VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
PROGRAMŲ SISTEMŲ STUDIJŲ PROGRAMA

Skruzdžių kolonijos algoritmas jūrinių konteinerių krovos optimizavimui

Ant Colony Algorithm for the Marine Container Loading Optimization

Magistro baigiamasis darbas

Atliko:	Algimantas Skuodis	(parašas)
Darbo vadovas:	Doc. dr. Algirdas Lančinskas	(parašas)
Recenzentas:	Prof. dr. Olga Kurasova	(parašas)

Vilnius – 2022

TURINYS

1. Įvadas

Konteinerių krovos uždavinys (angl. Container Loading Problem, CLP) yra kombinatorinio optimizavimo (angl. Combinatorial Optimization, CO) tipo uždavinys, kuriuo siekiama vienos fiksuoto dydžio stačiakampės gretasienės dėžės tūrį maksimaliai užpildyti mažesnėmis, skirtingo dydžio, stačiakampėmis gretasienėmis dėžėmis. Uždavinys priklauso pjaustymo ir pakavimo (angl. Cutting and Packing, CP) uždavinių klasei. Išsami pjaustymo ir pakavimo uždavinių tipologija pateikiama [Dyc90]. Pagal [Pis02], priklausomai nuo tikslo funkcijos ir galimų apribojimų, yra žinoma keletas CLP uždavinio variantų:

- Juostos pakavimo (angl. strip packing, STRIPPACK). Kuomet konteineris yra fiksuoto pločio ir aukščio, bet turi neribotą gylį. Tikslas yra užpildyti konteinerį taip, jog konteinerio gylis būtų minimalus.
- Kuprinės pakrovimo (angl. knapsack loading, KNAPSACK). Kuomet kiekviena pakraunama dėžė turi vertę. Tikslas yra užpildyti konteinerį taip, jog konteinerio vertė būtų didžiausia.
- Keleto vienodo dydžio konteinerių (angl. bin-packing, BINPACK). Kuomet yra keletas vienodo dydžio konteinerių. Tikslas yra užpildyti konteinerius taip, jog būtų sunaudota kuo mažiau konteinerių.
- Keleto skirtingo dydžio konteinerių (angl. multi-container loading, MULTICONT). Kuomet yra keletas skirtingo dydžio konteinerių. Tikslas yra užpildyti konteinerius taip, kad jų gabenimo kaštai būtų mažiausi.

Įvairūs šių uždavinių variantai labai dažnai pasitaiko pramonėje ir logistikoje.

Viena iš daugelio konteinerių užpildymo uždavinio pritaikymo sričių yra jūrinių konteinerių krova. Konteineriuose pervežamų krovinių apimtys auga Lietuvoje ir visame pasaulyje. Klaipėdos valstybinio jūrų uosto direkcijos duomenimis 2018-ais metais Klaipėdos uoste krovinių konteineriuose perkrauta 50 procentų daugiau nei 2017-ais metais [DIR19]. Pasauliniu mastu, 2018-ais metais uostuose perkraunamų konteinerių kiekis padidėjo beveik 5 procentais [UNC19].

Augant konteinerių krovos apimtims kartu auga greito ir efektyvaus konteinerių krovos planavimo poreikis. Kraunant jūrinius konteinerius yra siekiama kuo efektyviau išnaudoti konteinerių tūrį, t.y. konteinerį užpildyti maksimaliai atsižvelgiant į galimus apribojimus, tokius kaip maksimalus galimas konteinerio svoris ir tūris.

Tačiau optimalus konteinerio užpildymas nėra trivialus uždavinys, todėl efektyvus jo sprendimas reikalauja specifinių sprendimo metodų. Praktikoje šie uždaviniai aprašomi kaip kombinatorinio optimizavimo uždaviniai, kuriuose reikia parinkti geriausią krovinių krovimo į konteinerį eiliškumą atsižvelgiant į turimus ribojimus.

Kombinatorinio optimizavimo uždavinių sprendimo metodai skirstomi į tikslius ir apytikslius. Naudojant apytikslius metodus atsisakoma optimalaus sprendinio radimo garantijos, bet taupoma laiko priimant apytikslį sprendinį su toleruotina paklaida. Apytiksliai metodai dar yra klasifikuojami į *euristinius* (angl. heuristic) ir *metaeuristinius* (ang. metaheuristic). Euristiniai algoritmai dažniausiai taikomi kai nėra žinomas tradicinis algoritmas optimalaus sprendinio radimui,

tad tenkinamasi apytiksliu sprendiniu. Pagal [BR03, p 271] pateiktą apibendrinimą, metaeuristiniai algoritmai yra aukštesnio lygio strategijos, kurios, naudodamos skirtingus metodus, valdo optimalaus sprendimo paieškos procesą. Pavyzdžiui, šiai algoritmų klasei priklauso tokie algoritmai kaip dalelių spiečiaus optimizavimo (angl. Particle Swarm Optimization, PSO), modeliuojamo atkaitinimo (angl. Simulated Annealing, SA), tabu paieškos (ang. Tabu Search, TS) ir genetinis algoritmas (angl. Genetic Algorithm, GA).

Vienas įstabesnių gamtoje pastebimų reiškinių yra skruzdžių kolonijų gebėjimas sudaryti trumpiausius kelius tarp kolonijos ir maisto šaltinių. Ieškodamos maisto skruzdės koordinuoja savo veiksmus naudodamos *stigmergiją* t.y., netiesioginį bendravimo būdą, kuris yra paremtas aplinkos keitimu [DS04, p 1]. Trumpiausio kelio paieškos atveju, jos koordinuoja veiksmus išskirdamos ir pajusdamos tam tikrus cheminius junginius - feromonus. Ant kelio paliktas feromonų kiekis tiesiogiai proporcingas tikimybei, jog ir kitos skruzdės rinksis šį kelią. Naudojant šitą elgsenos modelį buvo atliekami tyrimai [DS04, p 22] ir kuriami *metaeuristiniai* algoritmai, skirti įvairiems optimizavimo uždaviniams spręsti [DS04, p 39]. Bendrai, šie algoritmai vadinami skruzdžių kolonijų optimizavimo algoritmais (angl. Ant Colony Optimization, ACO) ir yra taikomi įvairioms optimizavimo uždavinių klasėms: kelio parinkimo (angl. routing), paskirstymo (angl. assignment), planavimo (angl. scheduling) ir, kaip kad CLP uždavinio atveju, poaibio (angl. subset) parinkimo.

Šiame darbe skruzdžių kolonijos algoritmas bus pritaikytas jūrinių konteinerių krovos optimizavimo uždaviniams spręsti. Algoritmo efektyvumas bus vertinamas skaičiuojant pakrauto konteinerio užpildymą. Taikant randomizuotą feromonų kiekio ribojimą bei elitinės skruzdėlės strategiją, bus siekiama mažinti skruzdžių kelio stagnaciją bei didinti skirtingų kelių pasirinkimo tikimybę, taip pagerinant algoritmo efektyvumą.

1.1. Tyrimo objektas

Šio darbo tyrimo objektas yra skruzdžių kolonijos algoritmas jūrinių konteinerių krovos optimizavimui.

1.2. Darbo tikslas

Sudaryti ir ištirti skruzdžių kolonijos algoritmą efektyviam jūrinių konteinerių krovos optimizavimo uždavinio sprendimui su randomizuotu feromonų kiekio ribojimu bei elitine strategija.

1.3. Uždaviniai

Norint pasiekti aukščiau nurodytą tikslą, darbe iškelti sekantys uždaviniai:

- Ištirti esamas skruzdžių kolonijos algoritmo taikymo strategijas ir parinkti tinkamas jūrinių konteinerių krovos optimizavimo uždavinio sprendimui.
- Sudaryti skruzdžių kolonijos algoritmą jūrinių konteinerių krovos optimizavimo uždavinio sprendimui.

- Pasiūlyti galimas algoritmo modifikacijas.
- Ištirti sudaryto algoritmo efektyvumą.
- Palyginti algoritmo efektyvumą su kitais algoritmais.

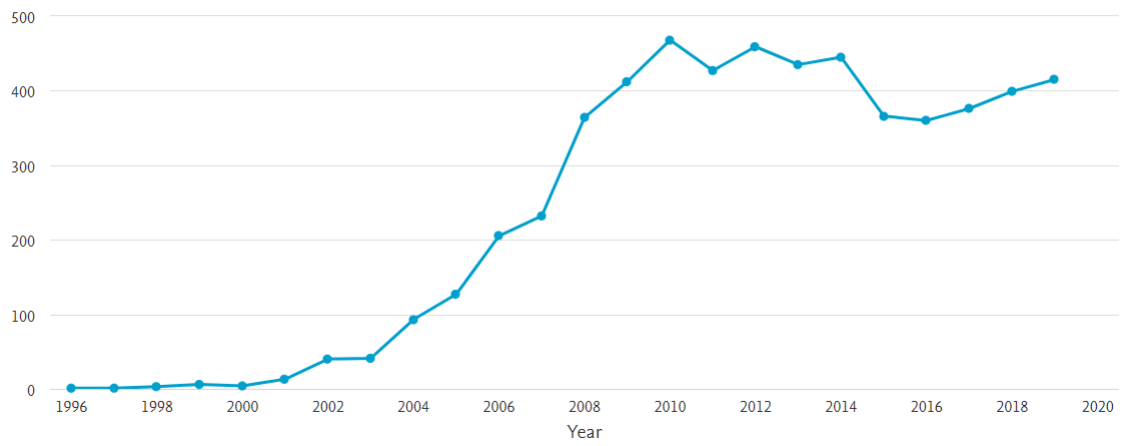
1.4. Laukiami rezultatai

- Sudarytas skruzdžių kolonijos algoritmas jūrinių konteinerių krovos optimizavimo uždavinių sprendimui su randomizuotu feromonų kiekio ribojimu bei elitine strategija.

2. Literatūros šaltinių apžvalga

[DS19] pateikia ACO algoritmų apžvalgą ir vystymo kryptis bei tendencijas. [DS19, p 331] pateikiami „Scopus" duomenų bazėje atliktos paieškos rezultatai. Buvo ieškoma straipsnių, kurių antraštėse vartojami „ant system", „ant colony system" arba „ant colony optimization" terminai, laikotarpis 1996-2016 metai. Pagal [DS19, p 331] atliktos paieškos rezultatus ir pateiktus duomenis, nuo 1996 metų kai buvo publikuotas pirmasis skruzdžių paieškos (angl. Ant search, AS) algoritmas, [DMC96a] tokių straipsnių kiekis stipriai augo iki 2010-ų metų ir, nuo to laiko, publikacijų kiekis išliko apie 400 straipsnių per metus.

Siekiant patikrinti [DS19, p 331] rezultatus ir sužinoti publikuojamų straipsnių kiekius nuo 2016-ų metų, buvo atlikta tokia pat paieška „Scopus" duomenų bazėje. Kaip matyti 4 grafike, publikuojamų straipsnių kiekis iš tikro pasiekė piką 2010-ais metais ir laikėsi apie 400 straipsnių per metus iki pat 2019-ų metų. Toks kasmet publikuojamų straipsnių kiekis rodo, jog su ACO algoritmais susijusios temos yra aktualios ir aktyviai plėtojamos.



1 pav. Publikuotų straipsnių, kurių antraštėse yra vartojami terminai „ant system", „ant colony system" arba „ant colony optimization" kiekis, 1996-2019metai. Šaltinis: „Scopus" duomenų bazė.

2.1. AS, ES, AS_{rank}, Maks-Min ir ACS strategijos

Kaip jau buvo minėta, AS buvo pirmas ACO algoritmas. Iš tikro, AS buvo sudarytas iš trijų algoritmų: „ant-cycle", „ant-density" ir „ant-quantity", pasiūlytų Dorigo daktaro disertacijoje [Dor92], kuri po kelių metų buvo publikuota „IEEE Transactions on Systems, Man, and Cybernetics" žurnale [DMC96a]. Minėtame straipsnyje buvo pasiūlytas AS algoritmo panaudojimas keliaujančio pirklio (angl. Traveling salesman problem, TSP) uždaviniui spręsti. TSP uždavinio tipas buvo pirmas, kuriam buvo pritaikytas AS algoritmas.

AS algoritmas susideda iš dvieju žingsnių: skruzdžių judėjimas bei feromonų atnaujinimas. Keliaujančio pirklio uždavinio atveju, pradžioje skruzdės išdėstomos atsitiktiniuose miestuose. Kiekvieno žingsnio metu, skruzdėlė k skaičiuoja kelio, vedančio į kitą miestą, pasirinkimo tiki-

mybę. Skruzdėlės k, esančios mieste i, miesto j pasirinkimo tikimybė yra lygi

$$p_{ij}^k = \frac{[\tau_{ij}]^\alpha \times [\eta_{ij}]^\beta}{\sum_{l \in N_i^k} [\tau_{il}]^\alpha \times [\eta_{il}]^\beta}, \text{ jei } j \in N_i^k \tag{1}$$

Kur τ_{ij} yra feromonų kiekis kelyje iš miesto i į miestą j, η_{ij} yra iš anksto žinoma kelio euristinė informacija, α ir β yra parametrai, įtakojantys feromonų kiekio ir euristinės informacijos svarbą, N_i^k yra skruzdės k, esančios mieste i, dar neaplankytų miestų aibė.

Kai kiekviena skruzdėlė suformuoja savo kelią, yra atnaujinamas feromonų kiekis: visose briaunose feromonų kiekis yra sumažinimas arba, kaip dar sakoma šio tipo algoritmų kontekste, feromonai yra išgarinami:

$$\tau_{ij} = (1 - \rho)\tau_{ij}, \forall (i,j) \in L \tag{2}$$

Kur L yra visų kelių aibė, $0 < \rho \leq 1$ yra feromonų išgarinimo greitis. Po išgarinimo, visose skruzdėlių aplankytose briaunose yra padidinamas feromonų kiekis, arba, kaip sakoma šio algoritmo kontekste, skruzdėlės išskiria papildomus feromonus:

$$\tau_{ij} = \tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^k, \forall (i,j) \in L \tag{3}$$

Kur $\Delta\tau_{ij}^k$ yra skruzdėlės k feromonų kiekis, kurį ji išskiria briaunoje ij, o briaunos ij priklauso briaunų aibei L. Keliaujančio pirklio uždavinio atveju, skruzdėlės išskiriamas feromonų kiekis yra lygus:

$$\Delta\tau_{ij}^k = \begin{cases} 1/C^k & \text{, jei briauna } (i,j) \text{ priklauso } T^k \\ 0 & \text{, kitu atveju} \end{cases} \tag{4}$$

Kur C^k yra skruzdėlės k keliui priklausančių briaunų aibės T^k dydis. Keliaujančio pirklio uždavinio atveju yra svarbu surasti trumpiausią kelią, todėl skruzdėlės, kurių sukonstruotame kelyje yra mažesnis briaunų skaičius, išskirs didesnį feromonų kiekį, o didesnį feromonų kiekį turinčio kelio pasirinkimo tikimybė bus didesnė. Tokiu būdu, naujos skruzdėlės bus labiau linkusios pasirinkti šį kelią ir taip dar labiau padidins šio kelio patrauklumą.

Tačiau, kaip pastebima [DS19, p 331], tuo metu algoritmas negalėjo varžytis su jau egzistuojančiais algoritmais skirtais TSP uždaviniu spręsti. Vėliau algoritmas buvo tobulinamas, pasiūlyti nauji algoritmo variantai [DS19, p 332]:

- Elitinės strategijos (angl. elitist strategy, ES) - [DMC96b], po kiekvienos, iteracijos geriausią rezultatą parodžiusiame kelyje, išskiriamas papildomas feromonų kiekis. Taigi, feromonų kiekis yra atnaujinamas sekančiu būdu:

$$\tau_{ij} = \tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^k + e\Delta\tau_{ij}^g \tag{5}$$

Kur e yra parametras, nurodantis feromonų kiekio, kuris paskaičiuojamas pagal globaliai

geriausią iki šiol rastą kelią ($\Delta\tau_{ij}^{g}$), įtaką:

$$\Delta\tau_{ij}^{g} = \begin{cases} 1/C^{k} & \text{, jei briauna } (i,j) \text{ priklauso } T^{g} \\ 0 & \text{, kitu atveju} \end{cases} \tag{6}$$

Taigi, skruzdėlės suformuoto kelio briaunose T^{k}, yra išskiriamas papildomas feromonų kiekis $e\Delta\tau_{ij}^{g}$, jei ta briauna priklauso iki šiol rasto geriausio kelio briaunų aibei T^{g}. Feromonų išgarinimas vyksta taip pat kaip ir aukščiau aprašyto AS algoritmo atveju (2).

Kaip teigia [DS04, p 73], elitinės strategijos algoritmo modifikacija, su tinkamai parinktu e parametru, parodė geresnius rezultatus naudodama mažesnį iteracijų kiekį.

- Reitingu paremta skruzdžių sistema (angl. rank based ant system, AS_{rank}) - skruzdėlių paliekamas feromonų kiekis tiesiogiai proporcingas skruzdėlės reitingui [BHS99]. Po kelio sudarymo, skruzdėlės yra rūšiuojamos pagal sukonstruoto kelio ilgį ir reitinguojamos. Kiekvienos skruzdėlės paliekamas feromonų kiekis koreguojamas pagal skruzdėlės reitingą r. Po kiekvienos iteracijos, $(w-1)$ geriausių skruzdėlių ir globaliai geriausia skruzdėlė išskiria feromonus (globaliai geriausia skruzdėlė gali ir nepapulti tarp geriausių šios iteracijos skruzdėlių). Globaliai geriausia skruzdėlė palieka didžiausią feromonų kiekį $(1/C^{g}) \times w$. O r eilės skruzdėlė pagal reitingą, palieką feromonų kiekį $(1/C^{r})$ padaugintą iš jos reitingo $\{0, w-r\}$. Taigi, šio algoritmo atveju, feromonai atnaujinami:

$$\tau_{ij} = \tau_{ij} + \sum_{r=1}^{w-1} \Delta\tau_{ij}^{r} + w\Delta\tau_{ij}^{g} \tag{7}$$

[DS04, p 74] teigia, jog pagal atliktus eksperimentus, reitingu paremta skruzdžių sistema parodė šiek tiek geresnius rezultatus nei elitinės strategijos (ES) algoritmas bei ženkliai geresnius rezultatus nei skruzdžių paieškos (AS) algoritmas.

- MAKS-MIN skruzdžių sistema (ang. MAX-MIN Ant System, MMAS) - [SH97] įvedami keturi AS algoritmo patobulinimai. Pirma, feromonų kiekį padidinti leidžiama tik iteracijos geriausiai arba globaliai geriausiai skruzdėlei. To neigiamas rezultatas yra tai, jog tokia strategija padidina skruzdžių kelio pasirinkimo stagnaciją t.y., kadangi feromonai išskiriami geriausiame kelyje (iteracijos ar globaliai), naujų kelių pasirinkimas skruzdėms tampa nepatrauklus.

  Taigi, antras patobulinimas - kad sumažinti stagnaciją, įvedamas feromonų kiekio ribojimas $[\tau_{min}, \tau_{max}]$.

  Trečia, pradžioje visiems keliams priskiriamas τ_{max} feromonų kiekis ir naudojama mažas feromonų išgarinimo greitis ρ. Tokių būdu, jau pradiniame etape skruzdėlės aplanko daugiau skirtingų miestų.

  Ketvirta, kai pasiekiama stagnacija arba, per tam tikrą iteracijų kiekį, rezultatas nėra pagerinamas, feromonų kiekis keliuose inicializuojamas iš naujo pradinėmis reikšmėmis.

Feromonų kelias atnaujinamas kaip:

$$\tau_{ij} = \tau_{ij} + \Delta\tau_{ij}^{g}, \text{ kur } \Delta\tau_{ij}^{g} = 1/C^{g} \tag{8}$$

Skruzdėlė, kuriai leidžiama išskirti feromonus, gali būti globaliai geriausia $\Delta\tau_{ij}^{g} = 1/C^{gg}$ arba iteracijos geriausia $\Delta\tau_{ij}^{g} = 1/C^{ig}$ (kaip ir ankstesnių algoritmų atveju, C žymimas vieno ar kito kelio ilgis).

