

**KLAIPĖDOS UNIVERSITETAS**

Jūrų technikos fakultetas

Informatikos inžinerijos katedra

**DIRBTINIO INTELEKTO TAIKYMAS  
GAMYBOS UŽDUOČIŲ SEKOS  
OPTIMIZAVIMO METODO KŪRIMUI**

**DEVELOPMENT OF METHOD FOR  
MANUFACTURING ORDER SEQUENCE  
OPTIMIZATION APPLYING ARTIFICIAL  
INTELLIGENCE**

Informatikos inžinerijos specialybė

magistro baigiamasis darbas

Autorius

TISI-11 gr. Ilja Gusiatin

Vadovas

prof. dr. Arūnas Andziulis

Klaipėda, 2013

## **ANOTACIJA**

Šiuolaikiniame gamybos valdymo procese labai svarbu atsižvelgti į gamyklinių užduočių planavimą, užtikrinant jų vykdymo sekos efektyvumą. Darbe yra ieškomas optimalus sprendimas analizuojamai gamybos užduočių sekos optimizavimui, buvo pasitelkus klasterių analizę ir dirbtinio intelekto technologijas. Kad pasiekti didesnę gamybos efektyvumą, užduočių seka yra optimizuojama, mažinant staklių prastovos laiką. Aprašoma realioje gamyboje (laidų rinkinių gamybos įmonė „Yazaki Wiring Technologies“) užduočių sekai optimizuoti naudojama programinė įranga, jos veikimo algoritmas ir efektyvumas, bei tobulinimo galimybės, taikant dirbtinio intelekto agentinių sistemų technologijas.

**PAGRINDINIAI ŽODŽIAI:** gamybos proceso optimizavimas, optimizavimo metodas, klasterių analizė, agentinė sistema.

## **ABSTRACT**

It is important to consider production tasks scheduling in modern manufacturing control process, to insure manufacturing process flow effectiveness. In this paper author looks for optimal approach for analyzed production tasks scheduling optimization, using cluster analysis techniques and artificial intelligence technologies. To achieve bigger production effectiveness, task schedule is being optimized, minimizing machine stoppage time. Paper describes software, which is used in real production process (Yazaki Wiring Technologies manufacturing company) to optimize production task schedule; for this purpose used algorithm and its effectiveness, its improving possibilities applying artificial intelligence technologies.

**KEYWORDS:** manufacturing process optimization, optimization method, cluster analysis, agent system.

## **SUMMARY**

Final thesis deals with a problem of optimization in real company “Yazaki Wiring Technologies”, working in wire harness manufacturing area. During the practice, fact of production planning algorithm improvement possibility was noticed. In analyzed area, production planning consists of making proper production process (wire cutting and terminal crimping) scheduling. Production order, in fact, is wire with terminals on its end should be cut. For each different terminal machine must be configured physically. Physical machine configuration cannot be done during cutting process, so production must be stopped for some time. So the main idea of final thesis is to create a propose algorithm, which can improve already existing optimization algorithm, to reach lesser production stoppage time.

First of all, it was decided to group similar orders in clusters and then to analyze these groups as one object. That leads to optimizing objects quantity minimization, what lead to optimization time minimizing. Also, further clustering let to know similarity of production orders by creating orders dendrogram.

Second, three-level intelligent agent system was proposed to make final optimization steps, turning created dendrogram into optimized flow of orders – production plan with as small as possible machine stoppage time. Every agent is responsible for dendrogram optimization on his level. Agents communicate to each other to reach better performance.

The experimental results showed, that newly created algorithm is better than currently existing one, which is actively used in all Yazaki Wiring Technologies factories. But, additional investigation and experiments are required to improve proposed algorithm for better efficiency.

## **ACKNOWLEDGEMENT**

I highly appreciate Yazaki Wiring Technology Lietuva for a given possibility to work and make practical experiments in production planning and control field, providing all necessary practical and theoretical information. My direct practice manager, actually production department manager, was always very helpful and supportive, answering my question regarding unclear procedures. Also, want to thanks all staff starting machine operators up to IT department manager for help in making experiments with hardware and software as well as clarifying specific details in required area. During practice, I had access to one of biggest ERP system SAP, well-known in whole world, given chance to improve my IT knowledge, improve my time scheduling and people management skills. This practice gave me an idea and possibility to write my final thesis.

## SANTRUPŲ IR TERMINŲ ŽODINĖLIS

CAO – (angl. *Cutting Area Optimization*) pagalbinė gamybos valdymo programinė įranga, skirta gamybos užduocių optimizavimui, bei kitų gamybos procesų palaikymui.

DB – duomenų bazė, organizuotas (susistemintas, metodiškai sutvarkytas) duomenų rinkinys.

UML – (angl. *Unified Modeling Language*) modeliavimo ir specifikacijų kūrimo kalba, skirta specifiikuoti, atvaizduoti ir konstruoti objektiškai orientuotų programų dokumentus.

ERP – ( angl. *Enterprise Resource Planning*) verslo valdymo sistema, padedančia planuoti įmonės resursus.

## TURINYS

ILIISTRACIJŲ SĄRAŠAS .....	7
LENTELIŲ SĄRAŠAS .....	8
ĮVADAS .....	9
1. ANALITINĖ DALIS .....	10
1.1. Optimizavimo uždavinių sprendimų tyrimas .....	10
1.2. CAO programinės įrangos apžvalga .....	16
1.3. Programinių įrankių apžvalga .....	18
1.3.1. Embarcadero RAD Studio XE programinis paketas .....	18
1.3.2. Microsoft Visio 2010 programinis paketas .....	20
2. METODINĖ DALIS .....	22
2.1. Klasterių analizė .....	22
2.1.1. Klasterių analizės metodai .....	23
2.2. Intelektualūs agentai .....	24
2.3. Duomenų struktūros C++ aplinkoje .....	25
2.3.1. Dinaminiai sąrašai .....	26
2.3.2. Sąrašų rikiavimas .....	29
2.4. Optimizacijos metodai ir technikos .....	30
2.4.1. Arčiausio kaimyno metodas .....	31
2.4.2. Atkarpų apkeitimo metodas .....	31
2.4.3. Perrinkimo metodas .....	32
3. PRAKTINĖ DALIS .....	33
3.1. Statistiniai duomenys .....	33
3.2. Pagalbinis įrankis .....	34
3.3. Dendrogramos kūrimas .....	35
3.4. Dendrogramos optimizavimo agentinė sistema .....	38
3.4.1. Ketvirtojo lygio agentas .....	38
3.4.2. Trečiojo lygio agentas .....	40
3.4.3. Antrojo lygio agentas .....	41
3.4.4. Antro lygio agento realizavimas .....	42
3.5. Optimizacijos rezultatai .....	43
3.6. Tolimesni tyrimai .....	44
IŠVADOS IR REKOMENDACIJOS .....	45
LITERATŪROS SĄRAŠAS .....	46

PRIEDAI .....	48
---------------	----

## ILIUSTRACIJŲ SĄRAŠAS

<b>1 pav.</b> Optimizavimo metodas, pagrįstas simuliacija [1] .....	10
<b>2 pav.</b> Dviejų planavimo metodų kombinacija, sprendžiant didelio sudėtingumo optimizacijos problema [1] .....	11
<b>3 pav.</b> Ganto diagrama optimizuotam gamybos procesui [1] .....	12
<b>4 pav.</b> Gamybos sistemos procesų atvaizdavimas [3] .....	14
<b>5 pav.</b> CAO programos pagrindinis langas .....	16
<b>6 pav.</b> Naujai sukurtos užduotys .....	17
<b>7 pav.</b> Optimizuota užduočių seka .....	18
<b>8 pav.</b> C++ Builder XE darbalaukis .....	19
<b>9 pav.</b> Microsoft Visio 2010 Premium darbalaukis .....	21
<b>10 pav.</b> Laidų karpymo abstraktus užsakymas.....	22
<b>11 pav.</b> K-vidurkių klasterizavimo algoritmas [10] .....	23
<b>12 pav.</b> Dendrogramos pavyzdys .....	24
<b>13 pav.</b> Dinaminio sąrašo tvarkymo žingsniai [12] .....	27
<b>14 pav.</b> Tiesinis sąrašas [12] .....	29
<b>15 pav.</b> Atstumų matrica .....	31
<b>16 pav.</b> Eksperimento metu naudojamas įrankis paskaičiuoti staklių derinimo laiką.....	35
<b>17 pav.</b> Nulinio lygio klasterizavimas.....	36
<b>18 pav.</b> Pirmojo lygio klasterizavimas .....	36
<b>19 pav.</b> Antro lygio grupė .....	36
<b>20 pav.</b> Trečio lygio grupė .....	37
<b>21 pav.</b> Dendrogramos sudarymo procesas.....	37
<b>22 pav.</b> Agentinė sistema .....	38
<b>23 pav.</b> Ketvirto lygio agento proceso algoritmas .....	39
<b>24 pav.</b> Antro lygio agento pasiūlymas.....	41

## LENTELIŲ SĄRAŠAS

<b>1 lentelė.</b> Atkarpų apkeitimo metodas.....	31
<b>2 lentelė.</b> Užsakymų paskirstymas palei laidą .....	33
<b>3 lentelė.</b> Kontaktų pasikartojamumas .....	34
<b>4 lentelė.</b> Staklių derinimo laikai .....	34



## IVADAS

Procesų valdymas šiuolaikinėse sistemose užima bene svarbiausią poziciją. Pasitelkus šiuolaikines IT technologijas galima užtikrinti efektyvų sistemų valdymą, leidžiantį pasiekti gamybos linijų, transporto srautų, logistikos terminalų ir kitų procesų aukštesnio produktyvumo.

Darbe nagrinėjama gamybos valdymo sistemos užduočių sekos optimizavimo programos efektyvumo tobulinimo problematika. Keičiant gamybinę užduotį, automatinėje kontaktų tvirtinimo mašinoje turi būti atlikti mechaniniai pakeitimai, reikalaujantys operatoriaus darbo. Kai mašina yra pertvarkoma, ji neatlieka savo tiesioginės paskirties (laidų pjovimas ir kontaktų tvirtinimas) yra prarandamas laikas. Užduočių sekos optimizavimo tikslas – sudėlioti užduotis taip, kad mašinos prastovos laikas tarp skirtingų užduočių būtų mažesnis. Yra labai aktualu tobulinti optimizacijos rezultatus, kuriu dėka bus sumažintas netiesioginis gamybos įrenginių darbo laikas, operatorių fizinis darbas, bet padidintas gamybos efektyvumas.

### ***Tyrimo objektas.***

Optimizavimo metodo, skirto gamybos užduočių sekos optimizavimui, netiesioginio gamybos laiko (automatinių mašinų prastovų) mažinti, remiantis klasterio analizių bei intelektualių agentinių sistemų technologijomis, kūrimas.

### ***Problematika.***

Šiuo metu CAO programos naudojamas optimizavimo metodas, veikiantis arčiausio kaimyno metodo pagrindu, yra ne labai efektyvus sudarinėjant gamybos užduočių tvarkaraštį. Darbe yra nagrinėjami metodo trūkumai, bet siūlomi variantai jų eliminavimui.

### ***Darbo tikslas.***

Sukūrus gamybos užduočių hierarchinę struktūrą, pasitelkiant klasterio analizės įrankiais, pateikti optimizavimo metodą, skirtą realios gamybos valdymo sistemos tobulinimui.

### ***Tikslui pasiekti keliami uždaviniai:***

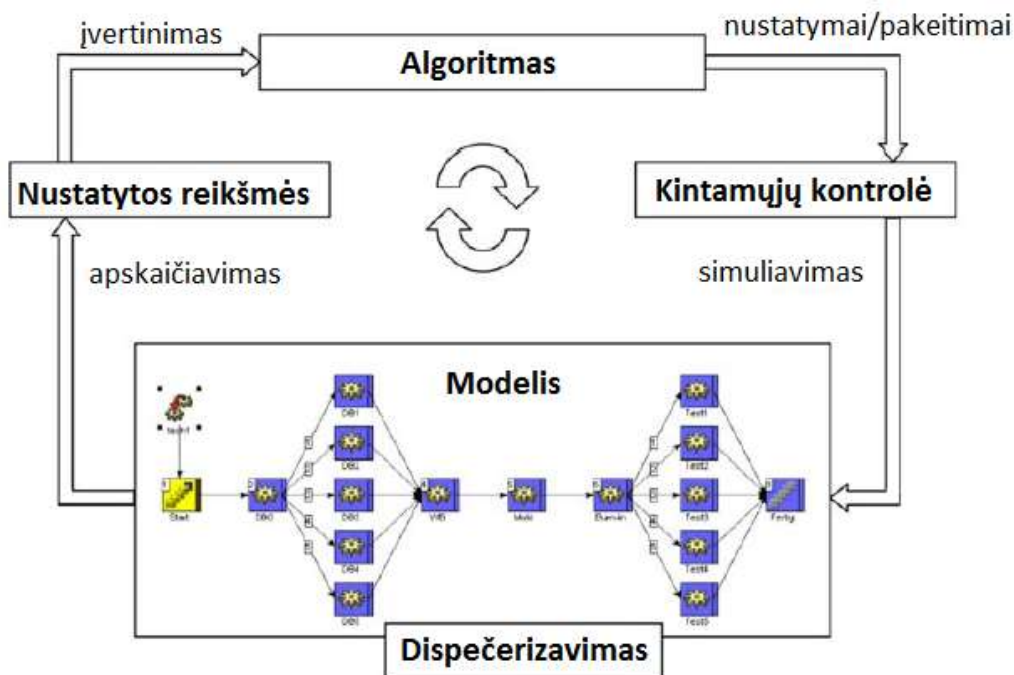
- 1) atlikti naujausių mokslinių pasiekimų analizę gamybos valdymo procesų optimizavimo srityje;
- 2) išanalizuoti gamybos valdymo programinės įrangos CAO pagrindinius veikimo principus, realiai naudojamos Yazaki Wiring Technologies įmonėje;
- 3) remiantis atlikta panašių gamybos valdymo procesų optimizavimo analize, pateikti naują patobulintą optimizavimo metodą, pagrįstą dirbtinio intelekto technologijomis.

## 1. ANALITINĖ DALIS

### 1.1. Optimizavimo uždavinių sprendimų tyrimas

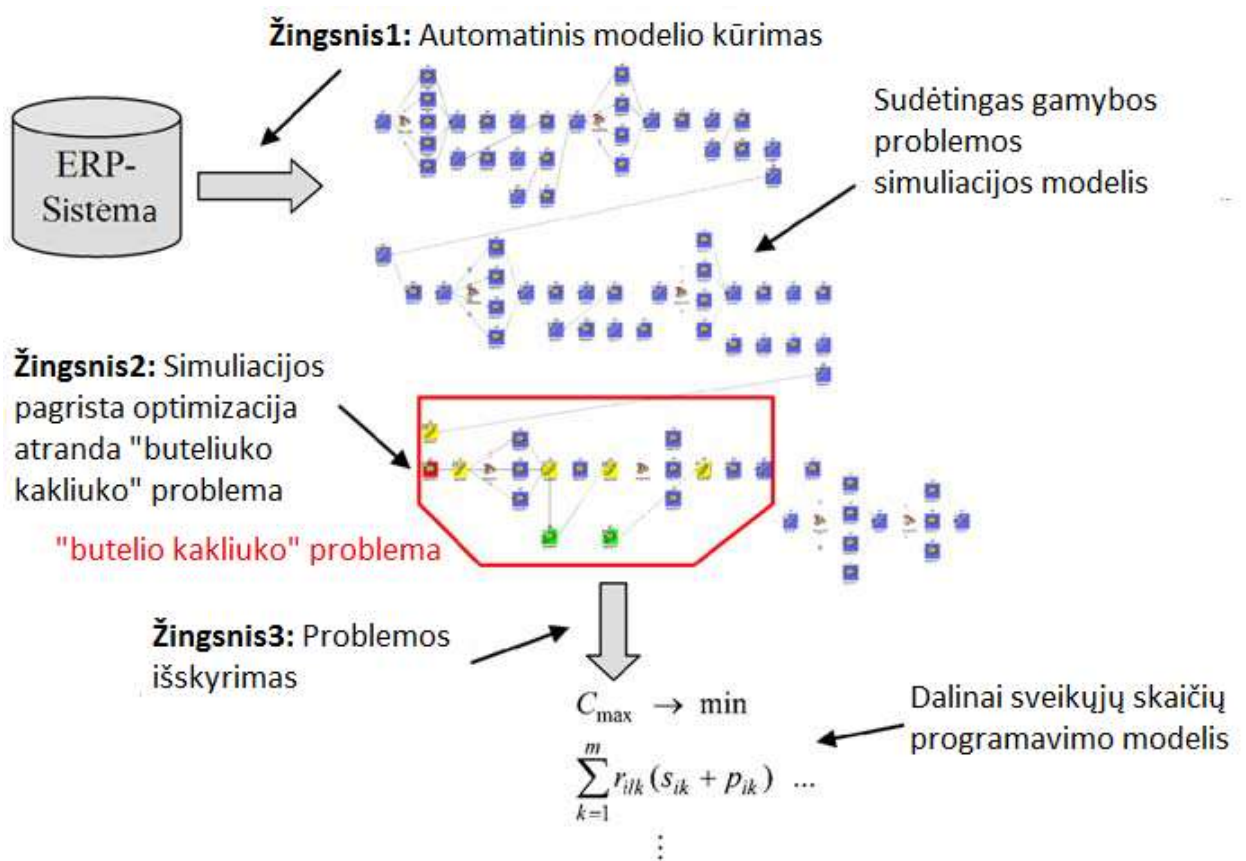
Gamybos procesų optimizavimas yra vieną aktualiausių temų šiuolaikiniame pasaulyje. Skirtingi metodai yra kuriami ir tiriama, kol kas, deja, geriausio rasti nepavyksta. Šios temos problematiką nagrinėja tokie mokslininkai, kaip A. Klemmt, C. Bohle, N. Boland [1, 3-5, 7].

