KAUNO TECHNOLOGIJOS UNIVERSITETAS
FUNDAMENTALIŲJŲ MOKSLŲ FAKULTETAS
TAIKOMOSIOS MATEMATIKOS KATEDRA

Viktoras Kravcovas

Genetinio algoritmo parametrų tyrimas sprendžiant gamybos planavimo uždavinį

Magistro darbas

Vadovas
doc. dr. N. Listopadskis

KAUNAS, 2013

KAUNO TECHNOLOGIJOS UNIVERSITETAS

FUNDAMENTALIŲJŲ MOKSLŲ FAKULTETAS

TAIKOMOSIOS MATEMATIKOS KATEDRA

TVIRTINU
Katedros vedėjas
doc. dr. N. Listopadskis

Genetinio algoritmo parametrų tyrimas sprendžiant gamybos planavimo uždavinį

Taikomosios matematikos magistro baigiamasis darbas

Vadovas
doc. dr. N. Listopadskis
2013 06 07

Recenzentas
doc.dr. K.Šutienė
2013 06 07

Atliko
FMMM 1 gr. stud.
V. Kravcovas
2013 06 07

KAUNAS, 2013

KVALIFIKACINĖ KOMISIJA

Pirmininkas: Rimantas Rudzkis, profesorius (VU)

Sekretorius: Eimutis Valakevičius, docentas (KTU)

Nariai: Jonas Valantinas, profesorius (KTU)

Vytautas Janilionis, docentas (KTU)

Vidmantas Povilas Pekarskas, profesorius (KTU)

Zenonas Navickas, profesorius (KTU)

Arūnas Barauskas, dr., direktoriaus pavaduotojas (UAB „Danet Baltic“)

Kravcovas V. Analysis of genetic algorithm parameters for manufacturing scheduling problem: Master's work in applied mathematics / supervisor dr. assoc. prof. N. Listopadskis; Department of Applied mathematics, Faculty of Fundamental Sciences, Kaunas University of Technology. – Kaunas, 2013. – 58 p.

SUMMARY

The main topic of this research paper is analysis of genetic algorithm parameters, when solving flexible flow shop (FFS) manufacturing scheduling problem. Scheduling, in general case, helps to solve the problems of efficient limited resources management. This research is based on approach of real manufacturing scheduling system and the data used in this paper is from currently working shop furniture factory.

The main objective of this paper is to compare two different combinatorial optimisation methods and determine which of them is the most suitable for solving FFS problem. The main objective consists of three major tasks:

1. We need to determine the best genetic mutation and genetic selection operator for genetic algorithm when solving FFS problem;
2. We need to determine the dependence between genetic algorithm parameters and the size of data matrix;
3. We need to compare two different algorithms – the genetic algorithm and the heuristic algorithm based on theory of constraints (TOC) and then determine which of them is the best for FFS problem.

The research paper consist of five main topics: the analysis of scheduling tasks and the methods used to solve this problem; the analysis of genetic algorithm operators and their adaptation to FFS problem; the research of genetic algorithm and TOC heuristic when solving FFS problem; the conclusion of the research; the application of achieved conclusions and recommendations for further research.

In the end author concludes that the best algorithm for solving FFS problem is genetic algorithm with $\mu + \lambda$ selection method and swap mutation operator. The parameters estimate for three different data matrix sizes were almost constant (see table 8). In distant future the application software used in this research will be updated for better representation of solution with some control mechanism of schedule progress and it will be used for scheduling real manufacturing proceses.

Turinys

ĮVADAS 8
1. TVARKARAŠČIŲ SUDARYMO UŽDAVINIO ANALIZĖ 10
1.1. Tvarkaraščių klasifikavimo metodai 10
1.2. Tvarkaraščių sudarymo programinės priemonės 11
1.3. Realios gamybos pavyzdys 13
1.4. Uždavinio matematinis modelis 14
1.5. Tvarkaraščių sudarymo metodai 17
2. Tyrimo metodai 21
2.1. Pradinių duomenų struktūra 21
2.2. Genetinio algoritmo paradigma 22
2.2.1. Sprendinio reprezentacija 23
2.2.2. Tikslo funkcija 24
2.2.3. Genetinio kryžminimo operatorius 25
2.2.4. Mutavimo operatorius 28
2.2.5. Pradinė populiacija 29
2.2.6. Genetinės atrankos operatoriai 30
2.3. Apribojimo teorijos pritaikymas sudarant gamybinius tvarkaraščius 32
2.4. Programinė įranga 35
3. TIRIAMOJI DALIS 38
3.1. Parametrų tyrimo schema 38
3.2. Tyrimo rezultatai 39
3.3. Genetinio algoritmo ir TOC euristikos palyginimas 49
IŠVADOS 51
Rekomendacijos 52
Literatūra 53
Priedas Nr1. Rekombinacijos parametrų įtakos rezultatai (inversinė mutacija) 55
Priedas Nr2. Rekombinacijos parametrų įtakos rezultatai (sukeitimo mutacija) 57
Priedas Nr3 Kompaktinis diskas 59

Paveikslėlių turinys

Pav. 1.1 Kiekių valdymo lentelės pavyzdys....13
Pav. 1.2 Srauto fabriko pavyzdys [models]....15
Pav. 1.3 Lankstaus srauto fabriko pavyzdys [models]....15
Pav. 1.4 Darbų fabriko pavyzdys....17
Pav. 1.5 Tvarkaraščių sudarymo metodų klasifikacija....17
Pav. 2.1. Negalimas ir neleistinas sprendinys [9]....24
Pav. 2.2 Vienataškis kryžminimas....26
Pav. 2.3. Dalinai susietas kryžminimas....26
Pav. 2.4 Kryžminimas pagal tvarkos sąrašą.....27
Pav. 2.5 Pozicinis kryžminimas....27
Pav. 2.6 Darbų sąrašo matricomis paremtas pozicinis kryžminimas....28
Pav. 2.7 Inversiška mutacija....29
Pav. 2.8 Sukeičiantis mutavimas....29
Pav. 2.9 Genetinio algoritmo struktūra [9]....31
Pav. 2.10. Srautinės gamybos modelis....33
Pav. 2.11 Programoje realizuotas algoritmas paremtas apribojimų teorija....34
Pav. 2.13. Programinė įranga – duomenų įvesties langas....35
Pav. 2.12. Programos duomenų bazė....35
Pav. 2.14. Programinė įranga – parametrų langas.....36
Pav. 2.15. Programinė įranga – Ganto diagrama....37
Pav. 2.16. Programinė įranga – Eksperimentinis modulis....37
Pav. 3.1. Ruletės rato tyrimas (300 generacijų)....40
Pav. 3.2 Ruletės rato tyrimas (600 generacijų)....40
Pav. 3.3 Turnyrinės atrankos tyrimas....41
Pav. 3.4 $\boldsymbol{\mu + \lambda}$ atrankos metodo tyrimas....41
Pav. 3.5 $\boldsymbol{\mu + \lambda}$ atrankos metodo darbo laikas....42
Pav. 3.6 $\boldsymbol{\mu + \lambda}$ atrankos metodo populiacijų standratiniai nuokrypiai....42
Pav. 3.7 Imigrantų metodo pritaikymas tiriant $\boldsymbol{\mu + \lambda}$ konvergavimą....43
Pav. 3.8 Rekombinacijos parametrų tyrimas $\boldsymbol{\rho C = 0.3}$....43
Pav. 3.9 Rekombinacijos parametrų tyrimas $\boldsymbol{\rho C = 0.7}$....43
Pav. 3.10 Rekombinacijos parametrų tyrimas $\boldsymbol{\rho C = 0.3}$....44
Pav. 3.11 Rekombinacijos parametrų tyrimas $\boldsymbol{\rho C = 0.7}$....44
Pav. 3.12 Rekombinacijos parametrų tyrimas : paviršius $\boldsymbol{\rho M \times \rho C = popaverage}$....45

Pav. 3.13 Aplinkos parametrų tyrimas : chromosomos ilgis $\boldsymbol{L = 200}$ 46
Pav. 3.14 Algoritmo darbo laikų palyginimas.....46
Pav. 3.15 Aplinkos parametrų tyrimas : chromosomos ilgis $\boldsymbol{L = 50}$ 47
Pav. 3.16 Algoritmo darbo laikų palyginimas.....48
Pav. 3.17 Genetinio algoritmo sprendinys, kai pateikta $\boldsymbol{L = 200}$ genų ilgio chromosoma 49
Pav. 3.18 TOC euristikos algoritmo sprendinys, kai pateikta $\boldsymbol{L = 200}$ genų ilgio chromosoma 49

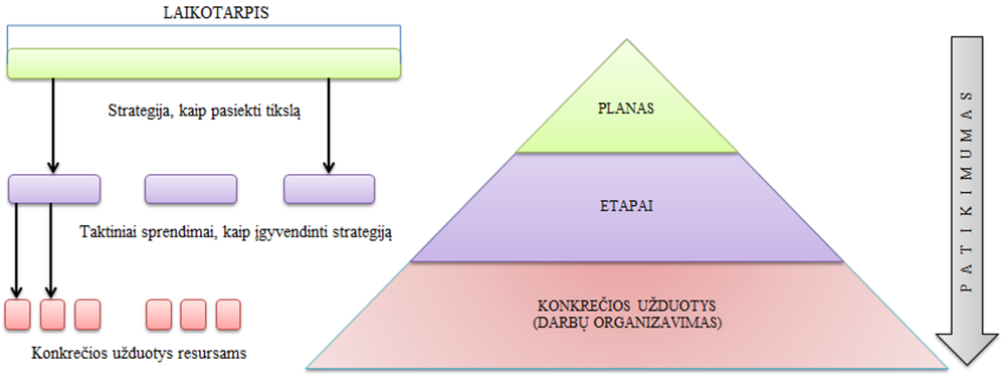
Lentelių turinys

Lentelė 1 Apribojimo identifikavimo pavyzdys.....32
Lentelė 2. Tyrimo schema.....38
Lentelė 3 Testuojamų duomenų matrica 40
Lentelė 4. $\boldsymbol{\mu + \lambda}$ parametrų palyginimas.....42
Lentelė 5. Rekombinacijos parametrų tyrimas : inversiškas mutavimas $\boldsymbol{-\eta}$.....44
Lentelė 6 Kokybės garantijos įverčiai kiekvienam parametrų rinkiniui (atlikus 40 generacijų) 46
Lentelė 7 Kokybės garantijos įverčiai kiekvienam parametrų rinkiniui (atlikus 20 generacijų) 48
Lentelė 8 Genetinio algoritmo parametrų apibendrinimas.....48
Lentelė 9 GA ir TOC algoritmų palyginimas.....49

ĮVADAS

Šiame darbe nagrinėsime gamybinių tvarkaraščių sudarymą, naudojant genetinį algoritmą. Požiūris į problemą ir jos sprendinius bus paremtas realiais pavyzdžiais iš prekybinės įrangos gamybos įmonės. Tvarkaraščių sudarymas, bendruoju atveju, padeda spręsti darbo su ribotais resursais optimizavimo problemas. Patys resursai gali būti labai įvairūs – darbo jėga, įrengimai, pinigai, įrankiai, medžiagos, energija ir t.t. Taip pat ir užduotys resursams gali būti labai įvairios – pradedant detalių, pusgaminių gamyba įvairiose pramonės šakose ir baigiant informacijos srautų valdymu kompiuterinėse sistemose. Dažniausiai resursams priskiriamas užduotis galima charakterizuoti tokiais faktoriais, kaip parengtumo momentas (laikas, kada užduotis gali būti pradėta vykdyti), baigties momentas (laikas, iki kurio užduotis turi būti pabaigta vykdyti), vėlavimo indeksas (arba svoris), apdorojimo laikas bei resurso panaudojimas. Be to, apjungiant tam tikras užduotis gauname struktūrą, kurią įprasta vadinti "darbu". Šiuo atveju darbu vadinsime tokią operaciją, kurios pagalba iš turimų pirminių žaliavų arba informacijos srauto, gaunamas galutinis produktas (rezultatai arba gaminiai). Priklausomai nuo pobūdžio, užduotys dažniausiai yra siejamos tam tikrais priklausomybės ryšiais (atlikimo eilės tvarka). Dėl to, kelių darbo centrų arba kompiuterių sistemų tvarkaraščiuose atsiranda prastovų. Šios prastovos tvarkaraščiuose leidžia sudaryti kiekvieno sprendinio įverčio funkciją, kurios tikslas dažniausiai būna trumpiausias visų darbų apdorojimo laikas, mažiausios prastovos, mažiausias bendras vėlinimas ir t.t.

Verta paminėti ir tai, kad šiame darbe sutapatinami du terminai – tai gamybos planavimas ir gamybos organizavimas. Iš tiesų, gamybos tvarkaraščių sudarymas priklauso gamybos organizavimo, o ne planavimo uždavinių grupei (1.1 pav).



Planavimo ir organizavimo sąryšis [TOCLT]

Šio tiriamojo darbo objektas yra lankstaus srauto gamybos tvarkaraščiai. Juos sudarinėsime pasinaudodami 2.4 skyriuje aprašyta programine įranga. Pagrindinis darbo tikslas yra palyginti du gamybinių tvarkaraščių sudarymo metodus (genetinį algoritmą bei apribojimų teorija pagrįstą euristinį metodą (2.11 pav.) ir iš jų atrinkti labiausiai tinkantį nagrinėjamų problemų sprendimui.

Šiame darbe spręsime 2 pagrindinius uždavinius. Pirmiausiai tirsime įvairias genetinio algoritmo parametrų modifikacijas :

a. genetinės atrankos operatoriai (ruletės ratas, turnyrinė atranka, $\mu + \lambda$ metodas);
b. genetinio kryžminimo operatoriai (pozicinis kryžminimas, suderintas su darbų sekos matricos kodavimu);
c. genetinės mutacijos operatoriai (sukeičianti mutacija ir inversijos mutacija);

bei ieškosime geriausių kiekvieno operatoriaus parametrų įverčių, sprendžiant lanksčios gamybos tvarkaraščių sudarymo uždavinius. Šiuos įverčius pasistengsime rasti kelių skirtingų dydžių pradinių duomenų matricoms.

Antrajame etape sudarysime apribojimų teorija (angl. *Theory of constraints –TOC*) pagrįstą euristinį metodą bei palyginsime jo efektyvumą su genetiniu algoritmu. Gauti rezultatai galės būtį taikomi sudarant lanksčios srautinės gamybos tvarkaraščius pagal 1.3 skyrelyje aprašytą matematinį modelį.

Tiriamąjį darbą sudaro 5 pagrindiniai skyriai

1. Tvarkaraščio sudarymo uždavinio analizė – šiame skyriuje nagrinėjami tvarkaraščių klasifikavimo kriterijai, tvarkaraščių resursai bei apribojimai. Analizuojama praktikoje taikoma programinė įranga. Aprašomi dažniausiai sutinkami gamybos modeliai bei tiriami tvarkaraščių sudarymo metodai.
2. Tyrimo metodai– šiame skyriuje įvarijama pradinių duomenų struktūra, nagrinėjami įvairūs genetinio algoritmo operatoriai bei pačio genetinio algoritmo mechanizmas. Sudaroma TOC euristika bei aprašoma sudaryta programinė įranga.
3. Tiriamoji dalis – šiame skyriuje vadovaudamiesi sudarytu eksperimento planu ieškosime geriausio iš tiriamų metodų tvarkaraščio sudarymo uždaviniui spręsti.
4. Išvados – šiame skyriuje pateiksime tyrimo metu gautus rezultatus
5. Rekomendacjos – šiame skyriuje pateiksime samprotavimus apie gautų rezultatų pritaikymą bei rekomendacijas tolimesniems tyrimams

1. TVARKARAŠČIŲ SUDARYMO UŽDAVINIO ANALIZĖ

Užduočių planavimo poreikis yra beveik kiekvienoje veiklos srityje. Dauguma planavimo uždavinių, kylančių įvairiose pramonės šakose yra pakankamai sudėtingi ir jų dažnai nepavyksta išspręsti standartiniais optimizavimo metodais, dėl to tokie uždaviniai neretai sprendžiami pasinaudojant kombinatorinio optimizavimo metodais. Kombinatorinis optimizavimas - tai įvairių modelių ir metodų taikymas sprendžiant diskrečios struktūros uždavinius. Ši matematikos šaka atsirado pakankamai neseniai ir didžiausią postūmį jai atsirasti lėmė spartus kompiuterinių technologijų vystymasis.

Nemaža dalis tvarkaraščių sudarymo uždavinių priklauso NP (angl. *nondeterministic polynomial time*) sudėtingumo kategorijai, t.y. tikslus sprendinys gaunamas tik atlikus pilną sprendinių perrinkimą, arba pritaikius metodus, kurie paremti šakų ir ribų metodo idėjomis. Tačiau, kadangi pilnas sprendinių perrinkimas reikalauja labai daug kompiuterio darbo laiko (dienom, mėnesiais ar net metais), dėl to tokio tipo uždaviniams spręsti naudojami apytiksliai euristiniai metodai, paremti tam tikromis taisyklėmis, evoliucijos idėjomis arba vietine paieška.[6]

1.1. Tvarkaraščių klasifikavimo metodai

Pakankamai nemaža dalis tvarkaraščių sudarymo uždavinių, kurie kyla įvairiose srityse, dažnai pasižymi panašiomis savybėmis. Atsiribojant nuo kiekvienos srities specifinių detalių, galima formuluoti tipinius tvarkaraščių sudarymo uždavinius. Toks uždavinių formalizavimas leidžia juos klasifikuoti bei parinkti tinkamus sprendimo metodus. Informacija, reikalinga tvarkaraščiui sudaryti dažniausiai apima šias sritis :

a. Užduočių trukmės bei tarpusavio sąryšiai (nuoseklumo, prioritetų ir t.t.);
b. Resursai, reikalingi užduotims atlikti, jų atsargos bei ribojimai ;
c. Tvarkaraščio realizavimo ypatybes ir t.t.