Kaip teigiama [DS04, p 75], eksperimentai parodė, jog, mažiems keliaujančio pirklio uždaviniams, geresni rezultatai buvo gauti feromonų kiekio atnaujinimui naudojant iteracijos geriausią skruzdėlę, o didesniems - globaliai geriausia skruzdėlę.

- Skruzdžių kolonijų sistema (ang. Ant Colony System, ACS) [DG97] algoritme atsiranda trys nauji AS algoritmo patobulinimai.

  1. Naudojama agresyvesnė kelio pasirinkimo strategija, taip priverčiant skruzdėles ištirti daugiau galimų kelių.
  2. Feromonų išgarinimas ir papildomų feromonų išskyrimas atliekamas tik ant briaunų, priklausančių globaliai geriausios skruzdėlės keliui.
  3. Ir trečia, kai skruzdėlė juda iš miesto i į miestą j, miestus jungiančioje briaunoje yra sumažinamas feromonų kiekis, taip siekiant padidinti alternatyvių kelių radimo tikimybę.

Skruzdėlė k juda iš miesto i į miestą j, kuris parenkamas naudojant pseudorandomizuotą kelio pasirinkimo taisyklę:

$$j = \begin{cases} argmax_{l \in N_i^k}\{\tau_{il} \times [\eta_{il}]^{\beta}\} & \text{, jei } q \leq q_0 \\ J & \text{, kitu atveju} \end{cases} \tag{9}$$

Kur J parenkamas taip pat, kaip ir AS algoritmo atveju (1) (su parametru $\alpha = 1$), q yra atsitiktinis dydis, tolygiai pasiskirstęs intervale $[0,1]$, q_0 yra parametras parenkamas eksperimentiniu būdu ($q_0 \in [0,1]$). Kitaip tariant, skruzdėlė, su tikimybe q_0, rinksis geriausia kelią vertindama tik feromonų kiekį ir euristinę informaciją, o, su tikimybe $(1 - q_0)$, rinksis kelią, vertindama kiekvieno kelio pasirinkimo tikimybę naudojant (1) taisyklę.

Šio algoritmo vykdymo metu tik globaliai geriausia skruzdėlė atnaujina feromonų kiekį ir feromonų kiekis atnaujinamas tik globaliai geriausios skruzdėlės kelyje (T^g) (ankstesnių algoritmo modifikacijų atveju feromonai garinami visuose keliuose):

$$\tau_{ij} = (1 - \rho)\tau_{ij} + \rho\Delta\tau_{ij}^{g}, \forall(i,j) \in T^{g} \tag{10}$$

Kur $\Delta\tau_{ij}^{g} = 1/C^{g}$, C^{g} geriausios skruzdėlės kelio ilgis, ρ yra feromonų išgarinimo greitis.

Kaip teigia [DS04, p 77], buvo atliekami eksperimentai naudojant iteracijos geriausią ir globaliai geriausią skruzdėlę. Esant nedideliam miestų skaičiui, skirtumas buvo nežymus, o

su 100 ir daugiau miestų, globaliai geriausios skruzdėlės naudojimas buvo ženkliai rezultatyvesnis. Papildomai, kaip jau buvo minėta, po kiekvieno žingsnio iš i miesto į j miestą, skruzdėlė iš karto sumažina briaunos (i,j) feromonų kiekį:

$$\tau_{ij} = (1 - \xi)\tau_{ij} + \xi\tau_0 \tag{11}$$

Kur ξ, $0 < \xi < 1$ ir τ_0 yra eksperimentiniu būdu parenkami parametrai. τ_0 gali būti naudojama pradinė feromonų kelio reikšmė.

Taigi, pagrindinis šios strategijos skirtumas yra tas, jog feromonų garinimas ir papildymas vyksta tik geriausios skruzdėlės kelyje bei papildomai vykdomas lokalinis feromonų kiekio mažinimas skruzdėlei žengiant kiekvieną žingsnį. Lokalinis feromonų kiekio mažinimas po kiekvieno skruzdėlės žingsnio sumažina stagnaciją bei padidina naujų kelių radimo tikimybę. Taip pat, lokalinis mažinimas yra tinkamas kai vienu metu juda kelios skruzdėlės t.y., vienai skruzdėlei žengus žingsnį, padidėja tikimybė, jog kitos skruzdėlės sekančiame žingsnyje pasirinks alternatyvų kelią.

2.2. Metaeuristika

[DD99], [DD99] ir [DS04] suformulavo pirmą ACO metaeuristinį algoritmą. Kaip teigiama [DS19, p 329], šių darbų tikslas buvo sudaryti pagrindą tolimesniems ACO algoritmo taikymams klasikinių NP-sunkių kombinatorinio optimizavimo uždavinių sprendimams. Pasiūlyta ACO metaeuristika susilaukė didelio susidomėjimo mokslo bendruomenėje ir buvo pritaikytas įvairiems kombinatorinio optimizavimo uždaviniams spręsti.

[DS04] yra geriausias pradinis šaltinis renkant informaciją apie ACO. Knygoje pateikiama ACO atsiradimo prielaidos ir istorija, pristatomas bendras kombinatorinio optimizavimo uždavinio formulavimas [DS04, p 34], supažindinama su ACO metaeuristinio algoritmo principais [DS04, p 36], pateikiamos ACO metaeuristinio algoritmo pritaikymo skirtingiems optimizavimo uždaviniams gairės. Taip pat, [DS04, p 39] pateikiama ACO metaeuristinio algoritmo pritaikymo naujiems uždavinių tipams spręsti apžvalga su nuorodomis į susijusius darbus (2004-ų metų duomenimis).

[DS04, p 34] ir [DS04, p 123] suformulavo kombinatorinio optimizavimo uždavinį, kurio sprendimui vėliau pritaikomas ACO metaeuristinis algoritmas. [DS04, p 34] uždavinys formuluojamas kaip (S,f,Ω), kur S yra sprendinių aibė, f yra tikslo funkcija priskirianti reikšmę $f(s,t)$ kiekvienam sprendiniui $s \in S$ ir kurios ekstremumo ieškome, $\Omega(t)$ yra aibė apribojimų, kurie apibrėžia leistinų sprendinių aibę. Tikslo funkcijos ir apribojimų parametras t parodo, jog tikslo funkcija ir apribojimai gali priklausyti nuo laiko (iteracijos žingsnio), pavyzdžiui, sprendžiant dinaminio tipo uždavinius.

Uždavinio tikslas yra surasti tokį įmanomą sprendinį s^*, su kuriuo tikslo funkcijos reikšmė būtų globaliai minimali (arba maksimali, priklausomai nuo uždavinio tipo):

- $C = \{c_1,c_2,...,c_{N_c}\}$ yra baigtinė aibė komponentų. N_c yra komponentų kiekis. Skirtingo tipo uždaviniuose komponentų prasmė skiriasi.

- X yra baigtinė aibė uždavinio būsenų, kur $x =< c_i,c_j,...,c_h,... >$.
- S yra baigtinė sprendinių aibė, kur $S \subseteq X$.
- $\widetilde{X}$ yra aibė įmanomų sprendinių, kur $\widetilde{X} \subseteq X$ ir parodo, jog iš principo $\exists x \in \widetilde{X}$ tenkina apribojimus Ω.
- S^* yra optimalių sprendinių aibė, kur $S^* \subseteq \widetilde{X}$ ir $S^* \subseteq S$ bei $S^* \neq \emptyset$.
- Kartais naudojama papildoma $g(s,t)$ funkcija priskirianti vertę kiekvienam $s \in S$. Daugeliu atvejų $g(s,t) \equiv f(s,t), \forall s \in \widetilde{S}$ kur $\widetilde{S} \subseteq S$ yra aibė leistinų sprendinių gautų iš S pritaikius apribojimus $\Omega(t)$.

Dirbtinės skruzdėlės formuoja sprendinius judėdamos pilnai jungiame grafe $G_c =< C,L >$, kurio viršūnėse C yra uždavinio komponentai, o briaunos L pilnai apjungia visus komponentus C. Viršūnės $c_i \in C$ bei briaunos $l_{ij} \in L$ gali turėti jiems priskirtus feromonų kelius τ (atitinkamai τ_i arba τ_{ij}) bei euristinę vertė η (atitinkamai η_i arba η_{ij}). Feromonų kelias yra tarsi kolektyvinė skruzdžių atmintis ir yra atnaujinamas pačių skruzdžių sprendinio formavimo metu. Euristinė vertė gali būti tam tikra išankstinė informacija apie uždavinį arba informacija iš kitų išorinių šaltinių.

Pasak [DS04], ACO metaeuristinį algoritmą sudaro trijų procedūrų vykdymas:

- FormuotiSprendinį - šios procedūros vykdymo metu skruzdės asinchroniškai juda grafo G_c viršūnėmis C. Kiekviena skruzdėlė parenka sekančią viršūnę įvertindama feromonų kiekį kaimyninėse viršūnėse (τ) bei naudodama euristinę informaciją (η). Didesnis feromonų kiekis padidina viršūnės pasirinkimo tikimybę. Taip pat, šios procedūros vykdymo metu atliekami papildomi patikrinimai, pavyzdžiui, tikrinama ar skruzdės nepateko į uždarą ciklą.
- AtnaujintiFeromonus - atnaujinami feromonų keliai. Feromonų kiekis gali didėti arba mažėti (išgaruoti). Kai skruzdės pasiekia tikslą ir grįžta atgal, kiekviena k skruzdėlė padidina feromonų kiekį konkrečioje briaunoje arba viršūnėje tam tikru feromonų kiekiu $\Delta\tau_{ij}$ t.y., $\tau_{ij} \leftarrow \tau_{ij} + \Delta\tau^k$. Didesnis feromonų kiekis padidina tikimybę, jog sekančio žingsnio metu kitos skruzdės irgi pasirinks šitą briauną ar viršūnę.

  Kai visos skruzdėlės atliko po žingsnį į sekančią viršūnę, yra atliekamas feromonų kiekio sumažinimas arba feromonų „išgaravimas” $\tau_{ij} \leftarrow (1-\rho)\tau_{ij}, \forall(i,j) \in L$ (tuo atveju jei feromonų kiekis susiejamas su grafo G briaunomis L), kur ρ yra parenkamas parametras $\rho \in [0,1)$. Feromonų kiekis mažėja eksponentiškai, todėl rečiau pasirenkami keliai greitai visai išnyksta t.y., yra „pamirštami”.
- BendriKolonijosVeiksmai - atliekami veiksmai, kurių negali atlikti kiekviena skruzdėlė atskirai. Pavyzdžiui, šio žingsnio metu gali būti surenkama kažkokia bendra informacija, kuri vėliau naudojama kiekvienos skruzdėlės kelio parinkimo metu arba šio žingsnio metu gali būti parenkamos kelios geriausiai pasirodžiusios skruzdės ir joms leidžiama naudoti didesnį $\Delta\tau^k$ feromonų kiekį, tokiu būdu vyksta natūrali geresnį rezultatą parodžiusių skruzdžių atranka.

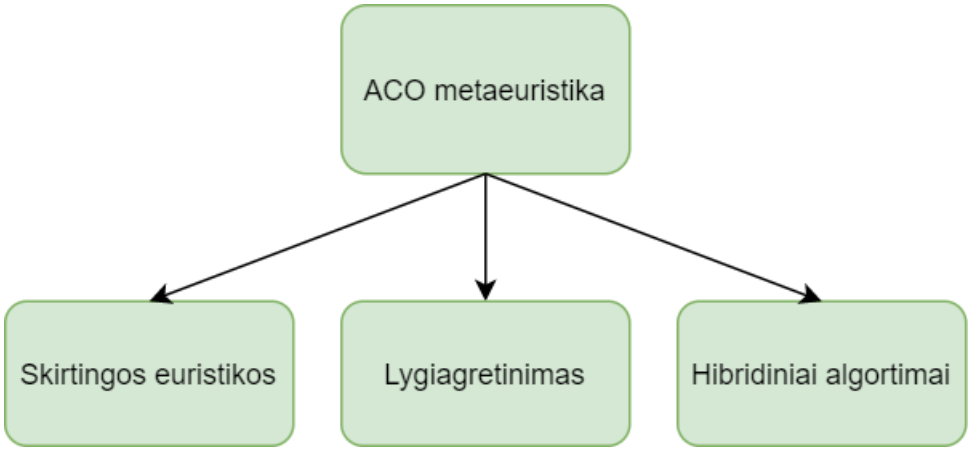
[DS04][p 38] pateikiamas labai abstraktus ACO metaeuristinio algoritmo pseudokodas:

```
procedure ACOMetaeuristika
        FormuotiSprendini
        AtnaujintiFeromonus
        BendriKolonijosVeiksmai
end procedure
```

Algoritmas 1. ACO metaeuristinio algoritmo pseudokodas

Tačiau, kaip teigia [DS04][p 38], ACO metaeuristinis algoritmas nedetalizuoja kokiu būdu yra kviečiamos ir sinchronizuojamos algoritmą sudarančios procedūros. Nėra netgi nurodyta ar procedūros vykdomos nuosekliai ar asinchroniškai. Taip pat, nėra pateiktos algoritmo vykdymo pabaigos sąlygos. Visi šie sprendimai paliekami algoritmų kūrėjams, taikantiems ACO metaeuristiką konkrečių uždavinių sprendimams. [DS04] pateikia algoritmo vykdymo rekomendacijas skirtingo tipo uždaviniams, kur pseudokodas yra praplečiamas pagal to uždavinio poreikius.

Pasak [DS19], pradžioje ACO algoritmų vystymo sritys buvo susijusios įvairiomis alternatyviomis feromonų kelių atnaujinimo taisyklėmis bei sprendinio generavimo mechanizmais. Vėliau tyrimai pakrypo link ACO algoritmų apjungimo su kito tipo algoritmais bei ACO algoritmo lygiagretinimu.



2 pav. Pagrindinės ACO vystymo kryptys.

2.3. CLP uždavinys

Šio darbo tikslas yra sukurti skruzdžių kolonijos algoritmą efektyviam jūrinių konteinerių krovos optimizavimo uždavinio (CLP) sprendimui su randomizuotu feromonų kiekio ribojimu bei elitine strategija. Bendrai, CLP uždavinys, kuriame siekiama maksimaliai užpildyti jūrinio konteinerio tūrį, gali būti formaliai aprašomas taip:

$max f(x) = \sum_{i=1}^{n} x_i v_i$, kur

- $\sum_{i=1}^{n} x_i v_i \leq W \times L \times H.$

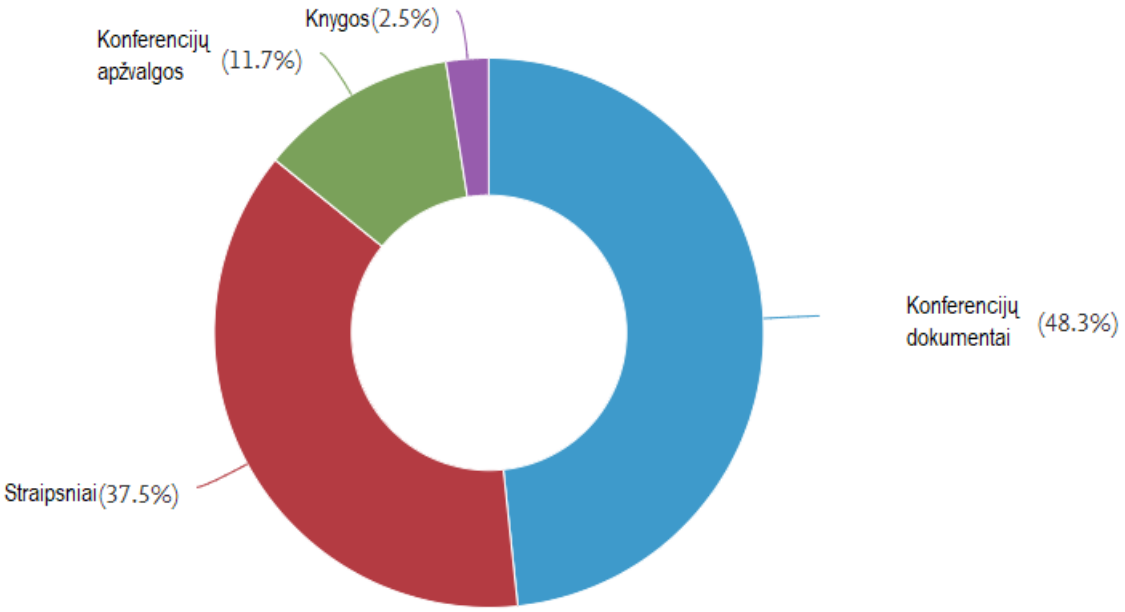
- W,H,L konteinerio plotis, aukštis ir ilgis.
- $i \in I$ yra baigtinė dėžių aibė.
- n dėžių kiekis $n = |I|$.
- x_i yra su dėže i susijęs kintamasis, kur $x_i = \begin{cases} 1, & \text{jei dėžė talpinama į konteinerį} \\ 0, & \text{jei nėra talpinama} \end{cases}$
- $x_i \in \{0,1\}, i = 1,...,n$.
- v_i yra i dėžės tūris.

2.4. Pritaikymas CLP uždaviniams

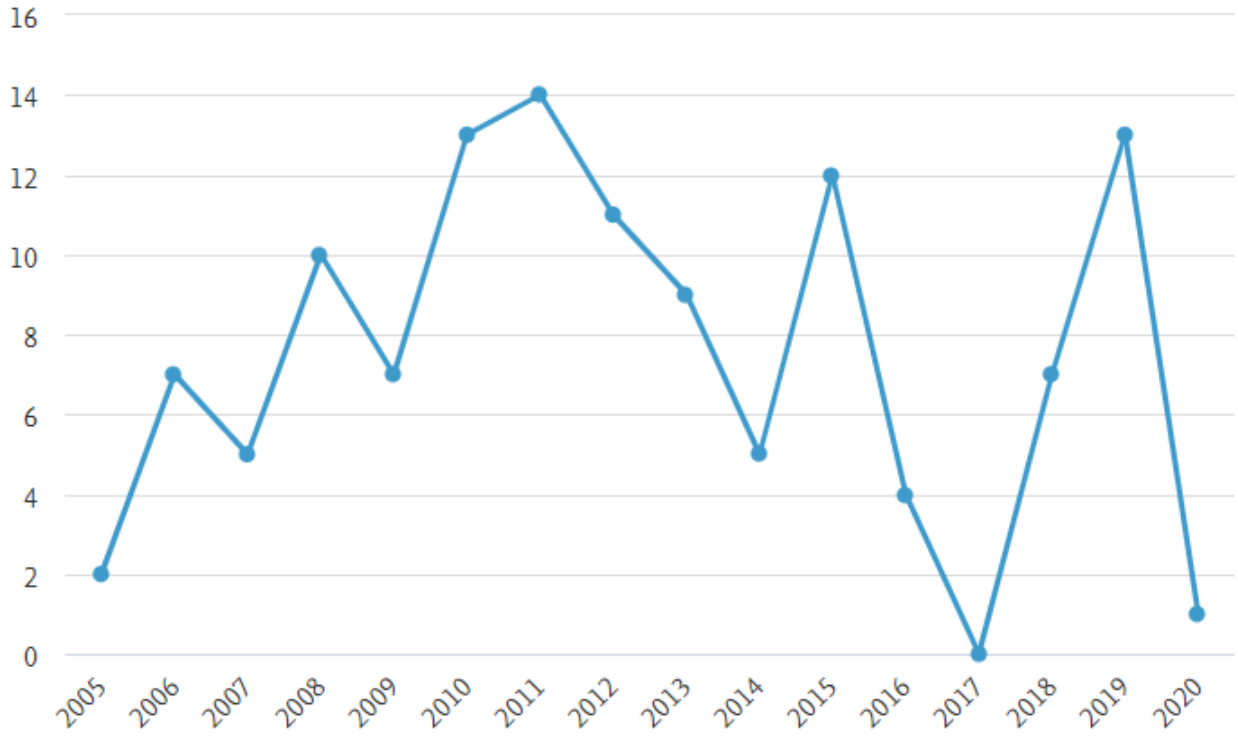
CLP uždavinio atveju, mus labiausiai domina darbai susiję su ACO algoritmo pritaikymu poaibio parinkimo uždaviniams spręsti. Tai galėtų būti KNAPSACK arba CLP tipo uždaviniai. [DS04, p 44] jau yra pateikiamas ACO metaeuristinio algoritmo adaptavimo kelių kuprinių užpildymo (angl. Multiple Knapsack Problem, MKP) uždaviniui gairės. MKP uždavinys skiriasi nuo KNAPSACK tuo, jog MKP atveju yra keletas konteinerių. Kaip ir KNAPSACK atveju, MKP kiekviena pakraunama dėžė turi vertę. Yra siekiama užpildyti visus konteinerinius dėžių poaibiu taip, kad bendra užpildytų konteinerių vertė būtų didžiausia. Šis algoritmas turėtų būti nesunkiai adaptuojamas ir CLP uždavinio sprendimui.