Su išaugusia šiuolaikinių kompiuterinių procesorių galia, planavimo optimizacijos problemų sprendimo metodu svarbumas tik didėja, teigia A. Klemmt ir autoriai [1]. Savo darbe jie apžvelgia du skirtingus metodu planavimo optimizavimui. Dalinai sveikųjų skaičių programavimas ir alternatyvi jam, euristiniu metodu, skirstomu pagal A. Gupta ir A. Savikumar į dispečerizavimo, euristinės paieškos ir dirbtinio intelekto technikas [2], kombinacija sudaryta iš dispečerizavimo ir paieškos technikų, vadinamas „optimizacija simuliacijos pagrindu“. Šitas metodas (1 pav.) dėl savo lankstumo, tampa vos svarbesnis pastaraisiais metais. Čia gamybos problemos, vietoj formulių ir grafų, yra apibrėžtos kaip objektų grindžiamas simuliacijos modelis. Optimizacija simuliacijos pagrindu siūlo apytikslį sprendimą, kuris yra gana lankstus bei aukšto tikslumo modelį. Deja, šitas metodas turi savo neigiamųjų savybių – nėra jokių garantijų, jog bus pasiektas optimumas ar netgi žemiausios nustatytos ribos. Taip pat cikliniai sprendimai yra labai laiko-reikalaujantis, ypač sudėtingu simuliacijos modelių atveju. Dėl to laiko ribos yra dažniausia nustatomos šitam optimizacijos procesui.



1 pav. Optimizavimo metodas, pagristas simuliacija [1]

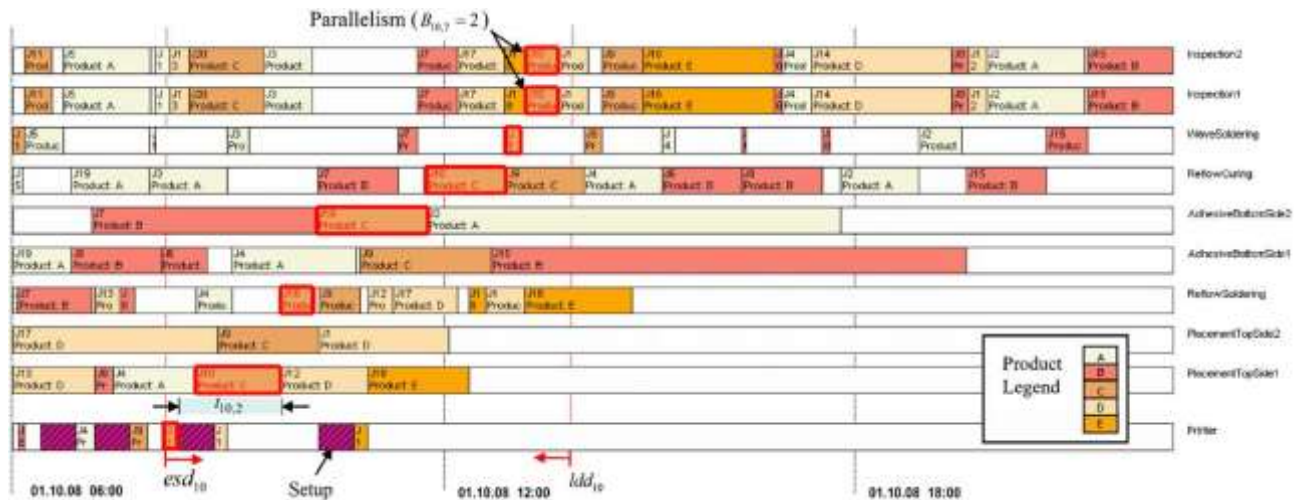
Lyginant du pateiktus metodus, yra prieinama išvados, jog dalinai sveikųjų skaičių programavimas, kuris išduota tikslų sprendimą, yra efektyviausia taikyti mažų problemų atvejais. Sudėtingos gamybos planavimo problemos gali būti efektyviai sprendžiamos simuliacijos pagrindu optimizavimu. Taip pat išskirtinio sudėtingumo lygmens (didesnis uždavinių, mašinų skaičius) gali būti taikomas abiejų metodų kombinacija. Iš pradžių, simuliacijos modelis yra automatiškai sudaromas iš gamybos ERP sistemos. Antrame žingsnyje, gautas sudėtingas modelis yra euristiškai optimizuojamas palei simuliacijos pagrindu metodą. Gamybos proceso „buteliuko kakliuko“ problema yra randama. Čia, skirtingos charakteristikos, tokios kaip mašinos utilizavimas ar didelės atsargos gali būti „butelio kakliuko“ problemos indikatoriais. Pagrindė, dėl trumpo planavimo horizontų, pamainų problemos gali būti nepaisomos. Dabar, simuliacijos modelis, kuris atvaizduoja „buteliuko kakliuko“ problemą, gali būti išskirtas. Problemos sudėtingumas „buteliuko kakliuko“ modelyje yra žymiai mažesnis negu bendros problemos sudėtingumas. To gana, kad šitai problemai spręsti, gali būti taikomas tam tikras matematinis modelis, pavyzdžiui dalinai sveikųjų skaičių programavimas (2 pav.).



**2 pav.** Dviejų planavimo metodų kombinacija, sprendžiant didelio sudėtingumo optimizacijos problema [1]

Beje, automatiškai yra gaunamas ne tik dalinai sveikųjų skaičių programavimo modelis, bet ir kita informacija tolimesniam jo vystymui. Pavyzdžiui, yra gaunamos viršutinė ir apatinė tikslo funkcijos ribos, apatinė laiko pradžios riba ir pradinis įmanomas sprendimas. Visa papildoma

informacija žymiai riboja matematinio modelio paieškos aprėptį, kuo pasėkoje greitesnė dalinai sveikųjų skaičių programavimo sprendiklio konvergencija. Gautas sprendimas gražinamas į simuliacijos modelį. Dabar egzistuoja globalus simuliacijos modelis su optimizuota „buteliuko kakliuko“ problemiška vieta. Visos simuliacijos modelio ypatybės, tokie kaip pavyzdžiui Ganto diagramos (3 pav.) arba ataskaitos, gali būti išgauti iš optimizuoto gamybos proceso.



**3 pav.** Ganto diagrama optimizuotam gamybos procesui [1]

Skrudėlyno kolonijos metodo tema vysto A. Berrichi ir kt [3]. Gamybos ir techninio aptarnavimo planavimo optimizavimo problemoms spręsti, užtikrinant geriausią sanglaudą tarp gamybos bei techninio aptarnavimo. Pagrindinė problema yra ta, jog tarp dviejų vienodo svarbumo procesų yra konfliktų, ir yra ieškoma sprendimo, tinkamo abiem pusėms. Nagrinėjamas straipsnis remiami ankstesniu autorių darbų optimizacijos srityje – integruotų dviejų tikslų modelių lygiagrečių mašinų procesų konfliktų spręsti. Yra sukurtas algoritmas, pavadintas PACO (Pareto Ant Colony Optimization), daugiatakslio skrudėlių kolonijos metodo pagrindu. Gautas rezultatas parodo, jog sukurtas algoritmas yra geresnis nei visuotinai žinomi daugiataksliai genetiniai algoritmai SPEA 2 ir NSGA II.

Iš pradžių, autoriai nagrinėja lygiagrečius gamybos planavimo ir techninio aptarnavimo procesus atskirai, o paskui yra siūlomas dviejų tikslų modelis. Gamybos procesas yra įsivaizduojamas kaip paprasto lygiagrečiai veikiančių mašinų procesas ir jos efektyvumo matas yra produktyvumas per tam tikrą laiko tarpą. Techninio aptarnavimo procesu yra pasirinktas sisteminis profilaktinis techninis aptarnavimas. Gamybos staklių efektyvumas yra apibrėžiamas kaip tikimybė, kad sistema atlieka savo tiesioginę funkciją tam tikru laiku  $t$  neapibrėžtą laiko tarpą.

Mašinos  $M_i$  efektyvumas yra apibrėžiamas kaip:

$$A_i(t) = P(M_i \text{ is operating at time } t)$$

O prastovos laiko tikimybė kaip:

$$\bar{A}_i(t) = 1 - A_i(t)$$

Autoriai priima mašinos  $M_i$  prastovos laiką (skaityk, remonto, arba techninio aptarnavimo), kaip atsitiktinį kintamąjį su eksponentiškai pasiskirsčiusia atsiradimo tikimybe. Taip pat priimtas kad po techninio aptarnavimo veiksmai sugrąžina mašiną į būsenos „gera, kaip nauja“ sąlygas.

Atsižvelgiant į visas priimtas nuostatas, pradinės mašinos  $M_i$  būsenos efektyvumas yra išreiškiamas formulę:

$$A_i(t) = \frac{\mu_i}{\lambda_i + \mu_i} + \frac{\lambda_i}{\lambda_i + \mu_i} \exp[-(\lambda_i + \mu_i)t]$$

kur  $\mu_i$  – remonto dažnis,

o  $\lambda_i$  – gedimų dažnis.

Efektyvumas  $A_i(t)$  laikui bėgant yra mažėjanti funkcija, kai prastovos laiko funkcija – didėjanti. Jeigu profilaktinė techninės apžiūros veiksmai nėra atliekami ant mašinos – jos sulaužymo tikimybė didėja.  $T_i$  yra laikas, kaip techninės apžiūros veiksmas baigėsi, taigi mašinos efektyvumo funkcija kai  $t > T_i$  yra:

$$A_i(t) = \frac{\mu_i}{\lambda_i + \mu_i} + \frac{\lambda_i}{\lambda_i + \mu_i} \exp[-(\lambda_i + \mu_i)(t - T_i)]$$

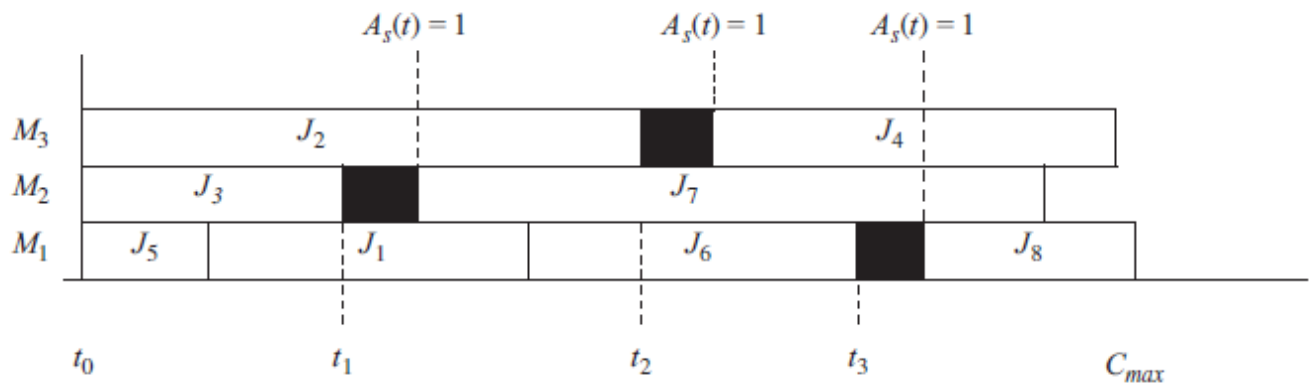
Bendras gamybos sistemos efektyvumas priklauso nuo struktūros (nuosekli, lygiagreti, hibridinė) ir nuo komponentų charakteristikų. Sistemai (4 pav.), susidedančiai iš  $m$  mašinų, pasiekiamumas yra apibrėžiamas tokią funkciją:

$$A_s(t) = 1 - \prod_{i=1}^{i=m} [1 - A_i(t)]$$

O prastova:

$$\bar{A}_s(t) = 1 - A_s(t) = \prod_{i=1}^{i=m} [1 - A_i(t)]$$

Integruotas modelis turi rūpintis kad du tikslai būtų optimizuojami sinchroniškai: gamybos produktyvumas didėtų ir mažėtų prastovų laikas dėl remonto darbų. Tam yra taikomas skruzdėlynų kolonijos metodas (Ant Colony Method) – vienas iš efektyvių algoritmų apytikslių sprendimų radimui, kur skruzdėlės (paprastieji agentai) bendradarbiauja kad rastų geriausią sprendimą sudėtinguose kombinatoriniuose atvejuose.



**4 pav.** Gamybos sistemos procesų atvaizdavimas [3]

Sukurtas PACO metodas yra aprobuotas ant sugalvotos gamybinės situacijos, beje palyginamas su kitais optimizacijos uždavinius spręsti taikomais algoritmais, SPEA 2 ir NSGA II. Gauti rezultatai (matuojami sistemos produktyvumu) aiškiai demonstruoja, kad autoriui siūlomas metodai yra geresnis negu žinomi algoritmai.

Gamybos planavimo optimizavimas yra aktualus kalnų atvirų šachtų iškasenų gavyboje. N. Boland ir kt. analizuoja planavimo problemas šioje srityje, sudedančių iš rūdos gabalų, kurie yra apibrėžiami kaip blokai, transportavimo iš šachtos tinkamos sekos radimo [4]. Praktikoje, dėl didelio blokų skaičiaus dažniausia jie yra apjungiami į didesnes planavimo formos vienetus. Autorių tikslas yra išspręsti atvirų šachtų iškasenų gamybos planavimo problemą formuluojama kaip dalinai sveikųjų skaičių programavimo problemą. Todėl dideli apjungti vienetai bus naudojami iškasenų gamybos proceso planavimui, o individualūs blokai – tarpiniams proceso sprendimams.

Yra siūlomas pasikartojamo išagregavimo metodas, kuris patobulina apjungtus blokus (atsižvelgiant į apdirbimą) iki tokio taško, kur patobulintiems apjungtiems blokams tinka tas pats optimalus sprendinys, kaip ir individualiems blokams. Taip pat yra siūlomas keletas patobulintų apjungtų blokų kūrimo strategijų dalinai sveikųjų skaičių programavimo procese, naudodami dualizmo rezultatus ir problemos struktūrą. Šitie patobulinti apjungti blokai leidžia kokybiškai išspęsti daug didelių problemų racionaliame laike.

C. Bohle ir bendraautoriai aprašo optimizavimo modelį vyno gamybos srityje – vynuogių surinkimo planavime [5]. Deja, neatsėjami neaiškumai agrokultūros srityje apsunkina planavimo optimizacijos taikymą. Tik neseniai atsirado patikima optimizacijos metodologija leidžianti efektyviai susidoroti su neaiškumais optimizacijos modeliuose. Savo straipsnyje autoriai tyria, kaip patikima optimizacija gali būti efektyvi praktikoje. Yra sukuriami ir pateikiami keli alternatyvus optimizacijos modeliai.

C. Bohle ir kt. teigia, kad optimizacijos modeliai yra daug metų plačiai taikomi įvairiuose srityse, daugiausia gamybos planavimo srityje. Tačiau optimizacijos modelių taikymas

agrokultūroje nebuvo toks sėkmingas. Savo ankstesniame straipsnyje [6] autoriai pristato modelį, skirtą padėti planuoti derliaus nuėmimo operacijas vynuogynuose ir darbo jėgos priskirimo prie operacijų. Šita problema yra ypač aktuali, net vynuogių derliaus nuėmimo laikas tiesiogiai įtakoja vyno kokybę. Aprašytas modelis atsižvelgia ne tik į tiesiogiai susietas išlaidas procese, bet ir į kokybės pablogėjimą dėl derliaus nuėmimo anksčiau arba vėliau vynuogių brandos laiko. Modelis balansuoja tarp vykdymo operacijos bei geros kokybės kainos, priklausomai nuo laukiamos vyno kokybės.

Dėl aukščiau aprašytų optimizacijos apsunkinimo priežasčių, autoriai savo naujame darbe [5] pasiūlo metodologiją, leidžianti susidoroti su neaiškumais. Vienas būdas vengti juos yra tiesiog ignoruoti juos, sprendžiant problemą naudojant vidutines arba daugiausia tikėtinas reikšmes. Kitas būdas – išreikšti modelio neaiškumus kaip problema ir apsvarstyti šiuos elementus kaip apribotus kartu su tikslo funkcija. Šitas sprendinys adresuotas stochastiniam programavimui. Deja, nevisos programos gali susidoroti su tokiu išreiškimu.

C. Bohle ir kt. siūlomas būdas – taikyti patikimą optimizaciją. Patikima optimizacija – tai sprendimo būdas, kuris yra nejautrus duomenų sutrikimams, bent jau tam tikrose ribose. Patikimas sprendimas yra galimas skirtingiems duomenų rinkiniams, tačiau nebūtinai yra optimalus konkrečiai kažkuriam vienam. Patikima optimizacija bando gauti įmanomus sprendimus visam diapazonui duomenų rinkinių su neaiškiais parametrais, subalansuotu ir kontroliuojamu būdu optimizuojant tikslo funkciją, atsižvelgiant į parametrų neaiškumus.

