypĺňania pásu s fixnou orientáciou obdĺžnikov, pre ktoré platí $h_{BF}h_{OPT} < 2\alpha + 3$.*

Pomocou lem 3 a 4 autori dokázali nasledujúcu vetu o aproximačnej garancii best–fit heuristiky:

Veta 5 *Best–fit heuristika je $\Theta(\alpha)$-aproximačný algoritmus pre 2D obdĺžnikový Problém vypĺňania pásu s fixnou orientáciou obdĺžnikov.*

Autori v práci [57] uviedli aj svoje výsledky vzhľadom na kvalitu riešenia a počet obdĺžnikov, ktoré treba uložiť. Experimentmi sa im podarilo dosvedčiť, že čím je počet obdĺžnikov väčší, tým je rozdiel medzi optimálnym riešením a riešením best–fit heuristiky menší. Inštancie, ktoré testovali, mali od $n = 16$ do 1048576 obdĺžnikov a obdĺžniky mohli byť rotované, aby sa predišlo vzniku veží. Optimálna výška pásu bola vopred známa.

Na obrázku 2.2 zo spomínanej práce [57] môžeme na horizontálnej osi vidieť počet ukladaných obdĺžnikov a na vertikálnej osi kvalitu riešenia best–fit heuristiky, ktorá je uvedená ako odchýlka (v percentách) od optimálneho riešenia $100(h_{BF} - h_{OPT})/h_{OPT}$. Teda menšia hodnota znamená lepšie riešenie. Z obrázka 2.2 vyplýva, že počet obdĺžnikov má zásadný vplyv na kvalitu riešenia a čím je počet obdĺžnikov väčší, tým je kvalita riešenia nájdeného best–fit heuristikou lepšia.



Obr. 2.2: Kvalita riešenia (% nad optimom) pre inštancie rôznych veľkostí

Účelová funkcia

Cieľom problému, ktorý riešime, je vyplniť minimálny počet obdĺžnikov všetkými objektami tak, aby z posledného obdĺžnika zostala nevyplnená čo najväčšia súvislá časť obdĺžnikového tvaru. Pri navrhnutom riešení sme obsah pod *skyline* identifikovali ako hodnotu, ktorú sa pri zvolenom algoritme snažíme minimalizovať. Keď sa na to pozrieme z predchádzajúceho pohľadu výšky riešenia pri 2D obdĺžnikovom Probléme vypĺňania pásu, vidíme, že čím je výška riešenia väčšia, tým je väčší obsah pod *skyline* a naopak, čím je výška riešenia menšia, tým je obsah pod *skyline* menší.

Túto hodnotu — obsah pod *skyline* — sme použili aj na merianie kvality riešení.

2.4 Riešenie – 2. fáza

V druhej fáze vytvárania nášho riešenia sme sa zamerali na to, že obalením súčiastok do obdĺžnikov môže vzniknúť voľný priestor– odpad. Best–fit algoritmus tento priestor ako odpad nepočíta, lebo objekty, ktoré ukladá, sú preň jednoduché obdĺžniky. V tejto fáze sme zmenili pohľad na problém. Od 2D obdĺžnikového Problému vypĺňania nádoby sme prešli k 2D nepravidelnému Problému vypĺňania nádoby.

Jedným zo základných pravidiel Problémov vypĺňania je, že objekty sa pri ukladaní nemôžu prekrývať. Keď sme riešili obdĺžnikový problém, obdĺžniky sme prekryť nemohli. Náš nápad bol, že pri nepravidelnom probléme obdĺžniky prekryť môžeme pod podmienkou, že sa neprekryjú objekty, ktoré sú v nich vpísané. Teda rozhodli sme sa modifikovať best–fit algoritmus tak, že sme upravili spôsob, ako pre *najnižší segment* nájsť najlepšie pasujúci objekt–súčiastku (už nie obdĺžnik), ktorá bude na dané miesto

uložená.

Pôvodne sme za najlepšie pasujúci objekt pre daný *najnižší segment* považovali najširší obdĺžnik, ktorý ešte nebol uložený, jeho šírka nie je väčšia ako šírka *najnižšieho segmentu* a jeho výška po uložení neprekročí výšku obdĺžnika, ktorý vypĺňame. My sme tento výber najlepšie pasujúceho objektu upravili nasledovne. Pre každú ešte neuloženú súčiastku obalenú do obdĺžnika vypočítame dve hodnoty–obsahy. Prvý obsah získame tak, že obdĺžnik so súčiastkou uložíme do ľavého rohu *najnižšieho segmentu* a potom skúsime posunúť súčiastku čo najviac doľava a následne čo najviac dole tak, aby sa súčiastky neprekryli. To, čo sa prekryje z obdĺžnikov, je spomínaný obsah. Druhý obsah získame podobne, najprv posun dole a potom posun doľava. Ak sa v niektorom prípade posúvania (doľava–dole, dole–doľava) súčiastka svojou šírkou nezmestí na *najnižší segment*, alebo svojou výškou presahuje výšku obdĺžnika, ktorý vypĺňame, tento prípad bude mať hodnotu 0. Následne súčiastke priradíme väčšiu z týchto dvoch hodnôt.

Definícia 6 (Najlepšie pasujúci objekt – nepravidelné objekty) *Pre daný najnižší segment je najlepšie pasujúci objekt obdĺžnik, ktorý ešte nebol uložený a prekryje čo najväčší obsah z priestoru vyplneného už uloženými obdĺžnikmi. Po posune sa musí šírkou zmestiť na najnižší segment a jeho výška po uložení nesmie prekročiť výšku obdĺžnika, ktorý vypĺňame.*

Ak je najväčšia hodnota 0, použijú sa pôvodné podmienky na výber najlepšie pasujúceho objektu. Naším spôsobom vyberania najlepšie pasujúceho objektu určite nevyrobíme riešenie s väčším celkovým obsahom pod *skyline* ako pri použití pôvodných podmienok.

Zhrnutie

V tejto kapitole sme predstavili problém, ktorým sa zaoberá naša práca. Navrhli sme riešenie, ktoré je modulárne (od pôvodného heuristického algoritmu [57] sa líši len v jednom kroku) a zároveň garantuje aspoň takú dobrú kvalitu riešenia ako algoritmus [57]. Takýmto prístupom sme chceli dosiahnuť, aby sme mali jednoduchú a dobrú kostru riešenia (best–fit heuristika), s ktorou sa, vďaka jej jednoduchosti, dá nárabať ďalej. V kapitole 3 detailnejšie popisujeme implementáciu nášho riešenia a v kapitole 4 prezentujeme výsledky experimentov.

Kapitola 3

Riešenie problému vypĺňania v priemyselnej aplikácii

Súčasťou tejto bakalárskej práce je program, ktorý implementuje riešenie navrhnuté v kapitole 2. V tejto kapitole bližšie opíšeme niektoré prvky tohoto programu.

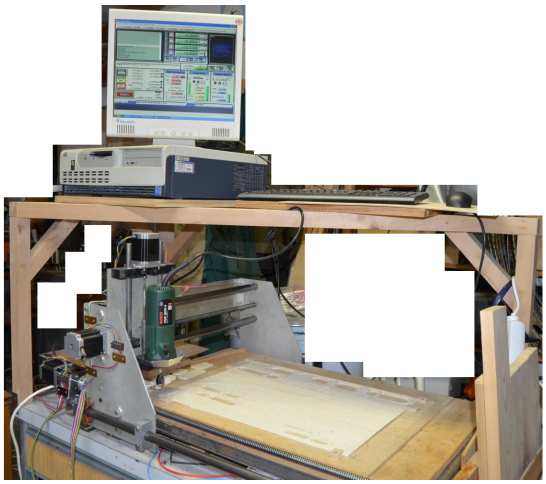
V sekcii 2.2 uvádzame, čo bolo motiváciou tejto práce, a na tomto mieste chceme pripomenúť špecifikácie problému, ktoré vyplývajú z priemyselnej aplikácie.

Cieľom programu je v prvom rade vypĺňať obdĺžniky, predstavujúce pláty dreva, súčiastkami modelu lietadla s cieľom minimalizovať obsah uložených objektov. Následne sa z drevených plátov vyrežú súčiastky programovateľnou CNC frézou (`Computer Numerical Control`). Pri ukladaní súčiastok modelu rotácia nie je povolená, lebo je potrebné dodržať požadovanú orientáciu kvôli štruktúre dreva, z ktorého budú vyrezané. Program potrebuje na vstupe parameter, ktorý udáva, aká má byť minimálna dovolená vzdialenosť uložených súčiastok v obdĺžniku, aby sa predišlo poškodeniu vrtákom frézy, ktorá súčiastky vyrezáva po vonkajšom obvode. Na obrázku 3.1a môžeme vidieť CNC frézu, plát dreva (obr. 3.1b) so súčiastkami a model lietadla (3.1c, 3.1d).

3.1 Základné informácie

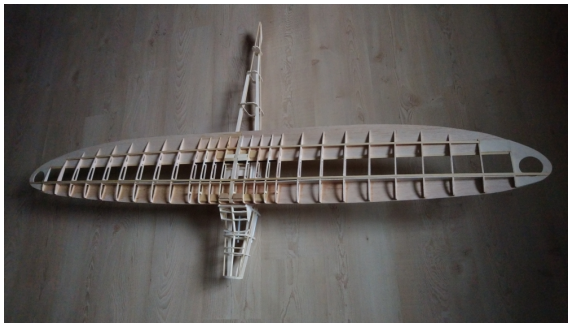
Program je napísaný v programovacom jazyku C a pre operačný systém Linux. Interakcia s programom prebieha cez príkazový riadok (`command line`). Dá sa použiť na riešenie šiestich Problémov vypĺňania: (1) 2D nepravidelný Problém vypĺňania nádoby, (2) 2D nepravidelný Problém vypĺňania pásu, (3) 2D nepravidelný Problém vypĺňania zvyškovej nádoby, (4) 2D obdĺžnikový Problém vypĺňania nádoby, (5) 2D obdĺžnikový Problém vypĺňania pásu a (6) 2D obdĺžnikový Problém vypĺňania zvyškovej nádoby. Program očakáva na vstupe šesť parametrov:

(i) názov súboru s neuloženými súčiastkami plánu lietadla vo formáte DXF,



(a) CNC fréza

(b) Súčiastky vyrezané z plátu preglejky



(c) Model lietadla bez poťahu

(d) Model lietadla

Obr. 3.1: Letecké modelárstvo

(ii) názov textového súboru s parametrami drevených plátov, z ktorých sa budú súčiastky vyrezávať,

(iii) názov súboru, v ktorom bude po zbehnutí programu jeden plát, vyplnený súčiastkami a pripravený na ďalšie spracovanie (DXF),

(iv) názov súboru, v ktorom bude po zbehnutí programu jeden plát, vyplnený farebnými obdĺžnikmi pre lepšiu vizuálnu predstavu uloženia súčiastok obalených do obdĺžnikov (DXF),

(v) názov súboru, v ktorom budú po zbehnutí programu všetky súčiastky, ktoré ešte nie sú uložené (DXF),

(vi) názov textového súboru, ktorý bude po zbehnutí programu obsahovať popis výslednej *skyline*.

Počas behu programu sa vytvorí ešte jeden súbor `value-log.txt`, do ktorého sa po každom uložení súčiastky pripíše hodnota obsahu pod *skyline*.

group code	vysvetlenie	poznámky
90	počet vrcholov	povinná hodnota
70	indikátor uzavretosti	povinná hodnota, 1 = uzavretá
10	súradnica vrchola na osi x	povinná hodnota pre každý vrchol
20	súradnica vrchola na osi y	povinná hodnota pre kažný vrchol
42	zakrivenie segmentu vychádzajúceho z vrchola	nepovinná hodnota pre každý vrchol

Tabuľka 3.1: Vybrané group code-y a ich vysvetlenie

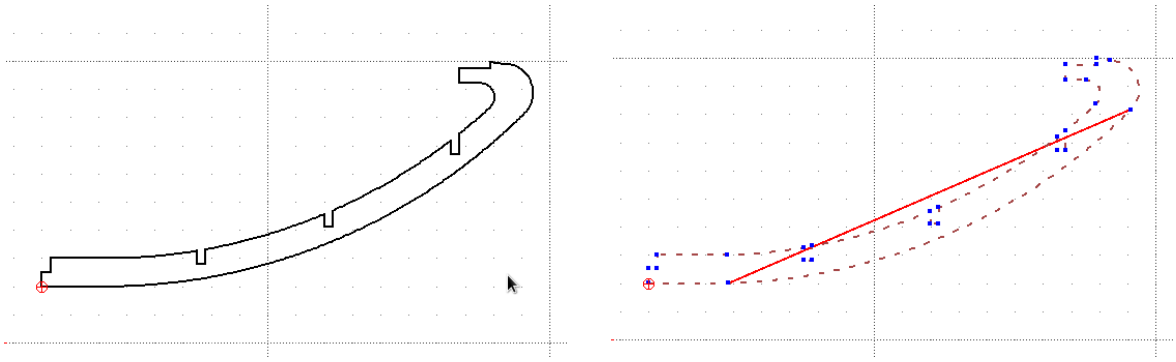
Na konci behu pri 2D Probléme vypĺňania nádoby sa na štandardný výstup vypíšu dve hodnoty: obsah plátu, ktorý bol vypĺňaný, a obsah obdĺžnikov, ktorými bol vyplnený. Jeden beh programu vždy vyplní jeden plát, čiže na uloženie všetkých objektov je niekedy potrebné spustiť program viackrát. Kvôli jednoduchšej obsluhe sme pridali k programu aj skript, ktorý tento proces automatizuje.