Norint susipažinti su naujausiais tyrimais susijusiais su ACO algoritmu bei KNAPSACK ir CLP problemomis, buvo atlikta tyrimų, publikuotų nuo 2005-ų metų, paieška „Scopus" duomenų bazėje. 2005 metai pasirinkti todėl, kad iki 2005-ų metų publikuotų reikšmingų tyrimų apžvalga jau buvo pateikta [DS04, p 39]. Kaip ir [DS04, p 39], pirmiausiai buvo išrinkti dokumentai, kurių pavadinimuose naudojami raktiniai žodžiai „ant system", „ant colony system", „ant colony optimization" bei „ant colony algorithm". Atlikus paiešką, rasta 11204 publikuotų dokumentų matematikos ir kompiuterių mokslų srityse. Paieška susiaurinta išrinkus tik tuos straipsnius, kurių antraštėse naudojami raktiniai žodžiai „kanpsack" arba „container loading problem". Susiaurinus paiešką pagal šiuos raktinius žodžius nuo 2005-ų metų, rasta 119 publikuotų dokumentų, o tiksliau: 45 straipsniai, 58 konferencijų pranešimai, 13 konferencijų apžvalgų ir 1 knyga. Paskutinis darbas paskelbtas 2020-ais metais.

Žemiau pateiktame grafike matyti publikuotų straipsnių, kuriuose yra naudojami raktiniai žodžiai „kanpsack" arba „container loading problem", pasiskirstymas pagal šaltinius nuo 2005-ų metų „Scopus" duomenų bazės duomenimis.



3 pav. Publikuotų straipsnių, kuriuose yra naudojami raktiniai žodžiai „kanpsack" arba „container loading problem", pasiskirstymas pagal šaltinius nuo 2005-ų metų. Šaltinis: „Scopus" duomenų bazė.



4 pav. Publikuotų straipsnių, kuriuose yra naudojami raktiniai žodžiai „kanpsack" arba „container loading problem", kiekis nuo 2005-ų metų. Šaltinis: „Scopus" duomenų bazė.

Surikiavus pagal citavimų kiekį buvo išrinkti naujausi laisvai pasiekiami tyrimai labiausiai susiję su nagrinėjama tema.

2.5. Naujausios ACO algoritmo modifikacijos

[KTK08] pateikė naują ACO algoritmą pavadinimu „dvejetainė skruzdžių sistema" (angl. Binary Ant System, BAS), skirtą MKP uždaviniui spręsti. Algoritmo esmė yra pasiūlytas visiškai naujas feromonų generavimo metodas, pritaikytas dvejetainei sprendinio paieškos struktūrai. Šiame darbe pasiūlyta galimą sprendinį interpretuoti kaip dvejetainę eilutę $x = x_1,...,x_n \in \{0,1\}^n$, kur n yra uždavinio dydis. Ankščiau minėto [DS04] uždavinio formulavimo atveju, tai būtų grafo viršunių skaičius. $x_j = 1$ reiškia, jog elementas j buvo pasirinktas ir, atitinkamai, $x_j = 0$ reiškia, jog elementas j pasirinktas nebuvo.

[KTK08, p 2681] pateiktas rezultatų palyginimas su kitais trimis ACO paremtais algoritmais. Teigiama, jog, lyginant su kitais trimis algoritmais, BAS vidutiniškai pateikė geresnius sprendinius, deja, [KTK08] rezultatai nėra lyginami su kitos rūšies algoritmais.

[ZZZ06] pasiūlė įdomų ACO algoritmo variantą KNAPSACK uždavinio sprendimui. Prieš feromonų kelio atnaujinimą vykdoma atsitiktine atranka paremta skruzdžių mutacija. Darbe teigiama, jog standartinio ACO algoritmo atveju geriausi feromonų keliai yra dažniau atnaujinami ir didėjant feromonų kiekiui, mažėja tikimybė, jog skruzdės pasirinks alternatyvų kelią ir, tokiu būdu, didėja tikimybė, jog rastas sprendinys yra lokalus. Jei kokiame nors feromonų kelyje pasiekiamas feromonų koncentracijos lapsnis viršijantis tam tikrą reikšmę, feromonų kelias yra modifikuojamas tam tikra atsitiktine reikšme.

Atliktas palyginimas su standartiniu ACO algoritmu ir išvadose teigiama, jog patobulinto ACO algoritmo rezultatai yra geresni. Tačiau tyrime nėra atliktas palyginimas su kito tipo algoritmais skirtais šio uždavinio sprendimui.

[CCW+19] į ACO algoritmą pažvelgė visiškai kitu kampu. Kaip teigia [CCW+19], ACO algoritmo sėkmė labai priklauso nuo algoritmo parametrų, tad šiame darbe buvo pasiūlyta evoliucinė ACO parametrų valdymo strategija (angl. ACO Parameters Control based on Evolutionary Strength, ACOP_ES). ACO parametrų parinkimo problema yra paverčiama optimalaus sprendinio paieškos problema, įvairiais kriterijais matuojamas algoritmo efektyvumas tampa tikslo funkcija, kurios pagalba siekiama rasti optimalius algoritmo parametrus. Kaip teigiama išvadose, ACOP_ES parinkti parametrai buvo geresni nei standartiniai.

[KFR+10] pasiūlė ACO algoritmą skirtą MKP uždaviniui spręsti, kuriame pritaikė MMAS algoritmo principus t.y, feromonų keliams pritaikyti minimalias ir maksimalias galimas feromonų reikšmes. Buvo pasiūlytas naujas metodas mažiausios galimos reikšmės parinkimui. Tokiu būdu, skirtumas tarp galimos minimalios ir maksimalios reikšmės tampa dinamišku. Atitinkamai, pasiūlytas algoritmas pavadintas dinaminiu MMAS (angl. dynamic MMAS, DMMAS). Taip pat buvo pasiūlytas hibridinis algoritmo variantas, kuriame papildomai panaudota lokali paieška. Algoritmas pavadintas DMMAS-ls.

Pirma, buvo atliktas DMMAS palyginimas su standartiniu MMAS algoritmu. Vėliau [KFR+10, p 78-81] pateikė DMMAS algoritmo rezultatų palyginimą su kitais dviem ACO paremtais algoritmais: Ant-Knapsack ([Ine04]) ir Aco_{LM} [LM99]. Visais trim atvejais, DMMAS parodė geresnius rezultatus. Vėliau, atliktas DMMAS-ls palyginimas su dviem visiškai kito tipo euristiniais algoritmais: GA ([CB98]) ir TS z^{LM} ([VH01]) algoritmais. Pasak [KFR+10], nors ir ne visais

atvejais rezultatai buvo geresni, DMMAS gali varžytis su nurodyto tipo algoritmais.

2.6. Naujausi ACO hibridiniai algoritmai

[KZB13] pasiūlytas MOEA/D-ACO (ang. multiobjective evolutionary algorithm using decomposition and ant colony, MOEA/D-ACO) hibridinis algoritmas skirtas spręsti kelių tikslų optimizavimo uždaviniams (anlg. multiobjective optimization problems, MOP). Algortmas apjungia ACO ir multifunkcinį evoliucinį algoritmą (anlg. evolutionary algorithm, EA). Pagrindinė algoritmo idėja: atliekama uždavinio tikslų dekompozicija iki vieno tikslo optimizavimo uždavinių, kurie yra padalinami skruzdžių grupėms. Vėliau rezultatai apjungiami ir kiekviena skruzdėlė papildomai atnaujina savo siūlomą sprendinį pagal kaimyninių skruzdžių sprendinius, jei jis labiau tenkina jų grupei priskirtą vieną iš tikslų.

Pateikiami eksperimentų rezultatai ir jų palyginimas su GA algoritmo rezultatais. Išvadose teigiama, jog MOEA/D-ACO pateikia geresnius rezultatus lyginant su GA.

[YLM+12] pasiūlė ACO algoritmo variantą CLP uždavinio sprendimui. Uždavinys sprendžiamas naudojant dvi ACO fazes. Pirmoje uždavinio sprendimo fazėje, ACO naudojamas dėžių eiliškumui sudaryti. Antroje ACO fazėje, dėžes talpinamos į konteinerį naudojant pasiūlytą bokšto statymo euristiką. Algoritmas pavadintas HACO (angl. Hybrid Ant Colony Optimization with tower building heuristic, HACO).

Pateikti algoritmo rezultatų palyginimai su kitais uždavinio sprendimo metodais. Kaip teigia autoriai, tyrimo rezultatai prilygsta rezultatams gautaiems naudojant kitus algoritmus [YLM+12, p 174].

[XZM+13] panaudojo ACO algoritmą konteinerių krovos planavime. Uždavinys susideda iš dviejų dalių: vietos parinkimas konteinerių aikštelėje ir krovos technikos planavimas. ACO panaudotas pirmo uždavinio sprendimui. Uždaviniui naudojamas standartinis ACO algoritmas. Šio darbo kontekste tyrimas yra įdomus dėl ACO panaudojimo optimalios vietos radimui dvimatėje erdvėje.

[ZD11] pasiūlė hibridinį ACO ir GA (toliau GAACO) algoritmą skirtą CLP uždaviniui. GA naudojamas sugeneruoti pirminį apytikslį sprendinį, kuris vėliau perduodamas ACO algoritmui optimalaus sprendinio radimui. Pateiktas algoritmo tyrimas užpildant 40 pėdų konteinerį 700 vienetų krovinių. Palyginti ACO, GA ir GAACO algoritmų rezultatai: GAACO algoritmas konteinerį užpildė optimaliausiai sugaišdamas mažiausiai laiko.

2.7. Literatūros apibendrinimas

Literatūros apžvalgoje buvo apžvelgta devyniolika literatūros šaltinių. Pradiniais šaltiniai renkant informaciją buvo [DS19], [DS04] bei naujausių darbų paieška „Scopus" duomenų bazėje.

Kaip matyti atlikus literatūros apžvalgą, nors pradinėje stadijoje skruzdžių kolonijų algoritmas negalėjo varžytis su jau tuo metu egzistuojančiais algoritmais, tačiau, išvystytos algoritmo modifikacijos ir skruzdžių kolonijų optimizavimo algoritmo variantai, jau pateikia rezultatus, kurie gali konkuruoti su alternatyviais tų uždavinių sprendimo metodais. 1 lentelėje pateiktas naujausių

algoritmo modifikacijų, skirtų konteinerių krovos ir panašaus tipo uždaviniams, sąrašas.

Algoritmas	Tipas	Uždavinys	Metai	Nuoroda
BAS	ACO	MKP	2008	[KTK08]
ACOknapsack	ACO	KNAPSACK	2006	[ZZZ06]
MOEA/D-ACO	Hibridinis	MOP	2013	[KZB13]
DMMAS	MMAS	MKP	2010	[KFR+10]
HACO	Hibridinis	CLP	2012	[YLM+12]
ACOclp	Hibridinis	Vietos parinkimo	2013	[XZM+13]
GAACO	Hibridinis	CLP	2011	[ZD11]
ACOP_ES	ACO	MKP	2019	[CCW+19]

1 lentelė. *Naujausi* ACO metaeuristinio algoritmo variantai CLP ir panašaus tipo uždavinių sprendimui.

Kaip matyti iš atliktos literatūros šaltinių apžvalgos:

- Galimos skirtingos ACO algoritmo pritaikymo to paties tipo uždaviniams strategijos,
- Todėl yra nuolat atliekami tyrimai susiję su skruzdžių kolonijų optimizavimo algoritmais,
- Pagal jau atliktų tyrimų rezultatus, ši tyrimų sritis, savo pritaikymo galimybėmis, yra perspektyvi,
- Didžiausia problema kurią stengiasi išspręsti tyrėjai: skruzdžių kelio stagnacija,
- Stagnacija yra sprendžiama ir algoritmo rezultatai gerinami taikant skirtingas euristikas arba naudojant hibridinius algoritmus,
- Dažniausiai algoritmas modifikuojamas ir pritaikomas keliaujančio pirklio tipo uždaviniams spręsti, todėl yra nemažai erdvės tyrimams, susijusiems su skruzdžių kolonijų optimizavimo algoritmo pritaikymu konteinerių krovos ar panašaus tipo uždaviniams,
- Darbuose, kuriuose ACO algoritmas yra pritaikomas CLP tipo uždaviniams, nėra nagrinėjamos skirtingos pakraunamų krovinių orientacijos,
- Yra labai mažai darbų lyginančių skruzdžių kolonijų optimizavimo algoritmo, skirto konteinerių krovos uždaviniams, rezultatus su rezultatais, gautais naudojant kito tipo euristinius algoritmus tam pačiam uždaviniui spręsti.

Norint įgyvendinti šiame darbe iškeltus uždavinius, sekantys žingsniai būtų: naudojant [DS04][p 44] pateiktas rekomendacijas skirtas ACO pritaikymui MKS uždavinio sprendimui bei literatūros apžvalgoje pateiktomis galimomis ACO algoritmo modifikacijomis, skirtomis susijusių uždavinių sprendimui, suformuluoti CLP uždavinį ACO terminais ir pritaikyti ACO algoritmą šio konkretaus uždavinio sprendimui. Rezultatų palyginimui bus naudojami [GML14] ir [DD10] darbai, kuriuose CLP uždaviniui panaudoti genetinis bei simuliuoto atkaitinimo algoritmai.

2.8. Konteinerių krovos optimizavimo uždavinys

Kadangi, tikslas yra palyginti sudaryto algoritmo rezultatus su [GML14] ir [DD10] darbuose gautais rezultatais bei ištirti algoritmo efektyvumą naudojant [GML14] ir [DD10] naudotus testinius duomenis, konteinerių krovos optimizavimo uždavinys suformuluojamas sekančiu būdu:

- Yra jūrinis konteineris, kurio plotį žymimas W, aukštis H, o ilgis L,
- Konteinerio tūris $V = W \times L \times H$,
- Yra N baigtinė aibė stačiakampių dėžių,
- Dėžių kiekis žymimas $n = |N|$,
- Kiekviena dėžė priklauso vienam iš m dėžių tipų iš dėžių tipų aibės $D = \{Ti,...,Tm\}$, kur T_i yra i dėžės tipo išmatavimas $\{w_i,l_i,h_i\}$, kur $w_i < W$, $l_i < L$ ir $h_i < H$,
- Kiekvieno dėžės tipo tūris yra $v_i = w_i \times l_i \times h_i$.
- [DD10] pateiktuose testiniuose duomenyse kai kuriose dėžėse laikomi skysčiai, todėl, kaip ir [GML14], laikoma, jog dėžes galima pasukti, tačiau negalima apversti. Tokiu būdu, kiekvienas dėžes tipas turi dvi orientacijas, kurios žymimos $o_i \in \{1,2\}$.

Uždavinys:

$$max f(x) = \sum_{i=1}^{n} x_i \times v_i, \tag{12}$$

kur x_i nurodo ar dėžė i yra talpinama į konteinerį ar ne $x_i \in [0,1]$, v_i yra i dėžės tūris, ir

$$\sum_{i=1}^{n} (x_i \times v_i) \leq V. \tag{13}$$

3. Bendras algoritmas

Kaip jau buvo minėta, [DS04][p 38] pateikiamas labai abstraktus ACO metaeuristinio algoritmo pseudokodas, tačiau ACO metaeuristinis algoritmas nedetalizuoja kokiu būdu yra kviečiamos ir sinchronizuojamos algoritmą sudarančios procedūros. Nėra netgi nurodyta ar procedūros vykdomos nuosekliai ar asinchroniškai. Taip pat, nėra pateiktos algoritmo vykdymo pabaigos sąlygos. Į visus šiuos klausimus bus atsakyta kuriant algoritmą.

Pradžioje, išplečiami abstraktaus algoritmo žingsniai ir suformuluojami klausimai į kuriuos bus atsakyta įgyvendinant šį algoritmą. Išplėstas skruzdžių kolonijų algoritmas atrodo taip:

1. Sudaryti skruzdžių kelio žemėlapį - grafą,
2. Kiekvienai grafo briaunai arba viršūnei priskirti pradinį feromonų kiekį,
3. Formuoti sprendinį:
   (a) Kiekvienai iš M skruzdėlių priskirti po atsitiktinę dėžę,
   (b) Kiekvienai iš M skruzdėlių, kol neviršyti ribojimai ir yra laisvų dėžių - pasirinkti sekančią dėžę iš, dar nepasirinktų, dėžių aibės,
   (c) Skruzdėlei, kuri nebegali pasirinkti dėžės - pritaikyti konteinerio pildymo euristiką ir apskaičiuoti konteinerio užpildymą,
   (d) Iš M savo kelią baigusių skruzdėlių pasirinkti geriausią rezultatą parodžiusią skruzdėlę.
4. Atnaujinti feromonų kelius:
   (a) Atnaujinti feromonų kiekį žemėlapyje pagal geriausios skruzdėlės rezultatą, tačiau neviršyti maksimalaus galimo feromonų kiekio,
   (b) „Išgarinti" feromonus, tačiau nesumažinti jų mažiau, nei mažiausias galimas feromonų kiekis.
5. Kartoti nuo trečio punkto iki tol, kol bus pasiekta algoritmo vykdymo pabaigos sąlyga.

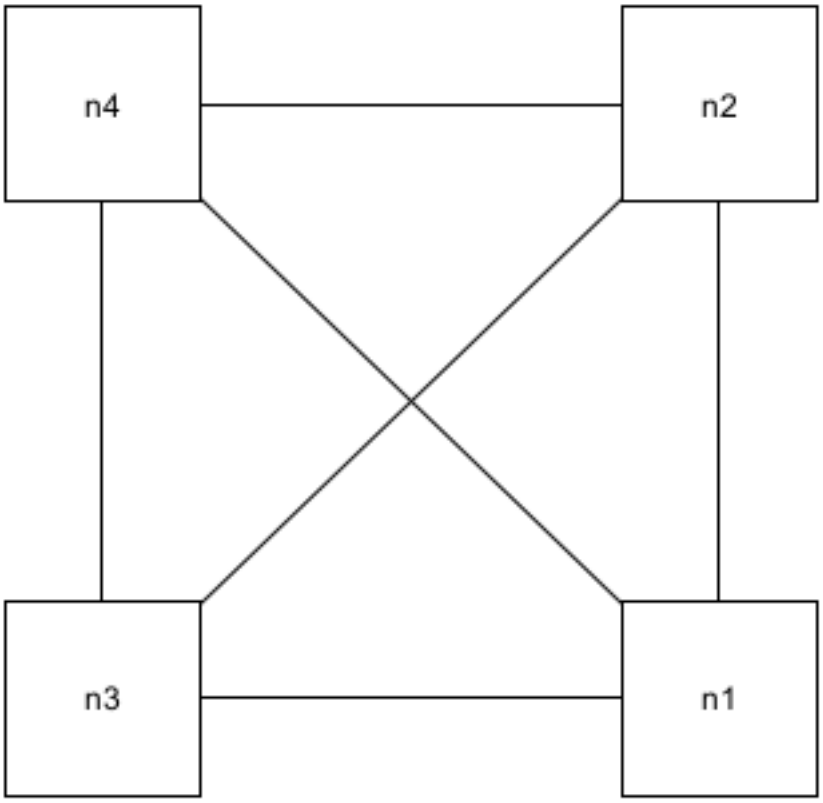
3.1. Kylantys klausimai

Norint įgyvendinti šį algoritmą, reikia atsakyti į sekančius klausimus:

- Kaip bus sudarytas grafas ir su kuo bus susietas feromonų kiekis?
- Kokiu principu bus pasirenkama sekanti dėžė?
- Kokia bus naudojama konteinerio pildymo euristika?
- Kokias nustatyti pradines, maksimalias ir minimalias feromonų reikšmes grafe?
- Kokiomis reikšmėmis bus mažinamas ir didinamas feromonų kiekis?
- Kokia bus algoritmo vykdymo pabaigos sąlyga?