Gamybos planavimo optimizavimo problemas nagrinėja ir Lietuvos mokslininkai. A. Andziulis ir kt. teigia, jog gamybos planavimo problemas traukia kombinatorinės optimizacijos ir optimizacijos programinės įrangos kūrimo specialistų dėmesį [7]. Autoriai palygina du plačiai naudojamus euristicinius metodus, taikomus gamybos planavimo problemoms spręsti, optimizavimo būdus: arčiausio kaimyno metodą ir skruzdėlių kolonijos optimizavimo metodą. Pateikti metodai yra patikrinami ant specifinės realios problemos, kuri priklauso asimetriškai keliaujančio pirklio problemų klasei, kuri yra žinoma kaip sudėtinga problema, kuriai iki šiol nėra efektyvaus sprendimo.

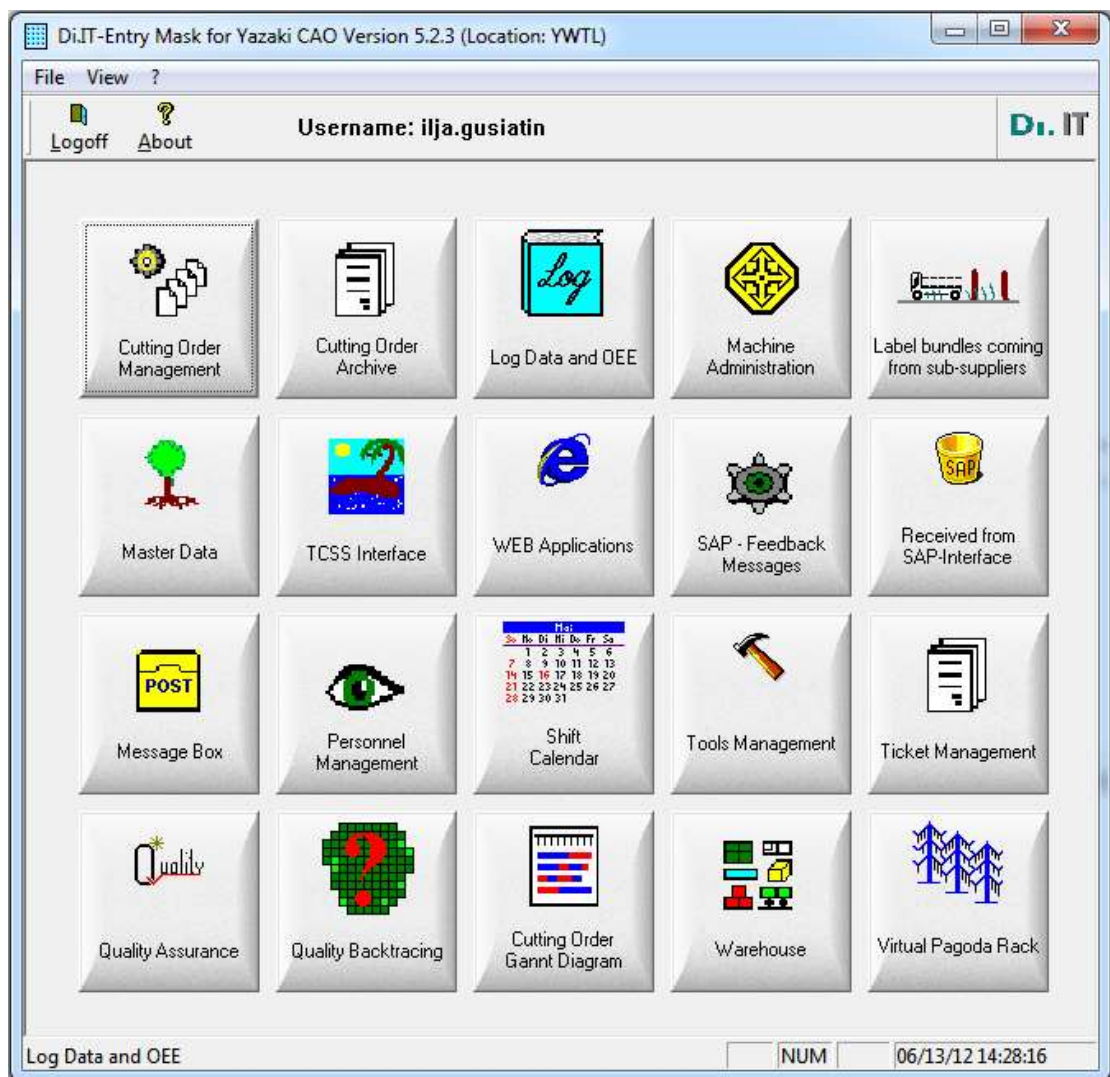
Savo straipsnyje A. Andziulis ir kt. apskaičiuoja arčiausio kaimyno ir skruzdėlių kolonijos optimizacijos metodų efektyvumą, lyginant juos palei du kriterijus: tikslo funkcijos mažiausią reikšmę ir laiką (procesoriaus veikimo), kuris buvo reikalingas tai reikšmei rasti [7]. Gauti rezultatai leidžia daryti išvadą, kad skruzdėlių kolonijos optimizacijos algoritmas yra geresnis tikslo funkcijos mažiausios reikšmės atžvilgiu, tačiau skaičiavimo laikas yra žymiai didesnis negu sprendžiant problemą arčiausio kaimyno metodu.



## 1.2. CAO programinės įrangos apžvalga

Vokiečių kompanijos Di.IT AG sukurtas 2006 metais sukurtas CAO programinis produktas, skirtas Yazaki laidų karpymo įmonėms visame pasaulyje. Produktas yra pastoviai tobulinamas, šiuo metu aktyvi yra 5.2.3 versija.

CAO – tai pagrindinės gamybos bei finansų valdymo programos (ERP) SAP interfeisas, kuris atsakingas už gamybos užduočių sekos optimizavimo ir paskirstymo tarp automatinių laidų karpymo mašinų, operatorių darbo produktyvumo sekimą, kontaktų tvirtinimo įrankių duomenų bazę, pamainų darbo laiko planavimą ir kt.



5 pav. CAO programos pagrindinis langas

Pagrindinis CAO programos langas pavaizduotas x pav. CAO programa susideda iš 20 atskirų modulių, kurie pagrindiniame lange vaizduojami kaip atskiri blokai:

- Cutting Order Managment – pjovimo užsakymo valdymas.
- Cutting Order Archive – padarytų užduočių archyvas.



- Log Data and OEE – operatorių darbo našumo skaičiavimo paprogramė.
- Machine Administration – automatinio karpymo mašinų nustatymai.
- Label bundles coming from sub-supplier – iš sub-kontaktorių atvežti laidai.
- Master Data – užduočių duomenų bazė.
- TCSS Interface – kontaktų tvirtinimo informacija.
- Web Applications – internetinės paprogramės.
- SAP – Feedback Messages – ataskaitos į pagrindinę SAP programą.
- Received from SAP-interface – per SAP sąsają gauti duomenys.
- Message Box – klaidų ir kitų pranešimų, susijusių su pjovimu, dėžutė.
- Personnel Management – personalo tvarkyklė.
- Shift Calendar – pamainų valdymo paprogramė.
- Tool Management – kontaktų tvirtinimo įrankių, sandariklių uždėjimo įrankių ir kt. administravimas.
- Ticket Management – etikečių valdymas.
- Quality Assurance – kokybės užtikrinimo paprogramė.
- Quality Backtracking – kokybės sekimo paprogramė.
- Cutting Order Gantt Diagram – pjovimo užduočių Ganto diagramos.
- Warehouse – sandėlio valdymas.
- Virtual Pagoda Rack – virtualus stelažas.

Užduočių duomenų bazė visą laiką sinchronizuojama su SAP duomenimis. Kai iš SAP programos yra sukuriamas užsakymas, jis yra siunčiamas į CAO, kur pasiima informaciją iš užduočių duomenų bazės (Master Data) ir kontaktų tvirtinimo informacijos (TCSS interface) paprogramių. Sukurtos užduotys atsiranda pjovimo užsakymų valdymo paprogramėje, optimizacijos lange. Naujai sukurti užduotys vaizduojami x pav.

Sequn. No.	CAO No.	Setup Costs	Product Costs	Cuttool No.	Prod. Vers.	Plantext	Tool Left	Tool Right	Prod. Progress	Target
0	4751808	9.50	0.01	5000648532	1	Circuit 7081 PT3DSP 0.5 D	7114545602		New	26.06.
0	4751809	1.50	0.01	5000648533	1	Circuit 7082 PT3DSP 0.5 G	7114545602		New	26.06.
0	4751810	14.00	0.02	5000648964	1	Circuit 4002 PT3DSP 4 R	7114412202	7116412202	New	26.06.
0	4751811	5.00	0.11	5000648965	1	Circuit 4009 PT3DSP 10 R			New	26.06.
0	4751812	14.00	0.01	5000648966	1	Circuit 1Y PT3DSP 1.5 W	7116816202	7116411202	New	26.06.
0	4751813	6.50	0.01	5000648967	1	Circuit 4005 PT3DSP 1.5 R	7116816202	7116415602	New	26.06.
0	4751814	14.00	0.02	5000651368	1	Circuit 5003 PT3DSP 0.75 L	7112508102	7112502202	New	26.06.
0	4751815	1.50	0.02	5000651369	1	Circuit 5004 PT3DSP 0.75 R/L	7112508102	7112502202	New	26.06.
0	4751853	1.50	0.02	5000651465	1	Circuit 7106 PT3DSP 0.75 GY	7116478602	7116410102	New	26.06.
0	4751856	1.50	0.02	5000651466	2	Circuit 7107 PT3DSP 0.75 W	7116478602	7116410102	New	26.06.
0	4751858	4.50	0.02	5000651467	1	Circuit 1AI PT3DSP 0.75 W	7116478602	7116411102	New	26.06.
0	4751859		0.02	5000651468	1	Circuit 1AK PT3DSP 0.75 W	7116478602	7116411102	New	26.06.
0	4751860	6.50	0.02	5000651469	1	Circuit 4004 PT3DSP 0.75 R	7116478602	7116478602	New	26.06.
<div>             List: MACH    Rec: 12    Selrec: 1    Costs Mach: 1711.50    Costs All: 2085.41    Next auto Opt: 14:11    CAP NUM    SCRL    DWR           </div>										

6 pav. Naujai sukurtos užduotys

Užduotis yra talpinami į CAO tokia tvarka, kokia ir buvo sukurti. Optimizacijos lange galime matyti trumpa informacija apie užduotį: sekos numeris, CAO numeris, paruošimo kaštai, pagaminimo kaštai, užduoties numeris, versija, aprašymas, reikalingi įrankiai ir kairės ir dešinės, būseną, pagaminimo pabaigos laikas ir kt.

Pjovimo užduotis – tai laido ir ant jo galų tvirtinamų kontaktų kombinacija. Kiekviena užduotis turi savo unikalų numerį – Cutlead No. Pagaminus vieną užduotį, ir pradėjus kitą, reikia pakeisti tiek įrankius, tiek komponentus: laidą ir kontaktus. Kiekvienas toks pakeitimas stabdo gamybą. Logiška yra daryti užduotis su vienodais kontaktais ir laidais vieną po kitos, kad atlikinėti mažiau keitimų.

Optimizacijos veiksmas – sudėlioti užduotis atsižvelgiant į jų komponentų (laidas ir kontaktai) skirtumus. Užduotys su vienodais kontaktais, laidais yra grupuojami, kuo pasekmėje, užduočių paruošimo laikas mažėja. Paveiksle x naujai sukurtų laidų paruošimas gamybai buvo įvertintas 1771,5 minučių. Paveikslas x+1 vaizduoja jau suoptimizuota užduočių seką, kurios paruošimo laikas yra 849,5 minučių. Tokiu būdu yra sutaupoma 922 minutės, arba 52% viso paruošimo laiko.

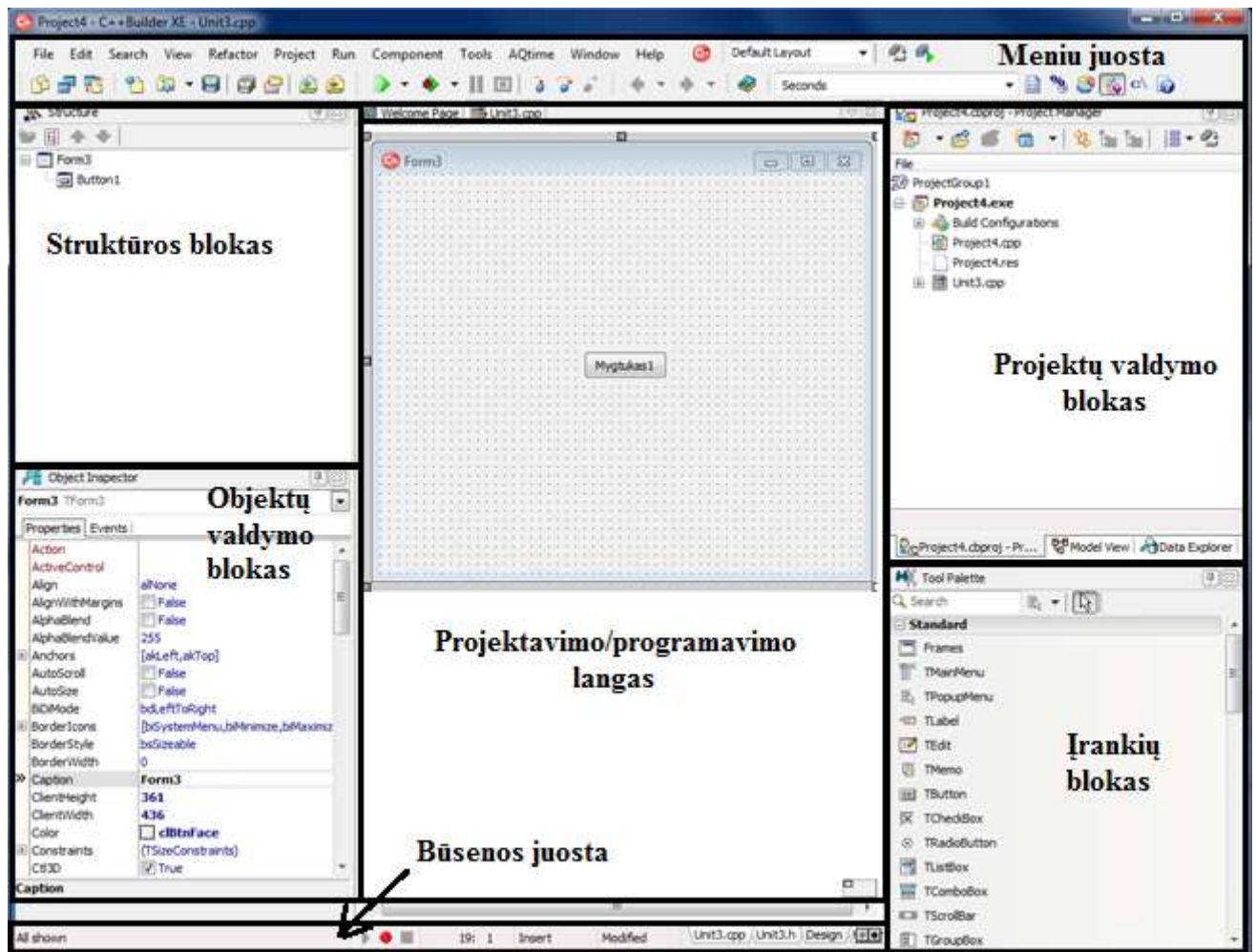
Seque. No.	CAO No.	Setup Costs	Product Costs	Cutlead No.	Prod. Vers.	Plaintest	Tool Left	Tool Right	Prod. Progress	Target
1	4751811	5.00	0.11	5000648965	1	Circuit 4009 PT3DSP 10 R			New	26.06.
2	4751839	5.00		5000649281	1	Circuit 7004M PT3DSP 0.75 Y			New	26.06.
3	4751840	1.50		5000649282	1	Circuit 7005M PT3DSP 0.75 G			New	26.06.
4	4751845	1.50		5000649287	1	Circuit 7006L PT3DSP 0.75 O			New	26.06.
5	4751846	1.50		5000649288	1	Circuit 7007L PT3DSP 0.75 G			New	26.06.
6	4752042	4.50	0.10	5000649137	2	Circuit 4000 P1551C 16 R			New	26.06.
46	4751905	2.00	0.01	5000649005	1	Circuit 7006M PT3DSP 0.75 O	7114334202		New	26.06.
47	4751911	2.00	0.01	5000649005	1	Circuit 7006M PT3DSP 0.75 O	7114334202		New	26.06.
48	4751912	1.50	0.01	5000649006	1	Circuit 7007M PT3DSP 0.75 G	7114334202		New	26.06.
49	4751946	1.50	0.01	5000649040	1	Circuit 7006H PT3DSP 0.75 R	7114334202		New	26.06.
50	4751937	9.00		5000649031	1	Circuit 2044A PT3DSP 1.5 R	7114334302		New	26.06.
51	4751938			5000649032	1	Circuit 2044B PT3DSP 1.5 R	7114334302		New	26.06.
52	4751808	9.00	0.01	5000648532	1	Circuit 7081 PT3DSP 0.5 O	7114545602		New	26.06.
53	4751809	1.50	0.01	5000648533	1	Circuit 7082 PT3DSP 0.5 G	7114545602		New	26.06.
54	4751810	1.50	0.01	5000648534	1	Circuit 7083 PT3DSP 0.5 G	7114545602		New	26.06.
I Int MACH Rec: 12 Sel:ec: 1 Costs Mach: 849.50 Costs All: 1203.41 Target: 1771.50 CAP: NUM: SCRL: DVR:										

7 pav. Optimizuota užduočių seka

### 1.3. Programinių įrankių apžvalga

#### 1.3.1. Embarcadero RAD Studio XE programinis paketas

Kūrimas C++ Builder XE yra tikrai vizualus, dėl savo *drag-and-drop* ir WYSIWYG įrankių pagalba. Modulio darbalaukis susideda iš meniu juostos, struktūros bloko, objektų valdymo bloko, būsenos juostos, projektavimo/programavimo bloko, projektų valdymo bloko ir įrankių bloko (8 pav.).