Pri 2D Probléme vypĺňania pásu sa vypíše len jedna hodnota – dosiahnutá výška kontajnera. Pri riešení tohoto problému sa program správa tak, že zo vstupného súboru s „parametrami drevených plátov“ zoberie iba šírku plátu–pásu a parameter vzdialenosti objektov a ako kontajner nainicializuje obdĺžnik s danou šírkou a nekonečnou výškou (16 miliónov milimetrov). V tomto prípade program na nájdenie kompletného riešenia potrebuje iba jeden beh.

3.2 Drawing eXchange Format

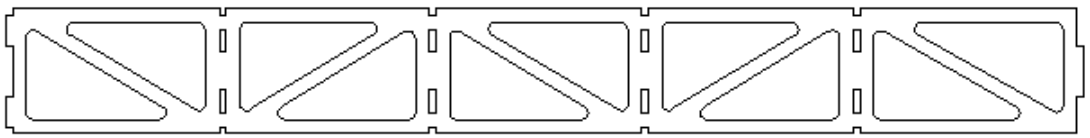
`Drawing eXchange Format` (DXF) je grafický formát vyvinutý spoločnosťou Autodesk na reprezentáciu informácií obsiahnutých v AutoCAD súbore–výkrese. Bol vytvorený za účelom prenášania dát medzi softvérovou aplikáciou AutoCAD a inými programami. Má štruktúru známu ako `tagged data`, čo znamená, že každý element dát má vopred číselné označenie nazývané *group code*, ktoré označuje, aký typ dátového elementu nasleduje. Toto označenie tiež udáva význam dátového elementu pre konkrétny záznam. Súbor DXF je organizovaný do sekcií, ktoré sa skladajú zo záznamov, a tie sa skladajú z dvojíc *group code* a *data*. Každý prvok *group code* a *data* je v DXF súbore na vlastnom riadku.

Náš program spracováva a vytvára DXF súbory ASCII formátu (popísaný vyššie, existuje aj binárny formát), dĺžky sú implicitne udané v milimetroch. Vstupné DXF súbory musia byť vo verzii 2007/2008/2009 (identifikátor AC1021). Program načítava informácie o objektoch zo sekcie `ENTITIES`. Práve táto sekcia obsahuje informácie o grafických objektoch na výkrese. Objekty, ktoré program spracováva, musia byť záznamy typu `LWPOLYLINE`. Pri tomto type záznamu sú pre nás dôležité dáta označené *group code*–mi vysvetlenými v tabuľke 3.1. *Group code* 42 popisuje zakrivenie čiary



(a) Súčiastka (b) Porovnanie krivky a rovnej čiary

Obr. 3.2: Porovnanie krivky a rovnej čiary medzi bodmi na súčiastke



Obr. 3.3: Príklad uzavretých vnorených polyline

vychádzajúcej z prislúchajúceho vrcholu. Naša implementácia číta a zapisuje hodnotu s *group code*–om 42, avšak ignoruje ju. To znamená, že so všetkými segmentmi polyline sa zaobchádza ako s rovnými čiarami. Na obrázku 3.2a je súčiastka, ktorej aproximácia (obr. 3.2b) je ďaleko od reality. Tento prístup môže pri prekrývaní obdĺžnikov spôsobiť problém (viac v kapitole 4).

Viac informácií o formáte DXF je možné nájsť napríklad v manuále [4].

3.3 Vstupné súbory

Vstupný súbor so súčiastkami musí byť vo formáte DXF. Súčiastky sa musia skladať iba z uzavretých kriviek, prislúchajúcich záznamom typu `LWPOLYLINE`. Jedna súčiastka sa môže skladať z viacerých uzavretých vnorených polyline (obr. 3.3). Všetky súradnice vrcholov musia byť kladné čísla. Súčiastky musia byť na vstupnom výkrese umiestnené tak, aby sa po obalení do obdĺžnikov tieto obdĺžniky neprekrývali.

Vstupný súbor s parametrami drevených plátov musí obsahovať tri čísla na jednom riadku oddelené medzerami. Prvé číslo je šírka plátu, druhé číslo je výška plátu a tretie číslo je vzdialenosť, v ktorej by uložené súčiastky mali byť od seba vzdialené. Táto vzdialenosť musí byť $\geq 1mm$. Všetky tri čísla musia byť uvedené v milimetroch. Algoritmus ukladania obdĺžnikov bez prekryvu garantuje, že vzájomná vzdialenosť súčiastok bude $\geq parameter$. Avšak algoritmus prekryvu obdĺžnikov dáva pri súčasnej implementácii túto garanciu len vtedy, ak vstupný DXF súbor nepoužíva v popise

objektov zakrivenie segmentov polylines (t.j. *group code* 42 v `LWPOLYLINE`).

Vstupné súbory zostanú po skončení behu programu nezmenené.

3.4 Výstupné súbory

Prvý výstupný súbor je vo formáte DXF a obsahuje iba záznamy typu `LWPOLYLINE` prislúchajúce uloženým objektom. Tento súbor je pripravený na ďalšie spracovanie, potrebné pred samotným vyrezávaním.

Druhý výstupný súbor je vo formáte DXF a reprezentuje plát dreva, ktorý je vyplnený obdĺžnikmi. Tento súbor je vytvorený nato, aby ponúkol iný pohľad na vyplnenie plátu a nebude v ďalšom procese opäť použitý.

Tretí výstupný súbor je tiež vo formáte DXF a obsahuje objekty, ktoré pri tomto behu programu neboli uložené. Ak tento súbor nie je prázdny, použije sa v nasledujúcej iterácii ako vstupný súbor.

Štvrtý výstupný súbor je textový súbor, ktorý má význam iba v prípade, že všetky objekty už boli uložené. Súbor vtedy obsahuje údaje o voľnej časti plátu v nasledujúcom formáte. Na prvom riadku sú rovnaké tri čísla ako pri vstupnom súbore s parametrami drevených plátov. Na ďalších riadkoch sú údaje o všetkých častiach *skyline*, jeden riadok pre každú časť. Tieto riadky tiež pozostávajú z troch čísel, oddelených medzerami: (1) ľavá súradnica na osi x, (2) pravá súradnica na osi x a (3) súradnica na osi y. Parameter vzájomnej vzdialenosti objektov musí zostať nezmenený, lebo od neho závisí, ako *skyline* vyzerá. Tento súbor sa dá potom použiť ako vstupný súbor s parametrami časti dreveného plátu na riešenie 2D Problému vypĺňania zvyškovej nádoby. *Skyline* sa inicializuje podľa údajov z tohoto súboru.

Posledný súbor, ktorý sa mení počas behu programu, sa nazýva `value-log.txt` a do tohoto súboru sa po každom uložení objektu pripíše na nový riadok obsah pod *skyline*. Tento súbor vznikne pri prvom behu programu a v ďalších iteráciách sa do súboru pripisuje hodnota pre každý plát od nuly.

Ak ostatné výstupné súbory existujú, ich obsah sa premaže a zapíšu sa do nich nové dáta. Ak výstupné súbory neexistujú, program tieto súbory vytvorí.

3.5 Príklady použitia programu

Keď sa program spustí s nesprávnym počtom argumentov, väčšinou nula, do príkazového riadku sa vypíše správne poradie argumentov, s ktorými je potrebné program spustiť, aby zbehol korektne.

```
./place-dxf
```

Správny spôsob, ako sputiť program s parametrami, je:

`./place-dxf` *<vstupný dxf súbor> <súbor s parametrami> <výstupný dxf súbor s uloženými súčiatkami> <výstupný dxf súbor s obdĺžnikmi> <výstupný dxf súbor s neuloženými súčiastkami> <výstupný súbor s parametrami>*

Príklad: `./place-dxf` vykres.dxf parametre.txt vykres-ready.dxf vykres-graph.dxf vykres-not-placed.dxf parametre-out.txt

Viac príkladov aj s obrázkami vstupných a výstupných súborov je v kapitole 4.

3.6 Pseudokód

V tejto časti uvedieme stručný pseudokód algoritmu, ktorý sme implementovali. Algoritmus je inšpirovaný prácou [57].

Data: DXF súbor s objektmi, txt súbor s parametrami
Result: DXF súbor s vyplneným plátom, grafický DXF súbor, DXF súbor s neuloženými objektmi, txt súbor s výslednou skyline, txt súbor so záznamami hodnotovej funkcie

```
načítaj parametre a inicializuj skyline;
načítaj objekty;
while existuje neuložený objekt a obdĺžnik nie je vyplnený do
    while pre aktuálne najnižší segment skyline existuje pasujúci objekt do
        nájdi najnižší segment skyline;
        nájdi najlepšie pasujúci objekt pre najnižší segment (alg2/alg3);
        umiestni tvar;
        uprav skyline;
    end
    zdvihni najnižší úsek skyline;
end
zapíš dáta do súborov;
```

Algorithm 1: Kostra algoritmu

```
utrieď objekty podľa šírky zostupne;
for objekt do
    if objekt nie je uložený a šírkou sa zmestí na segment a po uložení
     nepresiahne výšku kontajnera then
        vyber tento objekt ako najlepšie pasujúci;
        break;
    end
end
```

Algorithm 2: Výber najlepšie pasujúceho objektu – obdĺžnikový tvar

```
max-prekryv := 0;
for objekt do
    prekryv-LD = prekryv-DL = lokálny-max-prekryv := 0;
    ulož objekt do ľavého rohu najnižšieho segmentu;
    posuň objekt čo najviac doľava a potom dole (LD) bez prekrytia s už
     uloženými objektami;
    vypočítaj prekryv-LD;
    ulož objekt do ľavého rohu najnižšieho segmentu;
    posuň objekt čo najviac dole a potom doľava (DL) bez prekrytia s už
     uloženými objektmi;
    vypočítaj prekryv-DL;
    lokálny-max-prekryv := max{prekryv-LD, prekryv-DL};
    max-prekryv := max{max-prekryv, lokálny-max-prekryv};
end
if 0 = max-prekryv then
    použi na výber najlepšie pasujúceho objektu algoritmus 2;
else
    vyber objekt, ku ktorému prislúcha max-prekryv ako najlepšie pasujúci;
end
```

Algorithm 3: Výber najlepšie pasujúceho objektu – nepravidelný tvar

3.7 Počítanie posunu doľava

V tejto časti vysvetlíme, ako pre dve súčiastky s_1 a s_2 program vypočíta, o koľko môže byť súčiastka s_1, ktorú skúšame uložiť, prisunutá rovnobežne s osou x doľava k druhej súčiastke s_2, ktorá je už napevno uložená a viac sa nepohne. Pri počítaní posunu nahrádzame krivky medzi vrcholmi rovnými čiarami.

Princíp je v tom, že pre každú hranu h_1 súčiastky s_1 vypočítame jej vzdialenosť od každej hrany h_2 súčiastky s_2. Pre dvojicu hrán označíme túto hodnotu d. Výsledné posunutie D sa počíta ako minimum vzdialeností d cez všetky dvojice hrán h_1 a h_2, kde $h_1 \in s_1$ a $h_2 \in s_2$. T.j. $D = min\{d(h_1, h_2)\}$, kde $h_1 \in s_1$ a $h_2 \in s_2$. Hodnota D je dĺžka, o ktorú môžeme súčiastku s_1 prisunúť doľava k súčiastke s_2 bez toho, aby sa súčiastky prekryli. Nech p je hodnota parametra oddeľovacej vzdialenosti súčiastok zo vstupného súboru. Potom súčiastku s_1 môžeme prisunúť o vzdialenosť $D = max\{0, D - p\}$. Na obrázku 3.4 sú znázorené dvojice hrán s vyznačenou vzdialenosťou.

Pri hľadaní hodnoty D pre súčiastku s_1 je niekedy potrebné vypočítať vzdialenosť pre viac ako jednu súčiastku s charakteristikou s_2 a spomedzi všetkých týchto hodnôt musíme vybrať tú najmenšiu.