4. Grafo konstravimas

Dirbtinės skruzdėlės formuoja sprendinius judėdamos pilnai jungiame grafe $G_c =< C,L >$, kurio viršūnėse C yra uždavinio komponentai - kroviniai, o briaunos L pilnai apjungia visus komponentus C. Viršūnės $c_i \in C$ bei briaunos $l_{ij} \in L$ gali turėti jiems priskirtus feromonų kiekius (atitinkamai τ_i arba τ_{ij}). Taigi, šio uždavinio atveju, grafas yra pilnasis grafas, t.y., pasirinkusi vieną dėžę, skruzdėlė gali pasirinkti bet kurią iš visų likusių dėžių, išskyrus tas, kurias jau buvo pasirinkusi prieš tai.



5 pav. Pilnasis grafas, kuriame dėžės n1, n2, n3 ir n4 yra susietos su grafo viršūnėmis.

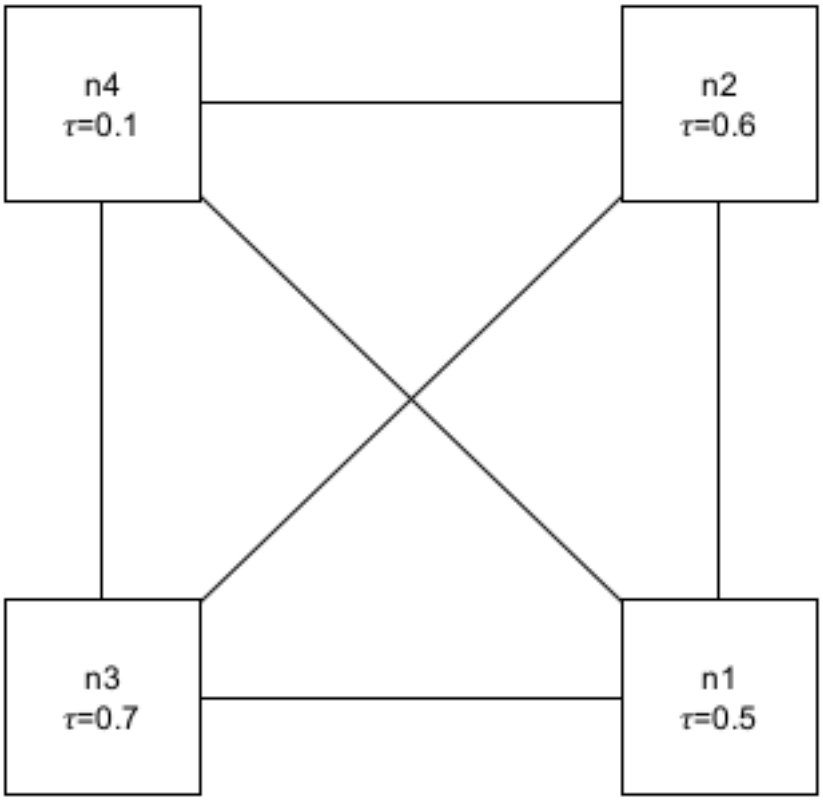
Yra keli būdai kaip galima sukonstruoti grafą šiam algoritmui:

- Galima susieti dėžes su grafo viršūnėmis ir feromonų kiekį priskirti kiekvienai viršūnei,
- Galima susieti dėžes su grafo viršūnėmis ir feromonų kiekį priskirti grafo kraštinėms,
- Galima susieti dėžių tipus su grafo viršūnėmis ir feromonų kiekį priskirti kiekvienai viršūnei,
- Galima susieti dėžių tipus su grafo viršūnėmis ir feromonų kiekį priskirti grafo kraštinėms.

4.1. Feromonų kiekis susietas su viršūnėmis

Jei feromonų kiekis yra susietas su grafo viršūnėmis, tokio atveju kiekviena dėžė turės tą pačią tikimybę būti pasirinkta, nepriklausomai nuo to kokia dėžė buvo pasirinkta prieš tai. Pavyzdžiui, jei skruzdėlė yra pasirinkusi n1 ar n2, abiem atvejais sekantis geriausias pasirinkimas yra n3

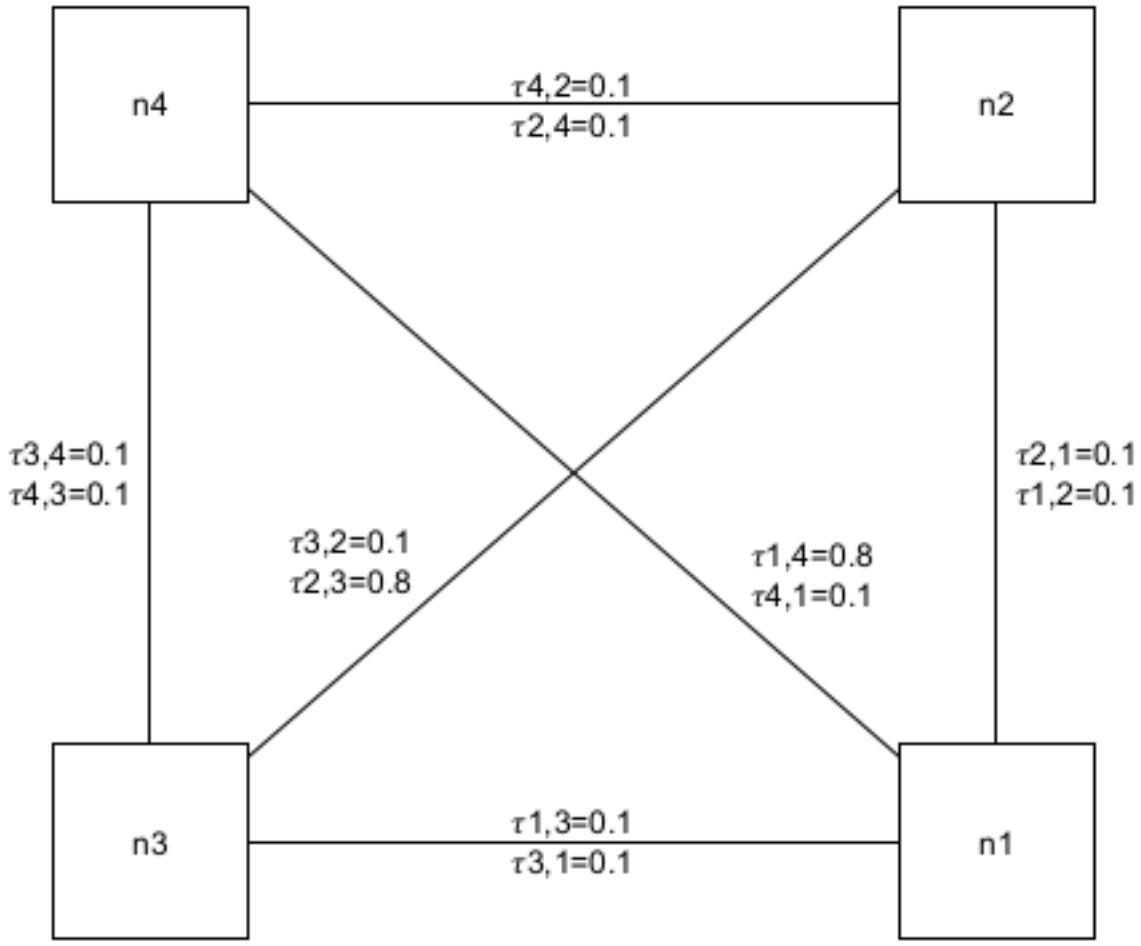
(6 paveikslėlis). Nors tokį variantą savo knygoje siūlo [DS04] ACO pritaikymui Knapsack uždavinio sprendimui, tačiau šiuo atveju jis netinka, nes Knapsack uždavinyje nėra svarbus elementų parinkimo eiliškumas, užtenka tik iš aibės išrinkti kriterijus atitinkantį poaibį ir įvertinti jį naudojant tikslo funkciją. Konteinerių užpildymo atveju yra svarbus dėžės ar, greičiau reikėtų sakyti, tam tikro tipo dėžės parinkimo eiliškumas (o šiuo atveju - svarbi ir orientacija), nes naudojamoje euristikoje parinkus dėžes skirtingu eiliškumu, gali būti gautas skirtingas rezultatas. Todėl šis grafo sudarymo būdas šio uždavinio sprendimui netinka.



6 pav. Pilnasis grafas, kuriame feromonų kiekis yra susietas su grafo viršūnėmis.

4.2. Feromonų kiekis susietas su briaunomis

Jei feromonų kiekis yra susietas su grafo briaunomis, tokių atveju kiekviena dėžė turi individualią informaciją apie tai, kokią sekančią dėžę geriau pasirinkti. Pavyzdžiui, jei paskutinė pasirinkta dėžė yra n1, tada sekanti dėžė, į kurią veda daugiausiai feromonų turintis kelias, yra n4, o jei paskutinė dėžė yra n2, tada sekantis, daugiausiai feromonų turintis kelias, veda į n3 (7 paveikslėlis).



7 pav. Pilnasis grafas, kuriame feromonų kiekis yra susietas su grafo briaunomis.

Tokiu būdu, dėžės parinkimo tikimybė priklauso nuo to, kokia dėžė buvo parinkta prieš ją. Taip pat, reiktų atkreipti dėmesį, jog tada kiekviena briauna turės dvi feromonų kiekių reikšmes, priklausomai nuo krypties kuria keliauja skruzdėlė. Pavyzdžiui, kaip pavaizduota 7-ame paveikslėlyje, feromonų kiekis, keliaujant iš n1 į n4, yra 0.8, o iš n4 į n1, tik 0.1.

Tokio tipo grafo sudarymas galėtų būti taikomas šiame algoritme, tačiau papildomai su briaunomis reikia susieti ir galimas dėžių orientacijas.

4.3. Dėžių tipai grafo viršūnėse

Dėžių tipų kiekis visada bus mažesnis arba lygus visam dėžių kiekiui. Pavyzdžiui, [DD10] pateiktame testiniame duomenų rinkinyje, kurį naudojant bus tikrinamas kuriamo algoritmo efektyvumas, yra 766 dėžės ir viso labo 12 dėžių tipų (2 lentelė).

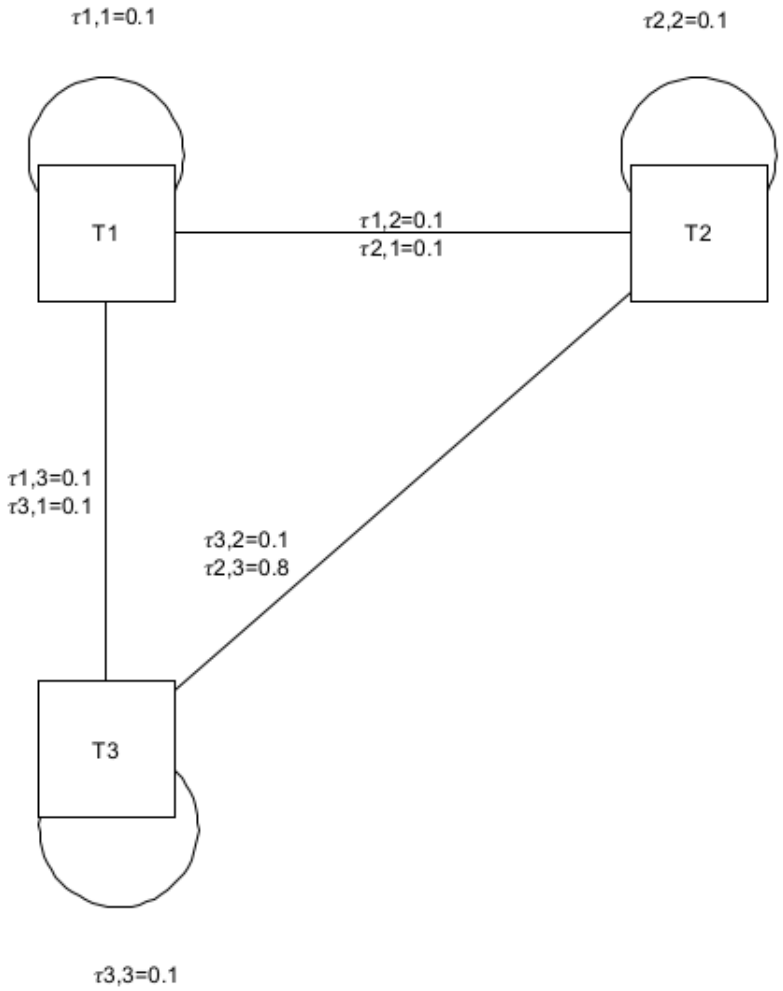
Tokiu būdu, grafo viršūnėmis galima būtų laikyti ne dėžes, o dėžių tipus. Tada kiekvienos kraštinės feromonų kiekis būtų proporcingas to tipo dėžės pasirinkimo tikimybei, o jei to tipo dėžių nebėra - būtų pasirenkamas sekantis dėžės tipas, kuris dar turi nepanaudotų dėžių ir į kurį veda daugiausiai feromonų turintis kelias. Šiuo atveju, galima būtų pasirinkti tą pačią viršūnę tiek kartų, kiek yra to tipo dėžių, tačiau tos pačio dėžės tipo pasirinkimo kelias visada turėtų tą patį feromonų

Kroviniai					
Nr	Krovinio tipas	Išmatavimai (cm) plotis/ilgis/aukštis	Kiekis (vnt)	Svoris (kg)	Bendras tūris (cm^3)
1	Ploviklis	40/36/28	325	20	13104000
2	Skystas baliklis	54/28/30	25	22	1134000
3	Asmeninės higienos priemonės	54/28/30	75	2,35	3402000
4	Ploviklis	39/29/32	75	20	2714400
5	Skutimosi priemonės	15/10/20	10	1,47	30000
6	Kūdikių priežiūros priemonės	42/37/25	150	2,8	5827500
7	Dantų pasta	36/18/18	3	3,72	34992
8	Šampūnas	18/17/22	25	4,79	168300
9	Šampūnas	22/17/22	50	5	13411400
10	Šampūnas	12/11/16	5	1,33	10560
11	Skystas baliklis	30/27/40	20	17,6	648000
12	Skutimosi priemonės	19/7/21	3	0,12	8379
Viso			766	9905,37	27493531

2 lentelė. Krovinių sąrašas.

kiekį (8 paveikslėlis).

Jei funkcija, naudojama įvertinti pasirenkamo kelio tikimybę, yra paremta kokiomis nors dėžės tipo fizinėmis savybėmis, pavyzdžiui, dugno pločiu, tūriu ar svoriu, tokiu atveju toks grafo sudarymo būdas atrodo irgi tinkamas, nes to pačio tipo dėžių parinkimo tikimybė visada būtų apskaičiuota tokia pat visoms to tipo dėžėms.

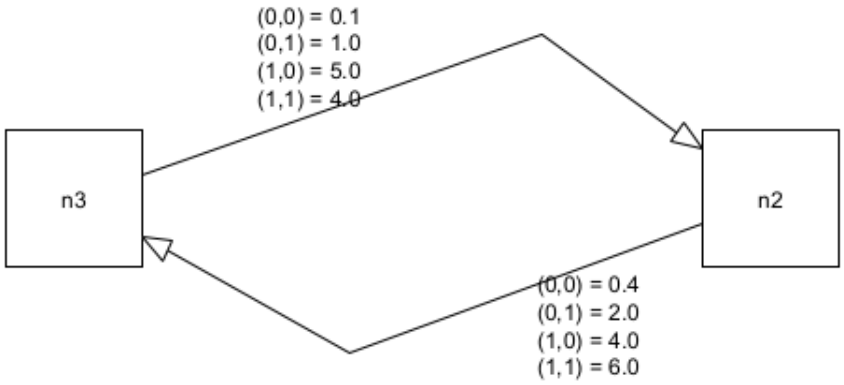


8 pav. Grafas su kurio viršūnėmis yra susieti dėžių tipais ir kuriame feromonų kiekis yra susietas su grafo briaunomis

Tačiau, jei siekiant išvengti kelio pasirinkimo sąstingio, renkantis dėžes bus naudojamas atsitiktinumo faktorius, tokiu atveju prarandame atsitiktinumą pasirenkant to paties tipo dėžes. Pavyzdžiui, jei yra 100 to paties tipo dėžių, pasirinkimo tikimybė, nepaisant atsitiktinumo faktoriaus, visoms šio tipo dėžėms bus vienoda. Kai, tuo tarpu, jei grafo viršūnėse yra dėžės, o ne jų tipai, vienos iš 100 dėžių pasirinkimo tikimybė gali būti didesnė, nepaisant to, jog visos dėžės yra to paties tipo.

Taip pat, kadangi kiekviena dėžė gali būti talpinama naudojant dvi skirtingas orientacijas, tai į konteinerį talpinant to paties tipo dėžes su skirtingomis orientacijomis, gali būti gauti skirtingi rezultatai. Taigi, turėtų būti ir skirtingas feromonų kiekis tarp skirtingų orientacijų to paties tipo dėžių.

Savo ruožtu, feromonų kiekis priklauso nuo prieš tai patalpintos ir sekančios talpinamos dėžės orientacijos. Todėl visų orientacijų kombinacijos grafe turėtų turėti atskiras briaunas. Arba, su kiekviena kiekvienos krypties briauna turėtų būti siejamos keturios feromonų reikšmės (9 paveikslėlis).



9 pav. Grafas, su kurio viršūnėmis yra susietos dėžės, o feromonų kiekis yra susietas su grafo briaunomis.

Apibendrinant:

- Feromonų kiekis bus siejamas su grafo briaunomis ir žymimas τ_{ij},
- Kiekviena briauna turės tiek feromonų reikšmių, kiek yra galimų kombinacijų tarp dviejų dėžių galimų orientacijų. Kadangi uždavinyje naudojamos tik dvi orientacijos, su kiekviena briauna bus susietos keturios feromonų reikšmės (9 paveikslėlis),
- Jei formuojant kelią buvo parinkta viena krovinio orientacija, tačiau neįmanoma šios orientacijos krovinio patalpinti į konteinerį naudojant konteinerio pildymo euristiką (6 skyrius), bus sumažintas krovinio galimų orientacijų kiekis, tačiau krovinys bus paliktas sekančioms iteracijoms,
- Jei formuojant kelią buvo parinkta viena krovinio orientacija, tačiau neįmanoma šios orientacijos krovinio patalpinti į konteinerį naudojant konteinerio pildymo euristiką (6 skyrius), o ši krovinio orientacija yra paskutinė galima, tokiu atveju krovinys bus pašalinamas iš galimų krovinių sąrašo,
- Atnaujinant (didinant ar mažinant) feromonų kiekį briaunose, jis bus atnaujinamas konkrečiai orientacijų kombinacijai.

5. Kelio pasirinkimas

Kiekviena skruzdėlė, būdama viršūnėje i, parenka sekančią viršūnę j įvertindama feromonų kiekį kaimyninėse viršūnėse τ bei euristinę informaciją. Didesnis feromonų kiekis padidina viršūnės pasirinkimo tikimybę. Kadangi siekiame maksimaliai užpildyti konteinerio tūrį, turėtume vertinti ir prioritetizuoti didesnį pasirenkamos dėžės tūrį, todėl, kaip euristinė informacija, bus naudojamas dėžių tūris v. Panašią strategiją pasirinko ir [YLM+12], tik dėl [YLM+12] algoritme naudojamos konteinerio pildymo specifikos, [YLM+12] kaip euristinę informaciją naudojo ne dėžės tūrį, bet jos dugno plotą.