8 pav. C++ Builder XE darbalaukis

**Meniu juosta.** Susideda iš dviejų juostų: pagrindinio meniu ir dažniausiai naudojamų mygtukų meniu.

**Struktūros blokas.** Šitame bloke yra pavaizduojami visuose projektuose naudojami objektai, bei jų hierarchinė struktūra.

**Objektų valdymo blokas.** Šiame bloke galima nustatyti pasirinkto objekto savybes, bei priskirti objektui veiksmą prie tam tikro įvykio.

**Būsenos juosta.** Šioje juostoje yra makrokomandų valdymo mygtukai, redagavimo režimas ir projektavimo/programavimo bloko perjungimo mygtukai.

**Projektavimo/programavimo blokas.** Pagrindinis programos langas. Būsenos juostoje pasirinkus „Design“, rodomas kuriamos programos projektavimo blokas. Pasirinkus „Unit1.cpp“ arba „Unit1.h“ sekcijas rodomas programos kodo redagavimo blokas.

**Projektų valdymo blokas.** Šitame bloke yra projektų valdymo dažniausiai naudojamų mygtukų juosta ir projektų sudėties langas.

**Įrankių blokas.** Šitame bloke yra visi galimi įrankiai, suskirstyti į kategorijas. Dėl to, kad įrankių yra daug, sukurtas paieškos dialogas.

Programinio paketo *Embarcadero RAD Studio XE* minimalūs sisteminiai reikalavimai:

- 1 GB RAM operatyvios atminties (2 GB RAM pageidautina);
- 5 GB atminties kietajame diske;
- vaizduoklio rezoliucija 1024x768 ar didesnė;
- 1,6 GHz dažnio procesorius (pageidautina 2 GHz ir daugiau);
- Microsoft Windows Server (2003 ar 2008), XP (Home ar Professional), Vista, 7 (palaikomos 32 ir 64 bitų sistemos);
- pelė, klaviatūra, DVD-ROM įrenginys.

*Embarcadero RAD Studio XE* yra komercinis produktas. Šitame darbe vartotojo sąsajos kūrimui buvo panaudota 30 dienų bandomoji versija.

### **1.3.2. Microsoft Visio 2010 programinis paketas**

*Microsoft Visio 2010 Premium* yra diagramų braižymo programinis įrankis, skirtas patogesniai vaizdavimui. Diagramų kūrimui naudojama vektorinė grafika. Iš 8 teikiamų ruošinių kategorijų, šitame darbe yra naudojamos dvi:

- struktūrinės schemos;
- programų ir duomenų bazės ruošiniai.

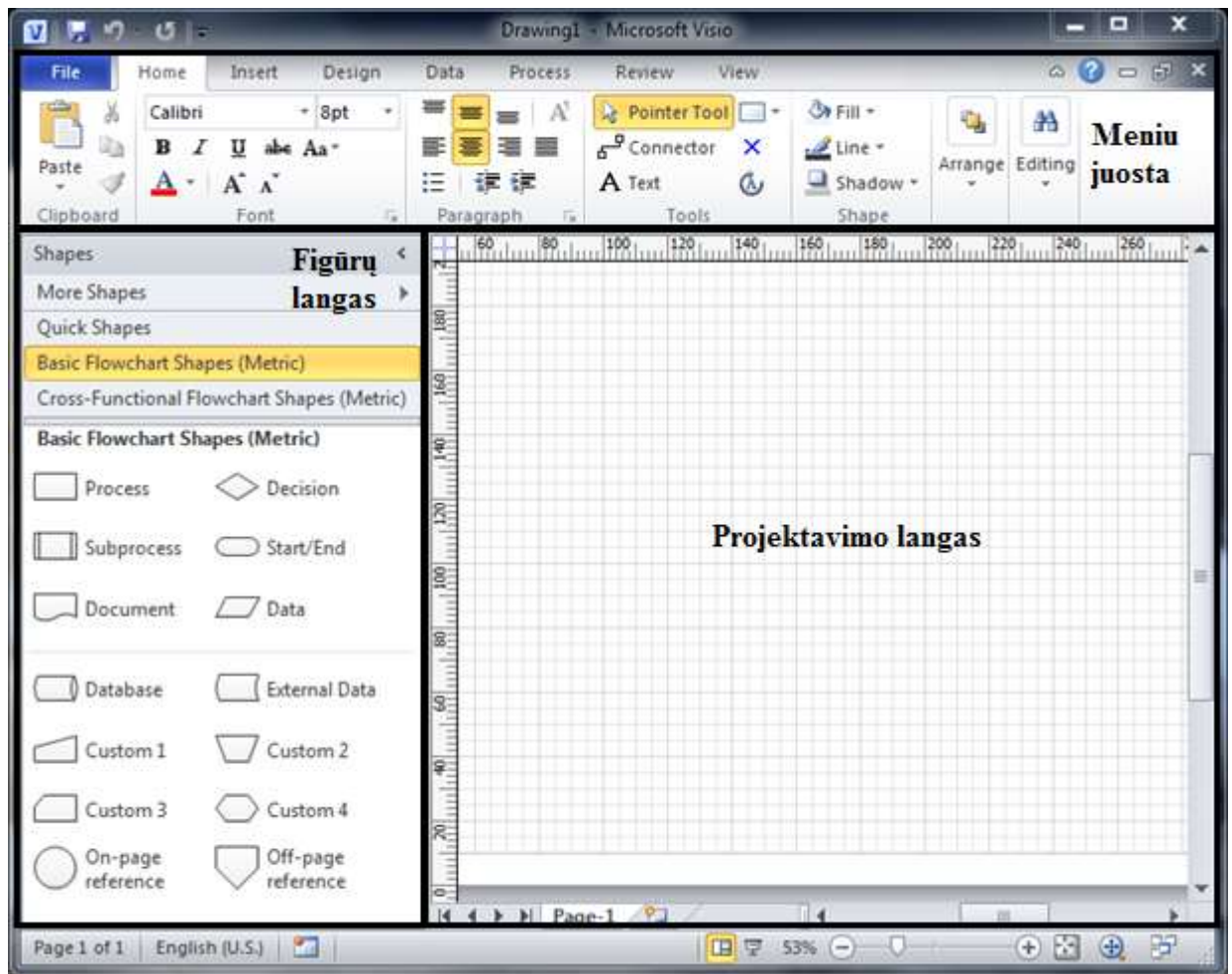
Pasirinkus *pagrindinės struktūrinės schemos* ruošinį, atsidaro darbalaukis, susidedantis iš 3 dalių: meniu juostos, figūrų lango ir projektavimo lango (9 pav.).

**Meniu juostoje** yra pasirinkto meniu dažniausiai naudojamų mygtukų blokai.

**Figūrų lange** yra pasirinkto ruošinio figūros, suskirstytos pagal kategorijas. Be to, į darbalaukį galima įtraukti figūras iš kitų ruošinių, pasirinkę kategoriją „*Daugiau figūrų*“. Dažniausiai naudojamos figūros: *procesas, sprendimas, pradžia/pabaiga, duomenys*.

Į **projektavimo langą** pertempus figūras, jas galima dėlioti, pavadinti ir sujunginėti.

UML diagramos (veiklos, bendradarbiavimo, komponentų, paplitimo/paskirstymo, sekos, būsenos, statinės struktūros ir panaudos atvejų) pateiktos *programų ir duomenų bazės* ruošiniuose.




**9 pav.** Microsoft Visio 2010 Premium darbalaukis

*Microsoft Visio 2010* programinis paketas skirtas tik *Microsoft Windows* operacinėms sistemoms. Kitoms operacinėms sistemoms ši programa nėra pritaikyta ir nė viena *MAC OSX* ar *Linux* operacinėse sistemose veikianti programa negali perskaityti *Visio* dokumentų.



## 2. METODINĖ DALIS

Ieškant optimalaus sprendimo analizuojamai gamybos užduočių sekos optimizavimui buvo pasitelkta klasterių analizė. Klasterių analizės metodai plačiai naudojami įvairiuose srityse, kur yra aktualu objektų, turinčių tam tikrų panašių komponentų, suskirstymas į grupes [8, 9]. Laidų karpymo užduotys yra laikomi objektais su trijų komponentais: laidas, kuris susidaro iš dviejų komponentų: laido skerspjūvio ploto ir spalvos, bei kairiojo ir dešiniojo kontakto (10 pav.). Iš bendros masės užsakymų galima sudarinėti grupės, turinčias vienodus komponentus, siekiant gaminti užsakymus iš grupės, kuriuos viduje esantys užsakymai skiriasi tik laido ilgiu (analizuojamame kontekste skaityk „nesikeičia“), paskui iš kitos grupės, kurios vidinių užsakymų komponentai skiriasi nuo prieš tai pagamintų užduočių tik vieno komponentų, ir taip išvengti didelių pakeitimų ant karpymo mašinų (tarkim, laido ir abiejų kontaktų pakeitimo) tarp daromo ir sekančio užsakymo.

<b>Užsakymas: SK00675592</b>		
<b>Kairinis kontaktas:</b> <b>7116532608</b>	<b>Laidas: G2803960,</b> kur "39" - skerspjūvio ploto kodas (0,50 mm <sup>2</sup> ) "60" - spalvos kodas (žalias)	<b>Dešininis kontaktas:</b> <b>7112508002</b>
		

10 pav. Laidų karpymo abstraktus užsakymas

### 2.1. Klasterių analizė

Klasterizavimas – tai duomenų grupavimas į panašių objektų klases ar klasterius. Tikslas - sugrupuoti objektus į klases ar klasterius taip, kad skirtumai tarp objektų klasterių viduje būtų labai maži, bet skirtumai tarp skirtingų klasterių objektų būtų kuo didesni. Duomenys (objektai) yra suskirstomi į grupes remiantis jų panašumais (naudojamas klasterizavimas) ir tuomet yra priskiriami klasių pavadinimai duomenims (objektams) [10].

Klasterizavimo etapai:

- 1) pasirinkti duomenų rinkinį;
- 2) nuspręsti, pagal kokius požymius duomenys bus grupuojami;
- 3) pasirinkti kiekybinį matą, kuriuo matuosime objektų panašumą;
- 4) pasirinkti skirstymo metodą;
- 5) peržiūrėti gautus rezultatus.

### 2.1.1. Klasterių analizės metodai

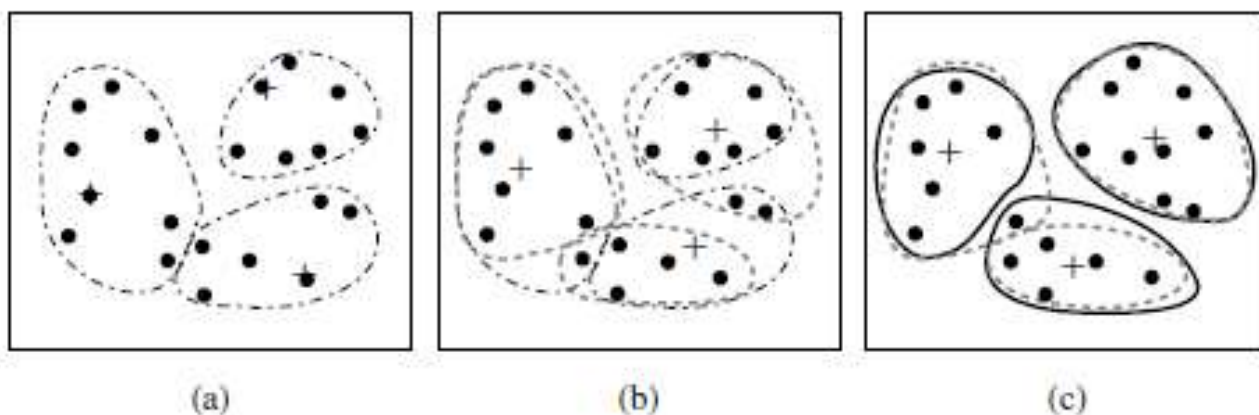
Yra keletas metodų, leidžiančių analizuoti klasterius.

- skaidymo metodai;
- hierarchiniai metodas;
- tankiu pagrįsti metodas;
- tinklu pagrįstas metodas;
- modeliu pagrįstas metodas.

Apžvelgsime du iš jų: skaidymo ir hierarchinį.

Skaidymo metodai yra taikomi tada, kai yra  $n$  objektų, kuriuos reikia išskirstyti į  $k \leq n$  klasterių. Duomenys yra skirstomi taip, kad vienas grupė turi bent vieną objektą, ir kiekvienas objektas priklauso tik vienai grupei. Duotas  $k$ , dalių į kurias skaidomas kiekis, skaidymo metodas sukuria priminius skaidinius. Toliau naudojamas iteracinis perskirstymo metodas, siekiant pagerinti suskaidymą perkeltant objektus iš vieno klasterio į kitą. Dauguma skaidymo metodų yra pagrįsta  $k$ -vidurkių arba  $k$ -medoidų algoritmais.

$K$ -vidurkių algoritmai yra pats populiariausias metodas. Kiekvieną klasterį atstovauja klasterio objektų vidurkių reikšmė (11 pav.). Algoritmo veiksmas yra minimizuoti klasterio taškų atstumus nuo klasterio centro taško suminius kvadratinis nuokrypius.



11 pav.  $K$ -vidurkių klasterizavimo algoritmas [10]

$$V = \sum_{i=1}^k \sum_{x_j \in S_i} (x_j - \mu_i)^2$$

Kur  $k$  – tai klasterių kiekis,

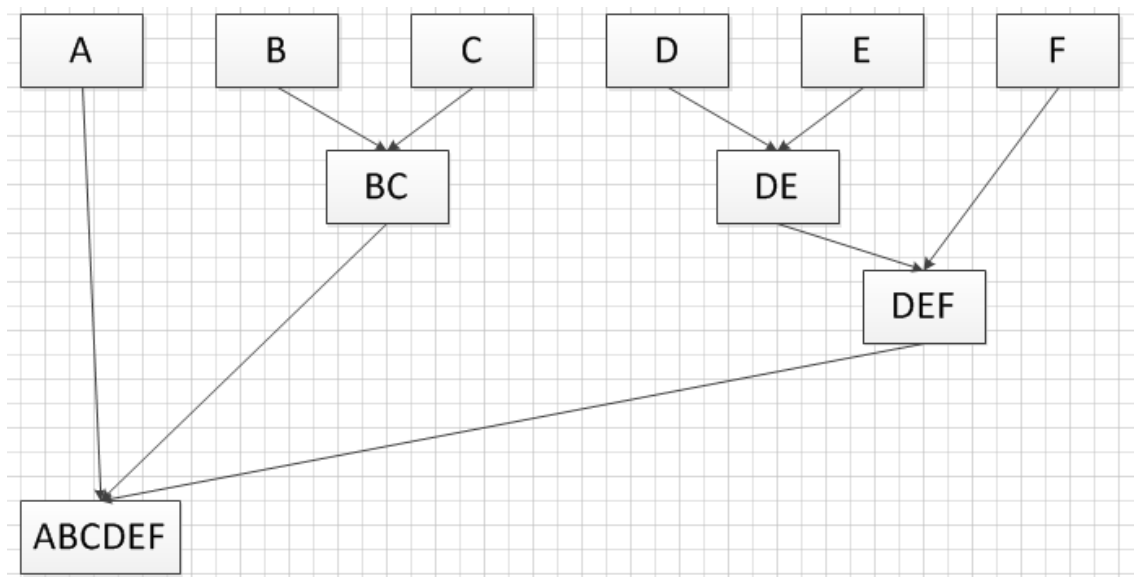
$S_i$  – gaunami klasteriai,

$\mu_i$  – klasterių masių centras.

Šitas algoritmas turi ir negatyvių savybių – reikia iš anksto nustatyti gaunamų klasterių kiekį, ir kaip gaunamas klasterizavimo rezultatas stipriai nuo tuo priklauso.

K-medoidų algoritmas skiriasi nuo k-vidurkių algoritmų tuo, kad klasterio centrų yra laikomas medoidas – klasterio objektas (skirtingai nuo centroido), kuris yra laikomas klasterio centru.