Pre dve hrany rôznych súčiastok vypočítame vzdialenosť nasledovne. Ak sa hrany pri posune nedotknú, vzdialenosť je ∞. Ak by koncový bod niektorej hrany pri posune zasiahol druhú hranu, použijeme vzorec na vypočítanie x súradnice bodu ležiaceho na úsečke s danou y súradnicou. Nech $v[x_1, y_1]$, $u[x_2, y_2]$ sú koncové body úsečky-hrany a $f[x_f, y_f]$ je hľadaný bod na tejto úsečke. Bod $g[x_g, y_g]$ je spomínaný koncový bod druhej hrany, teda $y_f = y_g$ (y_f je daná y súradnica). Hľadanú hodnotu x_f nájdeme pomocou vzorca [94]:

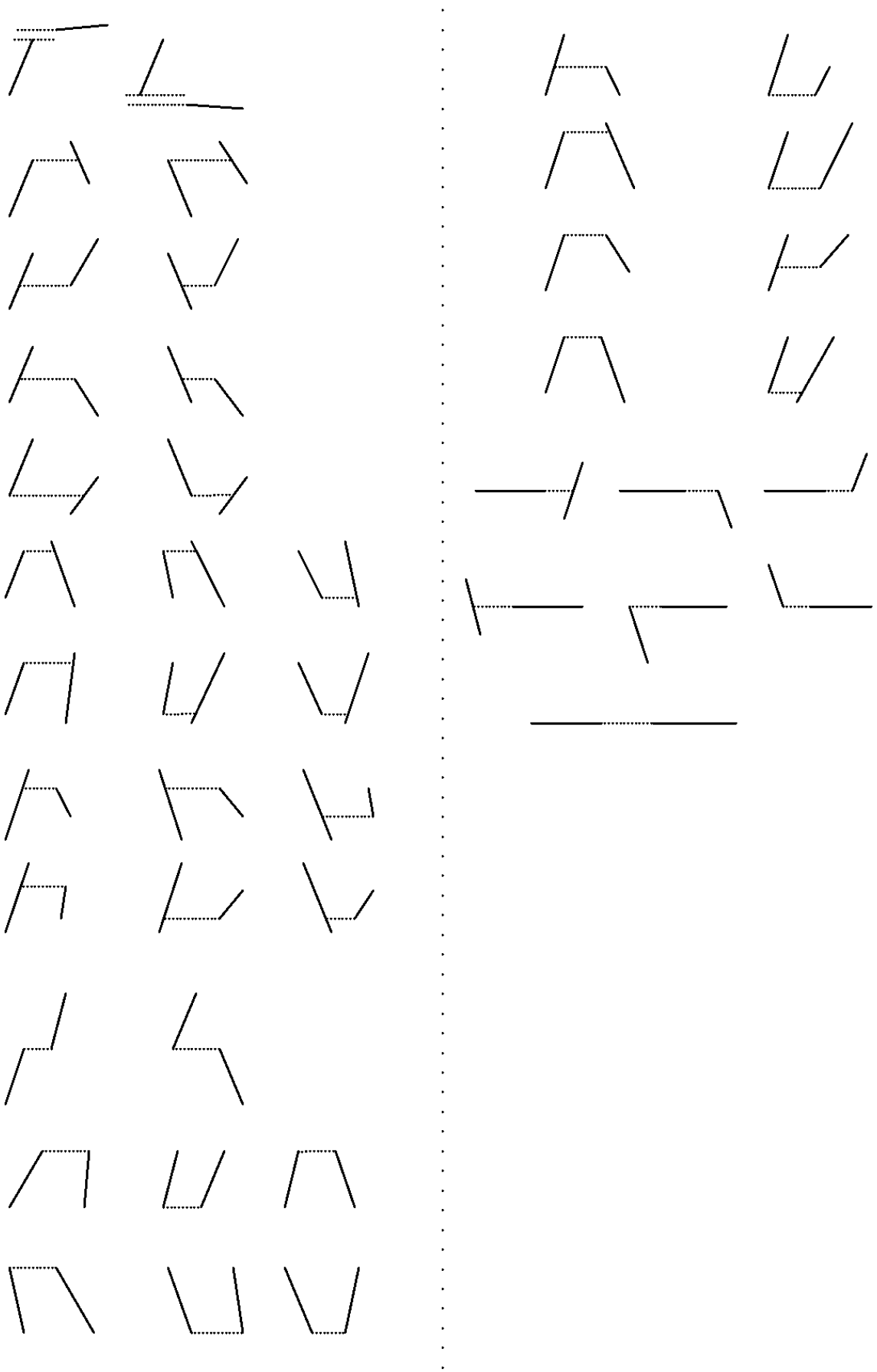
$$x_f = x_1 + \frac{(y_f - y_1) \cdot (x_2 - x_1)}{y_2 - y_1} \tag{3.1}$$

Následne vypočítame vzdialenosť bodov f a g ($max\{x_f, x_g\} - min\{x_f, x_g\}$). Keď sa vzorec 3.1 nedá použiť, vzdialenosť sa vypočíta spôsobom $max\{x, \bar{x}\} - min\{x, \bar{x}\}$ pre tie správne koncové body (y súradnica týchto bodov je rovnaká).

Na počítanie posunu smerom nadol program používa rovnakú metódu. Program hľadá vzdialenosť, o ktorú môže súčiastku posunúť rovnobežne s osou y smerom dole bez toho, aby sa súčiastky prekryli. Pred použitím metódy sa vytvoria kópie dvoch súčiastok, pre ktoré sa ide vypočítať vzdialenosť, a tie sa otočia o 90° v smere hodinových ručičiek. Potom sa vypočíta posun doľava pre tieto kópie presne podľa postupu opísaného vyššie.

Zhrnutie

V tejto kapitole sme popísali implementáciu riešenia navrhnutého v 2. kapitole. Uviedli sme pseudokód, ktorým sa snažíme objasniť úpravy pôvodného algoritmu. Popísali sme formát a význam vstupných a výstupných súborov. Nakoniec sme vysvetlili koncept nového algoritmu. V nasledujúcej 4. kapitole prezentujeme výsledky experimentov spolu s komentármi.



Obr. 3.4: Vzájomná poloha dvoch čiar
Pre danú dvojicu hrán je hodnota d dĺžka bodkovanej čiary.

Kapitola 4

Experimenty

V tejto kapitole prezentujeme výsledky experimentov. Obsahuje obrázky vstupných a výstupných súborov a grafy s komentármi na ich objasnenie. Demonštrujeme zlepšenia, ktoré priniesla modifikácia *skyline* algoritmu.

Experimenty boli robené na počítači s procesorom Intel Core 2 Duo CPU P8600 @ 2.40GHz, 4 GB RAM a operačným systémom Debian GNU/Linux 10 (buster). Nájdenie kompletného riešenia bez prekryvu obdĺžnikov trvalo na všetkých troch vstupoch pod 1.5 sekundy. Nájdenie riešenia po povolení prekryvu trvalo 3.3 sekundy na výkrese `drawing1`, 27.9 sekúnd na výkrese `drawing2` a 1.4 sekundy na výkrese `triangles`.

Na prezeranie a editovanie DXF súborov bol použitý program LibreCAD [75], verzia 2.1.3. Program bol úspešne odskúšaný aj na operačnom systéme Ubuntu 18.04.4 LTS.

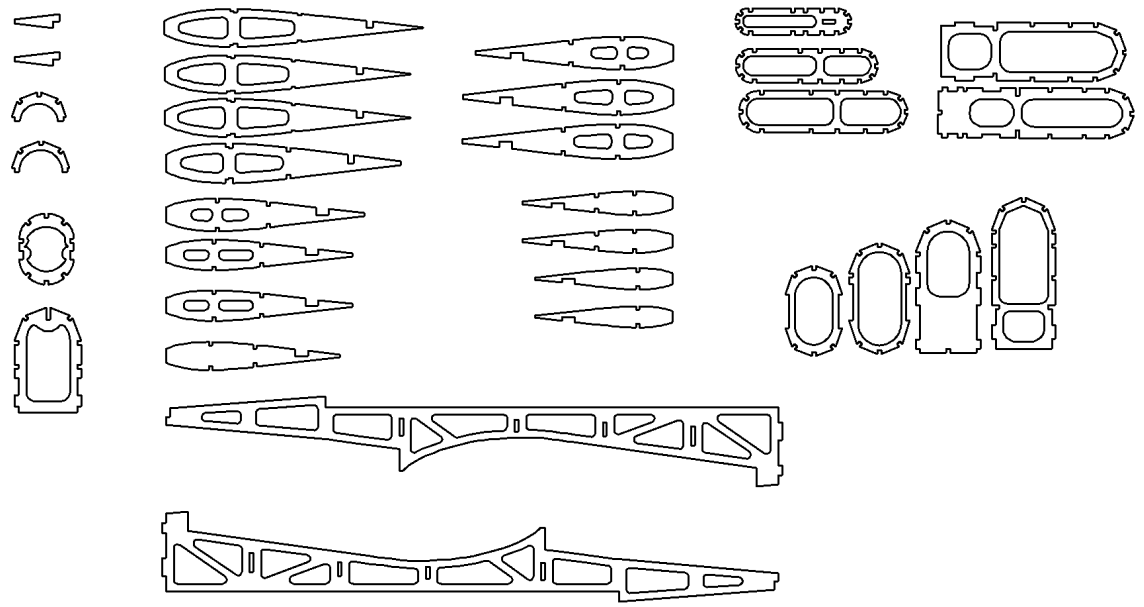
Nastavenie programu je možné pomocou makier, ktorými sa dá zapnúť/vypnúť prekrývanie obdĺžnikov a rovnako aj riešenie 2D Problému vypĺňania pásu. Viac informácii je v súbore `README.txt`, ktorý je súčasťou elektronickej prílohy.

4.1 Vstupné súbory

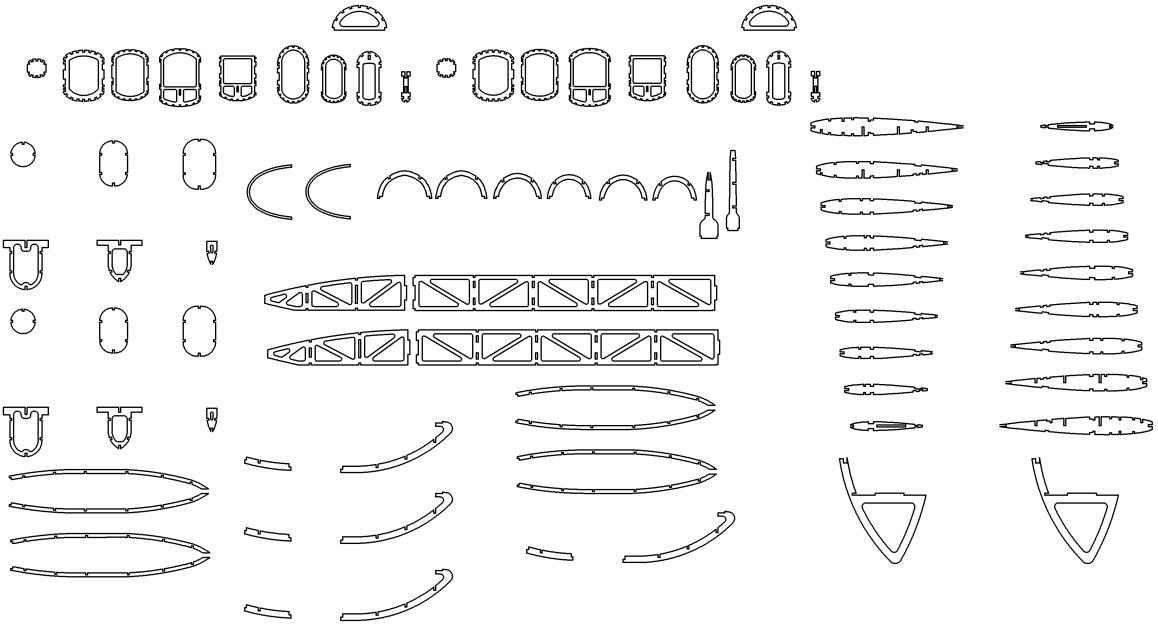
Prvé dva vstupné súbory sú výkresy so súčiastkami modelov lietadiel. Na obrázku 4.1a je výkres `drawing1` s menším počtom súčiastok (30+). Na obrázku 4.1b je výkres `drawing2` s väčším počtom súčiastok (80+), ktorý prislúcha modelu iného typu, ako je výkres na obrázku 4.1a. Modely lietadiel sa skladajú zo súčiastok rôznych veľkostí a tvarov. Modely sa okrem toho veľmi líšia aj počtom súčiastok, ktorý sa väčšinou pohybuje okolo niekoľkých desiatok.

Tretím vstupným súborom je výkres `triangles` s pármi trojuholníkov (50 párov) rovnakých rozmerov, ktoré do seba zapadajú (obr. 4.2). Trojuholník naľavo je o milimeter vyšší ako trojuholník napravo.