Kaip euristinę informaciją naudojant dėžės tūrį, viršūnės j pasirinkimo tikimybė, kai skruzdėlė yra viršūnėje i, bus apskaičiuojama naudojant 1 formulę. Tačiau šiuo atveju:

- p_{ij}^k yra tikimybė, jog skruzdėlė k, esanti viršūnėje i, kaip sekančią viršūnę pasirinks j,
- N yra aibė galimų pasirinkti viršūnių t.y., dėžių. Į šią aibę patenka grafo $G_c =< C,L >$ viršūnės, turinčios briaunas su viršūne i, išskyrus jau anksčiau skruzdėlės pasirinktas viršūnes,
- η yra algoritme naudojama euristine informacija. Kadangi, kaip euristinę informaciją, naudojame dėžių tūrius v, tai $\eta_{ij} = v_{ij}$ bei $\eta_{il} = v_{il}$,
  - Jei $\alpha = 0$, tai feromonų kiekis visiškai neįtakos kelio pasirinkimo ir skruzdėlės rinksis naudodamos euristinę informaciją, šiuo atveju, rinksis didžiausio galimo tūrio dėžes,
  - Jei $\beta = 0$, tai dėžių tūris visiškai neįtakos skruzdžių kelio pasirinkimo ir skruzdėlės rinksis kelią vadovaudamosi tik feromonų kiekiu,
  - Jei $\alpha > 0$, o $\beta = 0$, tikėtinas skruzdžių kelio pasirinkimo sąstingis, nes skruzdėlės visiškai nesivadovauja euristine informacija,
  - Jei $\alpha = 0$, o $\beta > 0$, tikėtinas skruzdžių kelio pasirinkimo sąstingis, nes skruzdėlės vadovausis tik euristine informacija ir nesivadovaus ankstesnių skruzdžių palikta informacija t.y., feromonais,
  - Pagal [DS04, p. 71] pateiktas, įvairiose algoritmų modifikacijose naudojamas, α ir β reikšmes matyti, jog šiame algoritme galima naudoti $\alpha = 1$ ir $\beta = 1$ ir vėliau, esant poreikiui, koreguoti šias reikšmes eksperimentiniu būdu.

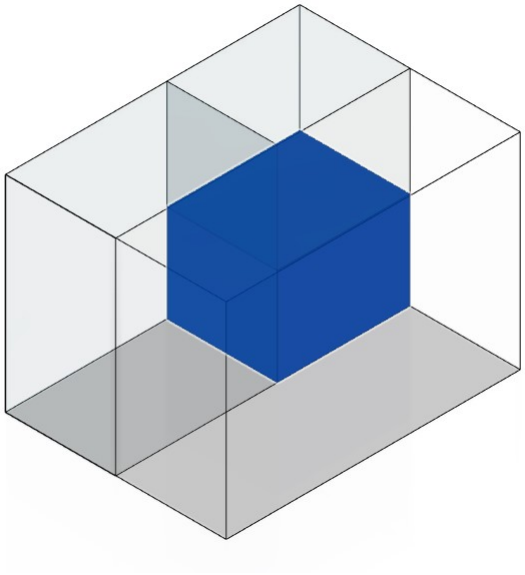
Skruzdėlė baigia savo pasirinkimus kai įvykdoma viena iš šių sąlygų:

- $D = \emptyset$ t.y., nebeliko nepasirinktų viršūnių arba
- $(\nexists c \in C)[c \notin P^k \wedge \sum_{l \in P^k} v_l + v_c \leq V]$, kur P yra skruzdės k anksčiau pasirinktų viršūnių (dėžių) aibė, v dėžės tūris, V yra konteinerio tūris, C visų dėžių aibė iš grafo $G_c =< C,L >$.

6. Konteinerio pildymo euristika

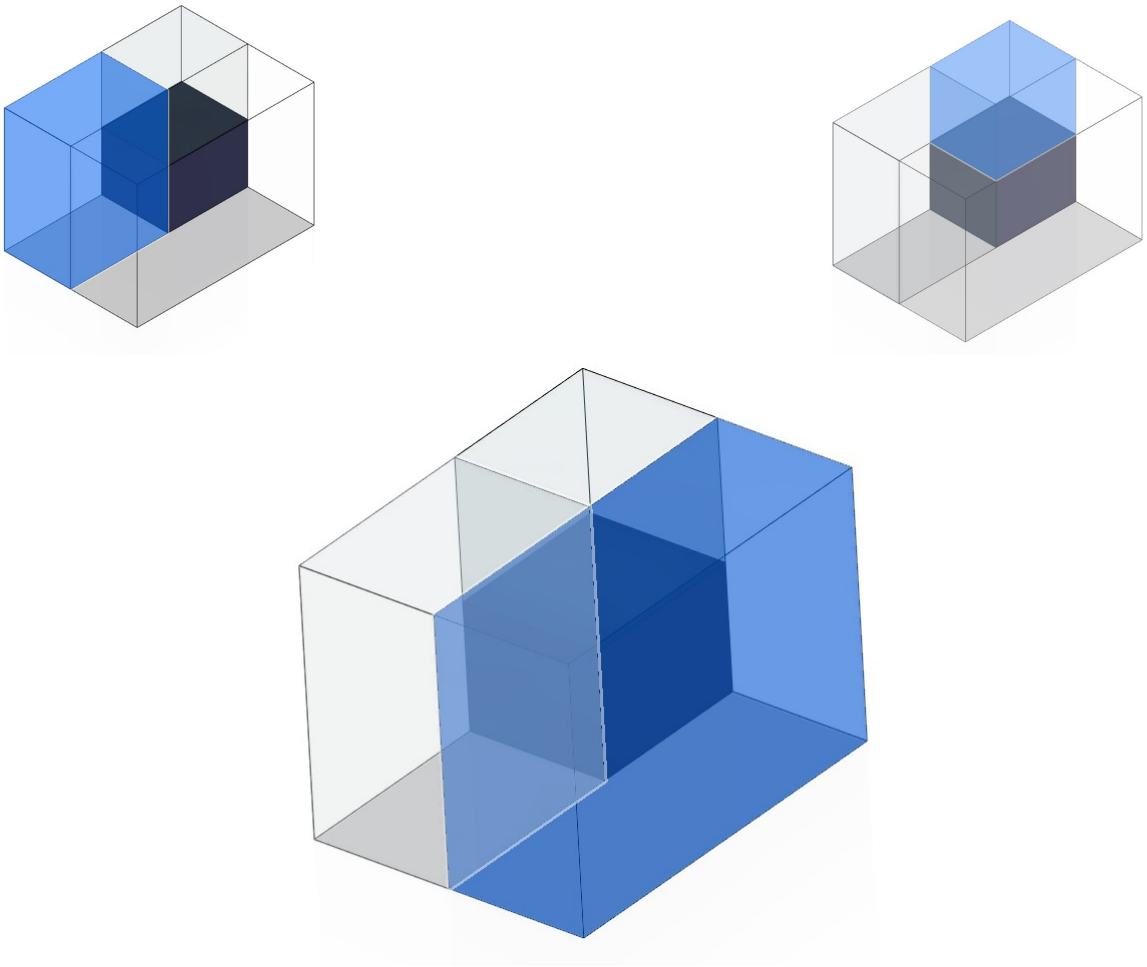
Vertinant konkrečios skruzdėlės efektyvumą, skaičiuojame konteinerio užpildymą. Kadangi konteinerio užpildymą įtakoja talpinamų dėžių eiliškumas bei orientacija, turi būti naudojamas deterministinis konteinerio užpildymo algoritmas. [GML14] panaudojo kelių pakopų užpildymo euristiką (angl. Multi-Level Filling Heuristics, MLFH). Siekiant sulyginti rezultatus, šiame algoritme bus naudojama ta pati konteinerio užpildymo euristika. Tokiu būdu konteinerio užpildymas apskaičiuojamas sekančiu būdu:

1. Kiekvienos skruzdėlės pasirinktas dėžes iš eilės talpiname į konteinerį,
2. Jei konteineris tuščias, pirma dėžė talpinama apatiniame kairiame konteinerio kampe (10 paveikslėlis),



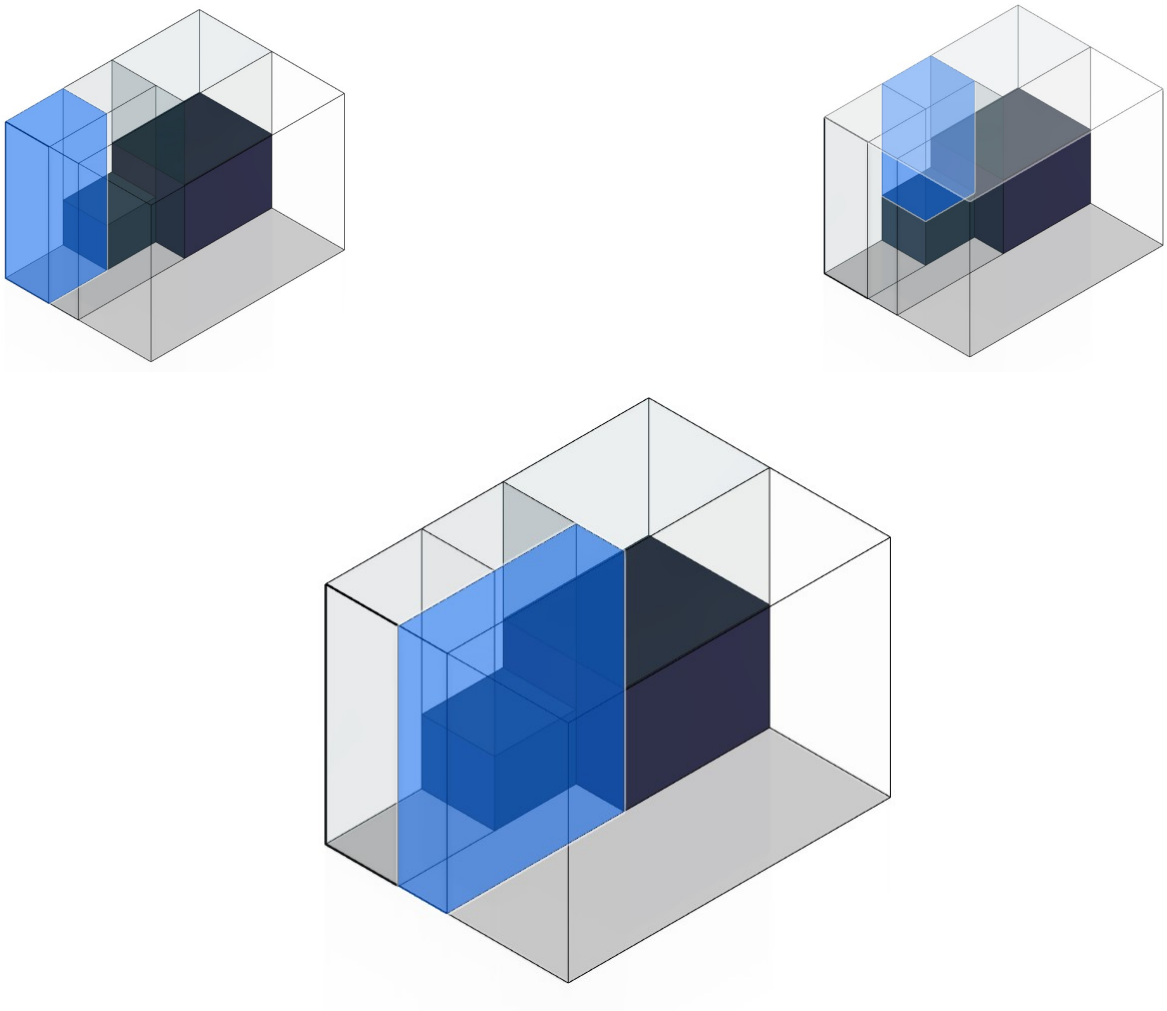
10 pav. Pirma dėžė patalpinta konteinerio apatiniame kairiame kampe

3. Pirma patalpina dėžė konteineryje sukuria tris naujas tuščias erdves: **priekyje** dėžės, **virš** dėžės ir **šalia** dėžės (11 paveikslėlis),



11 pav. Tuščios erdvės priekyje, virš ir šalia dėžės

4. Sekančią dėžę bandome talpinti naujai sukurtose **priekinėje**, **viršutinėje** ir **gretutinėje** tuščiose erdvėse,
5. Jei dėžė telpa į vieną iš erdvių, talpiname dėžę apatinėje kairėje erdvės dalyje ir tikriname ar erdvė užpildyta pilnai,
6. Jei erdvė užpildyta pilnai - ji bus pažymėta ir sekančios dėžės joje talpinamos nebus,
7. Jei erdvė nėra pilnai užpildyta - sukuriamos trys naujos tuščios erdvės (jei yra vietos), **priekyje** dėžės, **virš** dėžės ir **šalia** dėžės. Taip sukuriama sekanti tuščių erdvių pakopa (12 paveikslėlis),

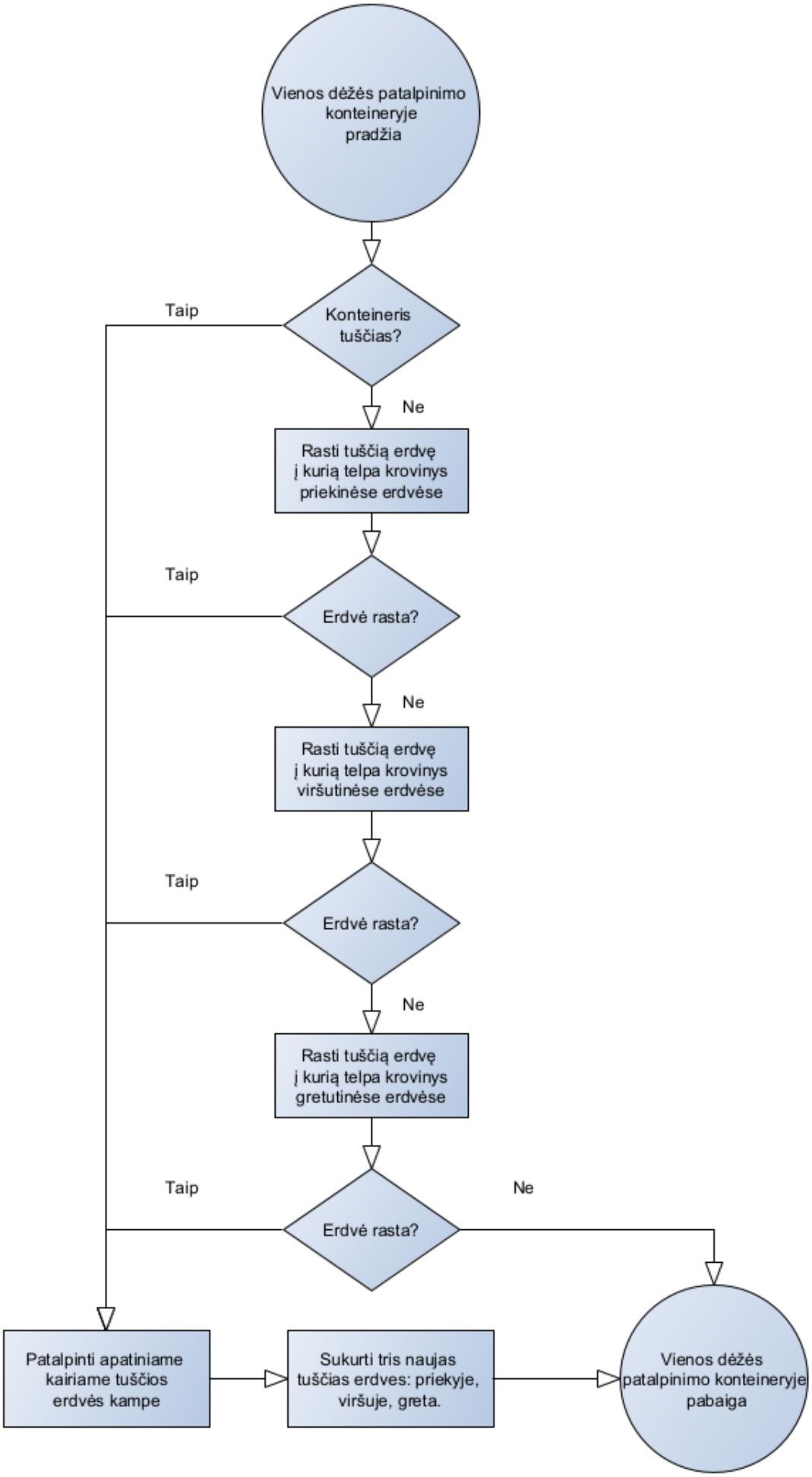


12 pav. Tuščios erdvės priekyje, virš ir šalia naujai patalpintos dėžės

8. Talpindami sekančią dėžę, tikriname visas tuščias erdves pagal jų sukūrimo eiliškumą,

9. Jei dėžė netelpa nei į vieną iš erdvių - algoritmas baigtas. Turint bendrą konteinerio tūrį V bei jame patalpintų dėžių tūrius v, galima apskaičiuoti konteinerio užpildymą $\frac{v}{V}$.

Apibendrinta konteinerio pildymo euristikos algoritmo blokine schema pateikta 14 paveikslėlyje. Pabrėžtina, jog algoritme pavaizduotas vienos dėžės talpinimas konteineryje. Kada, kokias ir kiek kartų talpinti dėžės - nustatys kuriamas skruzdžių kolonijų algoritmas.



13 pav. Vienos dėžės patalpinimo konteineryje algoritmas

7. Pradinės, maksimalios ir minimalios feromonų reikšmės

Nepriklausomai nuo to, su kuo siejamos feromonų reikšmes grafe - briaunomis ar viršūnėmis, yra labai svarbu parinkti algoritmui tinkamas pradines τ_0, maksimalias τ_{max} ir minimalias τ_{min} feromonų reikšmes.

7.1. Pradinės feromonų reikšmės

Jei bus parinkta per maža τ_0 reikšmė, tai feromonų kiekis greitai išgaruos ir algoritmo rezultatą stipriai įtakos pirmųjų skruzdėlių parinkti keliai t.y., naujos skruzdėlės rinksis pirmų skruzdžių kelius, nes visuose kituose keliuose feromonų kiekis išgaruoja per greitai.

Jei bus parinkta didelė τ_0 reikšmė, reikės daugiau algoritmo iteracijų kol skruzdžių pridedamas feromonų kiekis pradės įtakoti sekančių skruzdžių kelio pasirinkimą.

Vienas iš būdų išspręsti šią problemą yra nustatyti maksimalias ir minimalias feromonų reikšmes. Tokiu būdu, kai bus pasiektos minimalios ar maksimalios feromonų reikšmės, jos nebegalės būti mažinamos arba didinamos ir padidės tikimybė, jog skruzdėlės pasirinks kitus kelius.

[DS04, p 70] rekomenduoja susieti pradinį feromonų kiekį su skruzdžių kiekiu bei galimu sukonstruoto kelio ilgiu. Tačiau toks būdas tinka tik uždaviniams, kuriuose siekiame rasti trumpiausią kelią, pavyzdžiui, keliaujančio pirklio uždaviniui. Konteinerių krovos uždavinio atveju toks susiejimas netinka, nes trumpiausias kelias nebūtinai reiškia maksimaliai pakrautą konteinerį ir atvirkščiai.

Kitas būdas yra susieti τ_0 su maksimaliu galimu feromonų kiekiu τ_{max} kaip tai yra padaryta MMAS algoritme. Tokiu būdu, jau pradžioje yra sumažinama skruzdėlių stagnacijos tikimybė.

7.2. Maksimalios ir minimalios feromonų reikšmės

[SH97] pasiūlė algoritmo modifikaciją, kurioje pritaikė feromonų kiekio ribojimą $[\tau_{min},\tau_{max}]$. Siūlomame algoritme pradžioje buvo priskiriama maksimali galima pradinė feromonų reikšmė t.y, $\tau_0 = \tau_{max}$ ir naudojamas labai mažas feromonų garavimo greitis, t.y., mažas ρ. τ_{min} minėtame algoritme apskaičiuojama kaip $\tau_{min} = \tau_{max}/a$, kur a parenkama eksperimentiniu būdu ([DS04, p 75]). Tokiu būdu buvo siekiama sumažinti skruzdėlių kelio pasirinkimo sąstingį. $[\tau_{min},\tau_{max}]$ naudojimas apriboja p_{ij}^k iki intervalo $[p_{min}, p_{max}]$, kur $0 < p_{min} \leq p_{ij} \leq p_{max} \leq 1$.