Dažniausiai literatūroje pasitaiko šie tvarkaraščių klasifikavimo kriterijai :

a. Pagal duomenų srauto pobūdį (statinis / dinaminis):

   Statinis – šiuo atveju visi tvarkaraščio duomenys apie darbų trukmes ir darbų srautą žinomi iš anksto . Pagrindinis uždavinys – visų darbų sustatymas į eilę;

   Dinaminis – šiuo atveju visa informacija apie darbų srautą iš anksto gali būti nežinoma ir paaiškėti tik darbų vykdymo eigoje. Šie uždaviniai dažnai siejami su realaus laiko planavimo uždaviniais, kuomet sprendimą apie darbų organizavimą reikia priimti operatyviai.

Iš praktinės pusės žiūrint, gamyboje statinis tvarkaraštis neretai gali ir nepasiteisinti, kadangi bet kuriuo momentu gali pasireikšti atsitiktiniai įvykiai, kurie nebuvo įvertinti sudarant tvarkaraštį (t.y. įrengimų gedimas, gamybinis brokas ir t.t.). Dėl to tenka priimti operatyvinius sprendimus, kurie iškraipo visą tvarkaraštį. Šiuo atveju alternatyvų nėra daug, kadangi norint šiuos dalykus įsivertinti, reikia ištirti įvairius rizikos faktorius bei jų tikimybes, ko pasekoje uždavinys tampa komplikuotas ne tik sprendinio paieškos prasme, bet ir informacijos suvedimo. Dėl to plačiau taikomas sprendimas, kuomet tvarkaraštis formuojamas nedideliam laikotarpiui į priekį (priklausomai nuo įrengimų patikimumo – pusdienis, diena ir t.t.). Šiuo atveju įvykus bet kokiam nenumatytam atsitiktiniam įvykiui, pakankamai greitai bus galima atlikti darbų perplanavimą.

b. Pagal išteklių tipą :
    1. Darbo jėgos ištekliai
    2. Laiko ištekliai;
    3. Medžiagų ištekliai;
    4. Žinių ištekliai ir t.t.
c. Pagal ribojimų pobūdį :
    1. Resursų ribojimai;
    2. Laiko ribojimai;
    3. Nuoseklumo ribojimai ir t.t.

    Ribojimai taip pat gali būti skirstomi į laikinuosius , nenumatytuosius, nuolatinius.
d. Pagal duomenų apibrėžtumą:
    1. Deterministiniai (visa reikiama informacija tvarkaraščiui sudaryti žinoma iš anksto);
    2. Stochastiniai (visi duomenys aprašomi tikimybiniais modeliais);
    3. Nedeterministiniai (nestochastiniai) – kuomet dalis duomenų yra iš anksto žinomi, o likę aprašomi tikimybiniais modeliais;
    4. Nepilnai apibrėžti duomenys – šiuo atveju dalis duomenų gali būti klaidingi arba iš viso nežinomi. [6]

1.2. Tvarkaraščių sudarymo programinės priemonės

Programinių priemonių tvarkaraščių uždaviniams spręsti šiuo metu sukurta yra pakankamai nemažai. Tačiau kiekvienos įmonės gamybos specifika neleidžia sukurti universalaus produkto. Tad lieka trys variantai: arba ieškoti priimtinesnio sprendimo, arba kurti naują produktą, pritaikytą įmonės reikmėms arba pakoreaguoti įmonėje vykstančius procesus, taip kad jie būtų pritaikyti jau esamai programinei įrangai.

Bendruoju atveju tvarkaraščių sudarymo uždaviniams spręsti naudojama programinė įranga dažniausiai remiasi interaktyviu dialogu su vartotoju ir reikalauja pakankamai nemažai rankinio darbo pateikiant duomenis. Dėl šios priežasties tokia programinė įranga naudojama tik stambiaserijinėje gamyboje. Tačiau yra sugalvota ir pakankamai nemažai sprendimų įmonėms, kurios produkciją gamina nedidelėmis partijomis. Pavyzdžiui Lietuvių IT kompanijos UAB Euritecha sukurtas įrankis „GVS“ – gamybos valdymo sistema. Sistemos duomenų bazėje saugoma informacija apie kiekvieną gaminį (gamybos laikai, perderinimo laikai, reikalingos medžiagos ir t.t.). Gaminio detalizacijos lygis šiuo atveju apsprendžia kuriuose kontrolės taškuose mes seksime tvarkaraščio vykdymą (tai gali būti visas cechas, darbo baras, darbo centras ar darbo centrui priklausantis įrengimas). Kuo didesnis detalizacijos lygis, tuo galima sudaryti tikslesnį užduočių tvarkaraštį, tačiau kartu ir tuo daugiau reikia įdėti pastangų norint gaminį patalpinti į sistemą.

Šiuo atveju nagrinėjamoje prekybinės įrangos gamybos įmonėje, kurioje produkcija gaminama 1-100 vnt. partijomis, ir tiražavimo lygis yra pakankamai žemas, gaminius priimtina apsirašyti sustambintame darbo barų lygmenyje : Paruošimas (pjovimas, karpymas, lankstymas ir t.t.) → Surinkimas (virinimas, šveitimas) → Dažymas → Pakavimas. Į GVS sistemą suvedus užsakymus (nurodant kiekvieno gaminio gamybos termina) ir paleidus gaminius į gamyba, ši sistema generuoja kontrolinį lapą – kiekių valdymo lentelę (kuri yra barkodų skanerių sistemos analogas, pildomas rankiniu būdu) (1.1 Pav.). Lentelėje pateikiamas gaminių sąrašas (pagal vidutinę dienos normą). Gaminio poziciją sąraše apsprendžia vėlavimo indeksas, kurį galima būtų vertinti kaip tikimybę pavėluoti laiku atkrauti gaminius. Darbo baro vadovas gavęs tokį lapą turi organizuoti darbus taip, kad jie būtų vykdomi eilės tvarka ir taip, kad dienos planas būtų įvykdytas bet kokia kaina. Kiekvienas darbo baras gauna individualią lentelę. Dienos pabaigoje visos lentelės surenkamos ir suvedamos į sistemą. Nepagaminti gaminiai perkeliami į sekančios dienos kiekio valdymo lentelės viršų.

Nemaža dalis tokių programinių priemonių yra integruotos į didžiules ERP sistemas (angl. : *Enterprise Resource Management*). Vienas iš integracijos privalumų yra tai, kad šios sistemos įvairiais metodais (barkodiniai skaneriai, elektroninės kiekių valdymo lentelės ir t.t.) leidžia sekti plano vykdymą, vykdyti automatizuotą medžiagų apskaitą bei operatyviai identifikuoti sutrikimus, dėl kurių atsirado atsilikimas nuo plano. Tačiau dažniausiai tokio tipo programose yra realizuoti tik paprasčiausi optimizavimo algoritmai. Sudėtingi šiuolaikiniai metaeuristiniai algoritmai standartinėse sistemose beveik nėra taikomi [3]

KIEKIŲ VALDYMO LENTELĖ :
04M-PARUOSIMAS

DARBUOTOJAI	VAL	Lt/val	DARBUOTOJAI	VAL	Lt/val

Data	5/9/2013 19:25	
Laikas	8:00	17:00

Užsak. numeris	Maršrut. numeris	Paviršius	Gamybos užsakymo kodas	Gaminio kodas	Paveiksl.	Pavadinimas	VI	Kiekis	Pagaminta	Gamybos pradžia	Gamybos pabaiga	Gamybos laikas	Norma [val]	Skirtumas	Gamino
2377	25213 [1vnt.]	Balta RAL9003	4193	202.44.049		202.44.049 Aksesuarų stendas baltas, perforuotas	550%	1					0.003		
2374	25103 [40vnt.]	Balta RAL9003	4164	202.44.053		202.44.053 8 pakabų stendas su kugine apacia	83%	40					1.733		
2374	25101 [12vnt.]	Balta RAL9003	4162	202.44.050		202.44.050 3 pakabų centras	78%	12					4.340		
2374	25091 [14vnt.]	Balta RAL9003	4152	202.44.019		Deklas 3-ju aukstu (105446-01)	77%	14					7.700		
2397	25247 [12vnt.]	Pilka RAL7038 SK2	4234	202.44.031		Krepsiu centras su 850x800x530 skirtuku 117580-01	-43%	12					3.600		
2397	25241 [25vnt.]	Balta RAL9003	4228	202.44.018		T Stovas - pakaba reguliuojama koja (107297-01)	-46%	25					14.167		
2397	25252 [2vnt.]	Balta RAL9003	4239	217.44.004		217.44.004 - Basket round 105227 + 205164	-48%	2					5.200		
2397	25254 [3vnt.]	Balta RAL9003	4241	202.44.046		202.44.046 Aksesuaru stendas (117367-01)	-48%	3					4.250		
2397	25246 [1vnt.]	Balta RAL9003	4233	212.44.003		Stalas dvieju lygiu baltas tinklas (117569-01)	-49%	1					1.574		
2397	25242 [2vnt.]	Balta RAL9003	4229	204.44.023		Zurnalu stendas (110263-01)	-49%	2					2.533		
2397	25238 [2vnt.]	Balta RAL9003	4225	214.44.002		214.44.002 Veidrodis	-49%	2					2.033		
2397	25239 [2vnt.]	Balta RAL9003	4226	204.44.008		Krepsys apvalus baltas platus	-49%	2					2.633		
2397	25240 [4vnt.]	Balta RAL9003	4227	202.44.019		Deklas 3-ju aukstu (105446-01)	-49%	4					2.200		
2397	25250 [5vnt.]	Balta RAL9003	4237	202.44.045		202.44.045 Single stand narrow x-low white 119872	-49%	5					4.500		
2397	25248 [1vnt.]	Nerūdijantis plienas	4235	214.44.507		Veidrodis 2000x400 (117596-01)	-50%	1					0.833		
2397	25256 [1vnt.]	Balta RAL9003	4243	204.44.061		204.44.061 Acc board snakebasket white perf 12544	-50%	1					0.678		

Pav. 1.1 Kiekių valdymo lentelės pavyzdys

1.3. Realios gamybos pavyzdys

Šiame darbe daugiausiai remsimės prekybinės įrangos gamybos įmonės veiklos modeliu. Prekybine įranga šiuo atveju laikome įvairias metalo ir medžio konstrukcijas, baldus, be kurių neapsieina nei viena parduotuvė. Įmonėje jau eilę metų diegiama sinchroninės gamybos sistema (*LEAN manufactoring*). LEAN – tai verslo, gamybos procesų valdymo sistema, pagal kurią veikianti įmonė pasiekia didžiausią įmanomą pelną su mažiausiomis finansinėmis investicijomis ir mažiausiu darbuotojų skaičiumi. LEAN metodika pagrįsta veiklos procesų trumpinimu ir standartizavimu. Šios filosofijos – verslo metodo pagrindinis tikslas yra tai, kad visi įmonėje vykstantys procesai (ir gamybos, ir projektavimo, ir administravimo) veiktų kaip idealiai sureguliuotas konvejeris : sklandžiai, be prastovų, efektyviai išnaudojant didžiausius apribojimus [17]

Nagrinėjamoje įmonėje veikia du atskiri cechai – metalo ir medžio gaminių. Metalo ceche daugelis gaminių gaminami partijomis (dažniausiai iki 30 vnt.), bet labai dažnai pasitaiko ir vienetinių gaminių . Tuo tarpu medžio ceche gaminiai tiražuojasi labai retai. Taigi gamybos asortimentas šiuo atveju yra pakankamai didelis ir pastoviai kinta.

Paprastumo dėlei tiriamajame darbe nagrinėsime tik metalo ceche vykstančius gamybos procesus. Šiame ceche, dėl sinchroninės gamybos diegimo, gamyba tapo labai artima srautinei, t.y. kiekvienam darbo centrui galima priskirti tam tikra eilės numerį – visus įrengimus (iš visų barų)

galima sustatyti į vieną eilę – konvejerį. Tačiau gaminiai keliaudami šiuo konvejeriu nebūtinai yra apdorojami kiekviename darbo centre.

Nagrinėjamos įmonės metalo ceche šiuo metu veikia 4 pagrindiniai barai :

i. Paruošimo baras – žaliava tampa detalėmis;
ii. Surinkimo baras – detalės tampa gaminiu;
iii. Dažymo baras – gaminiai dažomi;
iv. Pakavimo baras – gaminiai paruošiami atkrauti.

Pagal gamybos filosofiją tam, kad atsirastų srautas, gamykloje turi būti pakankamas skaičius lygiagrečiai veikiančių įrengimų dažniausiai pasitaikančioms operacijoms. Šiuos įrengimus aptarnauja specialus žmogus – derintojas, kuris prieš kiekvieną gaminių partiją pereina atitinkamo gaminio technologijos maršrutą ir perderina visus įrengimus. Dauguma įrengimų įmonėje yra modernizuoti pagal SMED (*Single- Minute Exchange or die*) metodą, kuris leidžia įrengimą perderinti vienu rankos judesiu, dėl to perderinimo laikai įmonėje yra pakankamai nedideli.

Taip pat svarbų vaidmenį įmonėje atlieka ir logistas – tai žmogus, kuris perstumia gaminius nuo vieno darbo centro prie kito, taip taupydamas brangiai apmokamų darbininkų darbo laiką.

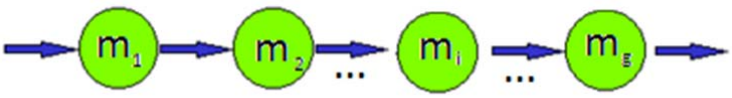
1.4. Uždavinio matematinis modelis

Kiekvieną gamybos rūšį pagal įrengimų išdėstymą ceche ir darbų srautus, galima suklasifikuoti į tam tikras grupes :

Srauto fabrikas: Šis gamybos tipas yra vienas iš efektyviausių – juo yra grindžiama visa LEAN gamybos filosofija. Pradines idėjas šiam gamybos tipui pirmasis užpatentavo Henris Fordas, savo gamykloje įdiegęs konvejerį. Pagrindinės srautinio fabriko idėjos remiasi tuo, kad kiekvienas darbas susideda iš tam tikro skaičiaus operacijų. Kiekvieno darbo operacijos turi būti vykdomos vienoda seka (bet nebūtinai kiekviename darbo centre) (1.2 pav). Taip pat pabrėžtina, kad prie kiekvieno darbo centro yra kaupiamas užduočių buferis – kiekviena užduotis pastatoma į eilę. Be to negalimi paraleliniai darbo centrai. Šiuo atveju planuojamos tik pirmojo darbo centro užduotys, visų likusių darbo centrų tvarkaraščiai yra priklausomi. Šio gamybos tipo matematinis modelis gali būti užrašomas taip:

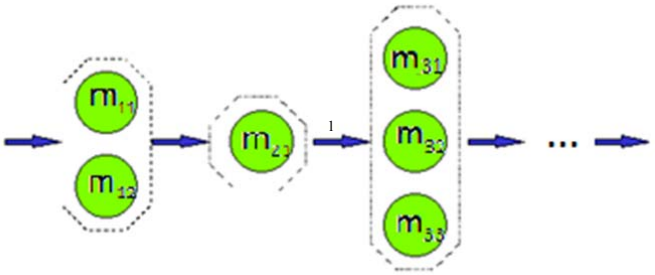
$$\forall i, j, m, n (S_{i,m} \leq S_{j,m} \rightarrow S_{i,n} \leq S_{j,n}) \quad (2.1)$$

Čia S- visų darbų pradžios momentų kiekviename darbo centre matrica. Indeksai i ir j žymi darbų numerius, o indeksai m ir n – darbo centrų numerius. Tuomet $S_{i,m}$~$C_{i,m}$ žymime $i-$ tojo darbo pradžią arba pabaigą darbo centre m. [2]



Pav. 1.2 Srauto fabriko pavyzdys [models]

Lankstus srauto fabrikas : Šis gamybos tipas yra kur kas dažniau sutinkamas. Tai yra tam tikra srautinio fabriko modifikacija, kuomet kiekviename etape gali buti daugiau nei vienas lygiagrečiai dirbantis darbo centras (1.3 pav). Kiekvienas gaminys keliauja per gamybos etapus tuo pačiu maršrutu, tačiau nebūtinai sustodamas prie kiekvieno darbo centro. Šiuo atveju planuoti tik pirmojo darbo centro užduotis nepakanka – reikalingas kiekvieno etapo užduočių planavimas.



Pav. 1.3 Lankstaus srauto fabriko pavyzdys [models]

Panagrinėkime plačiau šį gamybos modelį. Aptarkime pagrindinius modelyje naudojamus žymėjimus :

g : Darbo centrų skaičius

n : Planuojamų darbų skaičius

g_t : Darbo centrui t priklausančių paraleliai dirbančių įrengimų skaičius

m_j : Paskutinis darbo centras, kuriame buvo apdorotas darbas j

p_{ti} : Darbo i apdorojimo trukmė darbo centre t

S_i : Aibė darbo centrų, kuriuose apdorojamas darbas i

W_t : Aibė darbų, kurie yra apdorojami darbo centre t

c_{ti} : Darbo i pabaigos momentas darbo centre t

x_{tij}: $\{1 - \text{jei darbo centre } t \text{ darbas } i \text{ atliekamas iskart po darbo } j \text{ , 0 kitu atvėju}\}$

o_{tkj}: Laikas, per kurį darbas j pereina nuo darbo centro t į darbo centrą k

Pagrindine tikslo funkcija laikykime visų darbų atlikimo pabaigos momentą (*angl. makespan*).

$$\max_{j \in W_g}(c_{gj}) \rightarrow min \qquad (2.2)$$

Čia $\max_{j \in W_g}(c_{gj})$ – paskutinio darbo centro g visų užduočių $j \in W_g$ vykdymo pabaigos momentas.