Rozmery drevených plátov sú nastavené na `500 x 400 mm`, čo je štandardný rozmer na modelárske účely. Medzera medzi objektmi je nastavená na `4 mm`.

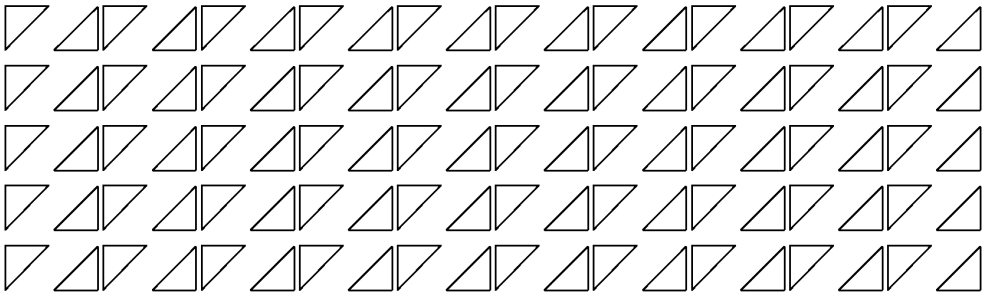


(a) Výkres `drawing1` (drawing1.dxf)



(b) Výkres `drawing2` (drawing2.dxf)

Obr. 4.1: Výkresy so súčiastkami modelov lietadiel



Obr. 4.2: Výkres `triangles` (triangles.dxf)

4.2 Výstupné súbory

Súčiastky z výkresu `drawing1` boli bez použitia prekryvu rozložené na dva pláty. Na obrázku 4.3a je prvý plát vyplnený súčiastkami a na obrázku 4.3b možeme vidieť uloženie z pohľadu obdĺžnikov. Druhý plát je na obrázkoch 4.4a a 4.4b. Po dovolení prekrývania obdĺžnikov boli súčiastky rozložené len na jeden plát (obrázky 4.5a, 4.5b).

Na obrázkoch 4.5a a 4.9a (resp. 4.10a, 4.11a) vidieť, že niektoré medzery medzi súčiastkami nie sú dostatočne veľké, alebo sa súčiastky prekrývajú. Tento problém je spôsobený ignoráciou zakrivenia čiar v súčasnej implementácii (*group code* 42 v `LWPOLYLINE`), o ktorom píšeme v časti 3.2. Program zatiaľ meria vzdialenosť len rovnobežne s osou x, resp. y, čiže aj toto môže byť dôvodom, prečo skutočná vzdialenosť môže byť menšia. Niekedy sa môže stať, že po posune doľava je vzdialenosť od súčiastky naľavo správna, ale po posune dole je menšia, ako má byť, lebo pri posune dole sa meria len vzdialenosť rovnobežná s osou y. Podobný problém môže nastať pri posune dole a potom doľava.

Súčiastky z výkresu `drawing2` boli bez použitia prekryvu rozložené na tri pláty. Prvý plát je na obrázkoch 4.6a a 4.6b, druhý 4.7a, 4.7b a tretí 4.8a, 4.8b. S použitím prekryvu sa výsledný počet plátov nezmenil. Prvý plát vidno na obrázkoch 4.9a a 4.9b, druhý 4.10a, 4.10b a tretí 4.11a, 4.11b.

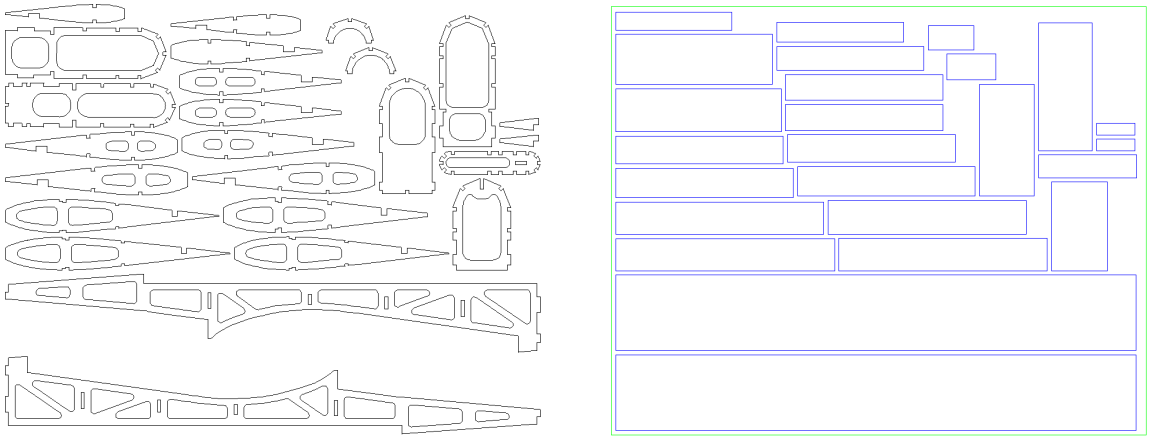
Uloženie trojuholníkov z výkresu `triangles` bez použitia prekryvu vyšlo na dva pláty, prvý je na obrázkoch 4.12a, 4.12b a druhý 4.13a, 4.13b. S použitím prekryvu sa trojuholníky podarilo uložiť na jeden plát, ktorý je na obrázkoch 4.14a a 4.14b.

Na obrázku 4.15a a 4.15b je ukážka riešenia 2D obdĺžnikového Problému vypĺňania pásu (`2D rectangular Strip Packing Problem`) pre výkres `drawing2`. Riešeniam tohoto problému veľkú pozornosť v tejto kapitole nevenujeme.

4.3 Výsledky

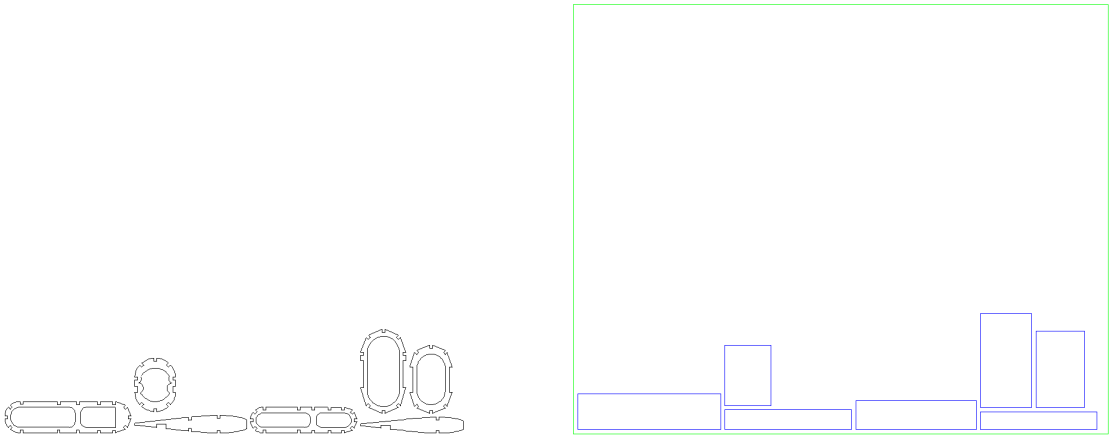
Na grafe 4.16a môžeme vidieť, že bez použitia prekryvu sa na prvý plát zmestilo 25 súčiastok z výkresu `drawing1` s celkovým obsahom (pod *skyline*) 191277.57 mm^2. Na druhom pláte bolo uložených zvyšných sedem s obsahom 26605.28 mm^2, čiže dokopy 217882.85 mm^2. S použitím prekryvu sa všetky súčiastky zmestili na jeden plát a obsah bol 177828.81 mm^2. Z toho vyplýva, že prekrývanie obdĺžnikov zlepšilo riešenie o 18.38 %.

Súčiastky z výkresu `drawing2` boli bez použitia prekryvu rozložené na tri pláty (obr. 4.16b): (i) 18 súčiastok – 191153.23 mm^2, (ii) 38 súčiastok – 188583.84 mm^2, (iii) 26 súčiastok – 164604.62 mm^2, dokopy 544341.69 mm^2. S použitím prekryvu sa počet plátov nezmenil: (i) 43 súčiastok – 190910.08 mm^2, (ii) 31 súčiastok – 188416.10 mm^2, (iii) 8 súčiastok – 75115.60 mm^2, dokopy 454441.78 mm^2. Prekrývanie obdĺžnikov v



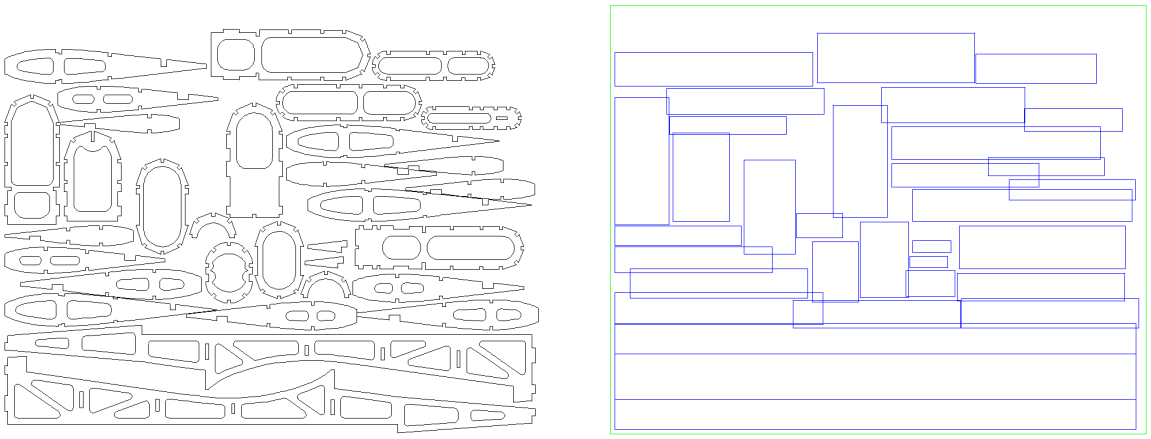
(a) ready-1.dxf (b) ready-1-graph.dxf

Obr. 4.3: Výstupné súbory výkresu `drawing1` (bez prekryvu) – 1. plát
Príkaz: `./place-dxf` drawing1.dxf parameters.txt ready-1.dxf ready-1-graph.dxf not-placed.dxf parameters-out-1.txt



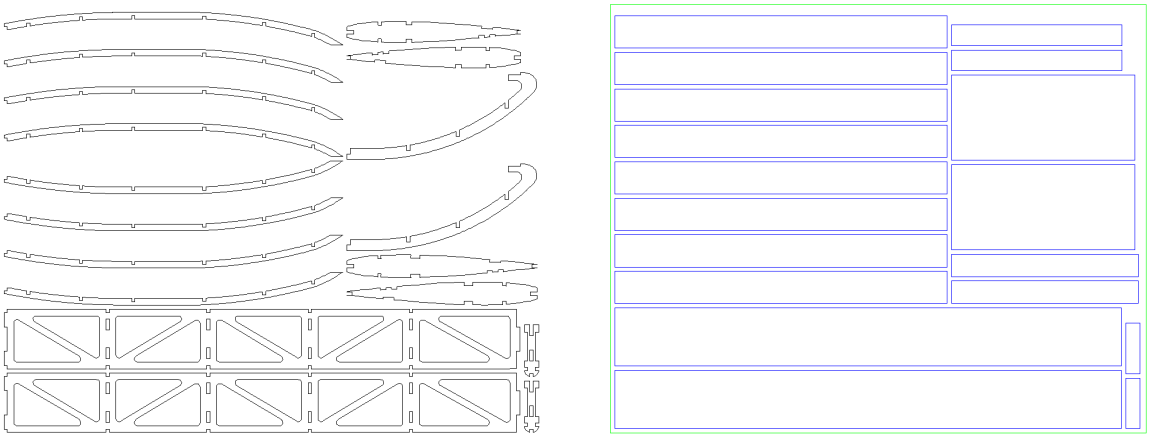
(a) ready-2.dxf (b) ready-2-graph.dxf

Obr. 4.4: Výstupné súbory výkresu `drawing1` (bez prekryvu) – 2. plát
Príkaz: `./place-dxf` not-placed.dxf parameters.txt ready-2.dxf ready-2-graph.dxf not-placed.dxf parameters-out-2.txt