Kadangi kuriamame algoritme nėra aišku kokią naudoti maksimalią feromonų reikšmę, šiame algoritme bus žengta vienu žingsniu toliau ir kiekvienai skruzdžių iteracijai bus naudojamos atsitiktiniu būdu parinktos $[\tau_{min},\tau_{max}]$ reikšmes:

- Pradžioje nustatome mažiausią galimą ir didžiausią galimą τ_{max} reikšmių intervalą. Pavyzdžiui, vadovaujantis [DS04, p 75] ir [SH97] rekomendacijomis, naudojame $[8,20]$,
- Pirmos iteracijos metu naudojama mažiausia galima τ_{max} reikšmė,
- Pradinis feromonų kiekis visada lygus pradiniam maksimaliam feromonų kiekiui t.y., $\tau_0 = \tau_{max}$,

- Minimalus feromonų kiekis apskaičiuojamas $\tau_{min} = \tau_{max}/a$, kur $a = 2$,
- Jei b kiekį iteracijų (kiekį b gali būti susietas su algoritmo vykdymo pabaigos sąlyga) nebus pagerintas globaliai geriausias iteracijų rezultatas:
  - Atsitiktiniu būdu parenkamas τ_{max} iš anksčiau minėto intervalo,
  - Pagal parinktą τ_{max} apskaičiuojami τ_{min} ir τ_0,
  - Visos feromonų reikšmės priskiriamos τ_0.

Pakartotino feromonų pradinių reikšmių priskyrimo metu naudojant atsitiktiniu būdu parinktą maksimalią feromonų reikšmę ir pagal ją paskaičiuotas pradinę bei minimalią reikšmes, siekiama papildomai sumažinti skruzdžių kelio sąstingį bei padidinti naujų kelių pasirinkimo tikimybę.

8. Feromonų „išgarinimas” ir papildymas

Po kiekvienos iteracijos feromonų kiekis atnaujinamas sekančiu būdu:

$$\tau_{ij} \leftarrow (1-\rho) \times \tau_{ij}, \forall (i,j) \in L \tag{14}$$

$$\tau_{ij} \leftarrow \tau_{ij} + \Delta\tau_{ij}^{g}, \forall (i,j) \in L \tag{15}$$

$$\Delta\tau_{ij}^{g} = \begin{cases} e \times \Delta\tau_{ij}^{gg} & \text{, jei briauna } (i,j) \text{ priklauso } L^{gg} \\ \Delta\tau_{ij}^{ig} & \text{, jei briauna } (i,j) \text{ priklauso } L^{ig} \\ 0 & \text{, kitu atveju.} \end{cases} \tag{16}$$

kur

- L yra grafo $G_c =< C,L >$ briaunų aibė,
- L^{ig} iteracijos geriausios skruzdėlės keliui priklausančių briaunų aibė,
- L^{gg} globaliai geriausios skruzdėlės keliui priklausančių briaunų aibė,
- e yra parametras, nustatantis globaliai geriausio iki šiol rasto kelio įtaką,
- ρ yra feromonų išgarinimo intensyvumo rodiklis $\rho \in (0,1]$,
- $\Delta\tau_{ij}^{gg}$ yra reikšmė, kuria didinamas feromonų kiekis τ_{ij} briaunose, priklausančiose L^{gg},
- $\Delta\tau_{ij}^{ig}$ yra reikšmė, kuria didinamas feromonų kiekis τ_{ij} briaunose, priklausančiose L^{ig}.

Kaip ir pradinis feromonų kiekis τ_0, feromonų išgarinimo intensyvumo rodiklis ρ stipriai įtakoja algoritmo efektyvumą: parinkus per mažą ρ, feromonų kiekis mažės per lėtai ir algoritmas bus vykdomas ilgiau, o jei ρ bus parinktas per didelis - algoritmo rezultatą stipriai įtakos pirmųjų skruzdėlių parinkti keliai t.y., naujos skruzdėlės rinksis pirmų skruzdžių kelius, nes visuose kituose keliuose feromonų kiekis išgaruoja per greitai.

[DS04] siūloma susieti $\Delta\tau_{ij}^{g}$ su geriausios skruzdėlės sukonstruoto kelio ilgiu. Tačiau toks būdas tinka tik uždaviniams, kuriuose bandome rasti trumpiausia kelią, pavyzdžiui, keliaujančio pirklio uždaviniui.

Šio uždavinio tikslas yra maksimalus konteinerio užpildymas, todėl τ_{ij} bus didinamas pagal geriausios (iteracijos arba globaliai, priklausomai nuo (16)) skruzdėlės rezultatą:

$$\Delta\tau_{ij}^{gg} = \frac{\sum_{c \in C^{gg}} v_c}{V}, \forall (i,j) \in L^{gg} \tag{17}$$

$$\Delta\tau_{ij}^{ig} = \frac{\sum_{c \in C^{ig}} v_c}{V}, \forall (i,j) \in L^{ig} \tag{18}$$

kur

- V yra konteinerio tūris,
- C^{gg} yra globaliai geriausios skruzdėlės viršūnių (krovinių) aibė,
- C^{ig} yra iteracijos geriausios skruzdėlės viršūnių (krovinių) aibė,
- L^{gg} yra globaliai geriausią rezultatą parodžiusius skruzdės briaunų aibė,
- L^{ig} yra iteracijoje geriausią rezultatą parodžiusius skruzdės briaunų aibė.

Kaip matyti iš (17) ir (18), didžiausia galima $\Delta\tau_{ij}^{g}$ reikšmė yra 1.

9. Algoritmo vykdymo pabaigos sąlyga

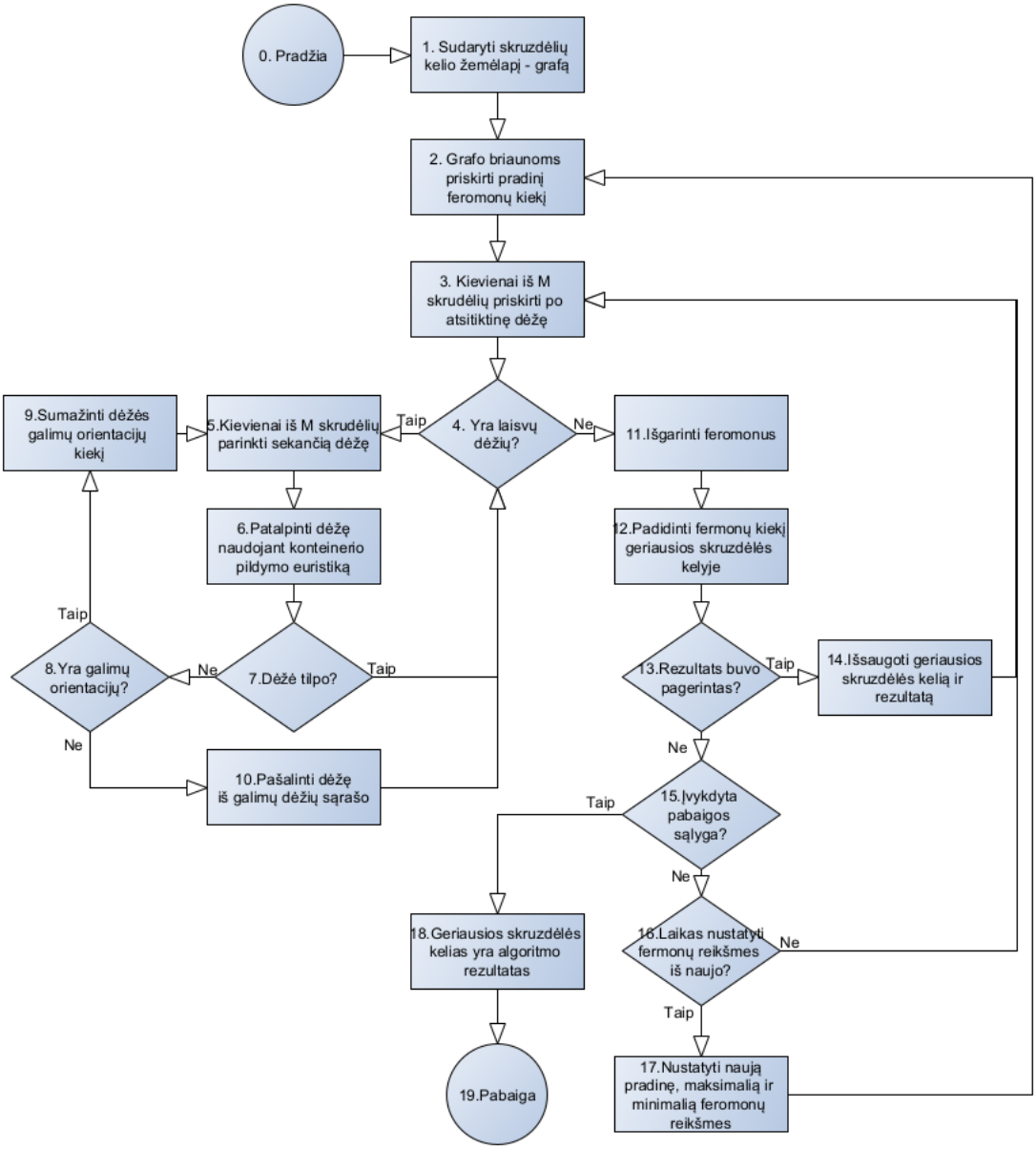
Yra svarbu nustatyti algoritmo vykdymo pabaigos sąlygą. Dėl šio tipo algoritmų specifikos negalima vienareikšmiškai įvertinti ar gautas rezultatas bus pagerintas sekančiose iteracijose. Todėl vienas paprasčiausių būdų yra skaičiuoti kiek kartų iš eilės rezultatas nebuvo pagerintas. Panašią strategiją naudojo ir [YLM+12] - algoritmas buvo stabdomas kai 2000 skruzdėlių iš eilės nepagerino rezultato.

Šio algoritmo atveju, kadangi kiekvienos iteracijos metu bus išrenkama tik viena geriausia skruzdėlė, galima skaičiuoti kiek iš eilės iteracijų rezultatas nebuvo pagerintas. Jei vienoje iteracijoje būtų naudojama 100 skruzdėlių, tada, remiantis [YLM+12] praktika, algoritmas būtų stabdomas jei rezultatas nepagerintas 10 ar daugiau iteracijų iš eilės.

Taip pat, šį parametrą galima naudoti ir iš naujo nustatant pradines, maksimalias ir minimalias feromonų reikšmes. Jei parinktume algoritmo vykdymo pabaigos sąlyga 30, tada 10 galėtų būti pradinių feromonų reikšmių nustatymo sąlyga. Tokiu atveju, algoritmas baigtų savo darbą jei 30 iteracijų iš eilės skruzdėlės nepagerino globaliai geriausio rezultato ir, tame tarpe, du kartus būtų iš naujo parenkamos ir nustatomos pradinės, maksimalios ir minimalios feromonų reikšmės (po 10-os ir 20-os iteracijų). Šios sąlygos galėtų būti koreguojamos eksperimentiniu būdu.

10. Algoritmas

Vadovaujantis aukščiau priimtais sprendimais, sudarytas sekantis skruzdžių kolonijos algoritmas jūrinių konteinerių krovos optimizavimo uždavinių sprendimui su randomizuotu feromonų kiekio ribojimu:



14 pav. Skruzdžių kolonijos algoritmas jūrinių konteinerių krovos optimizavimo uždavinių sprendimui su randomizuotu feromonų kiekio ribojimu.

Kur:

1. Skruzdžių kelio žemėlapis sudaromas vadovaujantis 4 skyriuje priimtais sprendimais,
2. Pradinis maksimalus feromonų kiekis τ_{max} yra 8, o mažiausia ir didžiausia τ_0 reikšmė bus $[8{,}20]$, kaip aprašyta skyriuje 7.2,
3. Vienos iteracijos metu naudojama 100 skruzdėlių,

4. Iteracija nesibaigia tol, kol nors viena iš šių skruzdėlių pasirenka naują dėžę,
5. Kiekviena skruzdėlė pasirenka sekančią dėžę naudodama 5 skyriuje aprašytą dėžės pasirinkimo tikimybės formulę 1, kur $\alpha = 1, \beta = 1$,
6. Pasirinkta dėžė bus talpinama naudojant 6 skyriuje aprašytą konteinerio pildymo euristiką ir taip paskaičiuojamas konteinerio užpildymas,
7. Jei, naudojant konteinerio pildymo euristiką, esamos orientacijos dėžės konteineryje patalpinti negalime, t.y., šios orientacijos dėžė negali būti patalpinta nei vienoje iš konteineryje sukurtų erdvių,
8. tačiau yra papildomų orientacijų,
9. dėžė nėra įtraukiama į skruzdėlės kelią, tačiau ir nėra pašalinama iš galimų pasirinkti dėžių - yra sumažinamas dėžės galimų orientacijų kiekis,
10. bet jei papildomų orientacijų neliko - dėžė pašalinama iš galimų pasirinkti dėžių sąrašo.
11. Pasibaigus iteracijai, kai skruzdėlėms nebeliko pasirinktinų dėžių, vykdomas feromonų išgarinimas naudojant 8 skyriuje pateiktą 14 formulę, kur $\rho = 0.1$,
12. Po feromonų išgarinimo, didinamas feromonų kiekis pagal 8 skyriuje pateiktas 15 ir 17 taisykles,
13. Jei paskutinėje iteracijoje geriausias rezultatas buvo pagerintas,
14. Jis išsaugomas kaip globaliai geriausias rezultatas,
15. Jei dar nepasiekta pabaigos sąlyga, kuri aprašyta 9 skyriuje,
16. Ir jei b kiekį iteracijų (kur b parenkamas eksperimentiniu būdu, pavyzdžiui 10) nebuvo pagerintas globaliai geriausias rezultatas,
17. Atliekamas feromonų kiekio reinicializavimas, atsitiktiniu būdu parenkant τ_{max} iš anksto apsibrėžtame intervale, pavyzdžiui $[8{,}20]$. Pagal jį apskaičiuojama τ_0 ir τ_{min}, kaip aprašyta skyriuje 7.2,
18. Jei pabaigos sąlyga buvo įvykdyta, globaliai geriausios skruzdėlės rezultatas bus algoritmo vykdymo rezultatas.

11. Bandymų rezultatai

Visi bandymai buvo atliekami kompiuteryje su Intel i5-8600K 3600Mhz taktinio dažnio 6 branduolių procesoriumi ir 32Gb operatyviosios atminties. Algoritmai suprogramuoti C++ programavimo kalba ir sukompiliuoti bei vykdomi Windows operacinės sistemos Linux posistemėje (ang. Windows Subsystem for Linux, WSL) naudojant G++ kompiliatorių ir OpenMP technologiją. Programos, naudojamos bandymuose, išeities tekstas pasiekiamas [22]. Bandymams buvo naudojami [GML14] ir [DD10] darbuose naudoti duomenys (2 lentelė).

Pirma, sukūrus algoritmą buvo atlikti bandymai siekiant įvertinti konteinerio pildymo euristikos ir viso algoritmo teisingumą. Algoritmo teisingumas įvertintas su lengvai apskaičiuojamais testiniais duomenimis, pavyzdžiui, turint konteinerio dydį 100x100x100 ir dėžę 100x100x50, turėtume gauti 50% konteinerio užpildymą ir t.t. Įsitikinus algoritmo vykdymo teisingumu, buvo atlikti konteinerio pildymo euristikos bandymai.

Konteinerio pildymo euristika buvo tikrinama trimis atskirais bandymais: pildant konteinerį pagal tūrį surūšiuotomis dėžėmis naudojant pirmą, antrą ir atsitiktinę dėžių orientacijas. Sekantis bandymas buvo konteinerio užpildymas atsitiktinai parinktomis dėžėmis naudojant atsitiktines dėžių orientacijas.

Pirmiausia buvo atliktas bandymas kuriame dėžės talpinamos į konteinerį surūšiuotos mažėjančia tvarka pagal dėžės tūrį. Kadangi siekis yra maksimaliai užpildyti konteinerį ir, kaip euristinę informaciją, dėžės pasirinkimo tikimybei naudojame dėžės tūrį, buvo siekiama įvertinti konteinerio užpildymą naudojant tik konteinerio pildymo euristiką be ACO algoritmo. Taip pat, buvo siekiama įvertinti dėžių orientacijų įtaką konteinerio pildymo euristikai. Kaip matyti iš 15 grafiko, pildant konteinerį naudojant tik vieną arba kitą orientaciją ir tą patį eiliškumą, visada gaunamas tas pats rezultatas, kas yra teisinga. Tuo tarpu, jei naudojame tą patį dėžių eiliškumą, bet atsitiktinai parinktas dėžių orientacijas, konteinerio pildymo euristikos rezultatai skiriasi ir yra prastesni, lyginant su rezultatais, kai naudojama tik viena orientacija. Apibendrinant, po 20 bandymų buvo gauti sekantys rezultatai: naudojant pirmą orientaciją konteinerio užpildymas - 86,30%, naudojant antrą orientacija konteinerio užpildymas - 83,62%, naudojant atsitiktines dėžių orientacijas konteinerio užpildymo bandymų vidurkis - 80,90%.

Pildant konteinerį dėžėmis atsitiktiniu eiliškumu ir naudojant atsitiktinai parinktas dėžių orientacijas (16 paveikslėlis), atlikus 20 bandymų konteinerio užpildymo vidurkis buvo 56,43%. Rezultatas yra prastesnis nei pildant konteinerį dėžėmis, surūšiuotomis mažėjančia tvarka pagal dėžės tūrį.

Iš šių bandymų galima daryti išvadą, jog siekiant maksimaliai užpildyti konteinerį, rinktis dėžes vadovaujantis dėžės tūriu yra teisinga strategija ir rezultatai, gaunami talpinant pagal tūrį surūšiuotas dėžes, yra geresni nei talpinant dėžes atsitiktiniu eiliškumu bei atsitiktinėmis dėžių orientacijomis.

Toliau, palyginimui, buvo atlikti bandymai naudojant skruzdžių paieškos (rezultatai 17 grafike), elitinės strategijos (rezultatai 18 grafike), skruzdžių kolonijos sistemos (rezultatai 19 grafike) ir Maks-Min skruzdžių sistemos (rezultatai 20 grafike) algoritmo variantus pritaikytus CLP uždaviniui ir naudojančius tą pačią konteinerio pildymo euristiką. Visais atvejais buvo atlikta 20 ban-

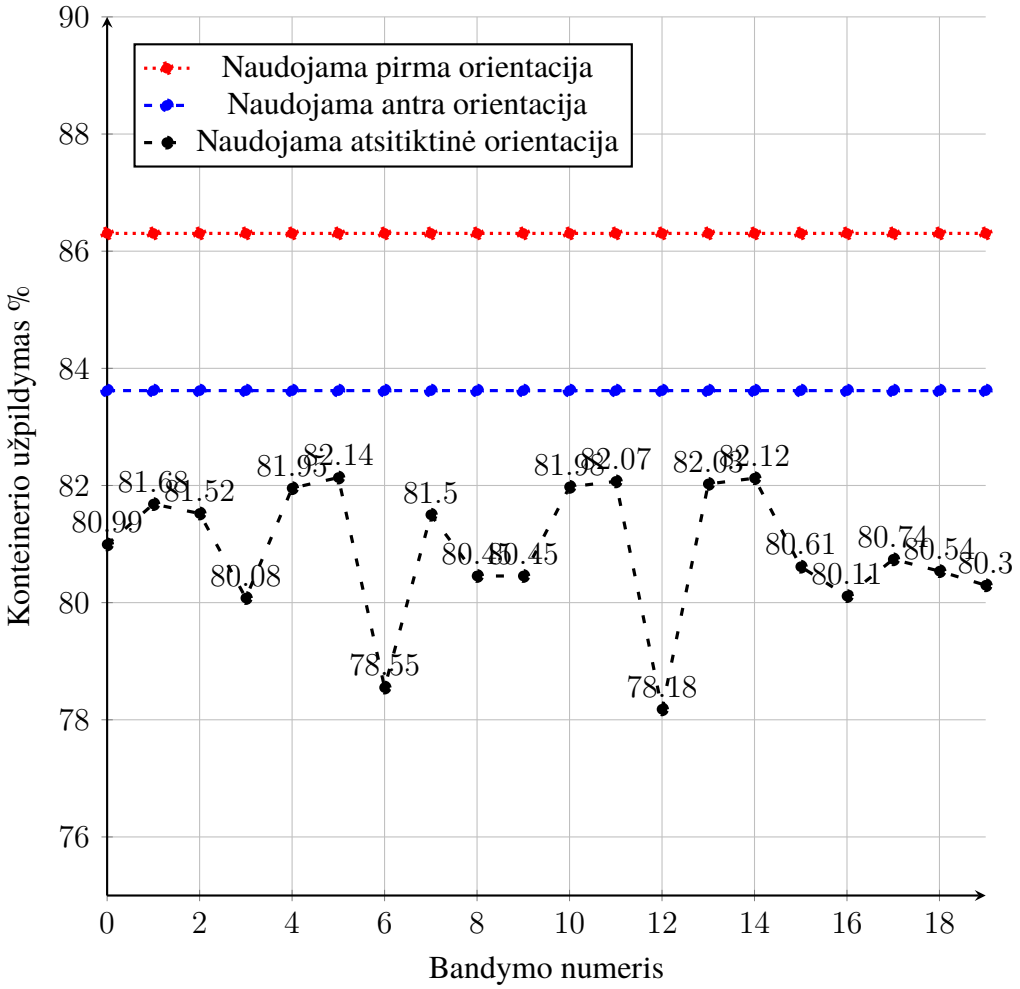
dymų naudojant 100 skruzdėlių ir 30 nepagerintų iteracijų pabaigos sąlygą. Visais atvejais naudota $\rho = 0{,}1$, $\tau_0 = 8$, $\alpha = 1$, $\beta = 1$ ir $\tau_{min} = 4$, $\tau_{max} = 0$ tais atvejais, kai τ_{min} ir τ_{max} yra taikomi.