Darbų tvarkaraščiui galioja šie apribojimai

- Darbo centras (visi lygiagrečiai veikiantys įrengimai) atlieka darbus, jeigu užduočių buferis nėra tuščias. [2]

$$\sum_{i=1}^{n}\sum_{j=1}^{n} x_{tij} = g_t \tag{2.3}$$

- Tas pats darbas gali būti priskirtas tik vienam lygiagrečiai dirbančiam įrengimui kiekviename darbo centre [2]

$$\sum_{j\in S_t} x_{tij} = 1,\ \forall i; t \in S_i\ ; \tag{2.4}$$
$$\sum_{i\in S_t} x_{tij} = 1,\ \forall j; t \in S_j.$$

- Nei vienas darbas negali būti pertrauktas bei vienu metu tas pats darbas gali būti apdorojamas tik viename darbo centre [2]

$$c_{tj} = \max\left(c_{ti}, c_{t-1,j} + o_{(t-1)tj}\right) + p_{tj},\ \forall x_{tij} = 1, t \in S_j,\ \ i,j \in W_t \tag{2.5}$$

- Jeigu darbas *j* nėra apdorojamas darbo centre t tuomet jo pabaigos momentas [11]

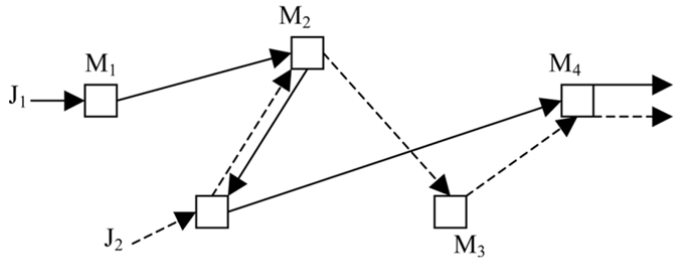
$$c_{tj} = c_{(t-1)j},\ \forall j \in n,\ \forall t \notin\ S_j \tag{2.6}$$

Šiame modelyje lieka neįvertinti perderinimo laikai. Šie laikai gali būti aprašomi labai įvairiai. Pavyzdžiui, darbų matricomis kiekvienam darbo centrui, kur kiekvienas matricos A_t langelis a_{ijt} reiškia stebimo darbo centro *t* perderinimo laiką, kai darbą *i* pakeičia darbas *j*. Šis būdas būtų bene tiksliausias, tačiau reikalauja labai daug papildomų duomenų pradiniame tvarkaraščio sudarymo etape. Kitas variantas, kad kiekvienam darbo centrui galima priskirti fiksuotą perderinimo laiką. Šiuo atveju nebus vertinama tai, ar iš eilės eina dvi partijos vienodų gaminių. Tačiau iš praktinės pusės žiūrint, nagrinėjamoje įmonėje, kur gaminiai gaminami nedidelėmis partijomis, šie atvejai nutinka pakankamai retai, dėl to bendrai tvarkaraščio kokybei šis modelis nepakenks. Aprašytame modelyje perderinimo laikas gali būti įtrauktas prie gaminių transportavimo laiko.

Darbų fabrikas : Šiuo atveju kiekvienas darbas turi atskirą technologinį maršrutą (1.4 pav) . Šiam uždaviniui būtina pradinė informacija – kiekvieno gaminio technologinio maršruto seka. Šią seką galima atvaizduoti matricos B pagalba. Matricos $B = [b_{it}]_{[n\times g]}$ kiekviens stulpelis nusako kokia tvarka i −tasis darbas juda gamybos procese. Šiuo atveju modelis įgauna tokį pavidalą

$$\forall i, x, y\ \left(b_{i,t} < b_{i,k} \rightarrow S_{i,b_{i,t}} \leq S_{i,b_{i,k}}\right) \tag{2.7}$$

čia b_{it} − darbo i gamybos proceso sekos numeris darbo centre *t* , $S_{i,b_{i,t}}$ - darbo i apdorojimo pradžios momentas $b_{i,t}$ technologiniame etape. [1]

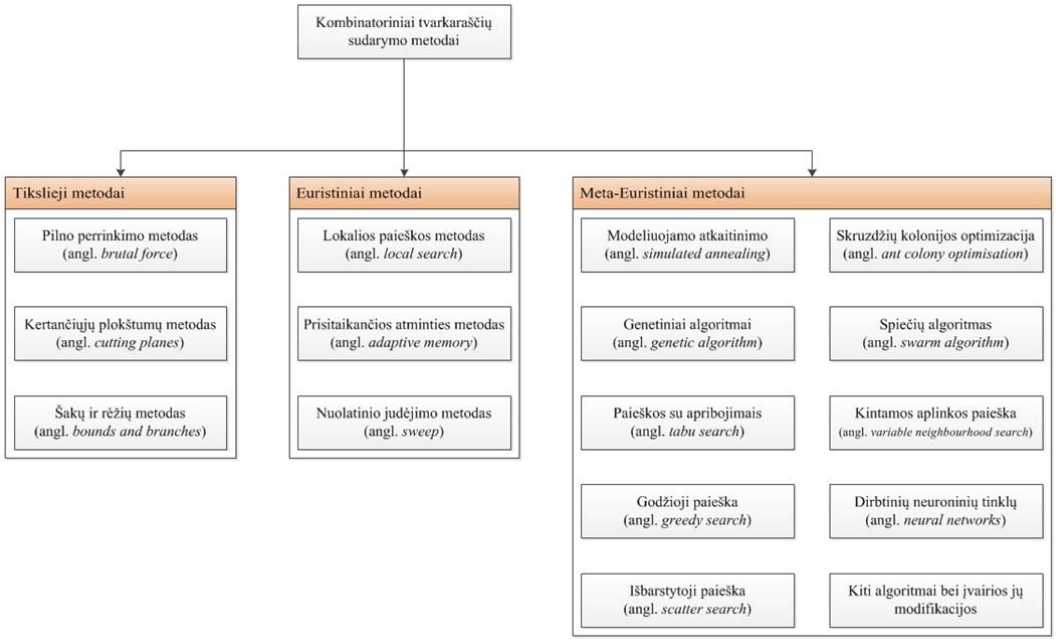


Pav. 1.4 Darbų fabriko pavyzdys

1.5. Tvarkaraščių sudarymo metodai

Šiuo metu egzistuoja pakankamai daug įvairių kombinatorinio optimizavimo metodų bei jų modifikacijų skirtų įvairaus pobūdžio tvarkaraščiams sudaryti. Didžioji dalis tvarkaraščių sudarymo uždavinių priklauso NP sudėtingumo klasei, todėl tikslus sistemos sprendinys gali būti gautas sprendžiant tik nedidelės apimties uždavinius. Kombinatorinio optimizavimo metodų tikslas – gauti artimą sprendinį tiksliajam, tačiau tai ne visada pavyksta. Deja, nėra priimtinų metodų patikrinti sprendinio tinkamumą visos sprendinių aibės kontekste, dėlto tampa neaišku, ar gautas sprendinys yra globalus ar lokalus optimumas. [6]

Praktikoje tvarkaraščių sudarymo metodus įprasta skirstyti į tiksliuosius, euristinius bei meta-euristinius (1.5. pav.)



Pav. 1.5 Tvarkaraščių sudarymo metodų klasifikacija

Tikslieji algoritmai pateisina savo vardą ir randą tikslų – optimalų sistemos sprendinį, tačiau jų konvergavimo greitis eksponentiškai priklauso nuo pradinių duomenų matricos dydžio. Dėl to šie metodai tinkami spręsti tik nedidelės apimties uždavinius. Tačiau neretai šių metodų pagrindines

idėjas galima pritaikyti generuojant pradinių sprendinių aibę, kuri vėliau gali būti tikslinama įvairiais Euristiniais ir Meta-Euristiniais metodais.

Euristinių metodų pagalba siekiama per priimtiną laiką rasti pakankamai auštos kokybės sprendinį, tačiau garantijų sprendinio kokybei šis metodas nesuteikia. Euristika, tai tam tikra taisyklė, metodika, kurios pagalba gerinamas pirminis sprendinys. Euristiniai metodai pritaikomi tik tam tikriem – konkretiems uždaviniams su tam tikrais apribojimais.

Metaeuristiniai algoritmai apibūdinami kaip tam tikro aukšto abstrakcijos lygio taisyklių rinkiniai. Skirtingai nuo euristinių algoritmų, šių taisyklių paskirtis yra formaliai aprašyti kurios nors uždavinių grupės sprendimo idėją, principą. Taigi šiuo atveju meta-euristinių algoritmų klasė yra apibendrinančioji – euristiniai metodai yra atskira šios klasės grupė. Meta-euristinių algoritmų klasifikavimas taip pat gali būti labai įvairus [6] :

a. **Paremti / Neparemti gamtos dėsniais :** dažnai yra įprasta meta-euristinius algoritmus grupuoti pagal tai, ar jie yra inspiruoti gamtos dėsnių (pvz. genetinis algoritmas (angl. *genetic algorithm*), skruzdžių kolonijos optimizavimas (angl. *ant colony optimisation*), modeliuojamas atkaitinimas (angl. *sumulated annealing*), arba neparemti gamtos dėsniais (pvz. Lokalios paieškos metodas (angl. *local search*), tabu paieška (angl. *tabu search*) ir t.t.[19]

b. **Vienu metu nagrinėjamas vienas ar daugiau sprendinių :** Kita charakteristika, kuri gali būti naudojama meta-euristinių metodų klasifikavimui, tai vienu metu nagrinėjamų sprendinių skaičius. Algoritmai, kurie dirba su vienu sprendiniu vadinami konstruktyviaisiais arba trajektorijos metodais (pvz. Tabu paieška (angl. *tabu search*), iteracinė lokali paieška (angl. *iterated local search*), kintančios aplinkos (angl. *variable neighborhood search*) metodai. Tuo tarpu sprendinių populiacijos nagrinėjimu paremti metodai dažniausiai tiria visos populiacijos evoliucionavimą. [19]

c. **Algoritmas naudoja atmintį ar nenaudoja:** Atminties naudojimas yra labai svarbi metodų klasifikacijos savybė. Šį bruožą galima tapatinti su algoritmo gebėjimu mokytis iš praeities – adaptuotis ir evoliucionuoti. Algoritmai, kurie nenaudoja atminties, dirba Markovo grandinių principu, t.y. kiekvienas naujas sprendinys yra gaunamas naudojant tik prieš tai buvusio sprendinio informaciją. Atminties panaudojimas taip pat gali būti įvairus. Vieni algoritmai naudoja trumpalaikę atmintį, kurioje saugoma informacija apie neseniai atliktas iteracijas ir tirtus

sprendinius, kitu atveju kaupiama informacija apie tam tikras sprendinių savybes. [19]

Nepaisant to, kad meta-euristinių metodų pagalba galima išspręsti daugelį kombinatorinio optimizavimo problemų, tačiau kaip teigiama „No free lunch“ teoremose

„.... visi algoritmai, kurie ieško įverčio funkcijos ekstremumų veikia vienodai tiksliai, kuomet dirbama įverčio funkcijos vidurkio aplinkoje. Trumpai tariant, jeigu algoritmas A veikia geriau nei algoritmas B tam tikrose įverčio funkcijose, tuomet egzistuoja lygiai tiek pat įverčio funkcijų, kuriose algoritmas B veikia geriau negu algoritmas A.“ [20];

„... bet kuriam optimizavimo algoritmui bet kuris jo veikimo pagerinimas vienai problemų klasei yra pabloginimas kitai uždavinių klasei.“ [20];

Apžvelkime keletą metodų plačiau.

Modeliuojamas atkaitinimas : Tai yra vienas seniausių meta-euristinių metodų ir vienas pirmųjų algoritmų gebėjusių išvengti konvergavimo į lokalų optimumą. Metodo veikimas paremtas fizikinio proceso kaitinimo arba grūdinimo idėja. Algoritmo veikimą galima būtų suskirstyti į kelis etapus

a. Metodo pradžioje sugeneruojamas sprendinio artinys $s \in N(s)$ bei pradinis temperatūros parametras T;
b. Kiekvienoje iteracijoje sugeneruojamas atsitiktinis šalutinis sprendinys $s' \in N(s')$. Šalutinis sprendinys s' tampa naujuoju sprendiniu, jeigu $f(s') < f(s)$ arba su tam tikra tikimybe $P\big(T, f(s') - f(s)\big)$, jeigu $f(s') \geq f(s)$. Šiuo atveju dažniausiai naudojama Boltsmano tikimybinė formulė

$$P\big(T, f(s') - f(s)\big) = \exp\left(-\frac{\big(f(s') - f(s)\big)}{T}\right) \tag{2.8}$$

Pagrindinis šios formulės vaidmuo algoritme – sprendinio gebėjimas ištrūkti iš lokalaus minimumo. Temperatūra T mažėja kiekvienos iteracijos žingsnyje. Apibūdintas dinaminis procesas yra Markovo grandinė. Tai reiškia, kad bazinis metodas nenaudoja atminties, nors [6] literatūros straipsnyje pateikiamos rekomendacijos atminties taikymui šiame algoritme.

Tabu paieška : Tabu paieškos metodas remiasi sprendinių uždraudimo idėja. Tai tam tikra trumposios atminties forma, vadinama tabu sąrašu. Pagrindinė tabu sąrašo funkcija yra įsiminti tam tikrą kiekį neseniai aplankytų sprendinių ir neleisti algoritmui vėl patekti į tuos pačius sprendinius. Tabu paieškos algoritmas pagrįstas kaimyninių sprendinių paieška, pereinant nuo vieno lokaliojo optimumo prie kito. Tabu paieška pradedama nuo pradinio, atsitiktinai sugeneruoto sprendinio $s \in N$. Kiekviename etape analizuojama einamojo sprendinio s aplinka $N(s)$ ir iš jos atrenkamas optimalus sprendinys (jeigu tiriamojoje aplinkoje nėra sprendinių su geresnėmis tikslo funkcijų

reikšmėm, tuomet pasirenkamas toks sprendinys, kuris mažiausiai pablogina tikslo funkcijos reikšmę). Kitaip tariant, perėjimas atliekamas į geriausią kaimyninį sprendinį $s' \in N(s)$. Norint išvengti grįžimo į neseniai nagrinėtą sprendinį, visi neseniai nagrinėti sprendiniai įtraukiami į Tabu sąrašą T, kur $|T| = h$ - tabu sąrašo ilgis. Šiuo atveju labai mažas tabu sąrašo ilgis gali grąžinti algoritmą į neseniai nagrinėtus sprendinius, ko pasekoje gali gautis tam tikras ciklas, dėl kurio algoritmas sukonverguos į lokalų optimumą. Kitą vertus labai ilgas tabu sąrašo ilgis gali apriboti paieška potencialiose aplinkose. Taigi perėjimas $s \to s'$ yra negalimas, jeigu $s' \in T$. Atskiru atveju esant tam tikrom aplinkybėm, vadinamom aspiracijos kriterijumi, perėjimas $s \to s'$ yra leistinas, net jeigu ir $s' \in T$ (pvz tuo atveju jeigu $f(s') < f(s^*)$, čia s^* yra geriausias iki šiol rastas sprendinys. [19]

Godžioji atsitiktinė adaptyvioji paieška, tai metodas apjungiantis konstruktyviąsias euristikas su lokalios paieškos algoritmais. Kiekvieną šio metodo iteraciją sudaro dvi pagrindinės fazės – sprendinio konstravimas ir jo gerinimas. Tarkime, kad uždavinio sprendinį sudaro tam tikrų elementų seka. Tuomet sprendinys konstruojamas žingsnis po žingsnio atsitiktinai formuojant sprendinio seką. Be to kiekvienas sprendinio elementas turi tam tikrą įvertį, kuris naudojamas lyginant prie dalinio sprendinio pridedant pasirinktą elementą. Sugeneruotas sprendinys tikslinamas pritaikius lokaliąją paiešką. [19]

Genetinis algoritmas : tai vienas iš gamtos dėsniais paremtų algoritmų, kurio veikimas pagrįstas evoliucijos teorija. Šiuo atveju evoliucionuoja ne vienas individas (sprendinys), bet visa individų populiacija (aibė). Pirmiausiai atliekant šį algoritmą sugeneruojama atsitiktinių (arba tikslingų) sprendinių aibė. Kiekvienam aibės elementui pagal atitinkamas tikimybes gali būti atliekamos genetinės modifikacijos – pritaikomi genetiniai operatoriai (kryžminimas bei mutavimas). Naujai gauti sprendiniai lyginami su senąja populiacija – taip įvairiais atrankos metodais formuojama nauja populiacija. Ciklas kartojamas, kol prieinama tam tikra konvergavimo sąlyga (pvz. iteracijų skaičius arba vidutinis procentinis pagerėjimas per tam tikrą laikotarpį – generacijų skaičių). Plačiau genetinį algoritmą aptarsime sekančiame skyriuje. [16]

Sprendžiant daugelį kombinatorinio optimizavimo uždavinių, galima naudoti praktiškai visų minėtų meta-euristinių algoritmų idėjas. Šiuo atveju universalumas yra vienas didžiausių pranašumų prieš bet kurį specializuotą euristinį metodą, tinkantį vienam konkrečiam uždaviniui spręsti. Siekiant pagerinti meta-euristiniais metodais gauto sprendinio kokybę gali būti apjungiamos kelių skirtingų meta-euristinių algoritmų idėjos. Dėl šios priežasties meta-euristiniai algoritmai, įvairios jų modifikacijos bei kombinacijos, kaip kad genetiniai algoritmai ir lokalioji paieška, tabu paieška ir

atkaitinimo modeliavimas yra pakankamai dažnas kombinatorinio optimizavimo mokslo šakos tyrinėjimo objektas. Pagrindinis šių tyrinėjimų tikslas – sukuri kuo efektyvesnius algoritmus įvairiems kombinatorinio optimizavimo uždaviniams spręsti.