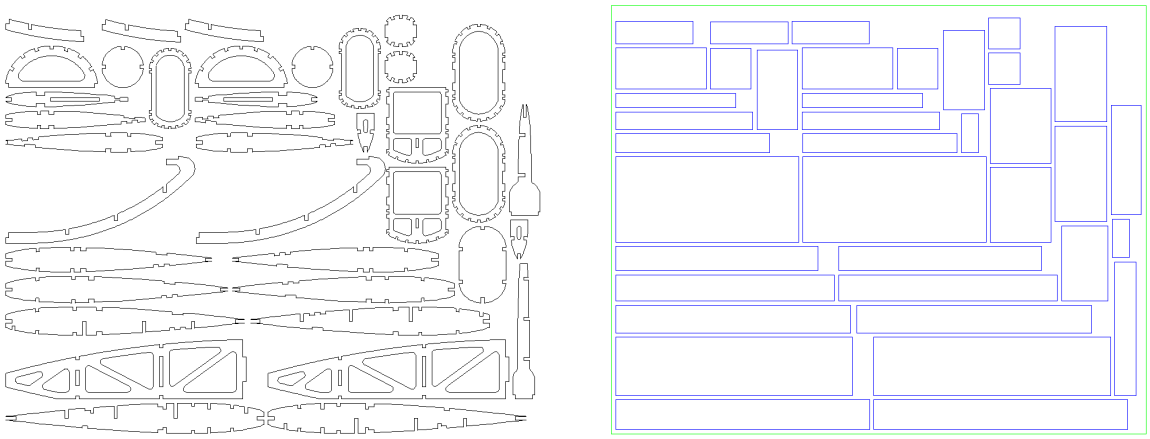
(a) ready-1.dxf (b) ready-1-graph.dxf

Obr. 4.5: Výstupné súbory výkresu `drawing1` (s prekryvom)
Príkaz: `./place-dxf` drawing1.dxf parameters.txt ready-1.dxf ready-1-graph.dxf not-placed.dxf parameters-out-1.txt



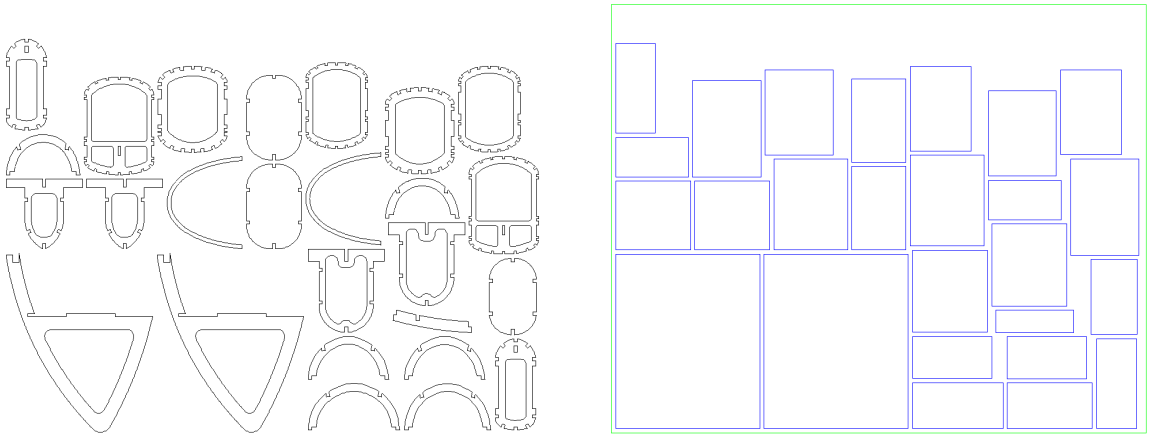
(a) ready-1.dxf (b) ready-1-graph.dxf

Obr. 4.6: Výstupné súbory výkresu `drawing2` (bez prekryvu) – 1. plát
Príkaz: `./place-dxf` drawing2.dxf parameters.txt ready-1.dxf ready-1-graph.dxf not-placed.dxf parameters-out-1.txt



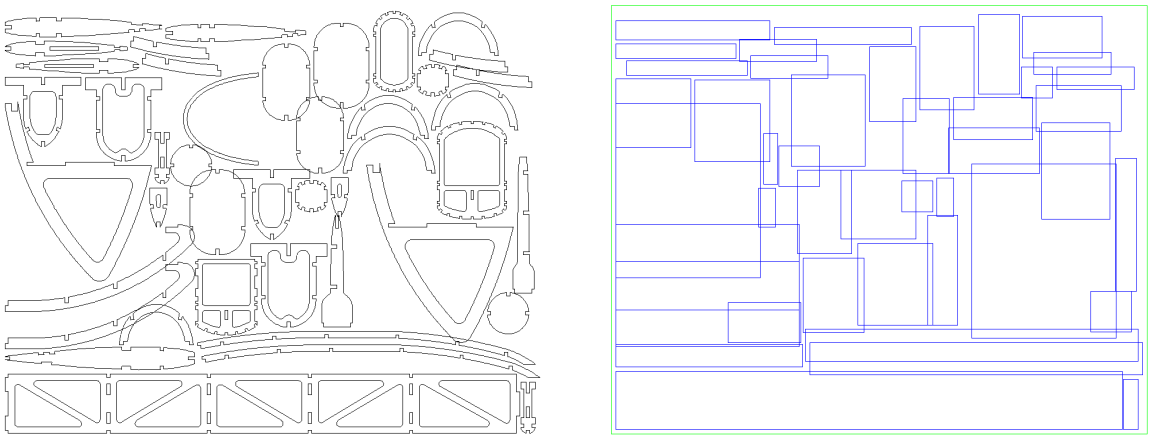
(a) ready-2.dxf (b) ready-2-graph.dxf

Obr. 4.7: Výstupné súbory výkresu `drawing2` (bez prekryvu) – 2. plát
Príkaz: `./place-dxf` not-placed.dxf parameters.txt ready-2.dxf ready-2-graph.dxf not-placed.dxf parameters-out-2.txt



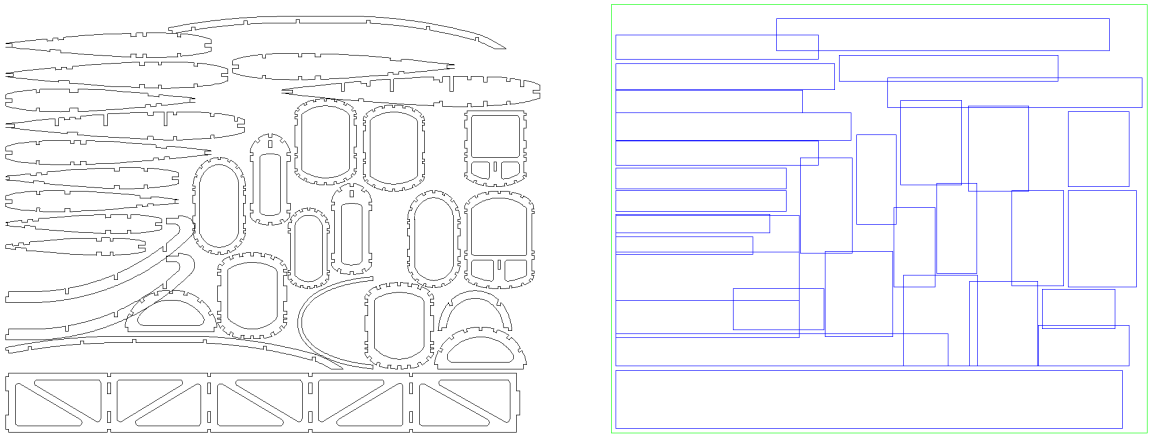
(a) ready-3.dxf (b) ready-3-graph.dxf

Obr. 4.8: Výstupné súbory výkresu `drawing2` (bez prekryvu) – 3. plát
Príkaz: `./place-dxf` not-placed.dxf parameters.txt ready-3.dxf ready-3-graph.dxf not-placed.dxf parameters-out-3.txt



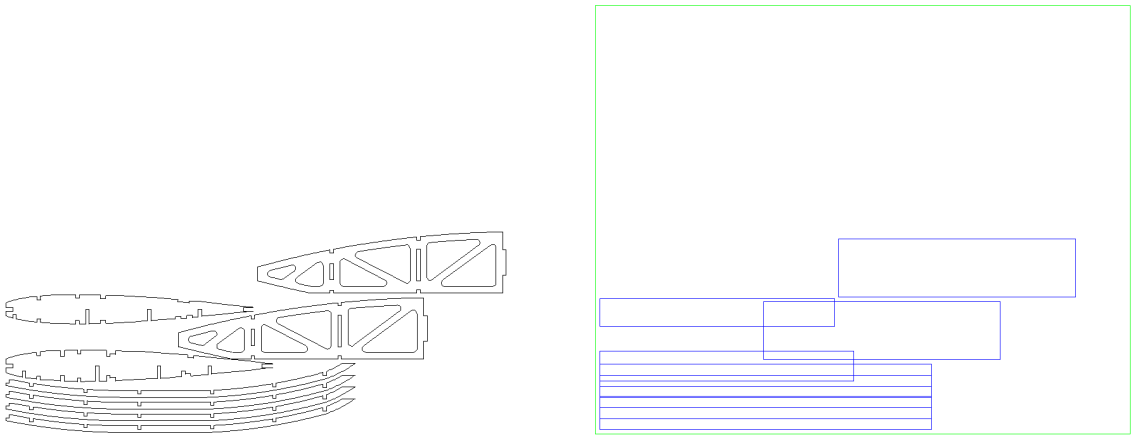
(a) ready-1.dxf (b) ready-1-graph.dxf

Obr. 4.9: Výstupné súbory výkresu `drawing2` (s prekryvom) – 1. plát
Príkaz: `./place-dxf` drawing2.dxf parameters.txt ready-1.dxf ready-1-graph.dxf not-placed.dxf parameters-out-1.txt



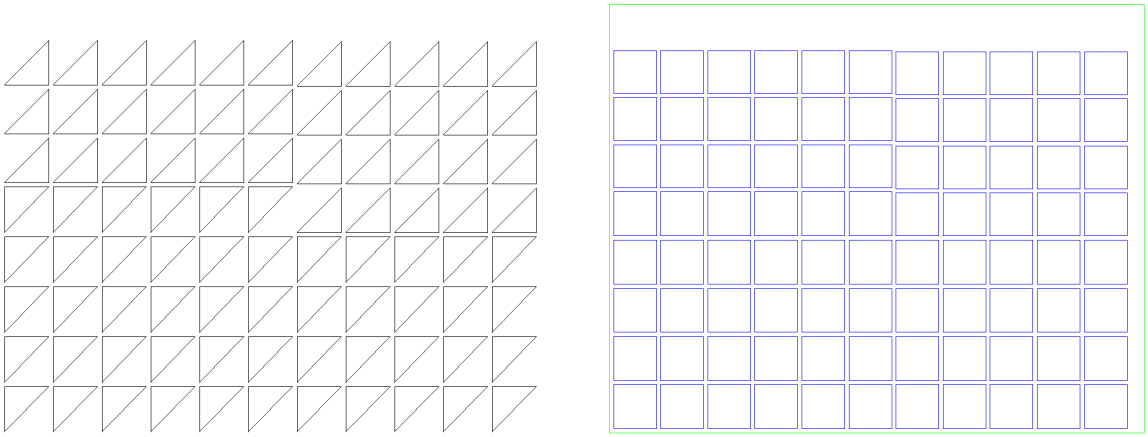
(a) ready-2.dxf (b) ready-2-graph.dxf

Obr. 4.10: Výstupné súbory výkresu `drawing2` (s prekryvom) – 2. plát
Príkaz: `./place-dxf` not-placed.dxf parameters.txt ready-2.dxf ready-2-graph.dxf not-placed.dxf parameters-out-2.txt



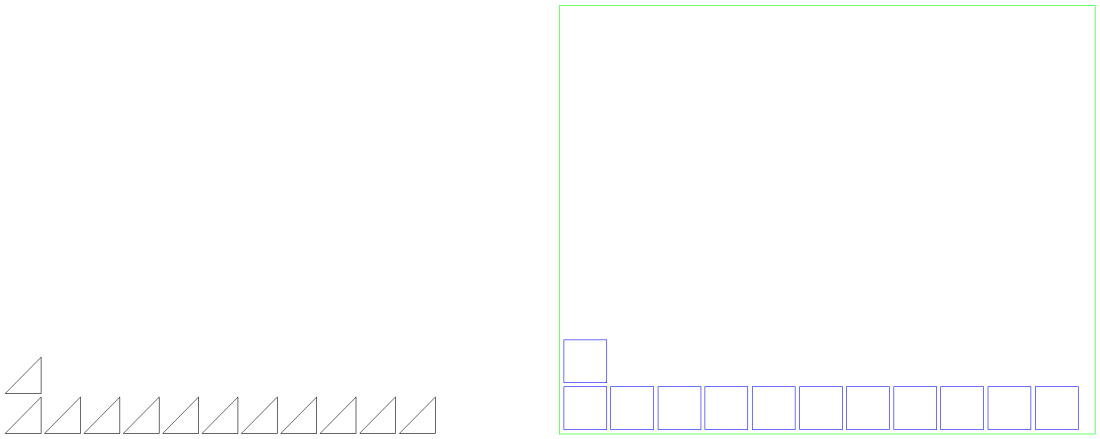
(a) ready-3.dxf (b) ready-3-graph.dxf

Obr. 4.11: Výstupné súbory výkresu `drawing2` (s prekryvom) – 3. plát
Príkaz: `./place-dxf` not-placed.dxf parameters.txt ready-3.dxf ready-3-graph.dxf not-placed.dxf parameters-out-3.txt