Ir, pabaigoje, buvo atlikti bandymai su skruzdžių kolonijų algoritmu jūrinių konteinerių krovos optimizavimui, naudojančiu randomizuotą feromonų kiekio ribojimą ir elitinę strategiją. Algoritmo vykdymo metu buvo naudojami sekantys parametrai: $\rho = 0{,}1$, $\alpha = 1$, $\beta = 1$, $\tau_{max} \in [8{,}20]$, $\tau_{min} = \tau_{max}/2$, $\tau_0 = \tau_{max}$, $b = 10$, $e = 1{,}5$. Atliktų bandymų rezultatai pateikti 21 grafike. Vieno iš atliktų bandymų konteinerio užpildymo iteracijų raida pateikta 22 grafike. Vertikalia punktyrine linija pažymėtos vietos kai feromonų kiekis grafe buvo inicializuojamas iš naujo atsitiktinai parinktomis pradinėmis ir maksimaliomis feromonų reikšmėmis. Kaip matyti 22 grafike, iš naujo inicializuojant feromonų kiekius, pirmose iteracijose gaunamas mažas konteinerio užpildymas, tačiau greitai rezultatai pasiekia globaliai geriausią rezultatą ir, tam tikrais atvejais, jį pagerina.

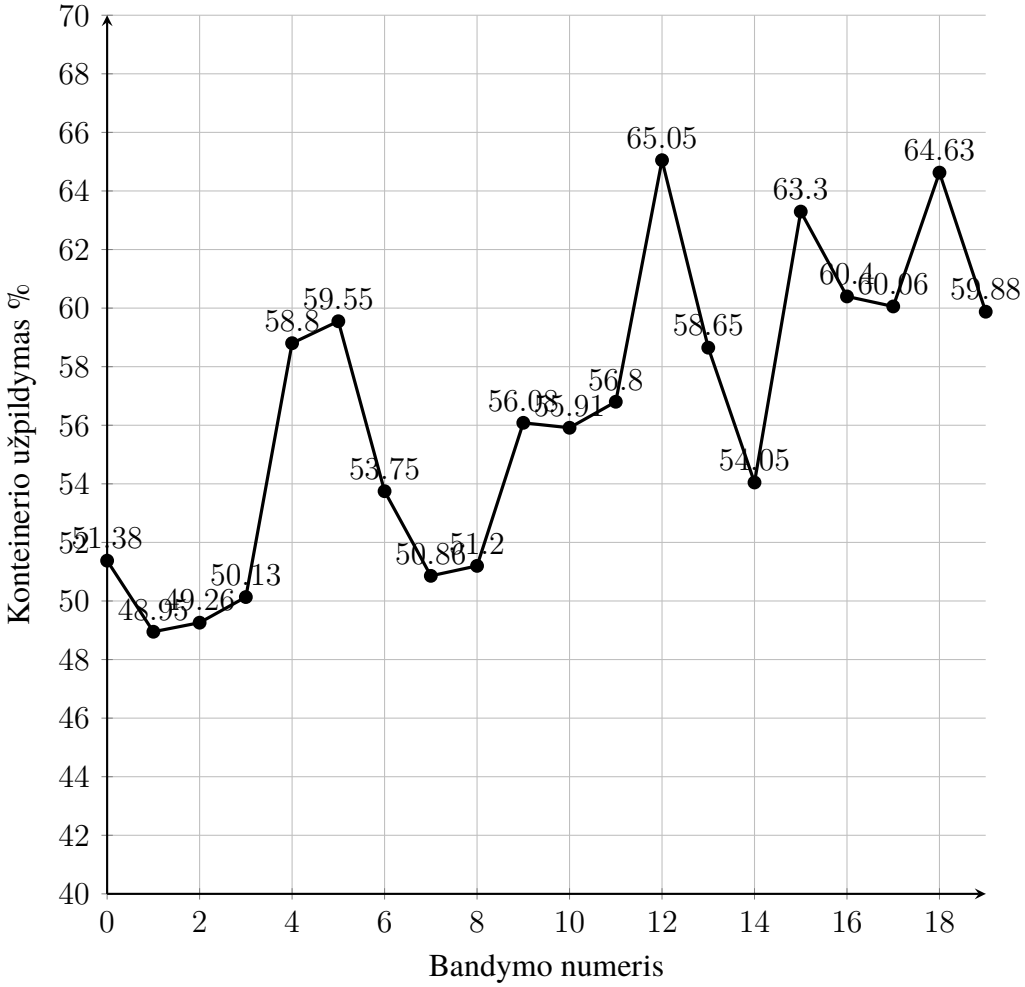
Algoritmas	Konteinerio užpildymas %	Rezultatas gautas
Atsitiktinis eiliškumas ir orientacija	56,43	Atlikus bandymus
Rūšiuotas, atsitiktinė orientacija	80,90	Atlikus bandymus
Rūšiuotas, antra orientacija	83,62	Atlikus bandymus
Rūšiuotas, pirma orientacija	86,30	Atlikus bandymus
Modeliuojamo atkaitinimo (SA)	87,51	Pateiktas [DD10] darbe
Skruzdžių kolonijos sistemos (ACS)	90,84	Atlikus bandymus
Elitinės strategijos (ES)	90,92	Atlikus bandymus
Skruzdžių paieškos (AS)	90,95	Atlikus bandymus
Maks-Min skruzdžių sistemos	91,06	Atlikus bandymus
Genetinis, SLFH	91,13	Pateiktas [GML14] darbe
Maks-MinRE	**91,23**	Atlikus bandymus
Genetinis, MLFH	91,4	Pateiktas [GML14] darbe

3 lentelė. Bandymų rezultatų palyginimas.

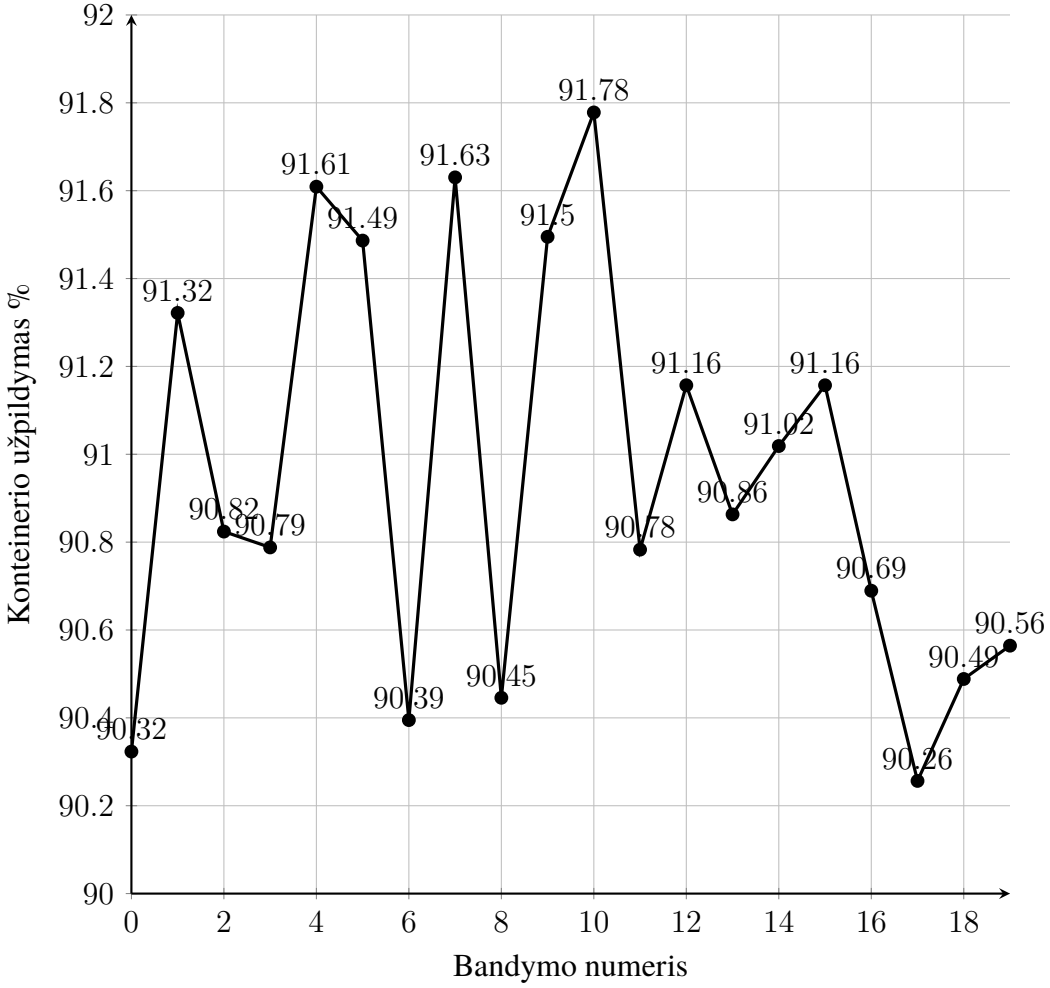
3 lentelėje pateiktas sukurto algoritmo (lentelėje paryškintas ir pažymėtas kaip Maks-MinRE) rezultatų palyginimas su skruzdžių paieškos (AS), elitinės strategijos (ES), skruzdžių kolonijos sistemos (ACS), Maks-Min skruzdžių sistemos rezultatais, bei rezultatais, gautais naudojant kito tipo algoritmus: genetiniais algoritmais, naudojančiais MLFH ir SLFH konteinerio pildymo euristikas ([GML14]), modeliuojamo atkaitinimo (SA) algoritmu ([DD10]). Visais atvejais lyginami rezultatai kai algoritmo prioritetas yra konteinerio tūrio užpildymas. Nurodytuose darbuose algoritmai buvo testuojami naudojant tuos pačius duomenis (2 lentelė). Taip pat, 3 lentelėje pateikti rezultatai, gauti pildant konteinerį atsitiktiniu būdu su atsitiktinėmis orientacijomis bei pildant kroviniais surūšiuotais pagal tūrį su pirma, antra ir atsitiktine orientacijomis.



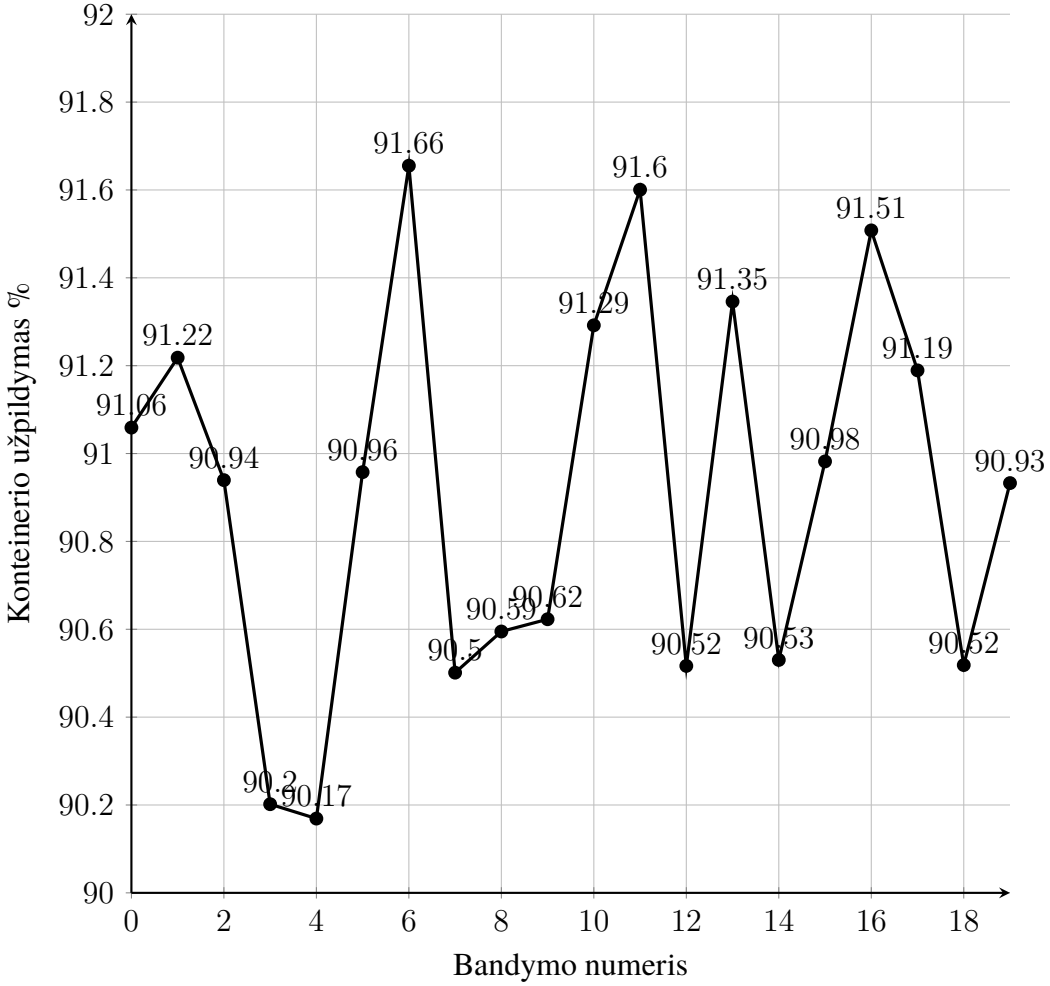
15 pav. Kroviniai talpinami į konteinerį surūšiavus ir naudojant pirmą, antrą bei atsitiktinę orientacijas. Naudojant pirmą orientaciją, konteinerio užpildymas - 86,30%, naudojant antrą orientacija, konteinerio užpildymas - 83,62%, naudojant atsitiktines dėžių orientacijas, konteinerio užpildymo bandymų vidurkis - 80,90%. Kaip matyti iš grafiko, pildant konteinerį naudojant tik vieną arba kitą orientaciją ir tą patį eiliškumą, visada gaunamas tas pats rezultatas, kas yra teisinga. Tuo tarpu, jei naudojame tą patį dėžių eiliškumą, bet atsitiktinai parinktas dėžių orientacijas, konteinerio pildymo euristikos rezultatai skiriasi ir yra prastesni, lyginant su rezultatais, kai yra naudojama tik viena orientacija.



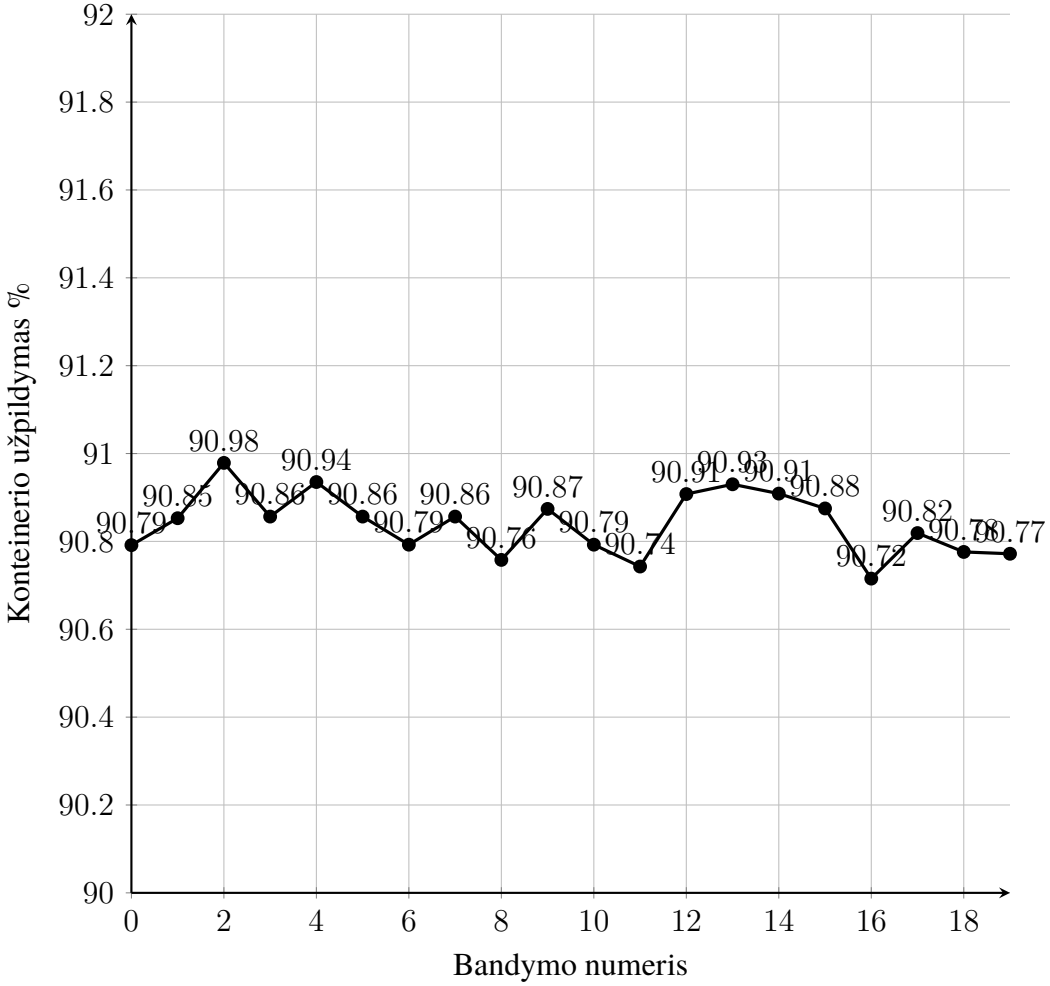
16 pav. Kroviniai talpinami į konteinerį atsitiktiniu eiliškumu ir naudojant atsitiktines dėžių orientacijas, konteinerio užpildymo vidurkis 56.43%. Rezultatas yra prastesnis nei pildant konteinerį dėžėmis, surūšiuotomis mažėjančia tvarka pagal dėžės tūrį.