2. Tyrimo metodai

Šiame darbe nagrinėsime gamybinių tvarkaraščių sudarymo uždavinį, bei jo sprendimą naudojant genetinį algoritmą. Kaip jau minėjome 1.2 skyrelyje, tyrimo metu bus naudojami realios gamyklos, dirbančios pagal lankstaus srauto fabriko modelį, duomenys. Atlikdami įvairius eksperimentus pasistengsime surasti rekomenduojamus genetinio algoritmo parametrų rinkinius įvairaus dydžio pradinių duomenų matricoms. Gavus optimalius parametrų rinkinius, genetinį algoritmą įvairiais aspektais palyginsime su tam tikru euristiniu metodu dirbančiu pagal apribojimų teorijos principus [17], Šiuo atveju algoritmas bus orientuotas į efektyvų apribojimo panaudojimą.

2.1. Pradinių duomenų struktūra

Eksperimentų metu dirbsime su realiais, šiuo metu praktikoje naudojamais duomenimis. Pradinė algoritmo informacija apibendrinus yra sudaryta iš trijų matricų A, B ir W:

$$A_{[N\times J]} = \begin{pmatrix} t_{1,1} & t_{1,2} & \dots & t_{1,J} \\ t_{2,1} & t_{2,2} & \dots & t_{2,J} \\ \dots & \dots & \dots & \dots \\ t_{N,1} & t_{N,2} & \dots & t_{N,J} \end{pmatrix} \tag{2.1}$$

čia N- darbo centrų aibė, o $J-$ darbų aibė. Darbo centru šiuo atveju vadinsime tam tikrą paraleliai dirbančių įrengimų grupę.Perdaug nenukrypstant nuo praktinio uždavinio, darysime prielaidą, kad kiekvienas šios grupės įrengimas dirba vienodu našumu. $t_{i,j}$ - žymėsime užduoties $i = \overline{1,N}$, apdorojimo trukmę darbo centre $j = \overline{1,J}$ Šiuo atveju žiūrint iš gamybinės pusės traktuosime, kad $t_{i,j}$ yra visos gaminio i partijos apdorojimo laikas darbo centre j. Uždavinio supaprastinimui įsivesime kelis apribojimus:

- Užduotis i gali būti apdorojama tik su vienu įrengimu iš darbo centro j, t.y. $t_{i,j}$ yra nedalomas.
- Jeigu q yra i gaminio partijos dydis, tuomet $t_{i,j}$ reprezentuoja visos partijos apdorojimo laiką. Be to, partija q yra nedaloma.

Matrica B reprezentuoja kiekvienos užduoties technologinę kortelę.

$$B_{[N\times J]} = \begin{pmatrix} r_{1,1} & r_{1,2} & \dots & r_{1,J} \\ r_{2,1} & r_{2,2} & \dots & r_{2,J} \\ \dots & \dots & \dots & \dots \\ r_{N,1} & r_{N,2} & \dots & r_{N,J} \end{pmatrix} \tag{2.2}$$

čia $r_{i,j}$ – įrengimų sekos numeris. $r_{i,j} \in \{1,2,\dots,N\}$. Kadangi nagrinėjamas srautinės gamybos modelis, todėl galioja dėsnis $\forall i,k,j : r_{i,j} > 0, r_{k,j} > 0, i < k \rightarrow r_{i,j} < r_{k,j}$. Šiuo atveju $r_{i,j} = 0$ jeigu užduotis i neturi būti apdorota darbo centre j.

Vektorius W reprezentuoja kiekvieno darbo centro pradinių darbų seriją, t.y. laiko momentą, nuo kurio darbo centras yra pasiruošęs vykdyti užduotis pagal sudarytą tvarkaraštį. Šiuo atveju

$$W_{[N]} = (h_1 \quad h_2 \quad \dots \quad h_J) \tag{2.3}$$

$$h_j = \left(e_{j,1}, e_{j,2} \dots e_{j,|h_j|}\right) \tag{2.4}$$

čia $e_{j,k}$ reprezentuoja atitinkamo paraleliai dirbančio įrengimo k, priklausančio darbo centrui j , pradinių darbų seriją.

2.2. Genetinio algoritmo paradigma

Genetinis algoritmas - tai stochastinis kombinatorinio optimizavimo metodas, paremtas gamtoje vykstančiu evoliucijos procesu. Algoritmui būdingi tam tikri terminai :

- Genas – struktūrinė sprendinio dalelė;
- Chromosoma – genų seka reprezentuojanti sprendinį;
- Populiacija – chromosomų (sprendinių) aibė;
- Fenotipas – dekoduotas sprendinys;
- Genotipas – užkoduotas sprendinys.

Pagrindines genetinio algoritmo idėjas pasiūlė J.Hollandas, labai didelį įnašą į evoliucinių strategijų kūrimą davė I.Rechenberg bei H.Schwefel [9]. Genetinis algoritmas sulaukė didžiulio susidomėjimo dėl savo pranašumų lyginant su kitais kombinatorinio optimizavimo metodais :

a. Pritaikomumas –genetinis algoritmas neturi jokių įgimtų matematinių apribojimų, kurie susiaurintų arba pablogintų optimizavimo problemos modelį. Algoritmas geba dirbti su įvairiom tikslo funkcijom ir jų apribojimais;
b. Robastiškumas – genetinių operatorių naudojimas leidžia tuo pačiu metu atlikti lokalią ir globalią paiešką, kai tuo tarpu dauguma tradicinių euristinių algoritmų atlieka tik lokalią paiešką. Remiantis dauguma mokslinių tyrimų buvo įrodyta, kad genetinis algoritmas daug efektyviau išnaudoja konvergavimo laiką optimalaus sprendinio paieškai, lyginant su kitais tokio tipo metodais;
c. Lankstumas – genetinį algoritmą nesunku papildyti įvairiomis euristikomis ir taip gauti efektyvų algoritmą specifinei problemai spręsti.[9]

Projektuojant genetinį algoritmą konkrečiai problemai spręsti, pradžioje reikia įvertinti pagrindinius algoritmo komponentus. Pirmiausiai turi būti apibrėžtos genotipo ir fenotipo erdvės. Fenotipo erdvės elementas - tai norimas gauti sprendinio pavidalas, šiuo atveju kiekvienos mašinos darbų sąrašas, kuris, pasinaudojant matricomis A, B ir W, tampa laike apibrėžtu tvarkaraščiu. Genotipo erdvė - tai užkoduotų sprendinių aibė. Tam tikrais atvejais, kodavimo transformacijos funkcija gali būti ir vienetinė funkcija. Sprendinių kodavimas siejamas su genetinių operatorių modeliais.

Labai svarbus vaidmuo šioje sistemoje tenka tikslo funkcijai. Šios funkcijos esmė genotipo erdvės sprendinį atvaizduoti į fenotipo erdvės sprendinį, o pastarąjį atvaizduoti realiųjų skaičių aibėje. Apibrėžus genotipo ir fenotipo erdves reikia suprojektuoti genetinius operatorius, tokius, kaip genetinį kryžminimą, mutavimą bei genetinės atrankos operatorių. Paskutiniame etape reikia apibrėžti genotipo erdvės elementų pataisymo funkcijas, t.y. genotipo erdvės elementui pritaikius kryžminimo bei mutavimo operatorius, šio elemento struktūra gali būti pažeista taip, kad vėliau jo nepavyks atvaizduoti fenotipo erdvėje.[16]

2.2.1. Sprendinio reprezentacija

J.Holland pasiūlytame genetinio algoritmo modelyje [9] sistemos sprendinys buvo užrašomas dvejetaine eilute 10110011010110. Deja, šis kodavimas turi kelis esminius trūkumus susijusius su „Hemingo uolos“ principu. Pavyzdžiui, dvi chromosomos iš genotipo erdvės 10000000000 ir 01111111111 yra pakankamai gretimi sprendiniai (pagal Euklidinio atstumo matą), tačiau transformavus šiuos sprendinius į fenotipo erdvę atstumas tarp sprendinių tampa labai dideliu. Ši problema gali labai smarkiai padidinti populiacijos diversifikaciją, dėl ko gali būti prarastas balansas tarp diversifikacijos ir intensifikacijos, tuo atveju algoritmo metu rasto sprendinio kokybės garantija smarkiai sumažėja. Žinoma yra pakankamai nemažai metodų šiai bei kitoms problemoms, susijusioms su sprendinio kodavimu, apeiti. [16]

Dvejetainis kodavimas naudojamas sprendžiant tik labai nedidelį kiekį realaus pasaulio uždavinių. Daugelyje pramonės šakų kylančioms problemoms šio kodavimo sistemos nepakanka. Dėl to per paskutiniuosius dešimt metų pakankamai nemažai dėmesio buvo skirta įvairiems kodavimo metodams, kurie padidindavo genetinio algoritmo efektyvumą sprendžiant įvairaus pobūdžio problemas. Priklausomai nuo to, kokia informacija saugoma kiekviename gene, išskiriamos kelios pagrindinės sprendinių kodavimo klasės [9] :

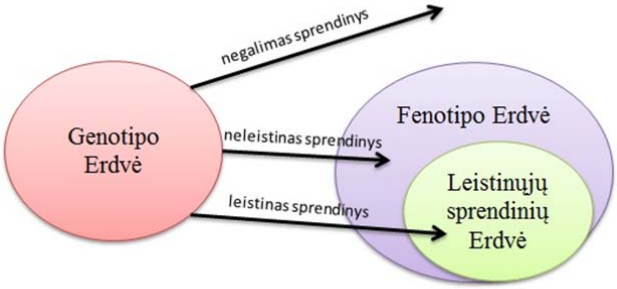
a. Dvejetainis kodavimas;
b. Kodavimas realiais skaičiais;
c. Kėlinių kodavimas;

d. Įvairios duomenų struktūros.

Kiekvienam sprendinio kodavimo metodui keliami tam tikri reikalavimai [9]:

a. Bijekcija. Sprendinio kodavimas turi būti bijektyvi transformacija;
b. Leistinumas. Visi fenotipo erdvės sprendiniai turi būti leistini (2.1 pav);
c. Užbaigtumas. Kiekvienas sprendinys iš fenotipo erdvės gali būti atvaizduotas genotipo erdvėje.

Šiame kontekste laikysime, kad sprendinys yra negalimas, jeigu jame dalis genų pasikartoja, arba dalies genų trūksta. Tokie atvejai labai dažnai pasitaiko atliekant genetinio kryžminimo operaciją. Norint išvengti negalimų sprendinių konstruojamos įvairios sprendinių atstatymo funkcijos, kurios gali labai ištęsti algoritmo konvergavimo laiką. Dėl to šios funkcijos yra vengtinos. Sprendinį laikysime neleistinu, jeigu fenotipo erdvėje užfiksuojami pažeidimai sprendinio apribojimų aibėje, pvz. užduoties technologinė seka ir t.t. [16]



Pav. 2.1. Negalimas ir neleistinas sprendinys [9]

Šiame darbe naudosime Ono Yamamura ir Kabayash pasiūlyta darbu sekos matricos kodavimą. Šiuo atveju chromosoma turi matricini pavidalą:

$$\begin{pmatrix} M1 & \{J_1\, J_2\, J_4\, J_5 \dots\} \\ M2 & \{J_1\, J_4\, J_5 \dots\} \\ \dots & \{\dots\} \\ M_n & \{J_1\, J_4 \dots\} \end{pmatrix}$$

čia pirmajame matricos stulpelyje pateikiami darbo centrų numeriai $M_1, M_2 \dots . M_n$, o antrajame stulpelyje pateikiamos atitinkamų darbo centrų užduočių sekos užduočių atlikimo tvarka. Šiuo atveju naudojant tokį sprendinių kodavimo modelį su sekančiame skyrelyje aprašytais kryžminimo ir mutacijos metodais, sprendinio išsigimimo tikimybė tampa nuline. Dėl to sutaupoma laiko, kuris būtų reikalingas naudojant išsigimusio sprendinio pataisymo funkcijas. [11]

2.2.2. Tikslo funkcija

Šiame darbe nagrinėjami duomenys yra statiniai – iš anksto žinomi, be to tvarkaraštyje nevertinami tokie atsitiktiniai įvykiai, kaip neplanuotos darbo centrų prastovos, gamybiniai defektai

ir t.t. Siekiant, kad tokiu principu formuojamas tvarkaraštis būtų kuo arčiau realybės, tvarkaraštis negali būti sudaromas ilgiems laikotarpiams. Sinchroninės gamybos sistemoje rekomenduojama dienos užduotis pateikti dviejų valandų ritmu. Žinoma, toks dažnas tvarkaraščių sudarymas reikalauja nemažai resursų, tačiau gaunama nauda yra žymiai tikslesnis dienos plano įvykdymas susietas su intensyvesniu kontrolės mechanizmu, lyginant su ilgesnio periodo tvarkaraščiais. Šiuo metu nagrinėjamoje prekybinės įrangos gamykloje formuojamas dienos užduočių tvarkaraštis (1.1 pav.). Be to, paruošiami ir dviejų sekančių dienų tvarkaraščiai, kurie gali būti redaguojami tik esant rimtiems gamybiniams sutrikimams. Taigi, šiuo atveju rekomenduosime tvarkaraštį sudaryti ne ilgesnį nei trijų dienų laikotarpiui.

Gamybinio tvarkaraščio sudarymui naudosime trumpiausio gamybinio laiko (angl. *minimum makespan*) minimizavimo funkciją :

Tarkime užduotis j yra apdorojama darbo centre i turinčiame n_i lygiagrečiai veikiančių įrengimų. Tarkime, C_i^j yra laiko momentas, kuomet užduotis j baigiama apdoroti darbo centre i, o c_{ik}^j yra laiko momentas, kuomet užduotis j baigiama vykdyti darbo centro i k- tajame įrengime $(k \in n_i)$. Kaip jau minėjome anksčiau, užduotis j gali būti apdorojama tik viename iš n_i lygiagrečiai dirbančių įrengimų. Dėl to įsiveskime papildomą fiktyvų kinamąjį x_{ik}^j

$$x_{ik}^j = \begin{cases} 1, & jeigu\ užduotis\ j\ atliekama\ darbo\ centro\ i\ k-tajame\ įrengime \\ 0, & kitu\ atveju \end{cases} \quad (2.5)$$

Tuomet įrengimas k užduotį j baigia vykdyti laiko momentu

$$c_{ik}^j = \max\left(C_{i-1}^j, c_{ik}^{j-1}\right) + p_{ij} \cdot x_{ik}^j \quad (2.6)$$

Čia p_{ij} – užduoties j apdorojimo trukmė darbo centre i. Tuomet darbo centro i užduoties j vykdymo pabaigos momentas yra

$$C_i^j = c_{ik}^j \quad (2.7)$$

Bendras visų užduočių vykdymo pabaigos momentas darbo centre i bus lygus

$$C_i = \max_j\{C_i^j\} \quad (2.8)$$

Bendras visų užduočių vykdymo pabaigos momentas

$$C = \max_i\{C_i\}$$

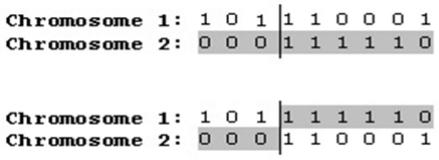
Taigi šiuo atveju jeigu ϕ yra bet kuris fenotipo erdvės elementas, tuomet jo įverčio funkcija

$$f(\phi) = C(\phi) \quad (2.9)$$

2.2.3. Genetinio kryžminimo operatorius

Genetinio kryžminimo operatorius šiuo atveju yra pagrindinis operatorius. Kryžminimo operatorius suporuoja dvi tėvinės chromosomas (genotipo erdvės elementus) ir taip atsiranda du palikuonys (chromosomos). Paprasčiausiu atveju, jeigu genotipo erdvės elementas yra reprezentuojamas

simbolių eilute, tuomet kryžminimo metu atsitiktinai pasirenkamas kirtimo taškas, kuris abi tėvines chromosomas padalina į dvi dalis- pradinį segmentą ir galinį segmentą. Abiejų tėvinių chromosomų pradiniai segmentai nukopijuojami į palikuonių chromosomas įprasta tvarka, o galiniai segmentai nukopijuojami sukeitus juos tarpusavyje. Taip gaunami du palikuonys turinys abiejų tėvų savybių (2.2 pav.)