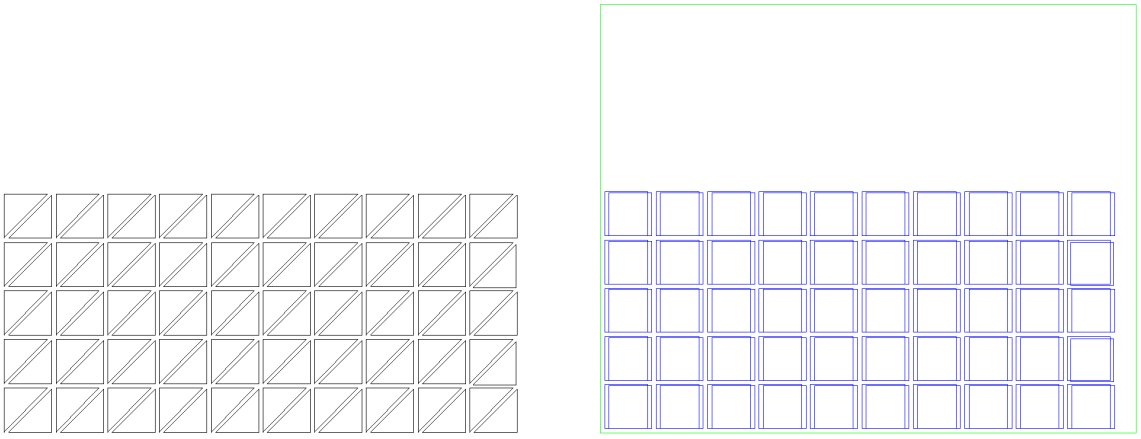
(a) ready-1.dxf (b) ready-1-graph.dxf

Obr. 4.12: Výstupné súbory výkresu `triangles` (bez prekryvu) – 1. plát
Príkaz: `./place-dxf` triangles.dxf parameters.txt ready-1.dxf ready-1-graph.dxf not-placed.dxf parameters-out-1.txt



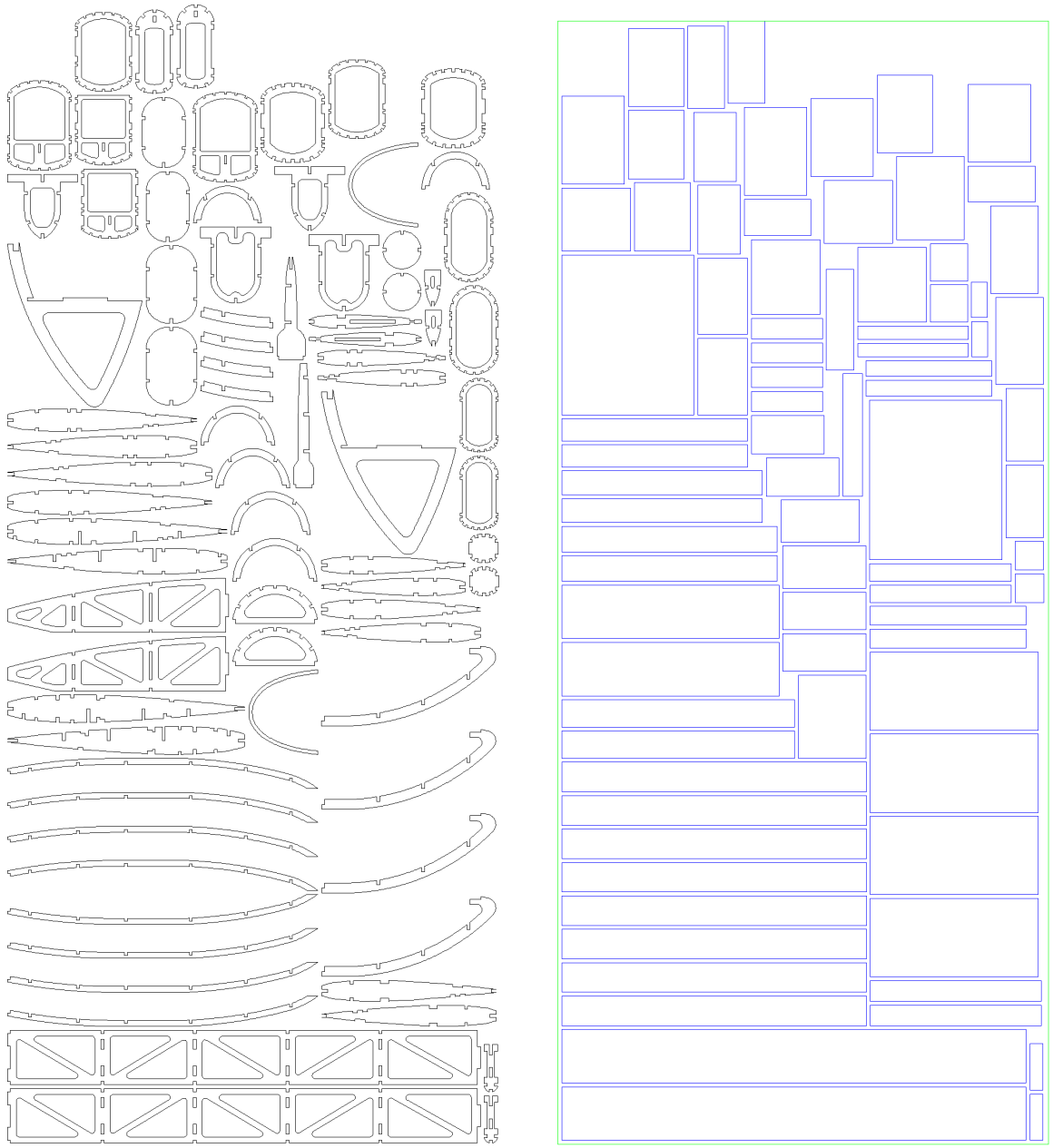
(a) ready-2.dxf (b) ready-2-graph.dxf

Obr. 4.13: Výstupné súbory výkresu `triangles` (bez prekryvu) – 2. plát
Príkaz: `./place-dxf` not-placed.dxf parameters.txt ready-2.dxf ready-2-graph.dxf not-placed.dxf parameters-out-2.txt



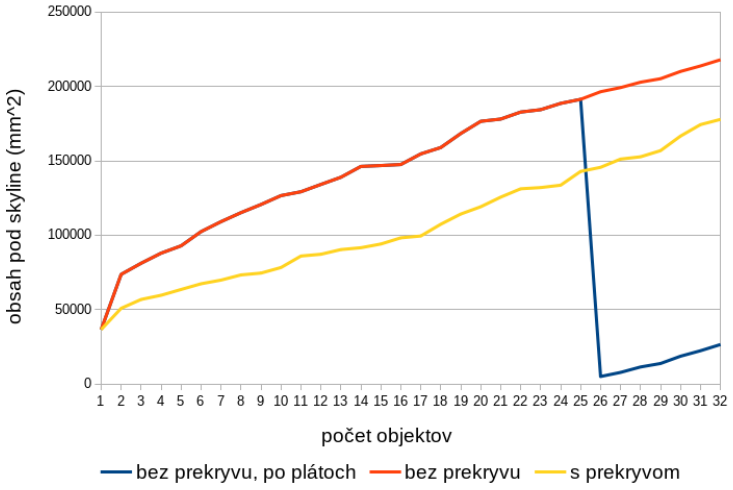
(a) ready-1.dxf (b) ready-1-graph.dxf

Obr. 4.14: Výstupné súbory výkresu `triangles` (s prekryvom)
Príkaz: `./place-dxf` triangles.dxf parameters.txt ready-1.dxf ready-1-graph.dxf not-placed.dxf parameters-out-1.txt

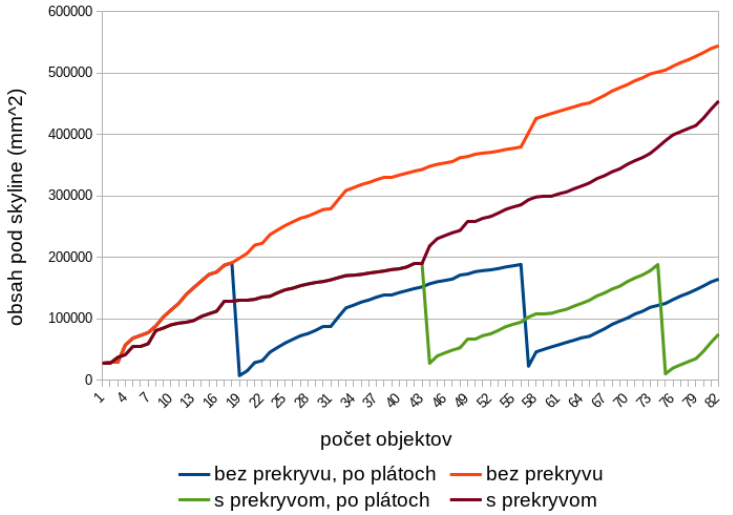


(a) ready-1.dxf (b) ready-1-graph.dxf

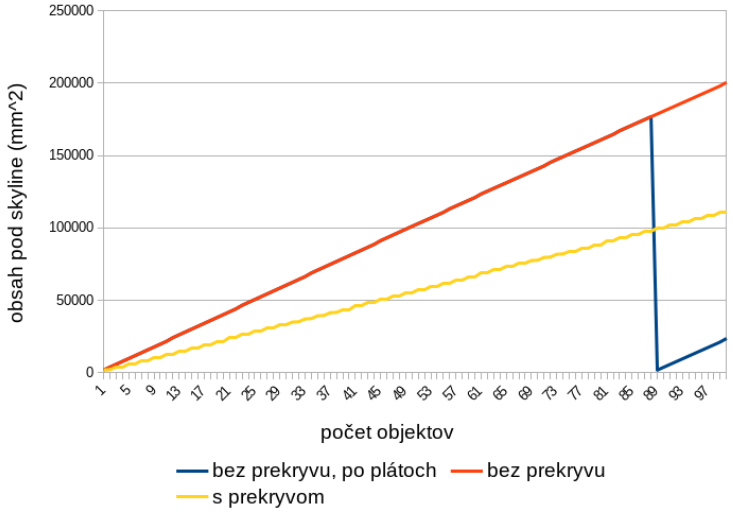
Obr. 4.15: 2D obdĺžnikový Problém vypĺňania pásu
Príkaz: `./place-dxf` drawing2.dxf parameters.txt ready-1.dxf ready-1-graph.dxf not-placed.dxf parameters-out-1.txt



(a) Výkres `drawing1`



(b) Výkres `drawing2`



(c) Výkres `triangles`

Obr. 4.16: Výsledky

tomto prípade zlepšilo riešenie o 16.52 %.

Trojuholníky z výkresu `triangles` boli bez prekryvu rozložené na 2 pláty (obr. 4.16c): (i) 88 trojuholníkov – 176840 mm^2, (ii) 12 trojuholníkov – 23760 mm^2, dokopy 200600 mm^2. S použitím prekryvu sa trojuholníky zmestili na jeden plát s obsahom 110840 mm^2. Teda prekrývanie obdĺžnikov prinieslo zlepšenie o 44,75 %.

Zhrnutie

V tejto kapitole sme experimentálne potvrdili, že modifikovaný *skyline* algoritmus dosahuje niekedy výrazne lepšie výsledky oproti pôvodnému *skyline* algoritmu na vstupoch s nepravidelnými objektmi.

Súčasná implementácia pracuje len s polygónovou reprezentáciou hraníc objektov. Program zatiaľ nepodporuje prácu so zakrivenými segmentami polygónov, avšak doplnenie zakrivených segmentov je lokálny problém, t.j. vyžaduje len zmenu implementácie jednej funkcie.

Záver

V tejto práci sme čitateľa bližšie oboznámili s triedou Problémov vypĺňania, v anglickej literatúre nazývanou `Cutting and Packing Problems`, ktorá zastrešuje množstvo problémov. Predstavili sme všeobecnú štruktúru týchto problémov a následne historický vývoj skúmania tejto témy. Spracovali sme zaužívanú hierarchiu Problémov vypĺňania a zhrnuli kritériá, podľa ktorých bola vytvorená. Potom sme sa pozreli na konkrétne typy odvodených problémov, ktoré vyplynuli z tejto hierarchie, a podrobnejšie sme vyvetlili vzájomné rozdiely. V tejto časti práce často uvádzame aj anglické názvy problémov kvôli lepšej referencii na anglickú literatúru. Prispeli sme aktualizovanou tabuľkou prehľadových prác v tejto oblasti za posledných tridsať rokov a doplnili ju o niektoré relevantné práce z obdobia posledných pätnástich rokov.