Hierarchiniame metode, arba grafiniame klasterizacijos algoritme, iš pradžių nustatoma bendra visų klasterių tarpusavio priklausomybių struktūra. Taikant jungimo metodus pradinį klasterį sudaro vienas elementas, ir jungiant kitus elementus daromi vis stambesni klasteriai, kol galų gale lieka vienas klasteris. Hierarchinis metodas vizualizuojasi dendrogramų pagalba (12 pav.). Dendrograma - tai yra medis (grafas be ciklų). Dendrograma leidžia atvaizduoti objektų tarpusavio ryšius duotame duomenų rinkinyje. Tyrėjas pats sprendžia, kuriuo etapu objektų paskirstymas į klasterius yra optimalus.



**12 pav.** Dendrogramos pavyzdys

## 2.2. Intelektualūs agentai

Intelektualūs agentai – tai programinės įrangos dalis, kurį dirbą vartotojui ar kitai programai tam tikroje aplinkoje, gali bendrauti su kitais agentais ar procesais. Intelektualus agentas turi tam tikrą dirbtinį intelekto pavidalą. Priklausomai nuo tipų jie gali mokytis, daryti išvadas, tarpusavyje komunikuoti siekdami bendrų tikslų, keisti vietą.

Agentų orientuotas programavimas evoliucionavo iš objektiškai orientuoto programavimo (toliau OOP). Pagrindinis skirtumas tarp jų yra tas, kad OO programavime skaičiavimo procesas suprantamas kaip sistema, surinkta iš modulių, sąveikaujančių tarpusavy ir turinčius savo būdus įeinančių pranešimų apdirbimui. Tuomet agentai turi iš principo kitą autonomijos lygį. Agentas



nebūtinai laukia kito agento ar vartotojo įsakymo, bet gali savarankiškai formuoti sau tikslą. Tai priklauso nuo aplinkos sąlygų, įskaitant kitų agentų tikslus. Agentas gali imtis įsipareigojimų, arba atvirkščiai atsisakyti atlikti darbą, remdamas, pavyzdžiui užimtumu kitomis užduotimis. Agentas gali kurti, naikinti ir savimi pakeisti kitus agentus, aktyvizuoti funkcijas (kaip savo, taip ir kitų agentų). Tai parodo kad agentas yra aukštesniame lygmenyje nei objektai [11].

S. Haag skirsto agentus į 4 pagrindinius tipus pagal jų paskirtį:

- 1) pirkimo agentai (pirkimo botai);
- 2) vartotojo (personalūs) agentai;
- 3) prognozavimo (prižiūrintys-ir-valdantys) agentai;
- 4) duomenų gavybos agentai.

**Pirkimo agentai** tai tokie robotai, peržiūrėdami tinklo resursus (dažniausia Internetą), renka informaciją apie produktus ir paslaugas. Šitie agentai efektyviai dirba su dažnai naudojamomis prekėmis, tokiomis kaip CD diskai, knygos, elektroprekės ir t.t. Amazon.com – geras tokio roboto pavyzdys. Ši interneto svetainė pasiūlys Jums sąrašą prekių, kurie gali būti Jums įdomios, remdamasis informaciją apie Jūsų ankstesnius pirkinius.

**Vartotojo ar personalūs agentai** – tai tokie agentai, kurie atlieka Jus interesuojančius veiksmus. Jų pritaikymo sritis labai plati. Jie tikrina elektroninius paštus, įspėdami Jus apie naują gautą laišką, renka naujienas internete, savarankiškai pildo formas, ir gali būti varžovų kompiuteriniame žaidime. Tokio agento pritaikymo pavyzdys – programa Mail@agent. Ji tikrina Jūsų el. pašto dėžutę serveryje www.mail.ru, automatiškai trina SPAMą (reklamos laiškus).

**Prognozavimo agentų**, kurie irgi žinomi tai valdymo ir stebėjimo agentai, tikslas yra prižiūrėti ir siųsti ataskaitas. Agentai gali prižiūrėti kompanijos inventorizacijos lygį, konkurentų kainas. Tokio tipo agento panaudojimo pavyzdys – NASA Jet Propulsion Laboratory (reaktyvinio judėjimo laboratorija) turi agentą, prižiūrintį inventoriaus būklę, planavimą ir tvarkaraščių sudarymą. Tokie agentai gali prižiūrėti ir kompiuterio tinklus, stebėdami kiekvieno prijungto prie tinklo kompiuterio konfigūraciją.

**Duomenų gavybos agentai** veikia duomenų saugyklose, rinkdami informaciją. Duomenų saugyklos apjungia skirtingų šaltinių informaciją. Tokie agentai gali aptikti pagrindinius pokyčius vystymosi tendencijose ir perspėti Jus apie naujos informacijos atsiradimą.

### 2.3. Duomenų struktūros C++ aplinkoje

Kuriamas algoritmas bus realizuotas C++ programavimo kalba, todėl būtina apžvelgti dinamines duomenų struktūrų bazinius principus, kurias bus pasitelkta, kaip pagrindinių

instrumentu. Dėl to, kad iš pradžių nėra žinomas nei užsakymų, nei jų busimų suklasterizuotų grupių, kiekis, reikia naudoti dinaminiais duomenų sąrašais [12].

Programavimo aplinkoje, sąrašais vadinami tokie duomenų rinkiniai, kurių elementus apibūdina ne tik jų reikšmės, bet ir jų perrinkimo bei tvarkymo taisyklės. Kiekvienam sąrašui būdingos tokios savybės:

- sąrašas gali būti tuščias arba turėti tam tikrą elementų skaičių;
- jame turi būti griežtai apibrėžta elementų perrinkimo tvarka;
- sąrašas privalo turėti griežtas papildymo naujais elementais ir elementų pašalinimo taisykles;
- turi būti galimybė kreiptis į bet kurį sąrašo elementą.

### 2.3.1. Dinaminiai sąrašai

Keičiant elementų tipus, ryšių tarp elementų būdus ir tvarkymo taisykles, galima sudaryti daugybę įvairių tipų sąrašų (eiles, stekus, dekus, medžius). Paprasčiausi sąrašai gali būti realizuojami masyvuose, papildant juos tvarkymo procedūromis ir užpildymo charakteristika. Universalesnė sąrašų sudarymo priemonė yra dinaminės struktūros, kurios sudaromos iš elementų su nuorodomis. Tokiuose elementuose saugomos ne tik jų reikšmės, bet ir ryšio su kitais elementais aprašymai. Dinaminės struktūros elemento sudėtinės dalys yra reikšmė ir ryšio dalis.

Dinaminės struktūros elemento reikšmė gali būti bet kokia statinė arba dinaminė struktūra, o ryšio dalyje yra rašomos rodyklės į kitus to paties sąrašo elementus. Kadangi dinaminių struktūrų elementų reikšmės ir nuorodos yra aprašomos skirtingų tipų duomenimis, sąrašų elementai realizuojami įrašais, kurių struktūra yra tokia:

```
struct <Vardas> {  
    <Reikšmės laukai>;  
    <Ryšio dalies laukai>;  
}
```

Homogeninėse dinaminėse struktūrose, kuriose visi elementai yra vienodo tipo, aprašant ryšio dalies rodykles, reikia nurodyti pačios naujai aprašomos struktūros tipą. Kreipimasis struktūros aprašyme į save pačią vadinamas rekursija, o struktūros, kuriose yra tokie kreipiniai, vadinamos rekursyviomis. Rekursyvios struktūros elemento aprašo pavyzdys:

```
struct list {  
    int data;  
    struct list *next;    // Rekursyvi nuoroda  
}
```

Dinaminiai sąrašai pasižymi tuo, kad visus jų elementų jungimo tvarkos pakeitimo veiksmus galima atlikti keičiant vien tikrai elementų ryšius aprašančias rodykles ir neperrašant juose saugomų

duomenų reikšmių. Tokia savybė ypač naudinga tada, kai sąrašuose saugomi sudėtingos struktūros duomenys.

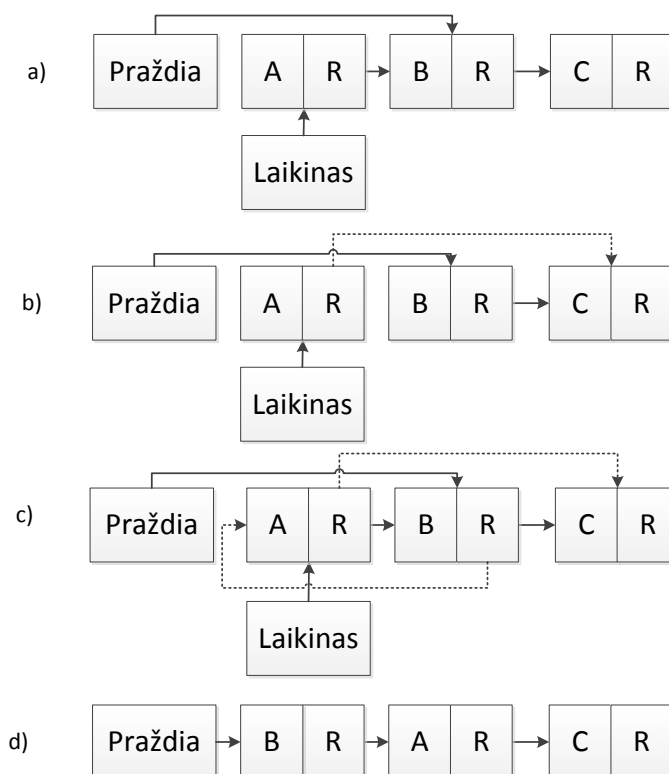
Elementų jungimo tvarkos keitimo aprašymą rodyklių reikšmių keitimo veiksmais pailiustruosime pavyzdžiu.

Tarkime, kad tiesiniame sąrašė reikia perrašyti rodykles taip, kad pasikeistų dviejų pirmųjų elementų A ir B ryšio tvarka. Norint tai padaryti, reikia pakeisti trijų su šiais elementais susijusių rodyklių reikšmes tokia tvarka, kuri parodyta a, b ir c paveikslėliuose. Juose punktyrines linijas pažymi naujai formuojamas rodyklių reikšmes.

Sukeitimas pradedamas pagalbinės rodyklės „*laikinas*“ sudarymu, kuriai perduodama pirmąjį elementą A rodančios rodyklės „*pradžia*“ reikšmė. Po to rodyklei „*pradžia*“ galima suteikti antrojo elemento B adresą (13a pav.).

Antrajame sukeitimo žingsnyje jau galima formuoti naują elemento A ryšio dalį, nes jos saugotas senasis adresas jau yra perduotas rodyklei „*pradžia*“ (13b pav.).

Trečiajame žingsnyje pagalbinės rodyklės „*laikinas*“ saugomas elemento A adresas perrašomas į elemento B ryšio dalį. Taip yra nurodoma, kad elementas A sąrašė turi būti už elemento B (13c pav.). Vizualiai sutvarkytas sąrašas dabar atrodo taip (13d pav.)



**13 pav.** Dinaminio sąrašo tvarkymo žingsniai [12]

Tiesinių sąrašų elementų rikiavimo tvarką galima įvertinti jų formavimo funkcijose ir jas sudaryti taip, kad, papildant sąrašą naujais elementais, jų rikiavimo tvarka išliktu. Tokios funkcijos

sudaromos iš dviejų dalių. Pirmojoje dalyje yra ieškoma sąrašo vieta, kurioje reikia įterpti naują elementą, o antrojoje - aprašoma įterpimo procedūra. Jeigu simbolių sąrašo elementai saugomi alfabeto tvarka, naujas elementas turi būti rašomas prieš pirmąjį sąrašo elementą, kurio kodas yra didesnis už papildančio simbolio kodą arba, jei tokio elemento nėra, sąrašo gale, įterpimo vietos paieškai galima naudoti rekursyvų sąrašo peržiūrėjimą arba paieškos ciklą. Įterpimui tinka standartinė steko papildymo funkcija.

Iš elementų, kurių ryšio dalyje yra viena rodyklė, galima sudaryti tik, tiesiniais sąrašais vadinamas, nuosekliai sujungtas grandines. Papildant tiesinius sąrašus tvarkymo procedūromis ir išorinio ryšio priemonėmis, galima sudaryti tokias tipines jų modifikacijas:

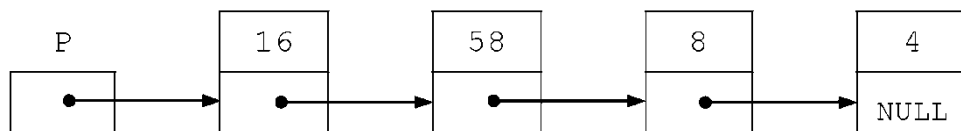
- tiesinius sąrašus, kuriuose nėra apribojimų sąrašo elementų analizei, tvarkymui ir apdorojimui;
- eiles, kuriuose nauji elementai prijungiami sąrašo gale, o skaitymui yra prieinamas tik pirmasis elementas (struktūra FIFO – *first in, first out*);
- stekus, kuriuose galima skaityti tik vėliausiai įrašytą elementą (struktūra LIFO – *last in, first out*);
- dekus, kuriuose elementai gali būti rašomi ir skaitomi tiek sąrašo pradžioje, tiek gale (struktūra DEQUE – *double-ended queue*).

Tokios sąrašinės struktūros labai plačiai vartojamos ne tik taikomosiose, bet ir sisteminėse programose. Pavyzdžiui, stekuose yra saugomi pertraukiamų procesų parametrai, organizuojami lokalūs duomenys. Eilės yra populiari pagalbinių (buferinių) atminčių, kuriuose kaupiami iš lėtų įrenginių gaunami arba į juos siunčiami duomenys, organizavimo priemonė.

Kiekvieno sąrašo tipo realizavimui skirta dinaminė struktūra turi turėti jos elementus aprašančią struktūrą, rodyklę arba rodykles į išoriniams ryšiams skirtus elementus ir tokias tvarkymo operacijas:

- naujo sąrašo kūrimo;
- naujo elemento prijungimo;
- elemento pašalinimo;
- sąrašo skaitymo ir analizės;
- sąrašo tvarkymo.

Sudarant sąrašų tvarkymo procedūras, reikalingas papildomas susitarimas apie tai, kaip bus žymima *sąrašo pabaiga* ir kaip bus žymimas *tuščias sąrašas*. Yra priimta tiek sąrašo pabaigą, tiek tuščią sąrašą dinaminėse struktūrose žymėti nuline rodykle *NULL*.



**14 pav.** Tiesinis sąrašas [12]

Tiesinio sąrašo grafinis vaizdas (14 pav.) parodytas pavyzdyje čia elementas vaizduojamas stačiakampiu, padalintu į tiek dalių, kiek yra laukų jo struktūroje: duomenų ir adreso. Duomenų lauke saugomos reikšmės. Rodyklės tipo lauke saugomas adresas (nuoroda) į kitą sąrašo elementą. Nuoroda pavaizduota linija su rodykle gale. Tos linijos pradžia yra nuorodos lauko viduje. Rodyklė remiasi į elementą vaizduojantį stačiakampį bet kurioje vietoje. Jeigu nuoroda neegzistuoja, tuomet linija nėra brėžiama ir laukas lieka tuščias. Tai reiškia, kad adreso reikšmė neapibrėžta, “šiukšlė”. Realiose programose tokia situacija yra neleistina.

### 2.3.2. Sąrašų rikiavimas

Sąrašo rikiavimu, vadinamas jo elementų išdėstymas saugojimo struktūroje rikiavimo rakto reikšmių didėjimo arba mažėjimo tvarka. Rikiavimo raktu vadinamas sąrašo elementų požymis, pagal kurį rikiuojami elementai. Koks požymis parenkamas rikiavimo raktu, priklauso nuo sąrašo elementų pobūdžio ir sprendžiamų uždavinių poreikių. Įrašų sąrašuose rikiavimo raktu paprastai būna kuris nors įrašo laukas. Pavyzdžiui, telefono abonentų sąrašai dažniausiai rikiuojami pagal abonentų pavardes alfabeto tvarka. Simbolių eilučių sąrašų rikiavimo raktas gali būti pirmasis eilutės simbolis arba pirmasis žodis.

Surikiuotuose sąrašuose gerokai greitesnė ir paprastesnė duomenų paieška pagal požymius, efektyviau atliekamos skaičiavimų ir įvairios duomenų tvarkymo operacijos. Rikiuojant sąrašus pagal tekstinius požymius, analizuojami jų simbolių kodai, kurie alfabeto tvarka didėja. Todėl rikiavimas pagal tekstinius požymius alfabeto tvarka atitinka rikiavimą skaitinių požymių didėjimo tvarka.

Paprasčiausi rikiavimo algoritmai aprašo nuoseklų surikiuotų laukų formavimą sąrašo gale arba pradžioje, siunčiant į juos iš nesurikiuotos dalies atrenkamus elementus su didžiausiomis arba mažiausiomis raktų reikšmėmis. Šiai grupei priklauso „*minimakso*“ algoritmas, kuriame naudojami tokie sąrašo tvarkymo principai:

- iš pradžių yra laikoma, kad nesurikiuotas laukas užima visą sąrašą;
- rikiavimas vykdomas žingsniais, kurie formuoja du surikiuotus laukus: vieną – sąrašo pradžioje ir kitą - gale;

- kiekviename žingsnyje nesurikiuotoje dalyje randami elementai su didžiausia ir su mažiausia rakto reikšme. Elementas su mažiausia rakto reikšme sukeičiamas su pirmuoju nesurikiuotos dalies elementu, o elementas su didžiausia rakto reikšme - su galiniu;
- rikiavimas baigiamas tada, kai nesurikiuotoje dalyje nebelieka elementų.