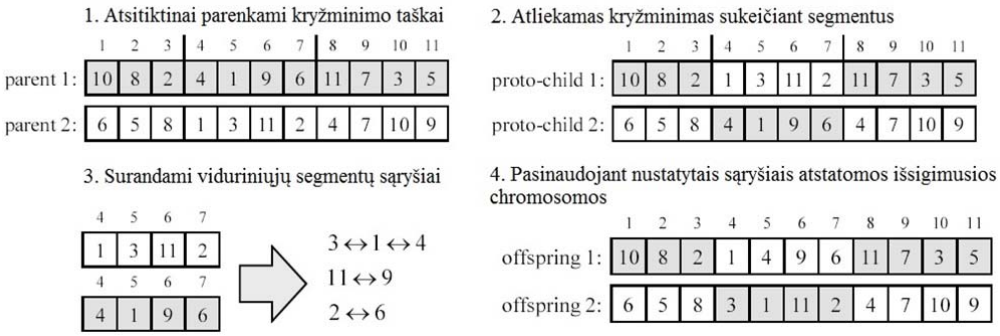
Pav. 2.2 Vienataškis kryžminimas

Galima teigti, kad bendras genetinio algoritmo veikimo laikas priklauso nuo kryžminimo operatoriaus efektyvumo laiko atžvilgiu. [9]

Nepriklausomai nuo pasirinkto kryžminimo operatoriaus modelio, pagrindinis parametras apibūdinantis šio metodo veikimo intensyvumą yra kryžminimo tikimybė ρ_C, kuri parodo, kad apytiksliai $\rho_C \cdot pop_{Size}$ palikuonių bus gauta kiekvienos generacijos metu, čia $pop_{Size}-$ populiacijos dydis. Didesnė kryžminimo tikimybė leidžia geriau ištyrinėti sprendinio aplinką, kas padidina tikimybę kiekvienoje sprendinio aplinkoje rasti tikruosius lokalius optimumus. Žinoma jei tikimybė yra per didelė, tuomet daugeliu atvejų yra eikvojamas skaičiavimo laikas tiriant nepotencialių sprendinių aplinkas.

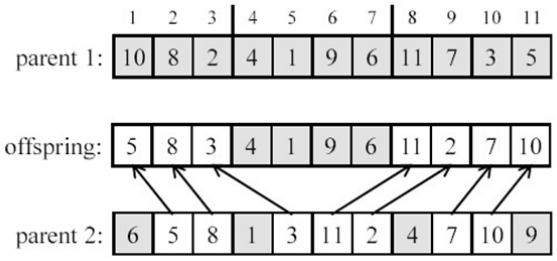
Kalbant apie kryžminimo operatoriaus modelį, galima pabrėžti, kad šiuo metu jų yra pakankamai nemažai, pavyzdžiui [16]:

a. Dalinai susietas kryžminimas (angl. *partial-mapped crossover* (PMX)) (2.3pav) - tai paprastas dviejų kirtimo taškų kryžminimas su integruota sprendinio pataisymo procedūra;



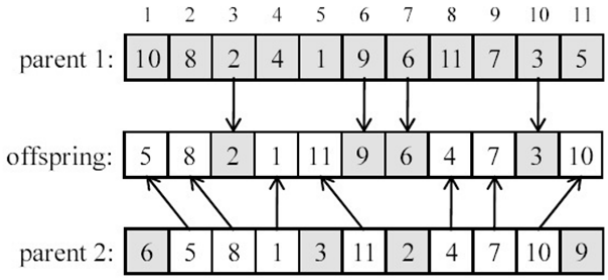
Pav. 2.3. Dalinai susietas kryžminimas

b. Kryžminimas pagal tvarkos sąrašą (angl. *order crossover* (OX)) (2.4 pav.) – tai modifikuota PMX kryžminimo versija su kitokia atstatomąja funkcija;



Pav. 2.4 Kryžminimas pagal tvarkos sąrašą.

c. Pozicinis kryžminimas (angl. *position-based crossover* (PX)). Tai modifikuota OX kryžminimo versija, kuomet naudojama kintamo dydžio kryžminimo taškų aibė C. Aibės dydis sugeneruojamas atsitiktinai $|C| \in [0, k]$, čia k – chromosomes ilgis. Kiekvienas kryžminimo taškų aibės elementas nurodo kuris chromosomos genas yra nejudinamas (tiesiogiai perkopijuojamas iš tėvinės chromosomos į palikuonį) - $C_i \in [0, k]$. Likę tušti palikuonio tarpai užpildomi antrojo tėvo dar neįterptais genais jų išsidėstymo tvarka (2.5 pav).



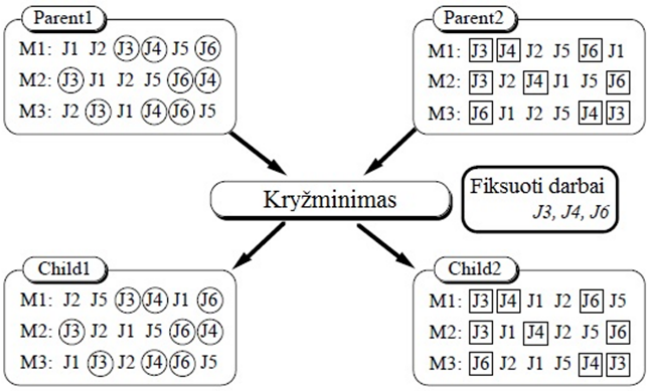
Pav. 2.5 Pozicinis kryžminimas

Kaip matome kryžminimo operatoriaus modelių yra pakankamai įvairių. Skirtingi kryžminimo operatoriai lemia skirtingus sprendinio aplinkos nagrinėjimo metodus. Literatūroje [5] pateikiamas apibendrintas šių modelių klasifikavimas paremtas modelių kanonine forma arba euristikų naudojimu. Kanoninės formos modelių klasei priklausantys kryžminimo operatoriai yra operatoriai su vienu, dviem ar daugiau kryžminimo taškų. Šie operatoriai neturi integruotų chromosomos atstatymo procedūrų. Nors ir naudojant tokias procedūras, nėra jokių garantijų, kad dviejų chromosomų palikuonis turės geresnį tikslo funkcijos įvertį lyginant su abiejų tėvinių chromosomų įverčiais. Tuo tarpu euristinei klasei priklausantys kryžminiai operatoriai savyje gali turėti integruotas chromosomų atstatymo procedūras bei įvairias kitokias euristikas, skirtas efektyviam kryžminimo taškų parinkimui. [16]

Šiame tiriamajame darbe naudosime klasikinį kanoninės formos modelių klasei priklausantį pozicinio kryžminimo operatoriaus modelį suderintą su darbų sekos matricos kodavimu genotipo erdvėje. Šį kryžminimo modelį pasiūlė Ono Yamamura ir Kabayash, teikdamas, kad šis operatorius palikuonims perduoda tik atsitiktinai atrinktas tėvinių chromosomų savybes, kas leidžia geriau ištirti kiekvieno sprendinio aplinką. Be to paveikus šiuo operatoriumi bet kurį genotipo erdvės

elementą, niekada nepasireikš elemento išsigimimas – visi gauti sprendiniai bus leistinieji [11]. Operatorius sukryžmina dvi darbų sekos matricas – tėvines chromosomas ir gauna du palikuonis, veikdamas pagal šį algoritmą (2.6 pav.):

1. Sudaromas atsitiktinis fiksuotų darbų sąrašas. Šie darbai su užimamomis pozicijomis bus paveldėtos nepakitusios;
2. Atrinkti darbai su atitinkamomis pozicijomis nukopijuojami iš pirmos tėvinės chromosomos į pirmą palikuonį ir iš antros tėvinės chromosomos į antrą palikuonį;
3. Nukopijuojami visi likę darbai išlaikant jų seką iš antros tėvinės chromosomos į pirmą palikuonį ir iš pirmos tėvinės chromosomos į antrą palikuonį.



Pav. 2.6 Darbų sąrašo matricomis paremtas pozicinis kryžminimas

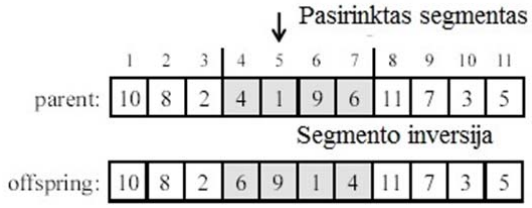
Šiuo atveju naudojant pozicinį kryžminimą suderintą su genotipo erdvės kodavimu sprendinio išsigimimo tikimybė tampa nuline. Dėl to sutaupomas laikas, kuris būtų reikalingas naudojant išsigimusio sprendinio atstatymui.[11]

2.2.4. Mutavimo operatorius

Mutavimo operatoriaus pagrindinis vaidmuo yra palaikyti populiacijos diversifikaciją bei padėti algoritmui pereiti prie naujų sprendinių ir tirti jų aplinkas. Pats paprasčiausias mutavimo operatoriaus atvejis yra dviejų atsitiktinai pasirinktų genų sukeitimas vienoje chromosomoje. Mutavimo operatorių šiame darbe apibūdinsime dviem pagrindiniais parametrais: mutavimo tikimybe ρ_M ir mutavimo lygiu r. Mutavimo tikimybė parodo, kad vidutiniškai $\rho_M \cdot popSize$ chromosomų mutuos kiekvienoje generacijoje. Labai mažas mutacijos koeficientas neleidžia pasiekti naujų tyrinėjimo sričių, dėl ko prastėja surasto sprendinio optimalumo kokybė. Kitą vertus per daug didelė mutavimo tikimybė labai padidina visos populiacijos diversifikacija, suprastėja lokalios paieškos rezultatai, gauti palikuonys praranda gerasias tėvų savybes ir algoritmas praranda gebėjimą panaudoti paieškos procese surinktą informacija. Mutavimo lygis r nurodo kiek kartų iš eilės ta pati chromosoma bus mutuojama – tai yra mutavimo stiprumo matas.

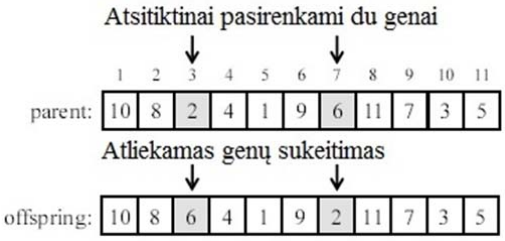
Mutavimo operatorių modelių taip pat yra pakankamai nemažai. Šiame darbe nagrinėsime du mutavimo operatorių modelius: [16]

a. Inversiškas mutavimas (angl. *inversion mutation*). Vienoje chromosomoje pasirenkami du atsitiktiniai taškai. Kraštiniai chromosomos segmentai neliečiami, o tuo tarpu vidurinio segmento pakeičia rikiavimo kryptį. Atsižvelgiant į genotipo elemento struktūrą, šiuo atveju mutavimas taikomas atsitiktinai pasirinkto darbo centro užduočių sekoje, kuri pateikiama kaip kėlinių sąrašas (2.7 pav); [16]



Pav. 2.7 Inversiška mutacija

b. Sukeičiantis mutavimas (angl. *swap mutation*). Šiuo atveju atsitiktinai pasirenkami du elementai ir jie apkeičiami vietomis (2.8 pav.). [16]



Pav. 2.8 Sukeičiantis mutavimas

Šiuo atveju pagal mutavimo poveikį galima teigti, kad didžiausias atstumas tarp sprendinių genotipo erdvėje atsiranda pritaikius inversišką mutavimą, antroje vietoje yra sukeičiantis mutavimas ir trečioje yra įterptinis mutavimas. Šio tyrimo metu bandysime parinkti tinkamiausią mutavimo metodą nagrinėjamai problemai bei surasti optimalius mutavimo tikimybės ir mutavimo lygmens įverčius.

2.2.5. Pradinė populiacija

Pradinės populiacijos sudarymui buvo sudarytas nesudėtingas algoritmas :

1. Pirmasis naujos populiacijos narys yra pradinių pateiktų duomenų atvaizdavimas genotipo erdvėje. Šiuo atveju visos darbų sekos yra surikiuotos didėjimo tvarka.

$$\begin{pmatrix} M_1 & \{J_{M_1}(1), J_{M_1}(2), J_{M_1}(3), \dots, J_{M_1}(|J_{M_1}|)\} \\ M_2 & \{J_{M_2}(1), J_{M_2}(2), J_{M_2}(3), \dots, J_{M_2}(|J_{M_2}|)\} \\ \dots & \\ M_n & \{J_{M_n}(1), J_{M_n}(2), J_{M_n}(3), \dots, J_{M_n}(|J_{M_n}|)\} \end{pmatrix}$$

Čia M_i- darbo centro numeris, o $J_{M_i}(k)$ – darbo centro M_i užduočių sekos $k-tasis$ elementas (užduotis);

2. Tarkime, kad populiaciją sudaro *popSize* skaičius narių. Pirmasis šios populiacijos narys aprašytas pirmajame etape. Visi sekantys nariai gaunami tokiu principu. Pirmiausiai į naujo populiacijos nario vietą nukopijuojamas prieš tai buvęs narys tuomet einamam populiacijos nariui s paleidžiamas daugkartinio mutavimo algoritmas:

```
For i = 1 To n
    For j = 1 To |J_{M_n}|
        l ← Atsitiktinis(1,2, … , |J_{M_n}|)
        s ← Sukeičianti_Mutacija (J_{M_i}(j), J_{M_i}(l))
    Next
Next
```

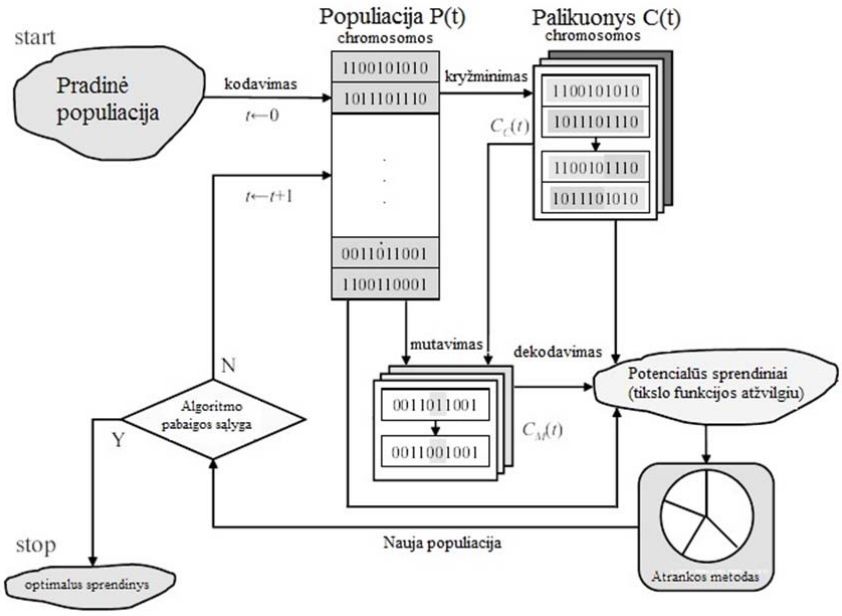
Kartojant šį algoritmą užpildoma visa populiacija, kurios atsitiktiniai sprendiniai pakankamai gerai padengs visų sprendinių erdvę.

2.2.6. Genetinės atrankos operatoriai

Genetinio algoritmo modelis pavaizduotas 2.9 pav. Genetinės atrankos operatorius yra šio algoritmo varančioji jėga. Šis operatorius veda genetinę paiešką į rezultatyvias paieškos aplinkas. Per pastarąjį dešimtmetį buvo sudaryta pakankamai nemažai atrankos operatorių modelių. Šiame darbe nagrinėsime tris dažniausiai naudojamus operatorius [16]:

a. Ruletės rato metodas;
b. Turnyrinės atrankos metodas;
c. $\mu + \lambda$ atrankos metodas.

Ruletės rato metodas. Šis metodas buvo pasiūlytas Holando [9] ir tai yra plačiausiai taikomas atrankos metodas. Pagrindinė šio metodo idėja yra įvertinti kiekvieno populiacijos elemento išgyvenimo tikimybę, kuri priklauso nuo tikslo funkcijos reikšmės. Įvertinus visas išgyvenimo tikimybes sudaroma skirstinio funkcija, kurios kiekvieno laiptelio intervalas atvaizduojamas menamame ruletės rate. Pasukus šį ratą išriedėjęs kamuoliukas patenka į vieną iš intervalų, šiam



Pav. 2.9 Genetinio algoritmo struktūra [9]

intervalui priklausantis atsitiktinis dydis priskiriamas naujai populiacijai. Panagrinėkime šio metodo matematinį modelį. Tarkime, kad G yra generacijų aibė, tuomet $G_t - t$ kartos populiacija, o $s_{t,i} -$ šios populiacijos elementai (genotipo erdvės sprendiniai), $1 \leq i \leq pop_{size}$. Tuomet pirmiausiai surandamos kiekvieno populiacijos nario išgyvenimo tikimybės $p_{t,i}$

$$p_{t,i} = \frac{f(s_{t,i})}{\sum_i f(s_{t,i})} \tag{2.10}$$

Tuomet sudarome skirstinio funkcija

$$F(i,t) = \sum_{k \leq i} p_{t,k} \tag{2.11}$$

Galiausiai sugeneruojame atsitiktinį dydį r , tolydžiai pasiskirsčiusį intervale (0,1), bei randame atvirkštinę skirstinio funkciją. Tuomet

$$F^{-1}(r,t) \to i \tag{2.12}$$

Taip gaunamas elementas, kuris išgyveno t generaciją ir pateko į naują kartą. Procedūra kartojama tol, kol pripildoma visa apibrėžto dydžio populiacija. [19]

Turnyrinės atrankos metodas. Šį metodą pasiūlė Goldbergas [5]. Pagrindinis šio metodo parametras yra turnyrinio sąrašo dydis. Algoritmas pirmiausiai užpildo turnyrinį sąrašą atsitiktiniu būdu atrinkdamas pirminės populiacijos chromosomas. Sudarius sąrašą iš jo atrenkama geriausią tikslo funkcijos įvertį turinti chromosoma, kuri patenka į naują generaciją. Neretai naudojamas turnyrinės atrankos dydis yra du elementai, šiuo atveju turnyras vadinamas binariniu. Taip pat buvo pasiūlytas ir stochastinis turnyro sudarymo metodas, kurio vienu iš galimų atveju gali būti ruletės

rato atrankos metodas. Panaudojus ruletės rato atrankos metodą užpildoma turnyrinis sąrašas ir iš šio sąrašo atrenkamas elementas, turintis geriausią tikslo funkciją.[19]

$\mu + \lambda$ atrankos metodas kaip kontrastas proporciniams atrankos metodams, buvo pasiūlytas deterministinis atrankos metodas. Metodo pagrindinė esmė yra sukryžminti λ tėvinių chromosomų ir taip gauti λ palikuonių. Šiuo atveju populiacijos dydis yra μ chromosomų. Atlikus kryžminimą gaunama $\mu + \lambda$ dydžio populiacija, iš kurios atrenkama μ geriausių chromosomų, kurios perkeliamos į sekančią generaciją. Šiuo atveju sudarytoje naujoje populiacijoje negali būti pasikartojimų. Galima ir kita algoritmo modifikacija (μ, λ), kuomet nauja populiacija sudaroma tik iš λ geriausių chromosomų. Šiuo atveju turi būti tenkinamas reikalavimas $\lambda > \mu$. [5]

2.3. Apribojimo teorijos pritaikymas sudarant gamybinius tvarkaraščius

Šiame skyrelyje panagrinėsime euristiką, specialiai pritaikytą lankstaus srauto gamybos fabrikui. Apribojimų teorija (angl. *theory of constraints*), tai efektyvi, pakankamai nesudėtinga ir todėl plačiai taikoma verslo valdymo filosofija. Šia filosofija vadovaujasi tokios žinomos kompanijos kaip Boening Inc., Motorola Inc., Ford Motor Company ir t.t. [17]. Gamybiniu aspektu žiūrint, apribojimų teorija teigia, kad bet kurios sistemos bendrąjį pralaidumą (šiuo atveju gaminiais, detalėmis ir t.t.) lemia didžiausias sistemos apribojimas, tiesiogiai įtakojantis greitį, kuriuo įmonė juda link pagrindinio tikslo – uždirbti pinigus. Dėl to sudarytos euristikos pagrindinis tikslas yra efektyviai išnaudoti apribojimą.