Analyzovali sme problém zo strojovej výroby a zaradili ho do hierarchie. Vďaka tomu sa nám podarilo zvoliť algoritmické riešenie, ktorého garancie už niekto dokázal a vyskúšal. Následne sme toto riešenie upravovali podľa špeciálnych požiadaviek, ktoré vyplynuli zo zadania nášho konkrétneho problému. Vytvorili sme tak unikátne algoritmické riešenie primárne 2D nepravidelného Problému vypĺňania nádoby (jedného tvaru) – `2D irregular SBSBPP`. Naše riešenie sme implementovali do aplikácie ovládanej cez príkazový riadok v operačnom systéme Linux. Popísali sme grafický formát DXF, ktorý sa používa na ukladanie vektorových 2D dát. Uviedli sme opis vstupných a výstupných súborov používaných v našom programe. Na pseudokóde sme vysvetlili naše úpravy algoritmu, z ktorého sme vychádzali. Prezentovali sme výsledky vyrobené naším programom a porovnanie pôvodného algoritmu s našou upravenou verziou. Hoci poukazujeme aj na niektoré nedostatky nášho riešenia, pôvodný algoritmus sme zlepšili.

Prirodzeným pokračovaním tejto práce je ďalšie vylepšovanie algoritmu, ktorý sme v tejto práci odprezentovali. Napríklad pri ukladaní súčiastky skúšame iba posuny doľava-dole a dole-doľava. Tento proces hľadania najlepšie pasujúceho tvaru by sa dal rozšíriť o posun doprava-dole a dole-doprava. Pomôcť môže tiež implementácia „vysokej stratégie“ (namiesto nami zvolenej „ľavej stratégie“), kedy sa vybraný najlepšie pasujúci obdĺžnik umiestni do rohu k vyššiemu susedovi najnižšieho segmentu *skyline*, nie k ľavému. Uvažovať sa dá aj o simulovanom žíhaní, tabu-prehľadávaní či iných diametrálne iných heuristikách—tie sú však vnútorne zložitejšie než prístup, ktorý sme zvolili my.

Literatúra

[1] Michio Abe. On the problem to cover simply and without gap the inside of a square with a finite number of squares which are all different from one another. *Nippon Sugaku-Buturigakkwai Kizi Dai 3 Ki*, 14:385 – 387, 1932.

[2] Selen B. Akkaya, Aykut Gül, Zeynep Coşkun, Coşku Karaman, Hande Öztop, and Gizem Mullaoğlu. The distributor's pallet loading problem: A case study. In *The International Symposium for Production Research*, pages 937 – 948, 2018.

[3] M.T. Alonso, R. Alvarez-Valdes, M. Iori, and F. Parreño. Mathematical models for multi container loading problems with practical constraints. *Computers & Industrial Engineering*, 127:722 – 733, 2019.

[4] Inc. Autodesk. Dxf reference. `https://damassets.autodesk.net/content/dam/autodesk/www/developer-network/platform-technologies/autocad-dxf-archive/acad_dxf_2007.pdf`, 2006. Accessed: 2020-05-14.

[5] Erhan Baltacioglu, James T. Moore, and Raymond R. Hill Jr. The distributor's three-dimensional pallet-packing problem: a human intelligence-based heuristic approach. *International Journal of Operational Research*, 1(3):249 – 266, 2006.

[6] Jørgen Bang-Jensen and Rune Larsen. Efficient algorithms for real-life instances of the variable size bin packing problem. *Computers & Operations Research*, 39(11):2848 – 2857, 2012.

[7] Sanjoy Baruah, Marko Bertogna, and Giorgio Buttazzo. *Multiprocessor scheduling for real-time systems*. Springer, 2015.

[8] G. Belov and G. Scheithauer. A branch-and-cut-and-price algorithm for one-dimensional stock cutting and two-dimensional two-stage cutting. *European Journal of Operational Research*, 171(1):85 – 106, 2006.

[9] Christian Blum and Verena Schmid. Solving the 2d bin packing problem by means of a hybrid evolutionary algorithm. *Procedia Computer Science*, 18:899 – 908, 2013. 2013 International Conference on Computational Science.

[10] Andreas Bortfeldt and Gerhard Wäscher. Constraints in container loading – a state-of-the-art review. *European Journal of Operational Research*, 229(1):1 – 20, 2013.

[11] R. L. Brooks, C. A. B. Smith, A. H. Stone, and W. T.Tutte. The dissection of rectangles into squares. *Duke Mathematical Journal*, 7(1):312 – 340, 1940.

[12] E. K. Burke, G. Kendall, and G. Whitwell. A new placement heuristic for the orthogonal stock-cutting problem. *Operations Research*, 52(4):655 – 671, 2004.

[13] Ignacio Castillo, Frank J. Kampas, and János D. Pintér. Solving circle packing problems by global optimization: Numerical results and industrial applications. *European Journal of Operational Research*, 191(3):786 – 802, 2008.

[14] Sara Ceschia and Andrea Schaerf. Local search for a multi-drop multi-container loading problem. *Journal of Heuristics*, 19(2):275 – 294, 2013.

[15] F. Chauny, R. Loulou, S. Sadones, and F. Soumis. A two-phase heuristic for strip packing: Algorithm and probabilistic analysis. *Operations research letters*, 6(1):25 – 33, 1987.

[16] Chandra Chekuri and Sanjeev Khanna. A polynomial time approximation scheme for the multiple knapsack problem. *SIAM Journal on Computing*, 35(3):713 – 728, 2005.

[17] M. Chen, K. Li, D. Zhang, L. Zheng, and X. Fu. Hierarchical search-embedded hybrid heuristic algorithm for two-dimensional strip packing problem. *IEEE Access*, 7:179086 – 179103, 2019.

[18] Luiz H. Cherri, Leandro R. Mundim, Marina Andretta, Franklina M.B. Toledo, José F. Oliveira, and Maria Antónia Carravilla. Robust mixed-integer linear programming models for the irregular strip packing problem. *European Journal of Operational Research*, 253(3):570 – 583, 2016.

[19] François Clautiaux, Ruslan Sadykov, François Vanderbeck, and Quentin Viaud. Combining dynamic programming with filtering to solve a four-stage two-dimensional guillotine-cut bounded knapsack problem. *Discrete Optimization*, 29:18 – 44, 2018.

[20] Jean Constant. Sacred mathematics: Japanese temple geometry by hidetoshi fukagawa and tony rothman; foreword by freeman dyson. *Mathematical Intelligencer*, 39(4):83 – 85, 2017.

[21] M. Helena Correia, José F. Oliveira, and J. Soeiro Oliveira. Cylinder packing by simulated annealing. *Pesquisa Operacional*, 20:269 – 286, 2000.

[22] Jean-François Côté, Mauro Dell'Amico, and Manuel IoriS. Combinatorial benders' cuts for the strip packing problem. *Operations Research*, 62(3):643 – 661, 2014.

[23] Teodor G. Crainic, Guido Perboli, and Roberto Tadei. Ts2pack: A two-level tabu search for the three-dimensional bin packing problem. *European Journal of Operational Research*, 195(3):744 – 760, 2009.

[24] Y. Cui, Gu Tianlong, and W. Hu. An algorithm for the constrained two-dimensional rectangular multiple identical large object placement problem. *Optimization Methods & Software*, 23:375 – 393, 2008.

[25] Yaodong Cui and Yuli Yang. A heuristic for the one-dimensional cutting stock problem with usable leftover. *European Journal of Operational Research*, 204(2):245 – 250, 2010.

[26] Everton Fernandes da Silva, Túlio AM Toffolo, Greet V. Berghe, and Tony Wauters. A study of exact methods for container loading problems. In *ORBEL, Date: 2017/02/02-2017/02/03, Location: Brussels*, 2017.

[27] Max W. Dehn. Über zerlegung von rechtecken in rechtecke. *Mathematische Annalen*, 57(3):314 – 332, 1903.

[28] Tansel Dokeroglu and Ahmet Cosar. Optimization of one-dimensional bin packing problem with island parallel grouping genetic algorithms. *Computers & Industrial Engineering*, 75:176 – 186, 2014.

[29] Kathryn A. Dowsland. An exact algorithm for the pallet loading problem. *European Journal of Operational Research*, 31(1):78 – 84, 1987.

[30] Kathryn A. Dowsland and William B. Dowsland. Solution approaches to irregular nesting problems. *European Journal of Operational Research*, 84(3):506 – 521, 1995.

[31] Harald Dyckhoff. A new linear programming approach to the cutting stock problem. *Operations Research*, 29(6):1092 – 1104, 1981.

[32] Harald Dyckhoff. A typology of cutting and packing problems. *European Journal of Operational Research*, 44(2):145 – 159, 1990.

[33] Harald Dyckhoff and Ute Finke. *Cutting and Packing in Production and Distribution*. Physica-Verlag Heidelberg, 1992.

[34] Harald Dyckhoff, Guntram Scheithauer, and Johannes Terno. Cutting and packing. *Dimensions*, 13:94 – 120, 1997.

[35] Michael Eley. A bottleneck assignment approach to the multiple container loading problem. *OR Spectrum*, 25(1):45 – 60, 2003.

[36] Ahmed Elkeran. A new approach for sheet nesting problem using guided cuckoo search and pairwise clustering. *European Journal of Operational Research*, 231(3):757 – 769, 2013.

[37] Gürdal Ertek and Kemal Kilic. Decision support for packing in warehouses. In *International Symposium on Computer and Information Sciences*, pages 115 – 124, 2006.

[38] Yanhong Feng, Gai-Ge Wang, Suash Deb, Mei Lu, and Xiang-Jun Zhao. Solving 0–1 knapsack problem by a novel binary monarch butterfly optimization. *Neural computing and applications*, 28(7):1619 – 1634, 2017.

[39] The free encyclopedia Wikipedia. Sangaku at Enmanji. `https://upload.wikimedia.org/wikipedia/commons/5/5f/Sangaku_at_Enmanji.jpg`, 2010. [Online; accessed 24 Apr, 2020], changed brightness and contrast.

[40] Fabio Furini and Enrico Malaguti. Models for the two-dimensional two-stage cutting stock problem with multiple stock size. *Computers & Operations Research*, 40(8):1953 – 1962, 2013.

[41] Fabio Furini, Enrico Malaguti, Rosa M. Durán, Alfredo Persiani, and Paolo Toth. A column generation heuristic for the two-dimensional two-staged guillotine cutting stock problem with multiple stock size. *European Journal of Operational Research*, 218(1):251 – 260, 2012.

[42] John A. George. A method for solving container packing for a single size of box. *Journal of the Operational Research Society*, 43(4):307 – 312, 1992.

[43] John A. George and David F. Robinson. A heuristic for packing boxes into a container. *Computers & Operations Research*, 7(3):147 – 156, 1980.

[44] Paul C. Gilmore and Ralph E. Gomory. A linear programming approach to the cutting-stock problem. *Operations Research*, 9(6):849 – 859, 1961.

[45] Paul C. Gilmore and Ralph E. Gomory. Multistage cutting stock problems of two and more dimensions. *Operations Research*, 13(1):94 – 120, 1965.

[46] Paul C. Gilmore and Ralph E. Gomory. The theory and computation of knapsack functions. *Operations Research*, 14(6):1045 – 1074, 1966.

[47] Juan C. Gomez and Hugo Terashima-Marín. Evolutionary hyper-heuristics for tackling bi-objective 2d bin packing problems. *Genetic Programming and Evolvable Machines*, 19(1 - 2):151 – 181, 2018.

[48] José F. Gonçalves and Mauricio G.C. Resende. A parallel multi-population biased random-key genetic algorithm for a container loading problem. *Computers & Operations Research*, 39(2):179 – 190, 2012.

[49] José Fernando Gonçalves and Mauricio G.C. Resende. A biased random key genetic algorithm for 2d and 3d bin packing problems. *International Journal of Production Economics*, 145(2):500 – 510, 2013.

[50] Roger B. Grinde and Tom M. Cavalier. A new algorithm for the minimal-area convex enclosure problem. *European Journal of Operational Research*, 84(3):522 – 538, 1995. Cutting and Packing.

[51] Murray J. Haims and Herbert Freeman. A multistage solution of the template-layout problem. *IEEE Transactions on Systems Science and Cybernetics*, 6(2):145 – 151, 1970.

[52] Thomas C. Hales. A proof of the Kepler conjecture. *Annals of Mathematics*, 162(3):1065 – 1185, 2005.

[53] David Hilbert. Mathematische probleme. *Nachrichten von der Gesellschaft der Wissenschaften zu Göttingen*, pages 253 – 297, 1900.

[54] Thom J. Hodgson. A combined approach to the pallet loading problem. *IIE Transactions*, 14(3):175 – 182, 1982.