Dideliu efektyvumu pasižymi populiarus „burbulo“ algoritmas, kuris jungia nuoseklaus surikiuoto lauko formavimo ir nuoseklaus tvarkomo elemento perstūmimo, atliekant dalinį sąrašo sutvarkymą, idėjas. Taikant šį būdą, kiekviename rikiavimo žingsnyje tvarkomo elemento indeksui nuosekliai suteikiamos visos galimos jo reikšmės. Suteikus kiekvieną naują reikšmę, elemento požymis lyginamas su jam gretimo tolimesnio elemento požymiu ir, jeigu jie prieštarauja rikiavimo tvarkai, elementai sukeičiami vietomis. Tokio proceso metu elementas su didžiausia rakto reikšme, lyg burbulas, „išstumiamas“ į masyvo pabaigą. Todėl šis rūšiavimo būdas ir yra vadinamas „burbulo“ metodu.

Metodas yra labai efektyvus, nes kiekvieno didžiausio nario „stūmimo“ į masyvo pabaigą žingsnio metu ne tik atrenkamas didžiausias tvarkomos dalies elementas, bet dalinai sutvarkomi ir likusieji elementai. Kadangi „išstumtą“ elementą kito žingsnio metu galima jau nebenagrinėti, kiekviename naujame rikiavimo žingsnyje tvarkoma masyvo dalis siaurinama, atmetant iš jos paskutinį elementą su ten patalpinta didžiausia reikšme. Iš atrinktų didžiausių elementų masyvo gale formuojamas surikiuotas laukas. Rikiavimas baigiamas tada, kai šis laukas užima visą masyvą. Jeigu masyve yra  $n$  elementų, didžiausias galimas rikiavimo žingsnių skaičius yra  $n-1$ .

## 2.4. Optimizacijos metodai ir technikos

Gamybos planavimo uždavinys, kuriama reikia išrikiuoti užsakymus taip, kad staklių derinimo laikas (prastovos laikas) būtų kuo mažesnis yra iš esmės tradicinė keliaujančio pirklio problema, kuriam reikia apeiti tam tikra miestų kiekį tokiu maršrutu, kad kelionės metu praeita distancija būtų kuo mažesnė. Analizuojamoje situacijoje gamybos užsakymai yra miestai, o staklių prastovos laikas, skirtas pasiruošimui sekančiam užsakymui – atstumas tarp miestų. Istoriškai,  $n$ -miestu keliaujančio pirklio situacija yra apibūdinama kaip:

$$x_{ij} = \begin{cases} 1, & \text{jeigu miestas } j \text{ yra pasiekiamas iš miesto } i \\ 0, & \text{jeigu miestas } j \text{ nėra pasiekiamas iš miesto } i \end{cases}$$

O optimizacijos tikslo funkcija:

$$\text{Minimize } z = \sum_{i=1}^n \sum_{j=1}^n d_{ij} x_{ij},$$

kur  $d_{ij} = \infty$  visiems  $i=j$ .

### 2.4.1. Arčiausio kaimyno metodas

Vienas iš populiariausių metodų spręsti keliaujančio pirklio problemą yra arčiausio kaimyno algoritmas [7, 15]. Arčiausio kaimyno euristinis pasiūlymas yra laikomas gana geru keliaujančio pirklio problemos sprendimu, atsižvelgiant į tai, kad sprendimas ne visada yra optimalus. Metodo esmė yra miestų (mazgų) jungimas į seką pradedant nuo bet kuriuo ir prijungiant arčiausia esantį, dar neaplangytą mazgą. Procesas tęsiasi kol nėra praeinami visi mazgai.

Pavyzdžiui, 5 miestų situacijai, atstumai tarp kurių yra pateikiami matricoje (15 pav.), yra pateikiamas toks sprendimas. Euristinis sprendimas gali prasidėti iš bet kuriuo miesto. Duotame pavyzdyje sprendimas pateikiamas pradedant iš miesto M3. Trečioje matricos eilutėje yra pateikiami atstumai iki kitų miestų, kur mažiausiai yra 80, iki miesto M2. Iš miesto M2 galima judėti į visus miestus, išskyrus aplankytą M3, ir arčiausias yra M4 su atstumu 110. Iš miesto M4 galima keliauti į miestus M1, M3 ir M5. Miestas M3 jau buvo aplankytas, todėl iš dviejų likusių M1 ir M5 arčiau yra iki M1 – atstumas 150. Iš M1 galimas tik vienas variantas, paskutinis M5 miestas.

$$\|d_{ij}\| = \begin{pmatrix} \infty & 120 & 220 & 150 & 210 \\ 120 & \infty & 100 & 110 & 130 \\ 220 & 80 & \infty & 160 & 185 \\ 150 & \infty & 160 & \infty & 190 \\ 210 & 130 & 185 & \infty & \infty \end{pmatrix}$$

15 pav. Atstumų matrica

Praeitas maršrutas: M3-M2-M4-M1-M5. Bendras atstumas yra lygus 80+110+150+210=550. Palyginimui, bendras atstumas sudarytas arčiausio kaimyno metodu pradedant iš miesto M1 (maršrutas M1-M2-M3-M4-M5) lygus 570.

### 2.4.2. Atkarpų apkeitimo metodas

Jau egzistuojančio realaus sprendimo, rasto arčiausio kaimyno metodu, patobulinimui, gali būti taikomas **atkarpų apkeitimo** euristinis metodas. Jo esmė yra patikrinti, ar galima apkeičiant nuo dviejų iki n-1 miestų vietomis, rasti optimalesnį kelią. Metodo optimizavimo veiksmai yra pateikiami 1-oje lentelėje.

1 lentelė. Atkarpų apkeitimo metodas

Tipas	Apkeitimas	Maršrutas	Ilgis
Pradinis	–	M1-M4-M3-M5-M2	625
Dviejų miestų apkeitimas	M4-M3	M1-M3-M4-M5-M2	700
	M3-M5	M1-M4-M5-M3-M2	605

	M5-M2	M1-M4-M3-M2-M5	520
Trijų miestų apkeitimas	M4-M5-M3	M1-M3-M4-M2-M5	$\infty$
	M5-M3-M2	M1-M4-M2-M5-M3	$\infty$
Keturių miestų apkeitimas	M4-M5-M3-M2	M1-M2-M4-M5-M3	605

Kaip galima matyti iš lentelės duomenų, pradinis maršrutas, kurio ilgis buvo 625, sutrumpėjo iki 520, kas, beje, yra geresnis rezultatas, negu rastas paprastai taikant arčiausio kaimyno metodą. Bet, tai reikalauja papildomu veiksmų algoritme, iš pradžių tikrinant visus egzistuojančius variantų pakeitimus, paskui mažiausio bendro atstumo paieška. Tai labai apsunkina procesoriaus skaičiavimus, ypač esant dideliame miestų kiekiui. Pakeitimų skaičius  $N$  priklauso nuo miestų skaičiaus  $n$  palei formulę:

$$N = \sum_{i=1}^{n-2} i,$$

kur  $i=1, 2 \dots n-2$ .

Beje, efektyviausias metodas yra arčiausio kaimyno metodo kombinavimas su atkarpų keitimo metodu. Iš pradžių, yra ieškomas geriausias sprendimas, rastas arčiausio kaimyno metodu, pradedant iš visų penkių miestų. Geriausiam sprendimui yra taikomas atkarpos keitimų metodas, ir žiūrima, ar pagerėjo rezultatas.

### 2.4.3. Perrinkimo metodas

Perrinkimo metodo esmė yra apskaičiuoti visus įmanomus maršrutus, ir pasirinkti iš jų trumpiausią. Perrinkimo metodu rastas sprendimas yra optimaliausias iš visų galimų. Yra labai efektyvus sprendžiant tik mažo sudėtingumo uždavinius, nes veiksmų skaičius  $N$  priklauso nuo miestų skaičiaus  $n$  palei formulę:

$N=n!$ , kai pirmas miestas gali būti pasirenkamas laisvai,

$N=(n-1)!$ , kai pirmas miestas yra fiksuotas.

Taikant šį metodą labai didelėms sekoms, sprendimo paieška gali užtrukti labai ilgą laiko tarpą, todėl yra būtina apriboti šito metodo veikimo laiką, arba tiesiog netaikyti didesniai negu tam tikrai nustatyto skaičiaus  $N$  miestų.



### 3. PRAKTINĖ DALIS

#### 3.1. Statistiniai duomenys

Praktinę dalį atlikti, iš užduočių duomenų bazės (Master Data) buvo atsitiktinai pasirinkti 500 užsakymų. Kad nustatyti, jų tarpusavio panašumus ir skirtumus, pažvelgsime į jų komponentų sąrašą. Iš antros lentelės matosi, jog analizuojamoje sekoje yra 41 unikalių laidų: 17 skerspjūvio ploto „39“, 23 skerspjūvio ploto „40“ ir vienas skerspjūvio ploto „60“. Tai leidžia priimti sprendimą, jog seka gali būti klasterizuojama palei užduočių komponento „laidas“, kaip pagal laido spalvą, taip ir pagal skerspjūvio plotį. Iš viso, užsakymų su skerspjūvio plotais „39“, „40“ ir „60“ yra 162, 337 ir 1, atitinkamai.

2 lentelė. Užsakymų paskirstymas palei laidą

Laidas	Kiekis	Laidas	Kiekis	Laidas	Kiekis
G2803911	7	1R604060	2	G2806050	1
G2803921	3	1R604070	2		
G2803940	28	G2804011	2		
G2803941	1	G2804040	99		
G2803945	9	G2804041	43		
G2803946	3	G2804045	6		
G2803948	2	G2804046	1		
G2803950	24	G2804047	4		
G2803951	15	G2804048	17		
G2803952	4	G2804049	4		
G2803956	2	G2804050	51		
G2803957	1	G2804051	6		
G2803960	45	G2804052	10		
G2803970	3	G2804056	3		
G2803980	9	G2804057	1		
G2803990	5	G2804059	8		
G2803992	1	G2804060	27		
		G2804070	27		
		G2804080	4		
		G2804090	6		
		G2804092	11		
		G2804031	1		
		G2804037	2		
<b>Skerspjūvio plotas "39":</b>	<i>162</i>	<b>Skerspjūvio plotas "40":</b>	<i>337</i>	<b>Skerspjūvio plotas "60":</b>	<i>1</i>

Dar vienas užduoties komponentas – kontaktas. Kontaktai yra tvirtinami prie laido iš dviejų pusių, todėl šitų komponentų suma yra 1000. Atvejis, kai kontaktas nėra tvirtinamas (gamybos

proceso savybė – laidas toliau bus suvirinamas), yra žymimas kaip „be kontakto“, ir nereikalauja papildomų mašinos derinimo procedūros. 3-iojo lentelėje matosi, kad tokių atvejų yra 169, kaip dažniausiai naudojamas kontaktas yra 7116478602.

**3 lentelė.** Kontaktų pasikartojamumas

Kontaktas	Pasikartojamumas	Kontaktas	Pasikartojamumas
be kontakto	169	7115246002	10
71161282	5	7116161002	1
7009122102	14	7116212202	29
7111508102	17	7116410102	23
7112502202	43	7116411102	21
7112508002	49	7116415702	56
7112508102	98	7116478602	189
7113272102	1	7116499802	32
7114334102	15	7116532608	2
7114334202	53	7116908802	14
7114411102	2	7116952202	8
7114545602	104	7116997402	45

Atsižvelgiant į 3-ią lentelę, yra daroma išvada, jog užsakymas gali būti apjungiami į klasterius pagal iš abiejų laido pusių tvirtinamų kontaktų. Kad nuspręsti, pagal kurį komponentą pirmiausia bus grupuojami užsakymai, reikia atsižvelgti į staklių derinimo laiką, norint pakeisti vieną ar kelis komponentus, pereinant nuo vienos prie kitos užduoties.

Eksperimentui pasirinktos sekos iš 500 užduočių CAO programos siūlomas optimizuotas tvarkaraštis užprašo **710,5 minutės** staklių prastovos laiko. Praktinės užduoties tikslas, suoptimizuoti užduočių seką taip, kad prastovos laikas būtų mažesnis, negu dabartinės sistemos.

### 3.2. Pagalbinis įrankis

Kad sėkmingai atlikti praktinę dalį, reikalinga sukurti prototipinį įrankį, kuris padės stebėti optimizavimo rezultato progresą. Staklių derinimo laikai nurodyti 4-oje lentelėje.

**4 lentelė.** Staklių derinimo laikai

Pakeitimas	Laikas (min)
Laido spalva	1,5
Laido skerspjūvio plotas	4
Kontaktas	4,5

Laidų karpymo mašinoje yra du kontaktų uždėjimo įrankiai ir paduodamas vienas laidas. Užsakymai vienas nuo kito gali skirtis tiek laido ilgiu – tokių atvejų jokių pakeitimų nėra, iki pilno užsakymo pakeitimo: kito skerspjūvio ploto laido su dviem kitais kontaktais.

Eksperimentui yra paimta 500 užsakymų seka. Apačioje matomas staklių derinimo laiko apskaičiavimo įrankis (16 pav.).

Užsakymas	Laidas	Laido skerspjūvis	Laido spalva	Kontaktas1	Kontaktas2	Laido sk. ploto pakeitimas	Laido spalvos pakeitimas	Kontakto1 pakeitimas	Kontakto2 pakeitimas	Staklių derinimo laikas	916
SK00701467	G2804040	40	40	7116997402	7116952202	1	0	1	1	13	
SK00701468	G2804041	40	41	7116997402	7116952202	0	1	0	0	1.5	
SK00701571	G2803945	39	45	7116997402	7116478602	1	0	0	1	8.5	
SK00701575	G2803980	39	80	7116997402	7116478602	0	1	0	0	1.5	
SK00701573	G2803990	39	90	7116997402	7116478602	0	1	0	0	1.5	
SK00701559	G2804048	40	48	7116997402	7116478602	1	0	0	0	4	
SK00559699	G2804040	40	40	7116997402	7116212202	0	1	0	1	6	
SK00562667	G2804040	40	40	7116997402	7116212202	0	0	0	0	0	
SK00562695	G2804040	40	40	7116997402	7116212202	0	0	0	0	0	
SK00562698	G2804040	40	40	7116997402	7116212202	0	0	0	0	0	
SK00559756	G2803945	39	45	7116997402	7114545602	1	0	0	1	8.5	
SK00559755	G2803952	39	52	7116997402	7114545602	0	1	0	0	1.5	
SK00559752	G2803950	39	50	7116997402		0	1	0	1	6	
SK00559753	G2803950	39	50	7116997402		0	0	0	0	0	

**16 pav.** Eksperimento metu naudojamas įrankis paskaičiuoti staklių derinimo laiką

Laido kodas yra sudarytas iš 8 skaitmenų. Pirmi 4 apibrėžia izoliacijos tipą (optimizacijai įtakos neturi ir yra ignoruojamas), antrame ketvirtuke yra užkoduoti skerspjūvio plotas bei laido spalva (po 2 skaitmenis). Sukurtos 4 kolonėlės-trigeriai, kurie lygina esamo užsakymo kombinaciją su prieš tai buvusią, ir sudeda vėliavas, parodančias pakeitimus. Staklių derinimo laikai yra sumuojami prie kiekvieno užsakymo. Galu galiausia gaunama visų pakeitimų suma.

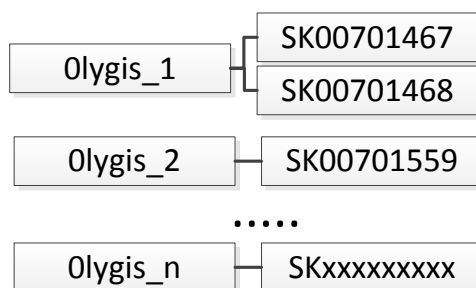
### 3.3. Dendrogramos kūrimas

Norint užtikrinti mažiausią staklių prastovos laiką, yra kuriamas algoritmas, taikant anksčiau aprašytą hierarchinį klasterizavimo metodą. Dabartinis CAO programos algoritmas yra pagrįstas tik matematiniais skaičiavimais. Tobulinant optimizacijos procesą, yra taikomos intelektualiosios technologijos gilesnei užsakymo sekos analizei. Kuriamas algoritmas yra pagrįstas dendrogramų teorija.

Algoritmas bus iš esmės kito pobūdžio procesas, taigi pradėdama yra nuo neoptimizuotos sekos, paminėtos anksčiau. Pirmas žingsnis, paprastas rūšiavimas, palei du kriterijus, iš pradžių palei kontaktas1, paskui palei kontaktas2. Rūšiavimo tvarka – mažėjimo. Rūšiavimo tikslas – išdėlioti užsakymus su tais pačiais kontaktais vienas šalia kito.

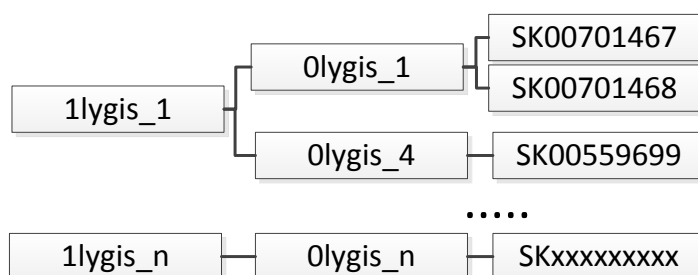
Antras žingsnis – nulinio lygio klasterizavimas (17 pav.). Visi užsakymai, tarp kurių Staklių derinimo laikas yra nulinis (užsakymai skiriasi tik laido ilgiu), yra sugrupuojami. Esmė – sumažinti

aibę užsakymu. Bendras staklių derinimo laikas lieka tas pats, nulinio lygio grupių skaičius – 239. Mažesnė aibė užsakymo yra lengviau bei greičiau analizuoti.