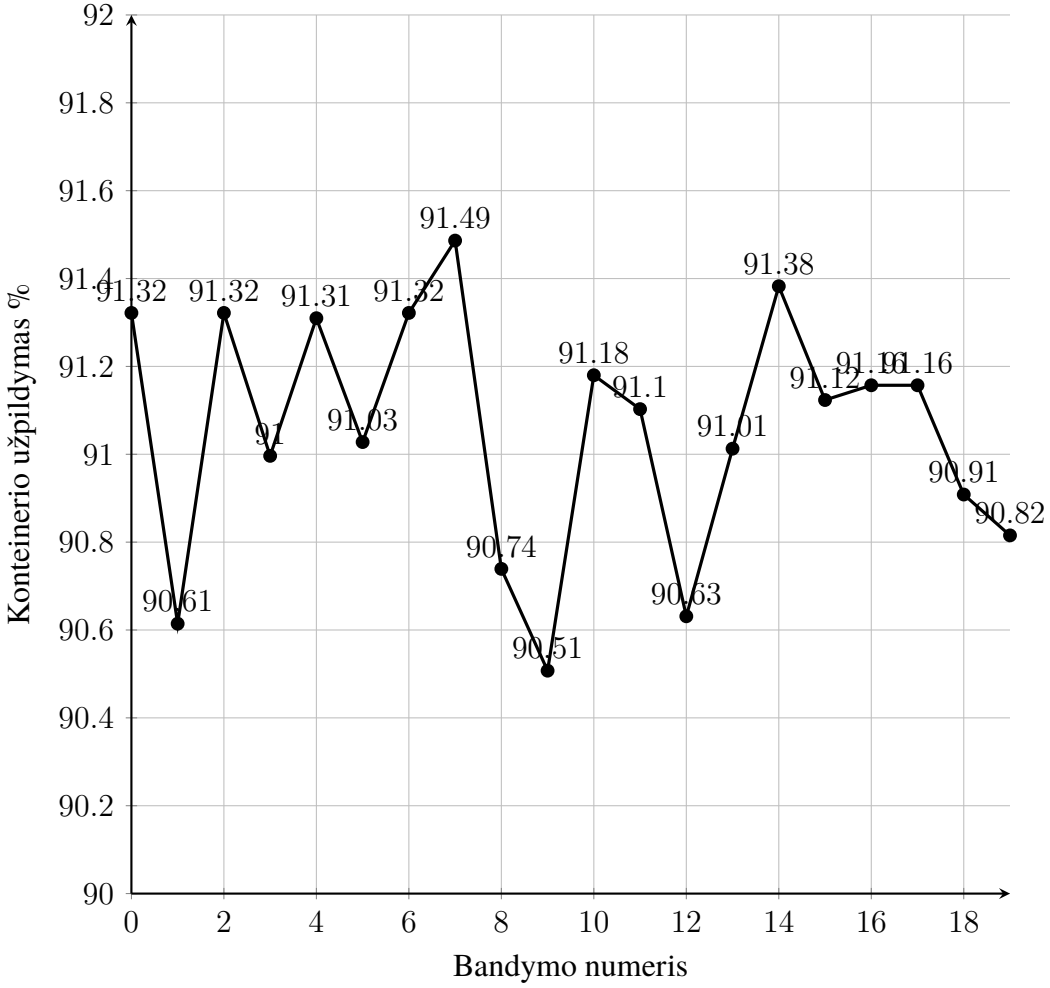
17 pav. Kroviniai talpinami į konteinerį naudojant skruzdžių paieškos algoritmo variantą, konteinerio užpildymo vidurkis 90,95%. Nors tai yra paprasčiausias skruzdžių kolonijų algoritmo variantas, tačiau jo rezultatai yra ženkliai geresni nei pildant konteinerį atsitiktiniu eiliškumu ir orientacijomis arba, surūšiuotomis dėžėmis su vienoda ar atsitiktinėmis orientacijomis.



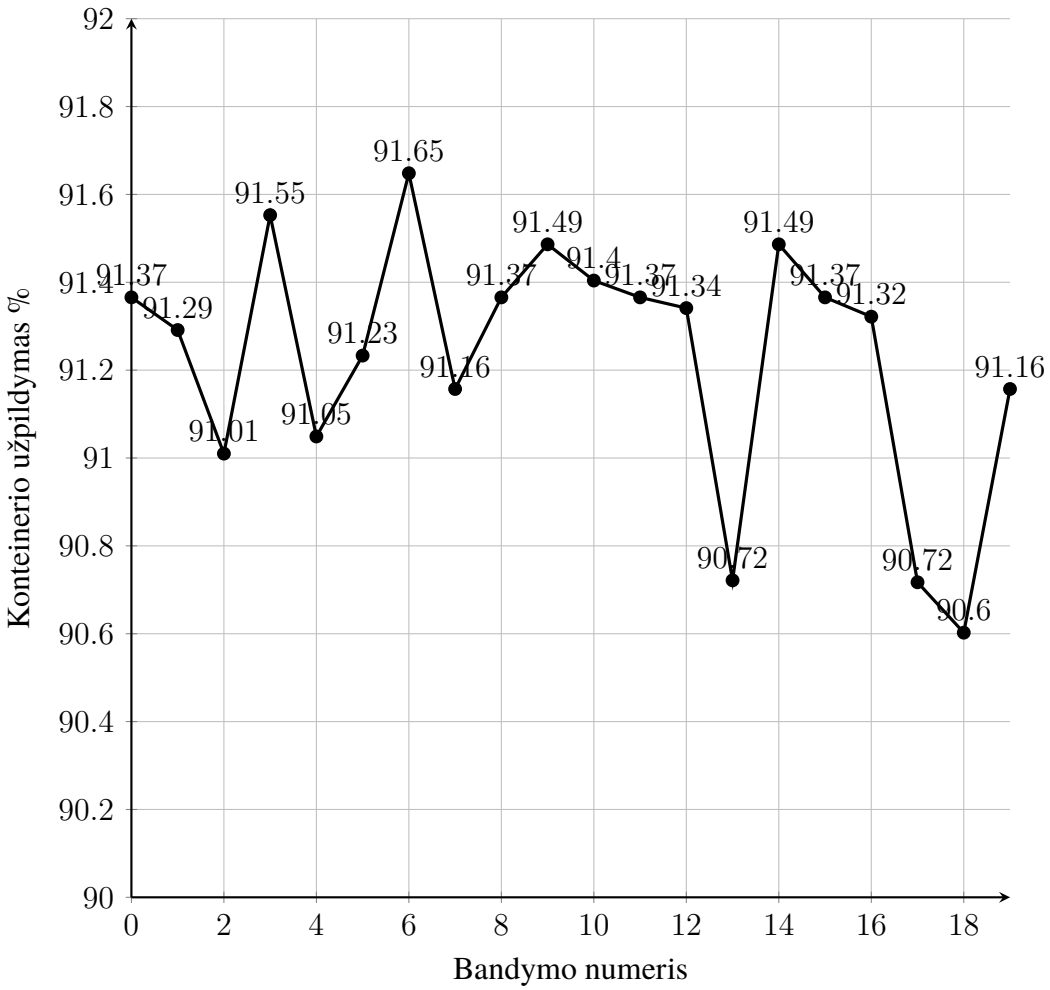
18 pav. Kroviniai talpinami į konteinerį naudojant elitinės strategijos algoritmo variantą, konteinerio užpildymo vidurkis 90.92%. Rezultato pagerinimo nepastebėta, elitinės strategijos algoritmo rezultatai yra panašūs į skruzdžių paieškos algoritmo rezultatus.



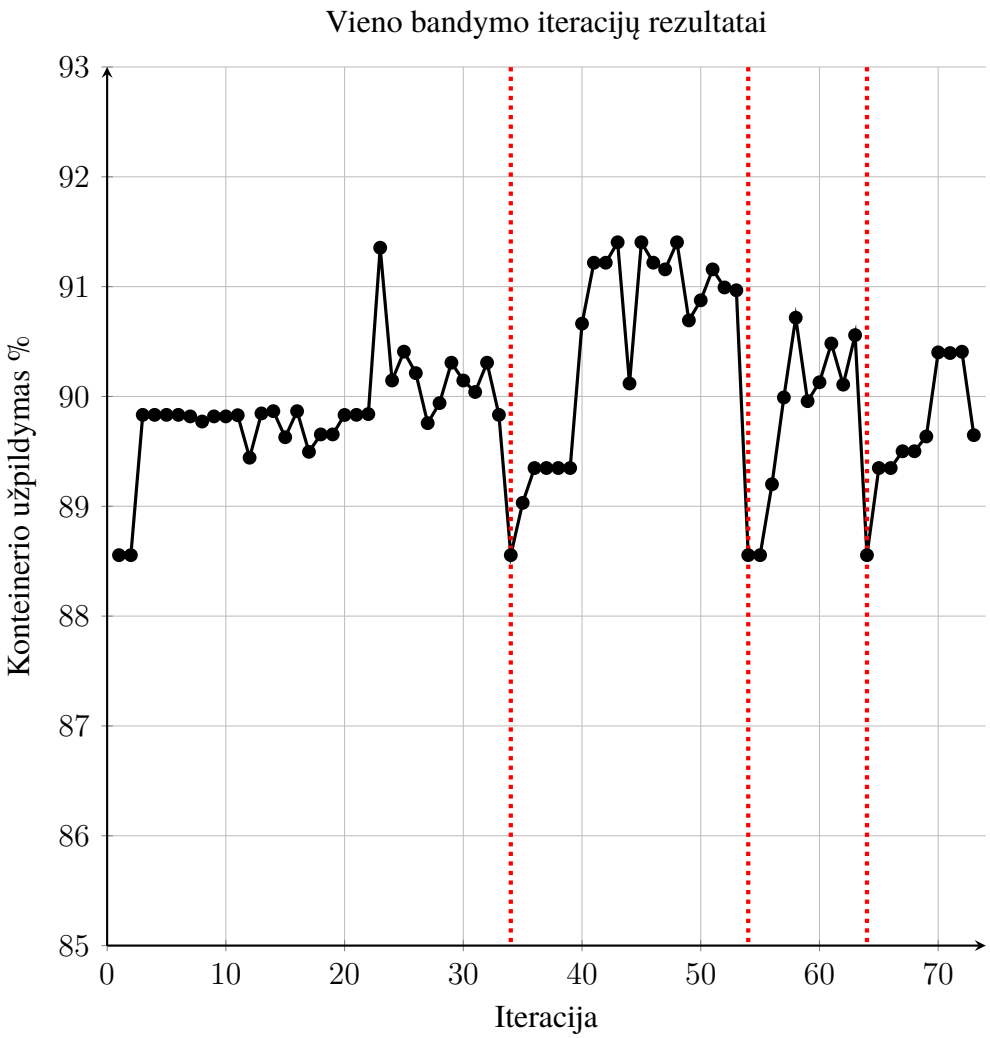
19 pav. Kroviniai talpinami į konteinerį naudojant skruzdžių kolonijos sistemos algoritmo variantą, konteinerio užpildymo vidurkis 90.84%. Rezultato pagerinimo nepastebėta, skruzdžių kolonijos sistemos algoritmo rezultatai yra panašūs į skruzdžių paieškos algoritmo rezultatus.



20 pav. Kroviniai talpinami į konteinerį naudojant Maks-Min skruzdžių sistemos algoritmo variantą, konteinerio užpildymo vidurkis 91.06%. Gautas geresnis rezultatas nei naudojant skruzdžių paieškos, elitinės strategijos ar skruzdžių kolonijos sistemos algoritmų variantus. Feromonų kiekio ribojimas padidina naujų kelių paieškos tikimybę.



21 pav. Kroviniai talpinami į konteinerį naudojant skruzdžių kolonijos algoritmą jūrinių konteinerių krovos optimizavimui su randomizuotu feromonų kiekio ribojimu ir elitine strategija, konteinerio užpildymo vidurkis 91.23%. Elitinės strategijos ir randomizuoto feromonų kiekio ribojimo strategija padidina naujų kelių paieškos tikimybę bei sumažina skruzdžių kelio stagnaciją, todėl gauti rezultatai yra geresni, lyginant su kitais skruzdžių kolonijų algoritmo variantais.



22 pav. Vieno iš bandymo iteracijų rezultatai, kai kroviniai talpinami į konteinerį naudojant skruzdžių kolonijos algoritmą jūrinių konteinerių krovos optimizavimui su randomizuotu feromonų kiekio ribojimu ir elitine strategija. Vertikalia punktyrine linija pažymėtos vietos kai feromonų kiekis grafe buvo inicializuojamas iš naujo atsitiktinai parinktomis pradinėmis ir maksimaliomis feromonų reikšmėmis. Kaip matyti grafike, iš naujo inicializuojant feromonų kiekius, pirmose iteracijose gaunamas mažas konteinerio užpildymas, tačiau greitai rezultatai pasiekia globaliai geriausią rezultatą ir, tam tikrais atvejais, jį pagerina.

12. Rezultatai ir išvados

Šiame darbe buvo sudarytas algoritmas jūrinių konteinerių krovos optimizavimo uždavinių sprendimui su randomizuotu feromonų kiekio ribojimu ir elitine strategija.

Darbo metu:

- Nustatyta algoritmo struktūra bei veiksmų eiliškumas,
- Atsakyta į klausimą kaip bus konstruojamas grafas,
- Kokiu principu bus parenkamas sekantis skruzdėlės žingsnis,
- Kokios priemonės bus naudojamos siekiant išvengti skruzdėlių kelio sąstingio,
- Kokia bus naudojama konteinerio pildymo euristika,
- Kaip bus ribojamas feromonų kiekis kelyje,
- Kaip bus mažinamas ir didinamas feromonų kiekis kelyje,
- Kokia bus algoritmo vykdymo pabaigos sąlyga,
- Atlikti algoritmo bandymai su testiniais duomenimis,
- Bandymų rezultatai palyginti su rezultatais, gautais naudojant kitus algoritmus.

Atlikus darbą buvo suformuluotos šios išvados:

- Algoritmo efektyvumas yra geresnis nei standartinių skruzdžių kolonijų optimizavimo algoritmų,
- Algoritmo efektyvumas yra geresnis nei naudojant modeliuojamo atkaitinimo algoritmą,
- Algoritmo efektyvumas yra panašus į genetinio algoritmo: geresnis už genetinį algoritmą su SLFH konteinerio pildymo euristika ir prastesnis nei genetiniame algoritme su MLFH konteinerio pildymo euristika,
- Atsižvelgiant į tai, algoritmas gali būti sėkmingai taikomas efektyviam jūrinių konteinerių krovos optimizavimo uždavinio sprendimui.

Šaltiniai

[22] Programos išeities tekstas. https://github.com/voidsky/aco4clp, 2022. Tikrinta 2022-05-11.

[BHS99] Bernd Bullnheimer, Richard Hartl **and** Christine Strauss. A new rank based version of the ant system - a computational study. *Central European Journal of Operations Research*, 7:25–38, 1999-01.

[BR03] Christian Blum **and** Andrea Roli. Metaheuristics in combinatorial optimization: overview and conceptual comparison. *ACM Comput. Surv.*, 35(3):268–308, 2003-09. ISSN: 0360-0300. DOI: 10.1145/937503.937505. URL: http://doi.acm.org/10.1145/937503.937505.

[CB98] P. Chu **and** J. A Beasley. A genetic algorithm for the multidimensional knapsack problem. *Journal of Heuristics*, 4:63–86, 1998.

[CCW+19] C. Chang, J. Cao, N. Weng **and** G. Lv. Ant colony optimization parameters control based on evolutionary strength. English. **in** *Proceedings - International Conference on Tools with Artificial Intelligence, ICTAI*, **volume** 2019-November, **pages** 448–455, 2019. URL: www.scopus.com.

[DD10] Türkay Dereli **and** Gülesin Sena Das. A hybrid simulated annealing algorithm for solving multi-objective container-loading problems. *Applied Artificial Intelligence*, 24(5):463–486, 2010.

[DD99] Marco Dorigo **and** Gianni Di Caro. The ant colony optimization meta-heuristic. *New Ideas in Optimization*:11–32, 1999-11.

[DG97] M. Dorigo **and** L. M. Gambardella. Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66, 1997.

[Dyc90] Harald Dyckhoff. A typology of cutting and packing problems. *European Journal of Operational Research*, 44:145–159, 1990-01. DOI: 10.1016/0377-2217(90)90350-K.

[DIR19] VĮ KLAIPĖDOS VALSTYBINIO JŪRŲ UOSTO DIREKCIJA. Vį klaipėdos valstybinio jūrų uosto direkcija, 2018 m. veiklos ataskaita. https://www.portofklaipeda.lt/veiklos-ataskaitos-2, 2019. tikrinta 2019-11-28.

[DMC96a] Marco Dorigo, Vittorio Maniezzo **and** Alberto Colorni. Ant system: optimization by a colony of cooperating agents. English. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 26(1):29–41, 1996. URL: www.scopus.com. Cited By :7804.

[DMC96b] Marco Dorigo, Vittorio Maniezzo **and** Alberto Colorni. Ant system: optimization by a colony of cooperating agents. ieee trans syst man cybernetics - part b. *IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics : a publication of the IEEE Systems, Man, and Cybernetics Society*, 26:29–41, 1996-02. DOI: 10.1109/3477.484436.

[Dor92] Marco Dorigo. Optimization, learning and natural algorithms. *PhD Thesis, Politecnico di Milano*, 1992.

[DS04] Marco Dorigo **and** Thomas Stützle. *Ant Colony Optimization*. Bradford Company, Scituate, MA, USA, 2004. ISBN: 0262042193.

[DS19] Marco Dorigo **and** Thomas Stützle. *Ant colony optimization: Overview and recent advances*. English, **volume** 272 **ofseries** *International Series in Operations Research and Management Science*. 2019, **pages** 311–351. URL: www.scopus.com. Cited By :78.

[GML14] Yanira González, Gara Miranda Valladares **and** Coromoto Leon. A multi-level filling heuristic for the multi-objective container loading problem. *Advances in Intelligent Systems and Computing*, 239:11–20, 2014-01. DOI: 10.1007/978-3-319-01854-6_2.

[YLM+12] Ching Nei Yap, Lai Soon Lee, Zanariah Abdul Majid **and** Hsin Vonn Seow. Ant colony optimization for container loading problem. **in** 2012.

[Ine04] Khaled Ghedira Ines Alaya Christine Solnon. Ant algorithm for the multidimensional knapsack problem. **in** *International conference on Bioinspired Methods and their Applications (BIOMA 2004)*, **pages** 63–72, 2004.

[KFR+10] Liangjun Ke, Zuren Feng, Zhigang Ren **and** Xiaoliang Wei. An ant colony optimization approach for the multidimensional knapsack problem. *J. Heuristics*, 16:65–83, 2010-02.

[KTK08] Min Kong, Peng Tian **and** Yucheng Kao. A new ant colony optimization algorithm for the multidimensional knapsack problem. *Computers and Operations Research*, 35(8):2672–2683, 2008.

[KZB13] L. Ke, Q. Zhang **and** R. Battiti. Moea/d-aco: a multiobjective evolutionary algorithm using decomposition and antcolony. *IEEE Transactions on Cybernetics*, 43(6):1845–1859, 2013.

[LM99] G. Leguizamon **and** Z. Michalewicz. A new version of ant system for subset problems. **in** *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, **volume** 2, 1459–1464 Vol. 2, 1999.

[Pis02] David Pisinger. Heuristic for the container loading problem. *European Journal of Operational Research*, 141:382–392, 2002-09. DOI: 10.1016/S0377-2217(02)00132-7.

[SH97] Thomas Stützle **and** Holger Hoos. Max-min ant system and local search for the traveling salesman problem. **in** *IEEE INTERNATIONAL CONFERENCE ON EVOLUTIONARY COMPUTATION (ICEC'97)*, **pages** 309–314. IEEE Press, 1997.

[UNC19] UNCTAD. United nations conference on trade and development, 2019 report. `https://unctadstat.unctad.org/wds/TableViewer/tableView.aspx?ReportId=13321`, 2019. tikrinta 2019-11-28.

[VH01] Michel Vasquez **and** Jin-Kao Hao. A hybrid approach for the 0-1 multidimensional knapsack problem. *IJCAI International Joint Conference on Artificial Intelligence*, 2001-05.

[XZM+13] Zhaojie Xue, Canrong Zhang, Lixin Miao **and** Wei Hua Lin. An ant colony algorithm for yard truck scheduling and yard location assignment problems with precedence constraints. English (US). *Journal of Systems Science and Systems Engineering*, 22(1):21–37, 2013. ISSN: 1004-3756. DOI: 10.1007/s11518-013-5210-0.

[ZD11] D. Zhang **and** L. Du. Hybrid ant colony optimization based on genetic algorithm for container loading problem. **in** *2011 International Conference of Soft Computing and Pattern Recognition (SoCPaR)*, **pages** 10–14, 2011.

[ZZZ06] P. Zhao, P. Zhao **and** X. Zhang. A new ant colony optimization for the knapsack problem. **in** *2006 7th International Conference on Computer-Aided Industrial Design and Conceptual Design*, **pages** 1–3, 2006.