Panagrinėkime šią euristiką plačiau. Apribojimo identifikavimo pavyzdys pateiktas lentelėje (lentelė 1). Čia kiekvieno gaminio apdorojimo laikas kiekviename įrengime nurodytas minutėmis. Turint tokio pavidalo pradinių duomenų lentelę bei vadovaujantis žemiau pateiktais apribojimo identifikavimo bei planavimo metodika, galime gauti sprendinį, kuriame apribojimo darbo centras bus išnaudotas efektyviausiai – darbo centras bus apkrautas maksimaliu pajėgumu su mažiausiom prastovom.

GAMINYS	Darbo centras 1	Darbo centras 2	Darbo centras 3	Darbo centras 4	Darbo centras 5	Laikas iki apribojimo	Apribojimo pėdsakas
Gaminys1	172			351	135	0	
Gaminys2	178			361	136	0	
Gaminys3		250	167	241	142	250	4663.75
Gaminys4		111	194	197	162	111	4802.75
Gaminys5		165	211	317	141	165	4748.75
Gaminys6		180	301	212	137	180	4733.75
Gaminys7		120	293	241	136	120	4793.75
Gaminys8		202	138	181	136	202	4711.75
Gaminys9		169	222	198	141	169	4744.75
Lygiagretūs įrengimai	1	1	1	4	1		
Bendras apkrovimas	350	1197	1526	574.75	1266	4913.75	

Lentelė 1 Apribojimo identifikavimo pavyzdys

1 etapas : Apribojimo identifikavimas

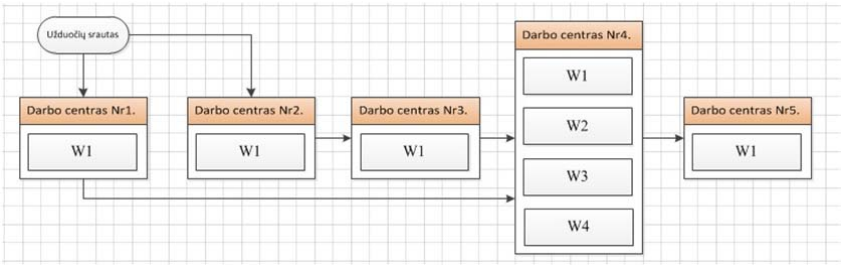
a. Pirmiausiai apskaičiuojamas kiekvieno darbo centro apkrovimas. Tai daroma sumuojant visų pro turimą darbo centrą einančių, užduočių apdorojimo laikus bei dalinant šį laiką iš darbo centro įrengimų skaičiaus. Bendras cecho arba baro apkrovimas bus įvardijamas kaip visų tam cechui arba barui priklausančių darbo centrų apkrovimų suma;
b. Darbo centras turintis didžiausią apkrovimą yra įvardijamas kaip apribojimas;
c. Kiekvienam darbui suskaičiuojamas gamybinis laikas iki apribojimo darbo centro;
d. Suskaičiuojamas apribojimo pėdsakas kiekvienam darbui, t.y. skirtumas tarp bendro sistemos apkrovimo ir turimos užduoties laiko iki apribojimo.

2 etapas : Apribojimo planavimas

a. Apribojimo darbų seka dėliojama apribojimo pėdsako didėjimo tvarka;
b. Jeigu kurie nors darbai pagal apribojimo pėdsaką yra lygiaverčiai, tuomet į sąrašą jie dedami duomenų pateikimo tvarka arba pagal papildomai nustatytą svarbos koeficientą;
c. Jeigu apribojime arba kuriame nors kitame darbo centre dirba daugiau nei vienas įrengimas, tuomet užduotys jiem skirstomos pirmiausiai atsilaisvinusio įrengimo tvarka;
d. Apskaičiuojami kiekvieno darbo pradžios ir pabaigos laikai apribojimo darbo centre.

3 etapas: Kitų darbo centrų planavimas

a. Jeigu *j* užduoties technologinio maršruto kelyje yra apribojimas, tuomet sudarinėjamas pusiau dinaminis tvarkaraštis, kurio metu visos užduotys suskirstomos į du sąrašus (kurios eina per apribojimo darbo centrą ir kurios ne). Jeigu užduotis iš antrojo sąrašo neįtakoja užduoties iš pirmojo sąrašo vykdymo pradžios laiko, tuomet į darbo centrui priskiriama užduotis iš antrojo sąrašo. Priešingu atveju iš pirmojo sąrašo. Pirmasis sąrašas surikiuotas apribojimo pėdsako didėjimo tvarka, o antrasis – parengtumo vykdymui laiko momento didėjimo tvarką. Ciklas kartojamas, kol priskiriami visi darbai iš abiejų sąrašų;
b. Jeigu *j* užduoties technologinio maršruto kelyje apribojimo nėra, tuomet visi nagrinėjamo darbo centro darbai surikiuojami parengtumo vykdyti laiko momento ir apdorojimo laiko didėjimo tvarka.



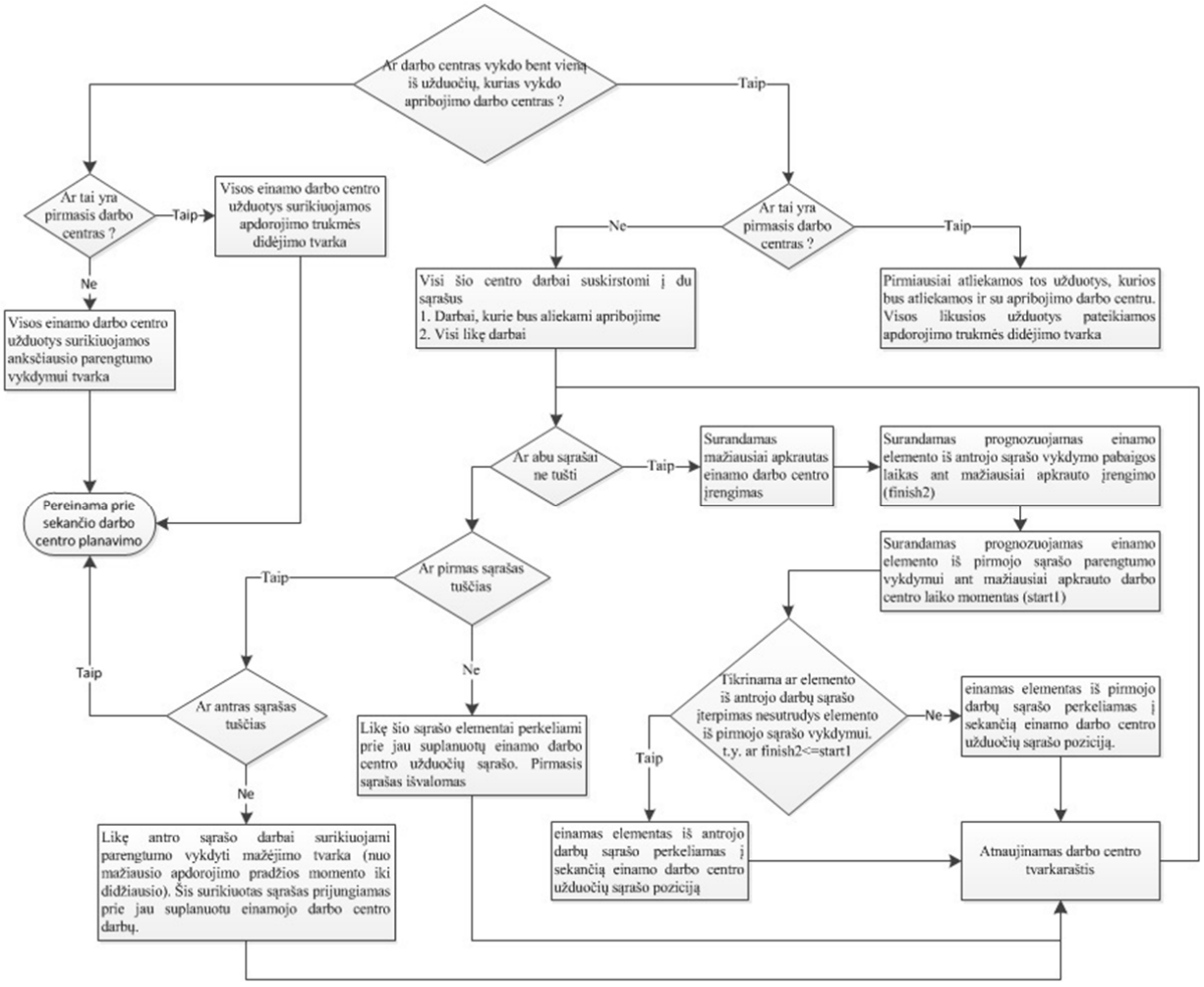
Pav. 2.10. Srautinės gamybos modelis

Lentelėje nr. 1 pateiktas srautinės gamybos pavyzdys, kurį galima iliustruoti 2.10 pav. Šiame pavyzdyje nagrinėjami 9 gaminiai. Gamybinę sistemą sudaro 5 darbo centrai ir kurių ketvirtajame dirba 4 lygiagretūs įrengimai. Kiekvieno gaminio apdorojimo laikai pateikti 1 lentelėje. Kaip matome pagal bendrąjį apkrautumą, darbo centras nr3 yra šio uždavinio didžiausias apribojimas. Tuomet šio darbo centro užduočių seka atrodytų taip :

$(Gaminys4, Gaminys7, Gaminys5, Gaminys9, Gaminys6, Gaminys8, Gaminys3)$.

Remiantis parengtumo bei apdorojimo laikais galima sudaryti tvarkaraštį apribojimui. Užduočių sekos visiems likusiems darbo centrams sudaromos remiantis 3 etape aprašytais žingsniais. Šios euristikos pagalba gautas bendras gamybos laikas yra 2040 min. Tyrimo metu sudarytoje programoje realizuotas algoritmas pateiktas 2.11 pav.

Sekančiame skyrelyje palyginsime šio euristinio metodo bei genetinio algoritmo efektyvumą sudarinėjant gamybinius tvarkaraščius.

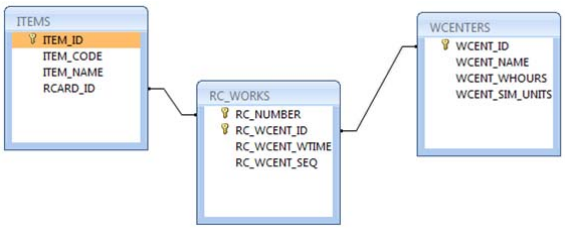


Pav. 2.11 Programoje realizuotas algoritmas paremtas apribojimų teorija

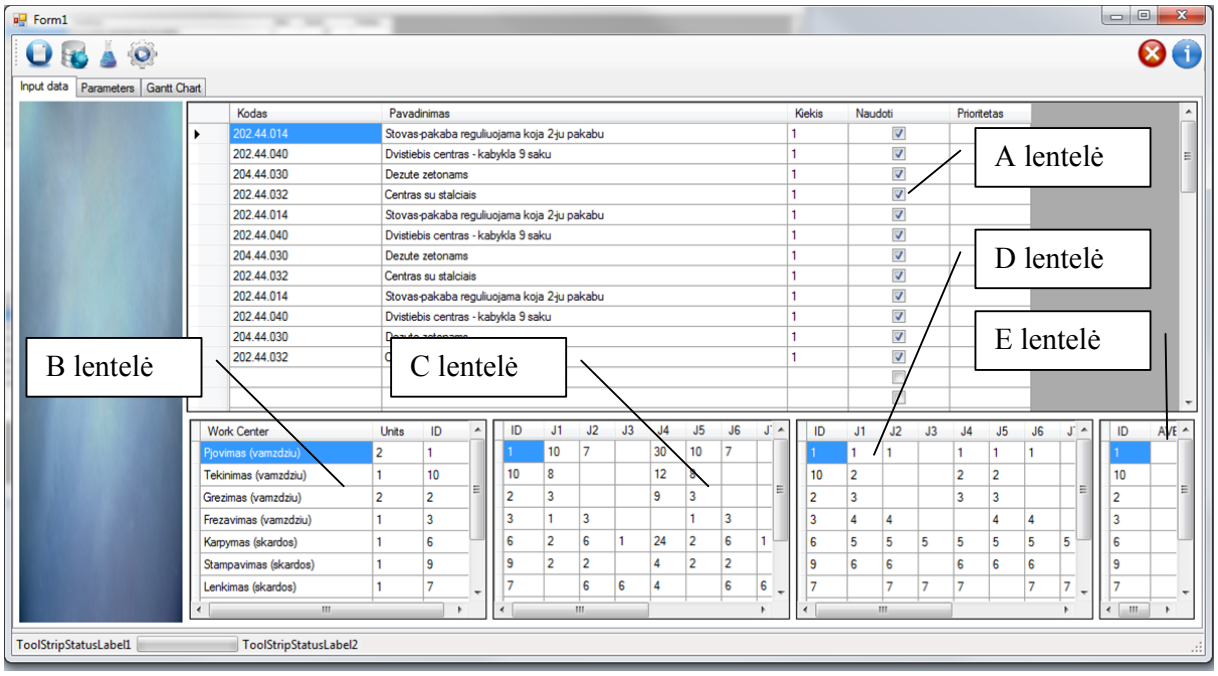
2.4. Programinė įranga

Tyrimo metu naudosime programinę įrangą, skirtą srautinio tipo gamybos tvarkaraščiams sudaryti bei tirti. Programinė įranga realizuota vb.net aplinkoje

Pradiniai duomenys : Programa naudoja realios gamybos duomenų bazę, kurios supaprastinta struktūra pavaizduota (2.12 pav.). Naudojant praktikoje taikomą pavyzdį, programai pateikiamas tik gaminių sąrašas, kurio raktinis laukas yra gaminio kodas (item_code). Pagal šį sąrašą iš duomenų bazės išgaunama kiekvieno gaminio technologinė kortelė (įrengimų seka ir apdorojimo laikai). Šiuo lentelėje ITEMS gaminio kodas nėra unikalus įrašas. Unikalus įrašas yra gaminio kodo ir jam priskirtos maršrutinės kortelės (RCARD_ID / RC_NUMBER) kombinacija, kadangi vienas gaminys gali turėti kelias maršrutines korteles priklausomai nuo gamybos tipo bei partijos dydžio. WCENTERS lentelėje pateikiamas įrengimų sąrašas. RC_WORKS lentelėje aprašomos kiekvienos maršrutinės kortelės gamybos technologija (laikai – RC_WCENT_WTIME bei darbo centro RC_WCENT_ID eilės numeris RC_WCENT_SEQ gamybos procese). Šiuo atveju prieš paleidžiant kiekvieną gaminį į gamybą būtina jį apsirašyti sistemos duomenų bazėje.