**17 pav.** Nulinio lygio klasterizavimas

Trečias žingsnis – pirmojo lygio klasterizavimas (18 pav.). Paieška nuliniu grupių, su kontaktu tik iš vienos pusės, kuriuos galima apjungti su kitomis grupėmis, turinčiomis tokius kontaktus ir to paties laido. Esmė – dar sumažinti užsakymu aibė. Tokių grupių kiekis – 175. Pirmo lygio grupės viduje esantys užsakymai neturi tokių skirtumų, kurie reikalautų staklių derinimo. Todėl tolimesniame optimizacijos procese bet kuria pirmo lygio grupė bus laikoma objektu. Prastovos laikas po šito žingsnio **781,5 minutės**.



**18 pav.** Pirmojo lygio klasterizavimas

Visos pirmojo lygio grupės skiriasi viena nuo kitos nors vienu komponentu. Sekančių žingsnių yra sujungiamos grupės, tarp kurių yra minimalus pakeitimas (staklių derinimo laiko atžvilgiu) – laido spalva. Gautų grupių kiekis – 76, kur į juos įeinančių pirmo lygio objektų varijuojasi nuo 1 iki 13. Antro lygio grupėse tarp komponentų yra nustatytas 1,5 minutės staklių derinimo laikas, neatsižvelgiant į vidinį išdėstymą.

Grupė2	Grupė1	Laidas	Laido skerspjūvis	Laido spalva	Kontaktas1	Kontaktas2
2lygis_16	1lygis_28	G2804047	40	47	7116478602	7116908802
2lygis_16	1lygis_29	G2804057	40	57	7116478602	7116908802
2lygis_16	1lygis_30	G2804059	40	59	7116478602	7116908802
2lygis_16	1lygis_31	G2804060	40	60	7116478602	7116908802
2lygis_16	1lygis_32	G2804070	40	70	7116478602	7116908802
2lygis_16	1lygis_33	G2804090	40	90	7116478602	7116908802
2lygis_16	1lygis_34	G2804017	40	17	7116478602	7116908802

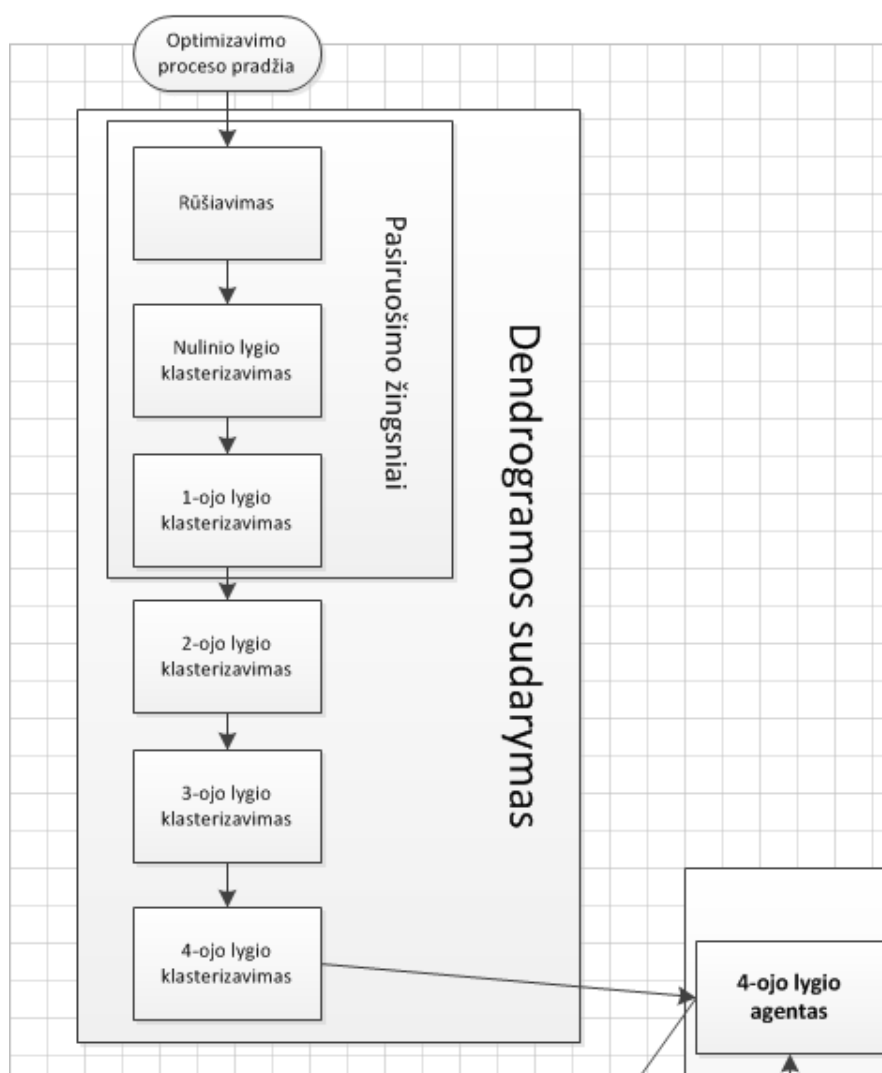
**19 pav.** Antro lygio grupė

Antro lygio grupės objektai bus analizuojami jau su modifikuotų komponentų rinkinių. Laido spalvos priderinimu tarp objektų užsiims žemiausia lygio agentas. Į trečio lygio grupės objektai yra suformuojami neatsižvelgiant į laido skerspjūvį. Staklių derinimo laikas grupės viduje tarp antro lygio grupių – 4 minutės. Po šito žingsnio tolimesniame klasterizavime atkreinta komponentas laidas.

Grupė3	Grupė2	Grupė1	Laidas	Laido skerspjūvis	Kontaktas1	Kontaktas2
3lygis_2	2lygis_2	1lygis_4	G2803980	39	7116997402	7116478602
3lygis_2	2lygis_2	1lygis_5	G2803990	39	7116997402	7116478602
3lygis_2	2lygis_3	1lygis_6	G2804048	40	7116997402	7116478602

20 pav. Trečio lygio grupė

Sekantis žingsnis – jungti trečio lygio grupes, nepaisant vieno kontakto skirtumą. Staklių derinimo laikas tarp trečio lygio grupėmis svyruoja nuo 4,5 minutės tik kontakto skirtumo atveju iki 9,5 minutės kai užduotys skiriasi vienu kontaktu ir laidu. Ketvirto lygio grupių kiekis – 14.

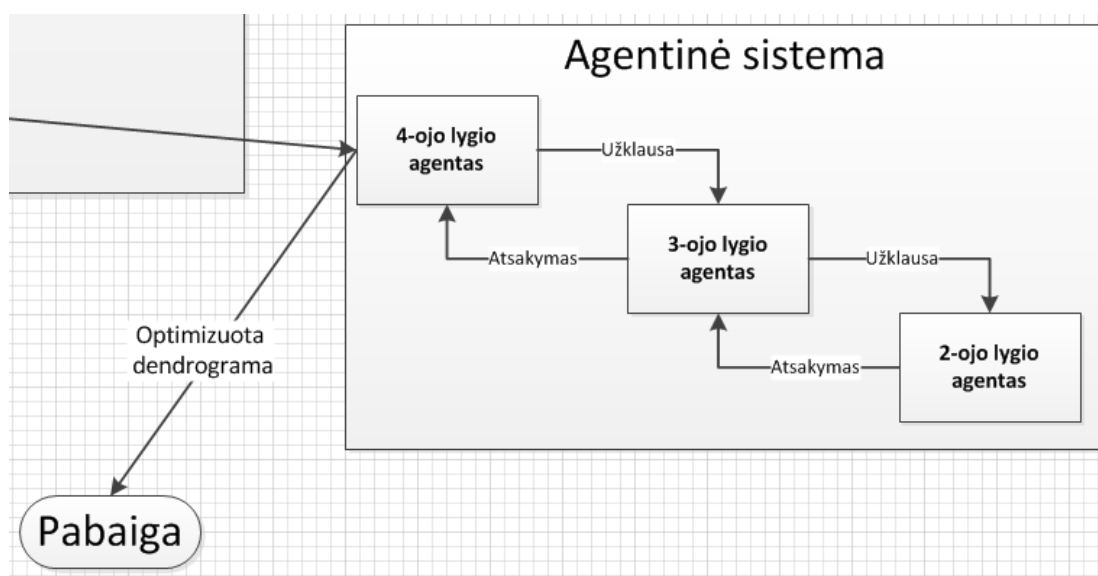


21 pav. Dendrogramos sudarymo procesas

Toliau klasterizavimo procesas nėra vykdomas. Ketvirto lygio objektai yra unikalūs, jie turi visus skirtingus komponentus. Tolesnis optimizacijos procesas pereina pas intelektualiuosius agentus. Bendras dendrogramos sudarymo procesas pavaizduotas 21 paveiksle.

### 3.4. Dendrogramos optimizavimo agentinė sistema

Po keturių klasterizavimo lygių neįmanoma daugiau optimizuoti seka taikant hierarchinį klasterizavimo metodą. Siekiant dar sumažinti bendrą sekos padaromų pakeitimų, yra siūlomas intelektualių programinių agentų sprendimas. Agentinė sistema sudaryta iš trijų agentų, pavaizduota 22 paveiksle. Agentai bendrauja tarpusavyje, siekdami užtikrinti optimaliai modifikuotą pirmo lygio grupės objektų seka. Jų tikslas – analizuoti jo lygyje esančių grupių komponentus ir teikti pasiūlymus dėl grupės arba objekto sekos vietos pakeitimų, pagrindžiant tai staklių derinimo laiko mažinimu.



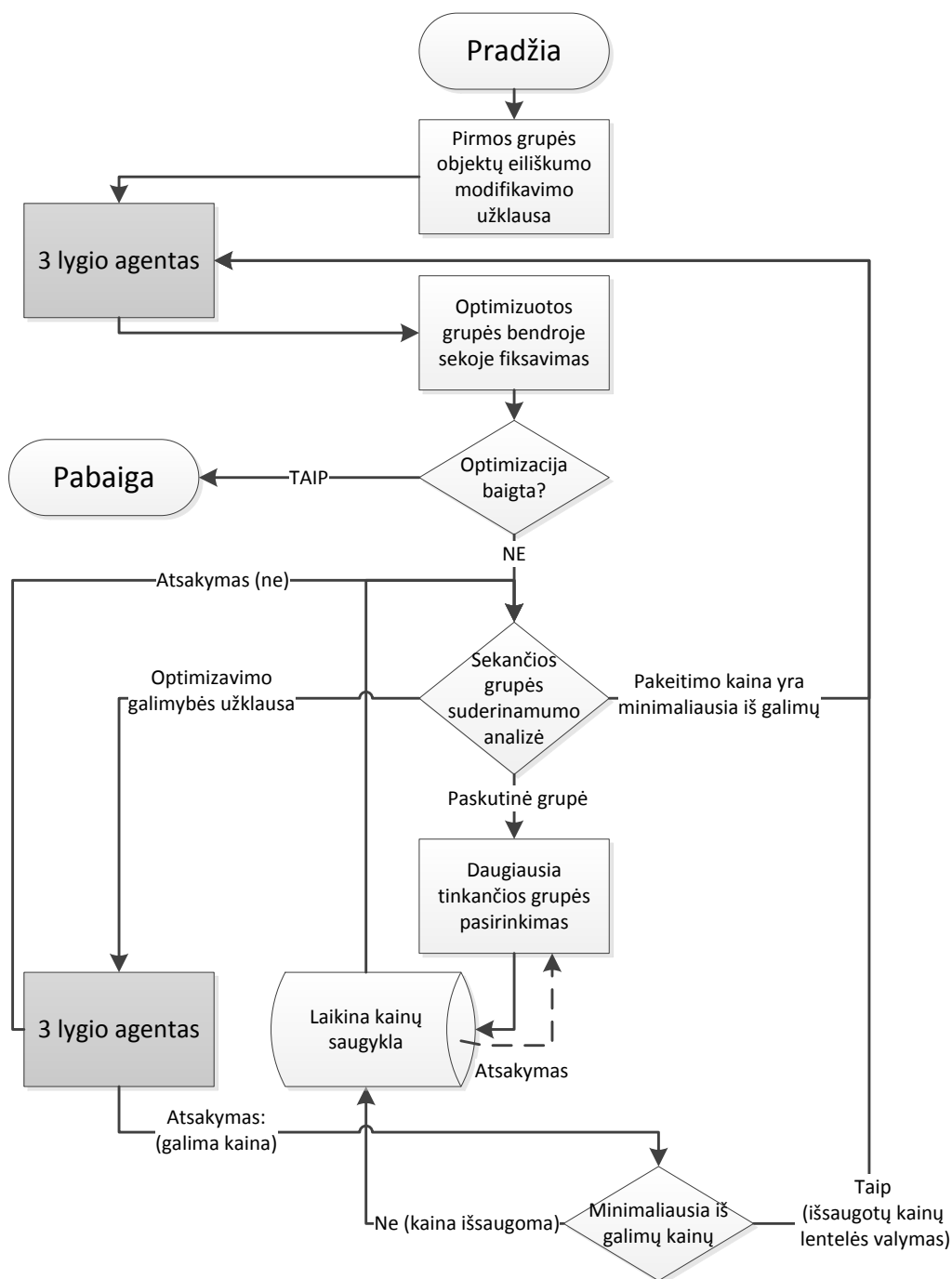
22 pav. Agentinė sistema

#### 3.4.1. Ketvirtojo lygio agentas

Generalinį dendrogramos pertvarkymą daro ir pradeda aukščiausio, ketvirto lygio agentas. Jo tikslas – sudėlioti ketvirto lygio grupės taip, kad bendras staklių derinimo laikas būtų mažiausias. Proceso eiga, pavaizduota 23 paveiksle:

- 1) užsakomas pirmos sekoje esančios ketvirto lygio grupės objektų eilės modifikavimas iš trečio lygio agento;
- 2) ta pati grupė yra fiksuojama bendroje sekoje;
- 3) analizuojamos sekoje esančios sekančios grupės suderinamumas su fiksuota seka;
- 4) apskaičiuojamas staklių derinimo laikas tarp paskutinio pirmos grupės objekto ir pirmo antro grupės objekto (jungties taškas);

- 5) jeigu jungties taške staklių derinimo laikas yra minimaliausias iš galimų (keičiamas tik vienas kontaktas), seka yra užfiksuojama, bet prieš tai yra nusiunčiama trečio lygio agentui vidiniam modifikavimui. Procesas prasideda iš naujo, nuo sekančio grupės derinimo prie fiksuotos sekos;



**23 pav.** Ketvirto lygio agento proceso algoritmas

- 6) jeigu staklių derinimo laikas yra didesnis už minimaliausiai egzistuojantį, siunčiama užklausa 3 lygio agentui – ar galima pamodifikuoti seką taip, kad staklių derinimo laikas būtų sumažintas;

- 7) jeigu gautas atsakymas yra ne, agentas užrašo grupės mažiausią staklių derinimo laiką ir nepriima grupes į fiksuotą seką, ir pradeda analizuoti sekančią grupę;
- 8) kai yra gautas atsakymas, agentas sprendžia ar jungties taške staklių derinimo laikas yra minimaliausias iš galimų seka yra užfiksuojama. Procesas prasideda iš naujo, nuo sekančio grupės derinimo prie fiksuotos sekos; jeigu staklių derinimo laikas yra didesnis už minimaliausiai egzistuojantį, agentas užrašo grupės staklių derinimo laiką ir nepriima grupes į fiksuotą seką, ir pradeda analizuoti sekančią grupę;
- 9) jeigu per visą seką neranda idealiai tinkančios grupės, pasirenka pirmą iš daugiausia tinkančių;
- 10) kai analizavimui lieka paskutinė grupė, yra kreipiamasi į trečio lygio agentą, kad grupė būtų pertvarkyta iš vidaus, siekiant užtikrinti minimalaus prastovos laiko dėl staklių derinimo darbų;
- 11) po paskutinės grupės pridėjimo prie fiksuotos sekos, optimizacijos procesas yra baigtas, užduočių seka yra perduodama staklėms, kur ir bus gaminama.

### **3.4.2. Trečiojo lygio agentas**

Trečio lygio agentas turi kito tipo tikslą. Veikdamas žemesniame lygmenyje, negu ketvirtojo lygio agentas, jis yra atsakingas už trečio lygio objektų suderinimą palei laido skerspjūvio plotą. Agentas nevykdo pastovios veiklos, įsijungdamas į optimizavimo procesą tik gavus užklausą iš ketvirto agento. Užklausa būna dviejų tipų:

- suderinamumo užklausa;
- vidinis grupėje esančių objektų sekos modifikavimas-optimizavimas.