[55] Robert Hooke, James Allestry, and John Martyn. *Micrographia, or, Some physiological descriptions of minute bodies made by magnifying glasses :with observations and inquiries thereupon /*. London :Printed by Jo. Martyn and Ja. Allestry, printers to the Royal Society ..., 1665.

[56] Wenqi Huang and Kun He. A caving degree approach for the single container loading problem. *European Journal of Operational Research*, 196(1):93 – 101, 2009.

[57] Shinji Imahori and Mutsunori Yagiura. The best-fit heuristic for the rectangular strip packing problem: An efficient implementation and the worst-case approximation ratio. *Computers & Operations Research*, 37(2):325 – 333, 2010.

[58] Klaus Jansen and Stefan E.J. Kraft. A faster fptas for the unbounded knapsack problem. *European Journal of Combinatorics*, 68:148 – 174, 2018.

[59] Zhang X. X. Jiangping. Evaluating spatial straightness error by approaching minimum enclosure cylinder [j]. *Journal of Huazhong University of Science and Technology (Natural Science Edition)*, 12, 2011.

[60] David S. Johnson. Fast algorithms for bin packing. *Journal of Computer and System Sciences*, 8(3):272 – 314, 1974.

[61] Leonardo Junqueira and Reinaldo Morabito. On solving three-dimensional open-dimension rectangular packing problems. *Engineering Optimization*, 49(5):733 – 745, 2017.

[62] Leonardo Junqueira, Reinaldo Morabito, Denise S. Yamashita, and Horacio H. Yanasse. Optimization models for the three-dimensional container loading problem with practical constraints. In *Modeling and Optimization in Space Engineering*, pages 271 – 293. 2012.

[63] Julia Kallrath, Steffen Rebennack, Josef Kallrath, and Rüdiger Kusche. Solving real-world cutting stock-problems in the paper industry: Mathematical approaches, experience and challenges. *European Journal of Operational Research*, 238(1):374 – 389, 2014.

[64] E. Kanniga, Sri. M. Srikanth, and Mahalingam Sundhararajan. Optimization solution of equal dimension boxes in container loading problem using a permutation block algorithm. *Indian journal of science and technology*, 7:22 – 26, 2014.

[65] Leonid V. Kantorovich. Mathematical methods of organizing and planning production. *Management Science*, 6(4):366 – 422, 1960. This is the English translation of the famous 1939 article by L. V. Kantorovich, originally published in Russian.

[66] Sarsij Kaystha and Suneeta Agarwal. Notice of retraction: Greedy genetic algorithm to bounded knapsack problem. In *2010 3rd International Conference on Computer Science and Information Technology*, volume 6, pages 301 – 305, 2010.

[67] Johannes Kepler. Strena seu de nive sexangula (the six-cornered snowflake). `http://www.thelatinlibrary.com/kepler/strena.html`, 1611. Accessed: 2020-01-17.

[68] D. J. Kleitman and M. M. Krieger. An optimal bound for two dimensional bin packing. In *16th Annual Symposium on Foundations of Computer Science (sfcs 1975)*, pages 163 – 168, 1975.

[69] Ravi Kothari and Diptesh Ghosh. Tabu search for the single row facility layout problem using exhaustive 2-opt and insertion neighborhoods. *European Journal of Operational Research*, 224(1):93 – 100, 2013.

[70] Ravi Kothari and Diptesh Ghosh. A scatter search algorithm for the single row facility layout problem. *Journal of Heuristics*, 20(2):125 – 142, 2014.

[71] Martine Labbé, Gilbert Laporte, and Silvano Martello. Upper bounds and algorithms for the maximum cardinality bin packing problem. *European Journal of Operational Research*, 149:490 – 498, 09 2003.

[72] H.C.W. Lau, T.M. Chan, W.T. Tsui, G.T.S. Ho, and K.L. Choy. An ai approach for optimizing multi-pallet loading operations. *Expert Systems with Applications*, 36(3, Part 1):4296 – 4312, 2009.

[73] Stephen C.H. Leung, Defu Zhang, and Kwang Mong Sim. A two-stage intelligent search algorithm for the two-dimensional strip packing problem. *European Journal of Operational Research*, 215(1):57 – 69, 2011.

[74] Greg Levin, Shelby Funk, Caitlin Sadowski, Ian Pye, and Scott Brandt. Dp-fair: A simple model for understanding optimal multiprocessor scheduling. In *2010 22nd Euromicro Conference on Real-Time Systems*, pages 3 – 13. IEEE, 2010.

[75] LibreCAD. LibreCAD Open Source 2D-CAD. `https://librecad.org/#`, 2020. Accessed: 2020-05-22.

[76] Kok-Hua Loh, Bruce Golden, and Edward Wasil. Solving the one-dimensional bin packing problem with a weight annealing heuristic. *Computers & Operations Research*, 35(7):2283 – 2291, 2008. Part Special Issue: Includes selected papers presented at the ECCO'04 European Conference on combinatorial Optimization.

[77] Kok-Hua Loh, Bruce Golden, and Edward Wasil. Solving the maximum cardinality bin packing problem with a weight annealing-based algorithm. In *Operations Research and Cyber-Infrastructure*, pages 147 – 164. Springer, 2009.

[78] Rita Macedo, Cláudio Alves, and J.M. Valério de Carvalho. Arc-flow model for the two-dimensional guillotine cutting stock problem. *Computers & Operations Research*, 37(6):991 – 1001, 2010.

[79] Silvano Martello and Michele Monaci. Algorithmic approaches to the multiple knapsack assignment problem. *Omega*, 90:102004, 2020.

[80] Silvano Martello, David Pisinger, and Paolo Toth. New trends in exact algorithms for the 0–1 knapsack problem. *European Journal of Operational Research*, 123(2):325 – 332, 2000.

[81] Laura A. McLay and Sheldon H. Jacobson. Algorithms for the bounded set-up knapsack problem. *Discrete Optimization*, 4(2):206 – 212, 2007.

[82] Ana Moura and José F. Oliveira. An integrated approach to the vehicle routing and container loading problems. *OR spectrum*, 31(4):775 – 800, 2009.

[83] Scientific Figure on ResearchGate. Application of tessellation in architectural geometry design. `https://www.researchgate.net/figure/Parts-of-M-C-Escher-Pictures-and-Its-Tessellation_fig1_325559370`. [Online; accessed 24 Apr, 2020].

[84] Frank G. Ortmann, Nthabiseng Ntene, and Jan H. van Vuuren. New and improved level heuristics for the rectangular strip packing and variable-sized bin packing problems. *European Journal of Operational Research*, 203(2):306 – 315, 2010.

[85] Marc Peeters and Zeger Degraeve. Branch-and-price algorithms for the dual bin packing and maximum cardinality bin packing problem. *European Journal of Operational Research*, 170(2):416 – 439, 2006.

[86] David Pisinger. An exact algorithm for large multiple knapsack problems. *European Journal of Operational Research*, 114(3):528 – 541, 1999.

[87] David Pisinger and Mikkel Sigurd. The two-dimensional bin packing problem with variable bin sizes and costs. *Discrete Optimization*, 2(2):154 – 167, 2005.

[88] Vincent Poirriez, Nicola Yanev, and Rumen Andonov. A hybrid algorithm for the unbounded knapsack problem. *Discrete Optimization*, 6(1):110 – 124, 2009.

[89] Kelly C. Poldi and Marcos N. Arenales. Heuristics for the one-dimensional cutting stock problem with limited multiple stock lengths. *Computers & Operations Research*, 36(6):2074 – 2081, 2009.

[90] Jakob Puchinger and Günther R. Raidl. Models and algorithms for three-stage two-dimensional bin packing. *European Journal of Operational Research*, 183(3):1304 – 1327, 2007.

[91] Vitória Pureza and Reinaldo Morabito. Some experiments with a simple tabu search algorithm for the manufacturer's pallet loading problem. *Computers & Operations Research*, 33(3):804 – 819, 2006.

[92] Günther R. Raidl and Gabriele Kodydek. Genetic algorithms for the multiple container packing problem. In *International Conference on Parallel Problem Solving from Nature*, pages 875 – 884. Springer, 1998.

[93] Glaydston M. Ribeiro and Luiz A. N. Lorena. Lagrangean relaxation with clusters and column generation for the manufacturer's pallet loading problem. *Computers & Operations Research*, 34(9):2695 – 2708, 2007.

[94] Alice Ryhl. Find x value on a line segment with given y. `https://math.stackexchange.com/questions/2297518/find-x-value-on-a-line-segment-with-given-y`, 2017. Accessed: 2020-05-18.

[95] Sartaj Sahni. Approximate algorithms for the 0/1 knapsack problem. *Journal of the ACM (JACM)*, 22(1):115 – 124, 1975.

[96] Guntram Scheithauer. A note on handling residual lengths. *Optimization*, 22(3):461 – 466, 1991.

[97] Guntram Scheithauer. Lp-bounds for the container and multi-container loading problem. In *Operations Research Proceedings 1996*, pages 123 – 128. Springer, 1997.

[98] Guntram Scheithauer and Johannes Terno. A heuristic approach for solving the multi-pallet packing problem. *Dresden university*, 1996.

[99] Elsa Silva, Filipe Alvelos, and J.M. Valério de Carvalho. An integer programming model for two- and three-stage two-dimensional cutting stock problems. *European Journal of Operational Research*, 205(3):699 – 708, 2010.

[100] Elsa Silva, José F. Oliveira, and Gerhard Wäscher. 2dcpackgen: A problem generator for two-dimensional rectangular cutting and packing problems. *European Journal of Operational Research*, 237(3):846 – 856, 2014.

[101] Elsa Silva, José F. Oliveira, and Gerhard Wäscher. The pallet loading problem: a review of solution methods and computational experiments. *International Transactions in Operational Research*, 23(1 - 2):147 – 172, 2016.

[102] Tina Sørensen, Søren Foged, Jeppe M. Gravers, Mukund N. Janardhanan, and Peter Nielsen. Input analysis of the distributor's pallet loading problem. In *Distributed Computing and Artificial Intelligence, 13th International Conference*, pages 545 – 554, 2016.

[103] Chanin Srisuwannapa and Peerayuth Charnsethikul. An exact algorithm for the unbounded knapsack problem with minimizing maximum processing time, 2007.

[104] H. S. Stone. Multiprocessor scheduling with the aid of network flow algorithms. *IEEE Transactions on Software Engineering*, SE-3(1):85 – 93, 1977.

[105] Paolo Toth and Silvano Martello. *Knapsack problems: Algorithms and computer implementations*. Wiley, 1990.

[106] Jung-Fa Tsai, Pei-Chun Wang, and Ming-Hua Lin. A global optimization approach for solving three-dimensional open dimension rectangular packing problems. *Optimization*, 64(12):2601 – 2618, 2015.

[107] Lijun Wei, Wee-Chong Oon, Wenbin Zhu, and Andrew Lim. A goal-driven approach to the 2d bin packing and variable-sized bin packing problems. *European Journal of Operational Research*, 224(1):110 – 121, 2013.

[108] Gerhard Wäscher, Heike Haußner, and Holger Schumann. An improved typology of cutting and packing problems. *European Journal of Operational Research*, 183(3):1109 – 1130, 2007.

[109] Horacio H. Yanasse, Alan S. I. Zinober, and Reginald G. Harris. Two-dimensional cutting stock with multiple stock sizes. *Journal of the Operational Research Society*, 42(8):673 – 683, 1991.

[110] Xiaoge Zhang, Shiyan Huang, Yong Hu, Yajuan Zhang, Sankaran Mahadevan, and Yong Deng. Solving 0-1 knapsack problems based on amoeboid organism algorithm. *Applied Mathematics and Computation*, 219(19):9959 – 9970, 2013.

[111] Wenbin Zhu, Weili Huang, and Andrew Lim. A prototype column generation strategy for the multiple container loading problem. *European Journal of Operational Research*, 223(1):27 – 39, 2012.

[112] Dexuan Zou, Liqun Gao, Steven Li, and Jianhua Wu. Solving 0–1 knapsack problem by a novel global harmony search algorithm. *Applied Soft Computing*, 11(2):1556 – 1564, 2011.

Príloha A: obsah elektronickej prílohy

V elektronickej prílohe priloženej k práci sa nachádza zdrojový kód programu a dáta na testovanie. Organizácia priečinku:

```
├── data
│   ├── parameters.txt
│   ├── drawing1.dxf
│   ├── drawing2.dxf
│   └── triangles.dxf
├── src
│   ├── <zdrojový kód>
│   └── Makefile
├── clean.sh
├── run-full.sh
├── run-test.sh
└── README.txt
```

Bližšie informácie o obsluhe programu sa nachádzajú v súbore `README.txt`. Potrebné informácie o skriptoch sú v tomto súbore, tak isto ako aj priamo v skriptoch.