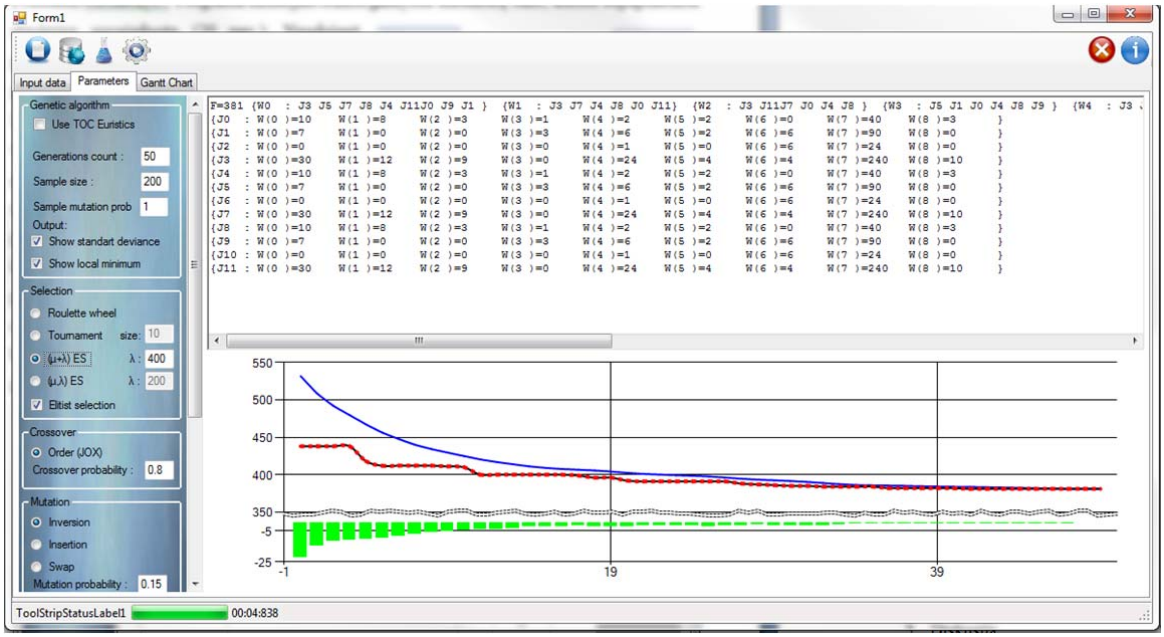
Pav. 2.12. Programos duomenų bazė



Pav. 2.13. Programinė įranga – duomenų įvesties langas

A lentelė (2.14 pav.) nurodo gaminių, kuriuos reikės gaminti, sąrašą. B lentelėje pateikiama visa informacija apie esamus darbo centrus (pavadinimas, lygiagrečiai dirbančių įrengimų skaičius, ir darbo centro kodas). C lentelėje pateikiami kiekvieno gaminio gamybos laikai kiekviename darbo

centre (t.y. matrica A pagal 2.1 skyrelį). D lentelėje pateikiami gamybos maršrutai (šiuo atveju tik pažymimas darbo centro numeris, prie kurio bus apdorojamas gaminys – technologinė seka pateikta iš apačios į viršų). E lentelėje įvedamas kiekvieno įrengimo darbų frontas.



Pav. 2.14. Programinė įranga – parametrų langas.

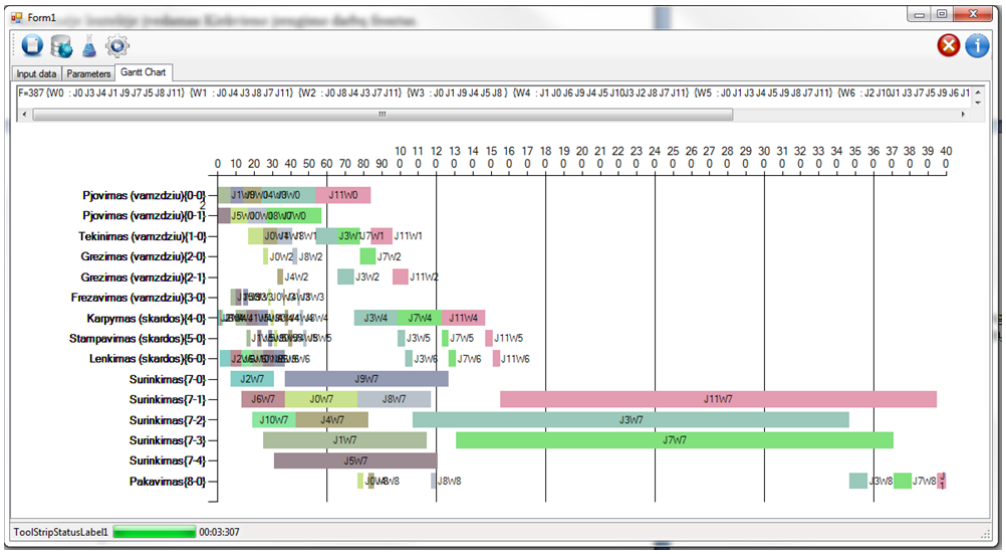
Parametrų lange galima pasirinkti kuris kombinatorinio optimizavimo metodas bus taikomas uždaviniui spręsti (genetinis algoritmas ar TOC euristika). Pasirinkus genetinį algoritmą galima manipuliuoti keturiomis parametrų grupėmis :

a. Aplinkos parametrai (generacijų skaičius, populiacijos dydis ir pirminės populiacijos mutavimo tikimybė);
b. Genetinės atrankos metodai ir jų parametrai;
c. Genetinio kryžminimo parametrai;
d. Genetinės mutacijos metodai ir parametrai.

Rezultatų lange (2.14 pav.) pateikiama pradinė duomenų matrica ir gautas sprendinys su tikslo funkcijos reikšme (šiuo atveju $F(s) = 381$). Grafinėje dalyje mėlynai pažymėta einamos populiacijos vidutinė tikslo funkcijos reikšmė. Juodai pažymėtas geriausio iki šiol rasto sprendinio tikslo funkcijos reikšmė (šiuo atveju minimalus tvarkaraščio vykdymo laikas). Žaliai stulpeline diagrama atvaizduotas einamos populiacijos standartinis nuokrypis (padaugintas iš -1).

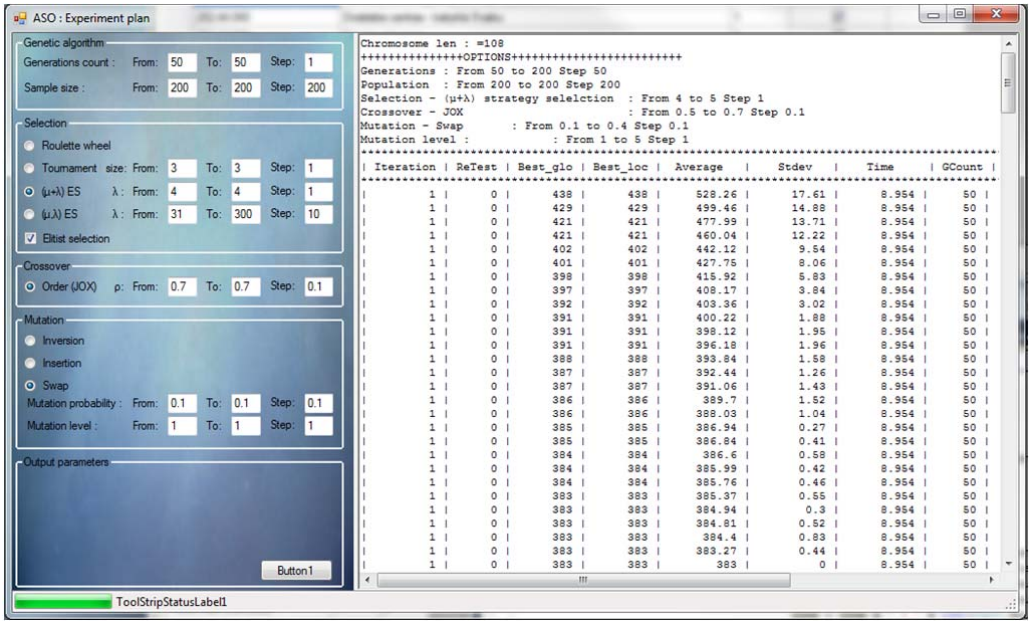
Pagal parinktus parametrus gautas sprendinys atvaizduojamas Ganto diagramos pavidalu (pav. 2.15.). Programoje yra įdiegtos grafiko tolinimo ir artinimo funkcijos, tačiau bendrajam kontekste Ganto diagrama padeda atskirti kuriais laiko momentais įrengimo apkrovimas būna didžiausias. Užduočių pateikimui šis grafinis metodas dažniausiai nėra tinkamas. Galutinis šio

uždavinio sprendinys, pritaikytas realiai gamybai būtų modifikuota kiekių valdymo lentelė (1.1 pav.), kurioje kiekvienam darbui priskiriamas pradžios ir pabaigos laikas bei kiti nurodymai.



Pav. 2.15. Programinė įranga – Ganto diagrama

Eksperimentinio plano modulyje atliekami intervaliniai parametrų testavimai. Kiekvienam parametrui galima nurodyti pradinę ir galinę reikšmes bei postūmio žingsnį. Kiekvienam parametrų rinkiniui programa imitaciją kartoja 50 kartų – tai yra pakankama imtis kokybiškiem įverčiams gauti. Rezultatų lange pateikiamas kiekvienos iteracijos numeris, kiekvienos generacijos metu pasiektas geriausias globalus ir lokalus sprendiniai. Taip pat pateikiamas populiacijos vidurkis, standartinis nuokrypis, sistemos darbo laikas tiriant kiekvieną parametrų rinkinį, bei pats parametrų rinkinys.



Pav. 2.16. Programinė įranga – Eksperimentinis modulis

3. TIRIAMOJI DALIS

Kaip jau minėjome anksčiau, pagrindinis šio darbo tikslas surasti tinkamus adaptyvius genetinio algoritmo parametrų rinkinius įvairaus dydžio pradinių duomenų matricoms, kai sprendžiamas lanksčios srautinės gamybos planavimo uždavinys. Gavus optimalius parametrų rinkinius genetinio algoritmo efektyvumą palyginsime su euristiniu metodu aprašytu 2.3 skyriuje.

3.1. Parametrų tyrimo schema

Genetinio algoritmo tyrimų rezultatas šiuo atveju turėtų būti kiekvieno parametro funkcinė priklausomybė nuo pradinių duomenų matricos dydžio arba kitaip – chromosomos ilgio. Tyrimo pradžioje pasirinksime fiksuotą chromosomos ilgį (100 genų) ir bandysime jam surasti optimalius parametrus. Vėliau ištirsime parametrų kitimą esant 50 ir 200 genų chromosomoms. Kiekvienam chromosomos ilgiui tyrimą skelsime į dvi pagrindines dalis – aplinkos parametrų tyrimą ir rekombinacijos parametrų tyrimą. Tiriamų faktorių schema pateikta 2 lentelėje.

		Aplinkos parametrų tyrimas		**Rekombinacijos parametrų tyrimas**	
		Stebimi parametrai	Fiksuoti parametrai	Stebimi parametrai	Fiksuoti parametrai
Aplinkos parametrai	Populiacijos dydis	{20,100,200}			Geriausias nustatytas
	Generacijų skaičius	300			Geriausias nustatytas
	Genetinės atrankos metodas:				
	Ruletės ratas	-			Geriausias nustatytas
	Turnyrinės atrankos metodas (parametras $T = pop_{size} \cdot k[\%]$)	5%,10%,15%,20%			
	$\mu + \lambda$ (parametras $\lambda = k \cdot \mu$)	$3\mu, 4\mu, 5\mu, 6\mu, 7\mu$			
Rekombinacijos parametrai	Kryžminimo tikimybė ρ_C		0.7	0.3, 0.4, 0.5, 0.6, 0.7	
	Mutavimo tikimybė ρ_M :		0.1	0.1, 0.2, 0.3, 0.4, 0.5	
	Mutavimo lygis r :		1	1, 2, 3, 4, 5	
	Mutavimo metodas				
	Sukeitimo mutavimas		x	x	
	Inversijos mutavimas			x	

Lentelė 2. Tyrimo schema

Aplinkos parametrų tyrimas. Šio tyrimo metu stengsimės parinkti geriausią genetinės atrankos metodą su jam priklausančiu parametrų rinkiniu. Taip pat nustatyti optimalų populiacijos dydį bei generacijų skaičių. Tyrimo metu visi likę parametrai bus fiksuoti pagal literatūroje [9] nurodomas rekomendacijas:

- Mutavimo lygis $r = 1$;

- Kryžminimo tikimybė $\rho_C = 0.7$;
- Mutavimo tikimybė ρ_M=0.1.

Atlikę visus bandymus gausime konkrečią vieną mažiausią tikslo funkcijos reikšmę, kurią laikysime globaliu optimumu. Ir su juo lyginsime visų eksperimentų rezultatus. Apibendrinus, tyrimų metu kiekvieną genetinio algoritmo parametrų rinkinį vertinsime šiais aspektais :

a. Ar su šiuo parametrų rinkiniu buvo pasiektas globalus optimumas. Jei taip, kokia yra sprendinio patikimumo garantija (tikimybė, kad esant atitinkamam parametrų rinkiniui bus pasiektas globalus minimumas):

$$\eta = \frac{N_{min}}{N} = \frac{kiek\ kartų\ buvo\ užfiksuotas\ globalus\ minimumas}{atliktų\ bandymų\ skaičius} \quad (3.1)$$

Analizuojant grafikus, šioje vietoje suprantamesnis bus kitas matmuo – vidutinė globalaus minimumo reikšmė, pasiekta iki $i-$tosios generacijos, eksperimentą kartojant N kartų.

$$\tau_i = \frac{1}{N}\sum_{j=1}^{N} \min_{k\le i} f(P_{k,j}) \quad (3.2)$$

čia $P_{k,j}$ –k kartų evoliucionavusi populiacija, atliekant j pakartotinį eksperimentą $j \le N$;

b. Laikas reikiamas globaliam minimumui pasiekti. Šiuo atveju vertinsime vidutinį populiacijos dydį ir vidutinį generacijų skaičių, kuriam esant globalus optimumas buvo pasiektas daugiausiai kartų.

Rekombinacijos parametrų tyrimas. Suradę optimalų aplinkos parametrų rinkinį pasirinktam chromosomos ilgiui šiuos parametrus fiksuosime ir ieškosime optimalaus rekombinacijos parametrų rinkinio (kryžminimo tikimybės, mutavimo tikimybės, mutavimo lygio). Taip pat bandysime parinkti tinkamą mutavimo metodą (inversiška mutacija arba sukeičianti mutacija).

3.2. Tyrimo rezultatai

Šiame skyriuje pateiksime genetinio algoritmo optimalaus parametrų rinkinio rezultatus. Kaip jau buvo minėta anksčiau pirmiausiai atliksime aplinkos parametrų vertinimą fiksuoto ilgio chromosomai. Šiuo atveju chromosoma bus sudaryta iš 100 $\pm$10% genų.

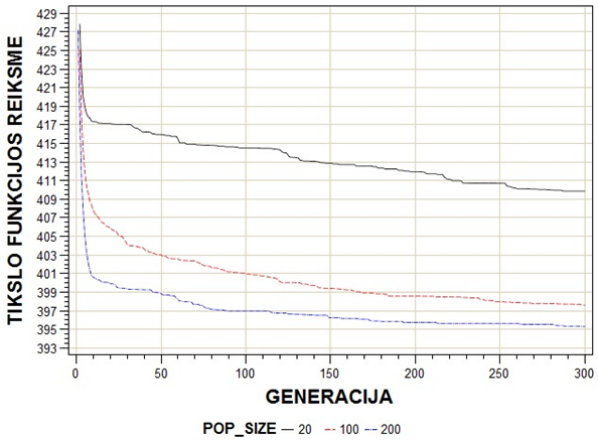
Atlikdami aplinkos parametrų tyrimą nagrinėsime įvairaus dydžio populiacijas {20,100,200}. Generacijų skaičių pasirinkome pakankamai didelį $G = 300$, kad pastebėtume po kurios generacijos atsiranda tikslo funkcijos nusistovėjimas. Šiuo atveju genetinio algoritmo pabaigos sąlyga yra iteracijų skaičius. Kiekvienam parametrų rinkiniui eksperimentą kartosime $N = 50$ kartų. Be to turime iš anksto pažymėti, kad atlikus visus eksperimentus su duotąja duomenų matrica (3 lentelė) gauname optimalų sprendinį, kurio tikslo funkcijos reikšmė yra 381 minutė (tiek laiko reikės norint įvykdyti visas 3 lentelėje išvardintas užduotis).

Darbo centras	Įrengimų skaičius	Vykdymo tvarka	J1	J2	J3	J4	J5	J6	J7	J8	J9	J10	J11	J12
Pjovimas (vamzdžiu)	2	1	10	7		30	10	7		30	10	7		30
Tekinimas (vamzdžiu)	1	2	8			12	8			12	8			12
Gręžimas (vamzdžiu)	2	3	3			9	3			9	3			9
Frezavimas (vamzdžiu)	1	4	1	3			1	3			1	3		
Karpymas (skardos)	1	5	2	6	1	24	2	6	1	24	2	6	1	24
Štampavimas (skardos)	1	6	2	2		4	2	2		4	2	2		4
Lenkimas (skardos)	1	7		6	6	4		6	6	4		6	6	4
Surinkimas	5	8	40	90	24	240	40	90	24	240	40	90	24	240
Pakavimas	1	9	3			10	3			10	3			10

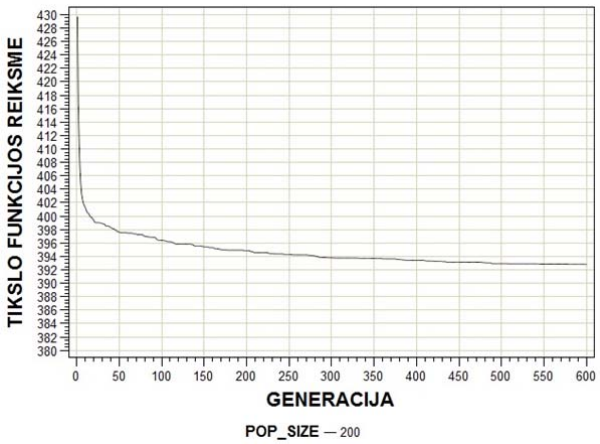
Lentelė 3 Testuojamų duomenų matrica

Aplinkos parametrų tyrimas

Pirmiausiai eksperimentą atlikome su ruletės rato metodu. Kaip matome tokio ilgio chromosomai reikalingas ne mažesnis nei dviejų šimtų individų populiacijos dydis. Be to kaip matome iš 3.1 pav. 300 generacijų skaičius nėra pakankamas. Eksperimentą pakartojome pasirinkę 200 individų populiacijos dydį ir generacijų skaičių padidinome iki 600 (3.2 pav.)