Suderinamumo užklausa gaunama iš ketvirto lygio agento. Užklausa sudėtyje yra pateikiamas paskutinio fiksuotos sekos pirmo lygio objekto komponentai ir derinamos grupės indeksas. Agentas turi išanalizuoti derinamos grupės trečio lygio grupės objektus. Jeigu analizuojamoje grupėje egzistuoja objektas su tokių pačių laido skerspjūvio ploto komponentu, siunčiama užklausa antro lygio agentui, dėl spalvos derinimo. Gavus atsakymą nuo antro lygio agento, yra apskaičiuojamas galimo staklių derinimo laikas, kuri yra toliau gražinama atgal ketvirto lygio agentui. Jeigu objektų su tokių pačių laido skerspjūvio ploto komponentu nerasta, yra apskaičiuojamas galimas staklių derinimo laikas, kuris yra toliau gražinamas atgal ketvirto lygio agentui.

Antra agento atliekama funkcija – grupės vidinė objektų eiliškumo optimizavimas. Visi grupėje esantys antro lygio objektai yra pertvarkomi taip, kad prie skerspjūvio ploto pakeitimo taško atsidurtų dvi tokios grupės, kurių kontaktų poros yra vienodos, jeigu tokios egzistuoja. Toliau, antro lygio agentui siunčiama užklausa suoptimizuoti objektų seka antro lygių grupės lygmenyje. Iš



esmės, funkcijos algoritmas yra labai panašus į ketvirto lygio agento atliekamos funkcijos algoritmą, tiesiog seka yra optimizuojama pagal kitus kriterijus. Gavus atsakymą apie darbų įvykdymą, siunčiamas atsakymas ketvirto lygio agentui apie darbų įvykdymą.

### 3.4.3. Antrojo lygio agentas

Žemiausio lygio agentą nuo trečio lygio agento skiria tik derinimo komponentas. Antro lygio objektas derina pirmo lygio grupės objektus palei vienodo skerspjūvio laidus palei spalvą.

Agentas nevykdo pastovios veiklos, ir įsijungia į procesą tik gavęs užklausą, kurį būna dviejų tipų (tokio pat, kaip ir trečio lygio agento):

- suderinamumo užklausa;
- vidinis grupėje esančių objektų sekos modifikavimas-optimizavimas.

Pirmo tipo užklausoje, agentas, radęs tinkamą pirmo lygio grupės objektą, raportuoja apie *minimalaus* staklių derinimo laiko egzistavimą tarp fiksuotos sekos paskutinio antro lygio galimos analizuojamos ketvirto lygio grupės antro lygio grupės objektų. Jeigu reikalingas objektas nėra randamas, siunčiamas pasiūlymas su mažiausia iš galimų staklių derinimo laiku, kuris yra vėliau persiunčiamas ketvirto lygio agentui, kuris savo ruožtu, priima sprendimą apie tos arba anos grupės fiksavimą optimizuotoje sekoje.

		Grupė	Laidas	Laido skerspjūvis	Laido spalva	Kontaktas1	Kontaktas2
3lygis_5	2lygis_13	1lygis_61	G2804047	40	47	7116478602	7116908802
		1lygis_62	G2804057	40	57	7116478602	7116908802
		1lygis_63	G2804059	40	59	7116478602	7116908802
		1lygis_64	G2804060	40	60	7116478602	7116908802
		1lygis_65	G2804070	40	70	7116478602	7116908802
	2lygis_14	1lygis_34	G2804048	40	48	7116478602	7116997402
		1lygis_35	G2804060	40	60	7116478602	7116997402
		1lygis_36	G2804070	40	70	7116478602	7116997402
	2lygis_15	1lygis_84	G2804040	40	40	7116478602	7116478602
		1lygis_85	G2804041	40	41	7116478602	7116478602
		1lygis_86	G2804050	40	50	7116478602	7116478602
		1lygis_87	G2804080	40	80	7116478602	7116478602



		Grupė	Laidas	Laido skerspjūvis	Laido spalva	Kontaktas1	Kontaktas2
3lygis_5	2lygis_13	1lygis_61	G2804047	40	47	7116478602	7116908802
		1lygis_62	G2804057	40	57	7116478602	7116908802
		1lygis_63	G2804059	40	59	7116478602	7116908802
		1lygis_65	G2804070	40	70	7116478602	7116908802
		1lygis_64	G2804060	40	60	7116478602	7116908802
	2lygis_14	1lygis_35	G2804060	40	60	7116478602	7116997402
		1lygis_34	G2804048	40	48	7116478602	7116997402
		1lygis_36	G2804070	40	70	7116478602	7116997402
	2lygis_15	1lygis_84	G2804040	40	40	7116478602	7116478602
		1lygis_85	G2804041	40	41	7116478602	7116478602
		1lygis_86	G2804050	40	50	7116478602	7116478602
		1lygis_87	G2804080	40	80	7116478602	7116478602

24 pav. Antro lygio agento pasiūlymas

Vykdamas antro tipo užklausą, agentas kaip tik galima derina antro lygio grupes, kad bendras trečio grupės objekto staklių derinimo laikas būtų kuo mažesnis. Vizualizuotas proceso pavyzdys pateiktas 24 paveiksle.

Agentas pastebi, kad dviejose antro lygio grupėse yra pirmo lygio objektai, turintys vienodą komponentą, į kurį nebuvo atžvelgiama trečio lygio klasterizavimo metu (šiuo atveju – laido spalva). Pakeitus vietomis pažymėtus objektus, trečio lygio objektų gamybai paruošimo laikas sumažėjo 1,5 minutės.

Sekos optimizacija antro lygio agento lygmenyje yra iš esmės keliaujančio pirklio uždavinys. Šitam uždaviniui spręsti buvo išanalizuoti 3 metodai: arčiausio kaimyno, arčiausio kaimyno su perstatymais ir perrinkimo. Efektyviausiai būtų taikyti visos trečio lygio grupės perrinkimą, tačiau tokia grupė gali būti sudaryta iš 13 pirmos grupės objektų. Analizuojamoje situacijoje, pirma užduotis yra fiksuota, todėl galimų variantų skaičius yra lygus 479001600. Iteracijos skaičius yra aiškiai per didelis, todėl yra nuspręsta netaikyti perrinkimo metodą grupėms, kuriuose yra daugiau negu 10 pirmo lygio objektų. Tokių atveju maksimalus variantų skaičius yra 362880. Grupėms, kurių pirmo lygio objektų skaičius yra didesnis negu 10, bus taikomas arčiausio kaimyno metodas.

#### 3.4.4. Antro lygio agento realizavimas

Kad sėkmingai realizuoti optimizavimo algoritmą, reikia aprašyti gamybinės užduoties struktūrą. Visos užduotys bus sujungtos į dinaminį sąrašą, dėl to kad užsakymų kiekis nėra žinomas. Abstrakčioje užduotyje yra 4 komponentai: laido skerspjūvio plotas, laido spalva ir abiejuose galuose tvirtinami kontaktai.

```
struct cutlead {  
    string cutlead_no;  
    int WireCrossSect;  
    int WireColor;  
    char TerminalL[10];  
    char TerminalR[10];  
    struct cutlead *next;  
}
```

Gavus komandą iš trečio lygio agento ieškoti geriausiai tinkančio prie fiksuotos sekos užsakymo, agentas analizuoja visą jam pavaldžią seką ir grąžina rezultatą.

```
cutlead find_best (cutlead *last_fixed){  
    float temp_result, best_result;  
    cutlead *temp_point, best_point;  
    temp_point=current_list;  
    best_result=13;  
    best_point=temp_point;  
    do  
    {  
        temp_result=compare(*last_fixed, *temp_point);  
        if (temp_result<best_result)  
        {
```

```

        best_result=temp_result;
        best_point=temp_point;
        temp_point=temp_point->next;
    }
    else
    {
        temp_point=temp_point->next;
    }
while (temp_point->next=NULL);
return best_point;
}

```

Funkcija *find\_best* taip pat yra naudojama arčiausio kaimyno ieškant. Išskviečiant ją, prie fiksuotos sekos yra pridamas gražintas rezultatas, kuris savo atveju yra šalinamas iš analizuojamos sekos. Tokiu būdu seka yra optimizuojama arčiausio kaimyno metodu. Toks algoritmas turi savo trūkumų, dėl to kad paima patį pirmą tinkantį užduotį, ir netgi jeigu tokiai egzistuoja kita užduotis, su tokiu pačiu staklių derinimo laiku, bet, galimai geresne tolima eiga.

### 3.5. Optimizacijos rezultatai

Pasirinktai sekai iš atsitiktinai paimtų užduočių iš bendros duomenų bazės *Master Data*, CAO programos sudarytas optimizuotas užduočių tvarkaraštis skiria **710,5 minutės** staklių derinimui ir reikalaujantis padaryti **187** pakeitimų. Optimizacija, daryta pagal sukurtą algoritmą, skiria **645,5 minutės** staklių derinimo darbams ir reikalauja padaryti **175** pakeitimus. Gautas rezultatas yra geresnis 65 minutėmis už dabar veikiantį, t.y. **9,18%** viso derinimo laiko yra sutaupoma. Tai buvo pasiekta eliminuojant CAO algoritmo trūkumus.

Anksčiau minėta, kad CAO algoritmas naudoja tikrai arčiausio kaimyno metodo. Atsitiktinai pasirinktoje sekoje egzistuoja užsakymai, kuriose užprašomas laidas be kontaktų arba su tvirtinamu kontaktu tik ant vieno laido galo. Tokie užsakymai, kaip taisyklė, turi mažesnę staklių derinimo laiką, t.y. reikalauja mažiau veiksmų ir laiko derinant stakles. Optimizuojant seką, iš pradžių yra suplanuojami kaip tik tie užsakymai (7 pav.). Sukurtame algoritme šita problema yra eliminuojama, taikant užduočių klasterizavimą. Dažniausia, „nepilnam“ užsakymui bus rastas kuris turi tokius pat komponentus plius dar vienas komponentas. Sekoje, planuojant „nepilną“ užsakymo po „pilno“ užsakymo, staklių derinimo laikas nebus reikalingas.

Taip pat klasterizacijos pagalba yra žymiai sumažinama optimizuojama seka. Užduotis, turintis vienodus komponentus, yra apjungiami į grupes, kurios yra analizuojamos kaip vienas sekos objektas. Tokiose grupėse tarp užsakymų staklių derinimo laikas yra lygi nuliui. Atliekant praktinius CAO programos algoritmo tyrimus, buvo pastebėta, jog dažniausia vieno kontakto užsakymai turi mažesnio kontakto kodą (pvz. 7112508002), kai kontaktai, turintys aukštą kodą (pvz. 7116997402) dažniausia pasitaiko pilno komponavimo užsakymuose. Taigi, rūšiavimas nuo

didžiausio iki mažiausio ir taikomi klasterizacijos žingsniai beveik panaikina mažo staklių derinimo laiko problemą.

### **3.6. Tolimesni tyrimai**

Atliekant sukurto algoritmo rezultatų tyrimą, buvo pastebėti keli algoritmo netobulumai, ties kurių galima vykdyti tolimesnius tyrimus bei tobulinimą. Pirmas liečia klasterizavimo lygių kiekį. Klasterizavimo žingsniai, atliekami po užduočių be pakeitimų, tik riboja žemesnių lygių agentų veikimo sritį, neleidžiant jiems ieškoti geriausio sprendimo. Reikalingas papildomas tyrimas, kuris padėtų nustatyti kuriam žingsnyje reikia sustoti grupuoti užsakymus.

Taip pat, agentinės sistemos efektyvumas irgi reikalauja tyrimo. Tolimesniuose tyrimuose planuojama išanalizuoti ir palyginti du algoritmus. Pirmas susidėtų iš dviejų klasterizavimo žingsnių, padedančių atsikratyti „nepilnų“ užsakymų ir veidrodinių užsakymų, ir sudarytiems objektams optimizuoti bus taikomi žinomi optimizacijos algoritmai: arčiausio kaimyno bei skruzdėlių kolonijos. Antras algoritmas susidėtų iš kelių klasterizacijos žingsnių (bus nustatytas efektyviausiai tinkantis skaičius), ir vieno besimokančio agento.

Per planuojamus tolimesnius tyrimus žadama rasti geriausiai konkrečiai situacijai tinkančio algoritmo, kuris galėtų rasti vieną iš optimaliausių užduočių tvarkaraščio.

## IŠVADOS IR REKOMENDACIJOS

- 1) Atlikti tyrimai parodė esminius šiuo metu CAO programinėje įrangoje ir panašiose gamybos valdymo ir optimizavimo programose naudojamų metodų ir jų vykdančių algoritmų trūkumus, buvo sukurtas naujas metodas, pritaikytas konkrečios gamybinės situacijos optimizacijos problemoms spręsti, pagrįstas klasterizacijos metodais kuriant gamybos užduočių hierarchinę struktūrą, kuri vėliau yra optimizuojama išmaniųjų agentų sistemos pagalba.
- 2) Eksperimento metu atsitiktinai pasirinktai 500 užduočių sekai buvo pritaikytas sukurtas metodas. Pasiūlytu metodu optimizuota gamybos užduočių seka leido pagerinti įrenginių panaudos efektyvumą **9,18%**, palyginus su procesų valdymu originaliu CAO programoje naudojamu metodu (**645,5** minutės staklių derinimo darbų vietoje **710,5** minučių, taip pat **175** staklių derinimo kartu vietoj **187**).
- 3) Atlikta naujai sukurto metodo taikymo gamyboje analizė parodė, kad tobulinimo procesas reikalauja gilesnių tyrimų, siekiant išsiaiškinti efektyviausią klasterizacijos žingsnių skaičių ir agentinės sistemos produktyvumą ir tobulinimo galimybes. Ateityje numatyti tyrimų darbai – sukurti besimokančių agentų sistemą gautos dendrogramos sekos optimizavimui, ir jos veikimo efektyvumo palyginimas su skruzdėlių kolonijos optimizacijos metodu.

## LITERATŪROS SĄRAŠAS

- 1) Klemmt, A.; Horn, S.; Wiegert, G.; Wolter, K.-J. 2009. Simulation-based optimization vs. mathematical programming: A hybrid approach for optimizing scheduling problems, *Robotics and Computer-Integrated Manufacturing* 25 (6): 917-925.
- 2) Gupta, A.K.; Sivakumar, A.I. 2006. Job shop scheduling techniques in semiconductor manufacturing, *International Journal of Advanced Manufacturing Technology* 27 (1): 1163-1169.
- 3) Berrichi, A.; Yalaoui, F.; Amodeo, L.; Mezghiche, M. 2009. Bi-Objective Ant Colony Optimization approach to optimize production and maintenance scheduling, *Computers & Operations Research* 37 (9): 1584-1596.
- 4) Boland, N.; Dumitrescu, I.; Froyland, G.; Gleixner, A. M. 2009. LP-based disaggregation approaches to solving the open pit mining production scheduling problem with block processing selectivity, *Computers & Operations Research* 36 (4): 1064-1089.
- 5) Bohle, C.; Maturana, S.; Vera, J. 2010. A robust optimization approach to wine grape harvesting scheduling, *European Journal of Operational Research* 200 (1): 245-252.
- 6) Ferrer, J.C.; Cawbey, A.M.; Maturana, S.; Toloza, S., Vera, J. 2008. An optimization approach for scheduling wine grape harvest operation, *International Journal of Production Economics* 112 (1): 985-999.
- 7) Andziulis, A.; Dzemydienė, D.; Steponavičius, R.; Jakovlev, S. 2011. Comparison of two heuristic approaches for solving the production scheduling problem, *Information Technology and Control* 40 (2): 118-122.
- 8) Cela, R.; Bollain, M.H. 2012. New cluster mapping tool for the graphical assessment of non-dominated solutions in multi-objective optimization, *Chemometrics and Intelligent Laboratory Systems* 114 (x): 72-86.
- 9) Guidi, L.; Ibanez, F.; Calcagno, V.; Beaugrand, G. 2008. A new procedure to optimize the selection of groups in a classification tree: Applications for ecological data, *Ecological Modelling* 220(x): 451-461
- 10) Duomenų gavyba ir daugiaagentinės sistemos [interaktyvus] [žiūrėta 2013-05-06] <http://www.oksl.ktu.lt/studijos/T120B130/index.html>
- 11) А. Н. Швецов, „Агентно-ориентированные системы: от формальных моделей к промышленным приложениям“, 2008
- 12) Stephen Haag, „Management Information Systems for the Information Age“, 2006

- 13) Пахомов, Б. 2008. Самоучитель C/C++ и C++ Builder 2007. Санкт-Петербург: “БХВ-Петербург”.
- 14) Blonskis, J.; Bukšnaitis, V.; Jusas, V.; Marcinkevičius, R.; Misevičius, A. 2005. C++ Builder. Kaunas: “Smaltijos leidykla”.
- 15) Taha, A. H. 2007. Operation research: an introduction. New Jersea: “Pearson Education Inc.”.

## **PRIEDAI**



### Kompaktinė plokštelė

Kompaktinis diskas, kuriame pateikiami:

Darbo katalogas: „Ilja\_Gusiatin/..“.

Darbo aprašymas: „../Ilja\_Gusiatin\_baigiamasis\_darbas.docx“.

Darbo aprašymas: „../Ilja\_Gusiatin\_baigiamasis\_darbas.doc“.

Darbo prezentacija: „../Ilja\_Gusiatin\_prezentacija.pptx“.

Darbo prezentacija: „../Ilja\_Gusiatin\_prezentacija.ppt“.

Darbo diagramos: „../diagramos ir paveikslai/..“.