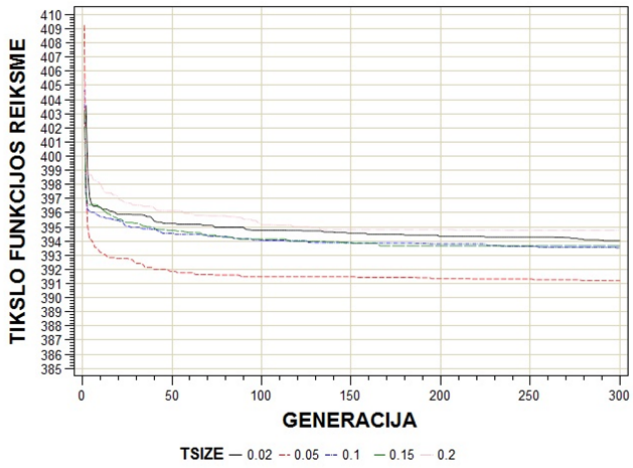


Pav. 3.1. Ruletės rato tyrimas (300 generacijų)



Pav. 3.2 Ruletės rato tyrimas (600 generacijų)

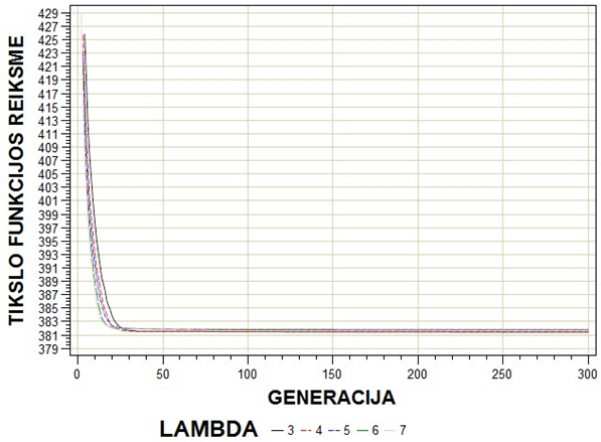
Galime pastebėti kad maždaug po 500 generacijų vidutinė globalaus sprendinio reikšmė nusistovi ties ≈ 392.8 riba, kai tuo tarpu atlikus 300 generacijų ši riba buvo ties ≈ 395.28. Sprendinio kokybę pagerinome, tačiau algoritmas sunaudojo dvigubai daugiau laiko – 500 generacijų algoritmas dirbo 5 sekundes. Žinoma šis laikas yra priimtinas, tačiau žiūrint iš kitos perspektyvos sprendinio kokybės garantija yra labai nedidelė. Šiuo atveju ruletės rato atrankos metodas optimalų surastą sprendinį, kurio tikslo funkcijos reikšmė yra 381 minutė, pasiekė tik 2 kartus iš 50, t.y. $\eta = \frac{2}{50} = 4\%$. Dėl šios priežasties ruletės rato algoritmą laikysime netinkamu ir prie jo grįšime tik tuo atveju, jeigu sekantys algoritmai neduos siekiamų rezultatų.



Pav. 3.3 Turnyrinės atrankos tyrimas

Tirdami ruletės rato metodą aptikome, kad populiacijos skaičius turi būti ne mažesnis negu 200 (su didesniu populiacijos kiekiu rezultatai pagerėja labai nedaug, tačiau tam sunaudojama kur kas daugiau laiko). Todėl ir turnyrinės atrankos metodą tyrėme esant 200 narių populiacijai. Kaip matome iš 3.3 paveikslėlio, ryškiausiai išsiskiriantis turnyrinio sąrašo parametras yra 5% populiacijos (šiuo atveju 10 elementų). Tiek mažesnės, tiek didesnės talpos turnyriniai sąrašai parodė kur kas blogesnius rezultatus. Generacijų dydį padidinus iki 600, vidutinė tikslo funkcijos reikšmė esant 5% dydžio turnyriniam sąrašui pagerėja labai nedaug $f_{300} \approx 391.2 \rightarrow f_{600} \approx 389.3$. Be to, šis metodas geriausio sprendinio su tikslo funkcijos reikšme 381 taip ir nepasiekia.

Sekančiame etape apžvelkime $\mu + \lambda$ metodą. Šiuo atveju parametrą λ irgi keitėme proporcingai populiacijos dydžiui $\lambda \in \{3\mu, 4\mu, 5\mu, 6\mu, 7\mu\}$. Kaip matome pagal 3.4 pav., konvergavimas į globalų optimumą yra pakankamai greitas ir tikslus. Sprendinio kokybė nuo parametro λ beveik nepriklauso. Todėl šiame kontekste svarbiausiu vertinimo kriterijumi tampa algoritmo darbo laikas. Kaip matome iš 3.5 pav. šis metodas yra pakankamai imlus laikui.



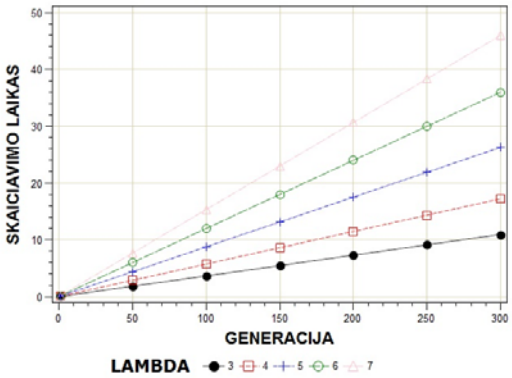
Pav. 3.4 $\mu + \lambda$ atrankos metodo tyrimas

Iš 3.6 pav. galima aiškiai pastebėti, kad algoritmas beveik visada sukonverguoja į globalų optimumą praėjęs 40 generacijų. Tuomet sprendinio kokybės garantija kiekvienam parametrų rinkiniui atrodys taip (4 lentelė):

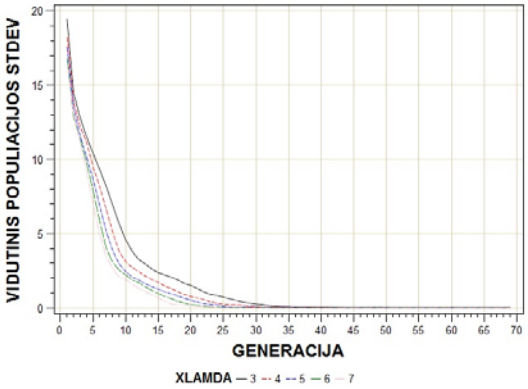
$\lambda = 3\mu$	$\lambda = 4\mu$	$\lambda = 5\mu$	$\lambda = 6\mu$	$\lambda = 7\mu$
$\eta_{40} = 38/50$	$\eta_{40} = 40/50$	$\eta_{40} = 37/50$	$\eta_{40} = 41/50$	$\eta_{40} = 39/50$

Lentelė 4. $\mu + \lambda$ parametrų palyginimas

Kaip matome ryškių priklausomybės tendencijų tarp parametro λ ir sprendinio kokybės garantijos mato η įžvelgti negalime. Tačiau $\lambda = 6\mu$ parametro atveju $\eta = 82\%$, kitavertus $\lambda = 4\mu$ parametro atveju $\eta = 80\%$. Pirmuoju atveju sistemos darbo laikas buvo ≈ 5.67 sekundės, kai tuo tarpu antruoju atveju ≈ 2.87 sekundės (atlikus 50 generacijų). Taupydami sistemos darbo laiką pasirinksime $\lambda = 4\mu$ parametro įvertį,

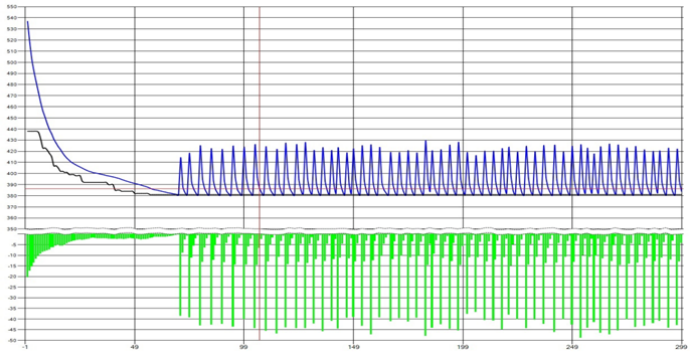


Pav. 3.5 $\mu + \lambda$ atrankos metodo darbo laikas



Pav. 3.6 $\mu + \lambda$ atrankos metodo populiacijų standratiniai nuokrypiai

Tačiau šiuo atveju gali būti kita problema. Kadangi evoliucijos strategija atrenka tik geriausias generacijos chromosomas gali būti pažeidžiamas diversifikacijos ir intensifikacijos balansas. Diversifikacija šiuo atveju leidžia algoritmui patekti į naujas, dar neaplankytas sprendinių sritis, kai tuo tarpu intensifikacija vykdo paiešką aplink geriausius sprendinius. Norėdami ištirti šį disbalansą turime į sistemą integruoti papildomus metodus, kurie leistų dirbtinai sukelti dispersiją. Vienas iš tokių metodų yra „imigrantų“ metodas. Šiame darbe naudosime šio metodo modifikaciją, kurios pagrindinė esmė esant kritiniam visos populiacijos standartinio nuokrypio lygiui (šiuo atveju pasirinktas $\sigma = 0.1$), dalis blogiausių populiacijos narių (šiuo atveju pasirinktą $N = 0.4\mu$) pakeičiami naujais nariais, kitaip tariant šio metodo tikslas – palaikyti populiacijos diversifikaciją. Tačiau, kaip matome iš 3.7 pav. diversifikacijos padidinimas naudos neatnešė (čia juodai pažymėtas globalus minimumas, mėlynai – vidutinė tikslo funkcijos reikšmė, o žaliai – populiacijos standartinis nuokrypis (atvaizduotas plokštumoje $y \leq 0$). Algoritmas pastoviai grįžta prie surasto optimalaus taško. Labai tikėtina, kad šis taškas ir bus tikrasis globalus optimumas.



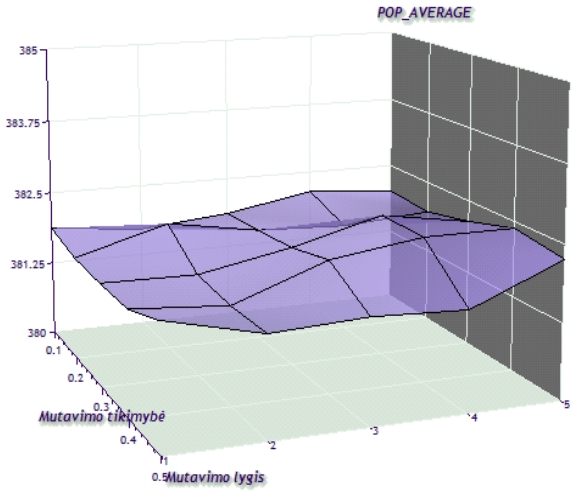
Pav. 3.7 Imigrantų metodo pritaikymas tiriant $\mu + \lambda$ konvergavimą

Atlikę aplinkos parametrų tyrimus nustatėme, kad geriausias genetinės atrankos metodas yra $\mu + \lambda$ atrankos metodas, kai $\lambda = 4\mu$. Laikysime, kad optimalus populiacijos dydis yra 200 individų, o pakankamas generacijų skaičius yra 40. Įdomus faktorius yra tas, kad generacijos skaičių pratęsus iki 300, esant parametrui $\lambda = 4\mu$ gauname, kad $\eta_{300} = 41/50$ – sprendinio kokybės garantija beveik nepakito.

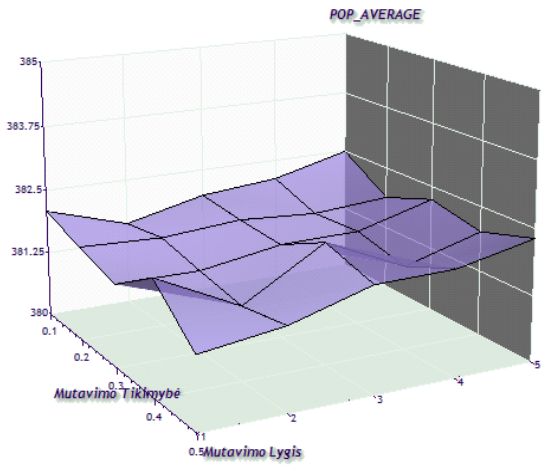
Rekombinacijos parametrų tyrimas

Sekančiame etape bandysime surasti optimalias rekombinacijos parametrų (kryžminimo tikimybės ρ_C, mutavimo tikimybės ρ_M ir mutavimo lygio r) reikšmes bei atrinkti tinkamiausią mutavimo metodą (inversinį arba sukeitimo).

Šio tyrimo metu naudosime rastas optimalias aplinkos parametrų reikšmes. Keisdami kryžminimo tikimybę ($\rho_C \in \{0.3 - 0.7\}$), mutavimo tikimybę ($\rho_M \in \{0.1 - 0.5\}$), bei mutavimo lygio reikšmes ($r \in \{1 - 5\}$) tirsime sprendinio kokybės garantijas. Sistemos darbo laikas šiuo atveju nėra svarbus, kadangi rekombinacijos operatorių grupė šį atsitiktinį dydį įtakoja nestipriai – didžiausią įtaką sistemos darbo laikui daro genetinės atrankos operatorius bei kiti aplinkos parametrai.



Pav. 3.8 Rekombinacijos parametrų tyrimas $\rho_C = 0.3$



Pav. 3.9 Rekombinacijos parametrų tyrimas $\rho_C = 0.7$

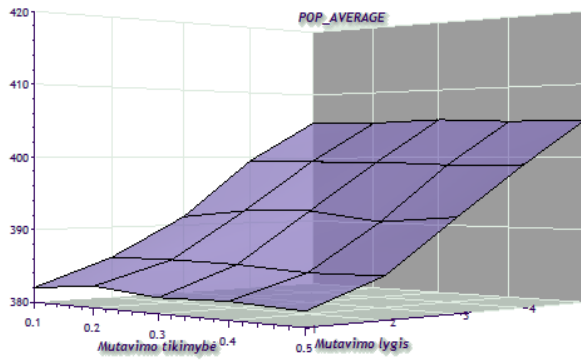
Kaip matome iš 3.8 ir 3.9 paveikslėlių, keičiant rekombinacijos parametrus, bendra sprendinio kokybės garantija beveik nesikeičia. Iš 5 lentelės pastebime, kad parametrų ρ_C, ρ_M ir r dalinės koreliacijos su η yra tokios mažos, jog yra tikėtina, kad šių koreliacijų kilmė yra imties atsitiktinumas

r	ρ_M	ρ_C	η^*
5	0.2	0.5	31
3	0.5	0.6	32
1	0.4	0.7	33
…	…	…	…
1	0.4	0.6	43
1	0.5	0.7	43
3	0.5	0.4	44

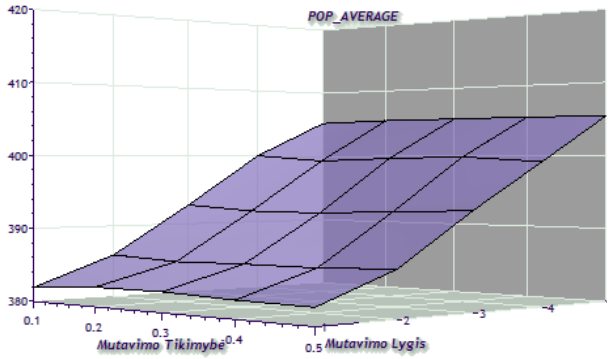
Lentelė 5. Rekombinacijos parametrų tyrimas : inversiškas mutavimas $-\eta$

Šiuo atveju parametrus palyginti galima tik pagal kokybės garantiją η^*, kuri parodo kiek kartų iš 50 bandymų, sprendinys sukonvergavo į žinomą minimumą. Šiuo atveju matome, kad geriausia parametrų kombinacija buvo $r = 3, \rho_M = 0.5\,, \rho_C = 0.4$. Mažiausias stebimas sprendinys užfiksuotas $44/50$ kartų. Deja yra tikimybė, kad pakartojus eksperimentą dar 50 kartų rezultatai gali pasikeisti. Išsamesni tyrimo rezultatai pateikiami 1 priede.

Kitame etape nagrinėkime sukeitimo mutaciją

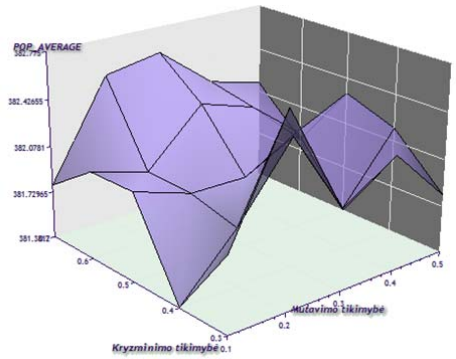


Pav. 3.10 Rekombinacijos parametrų tyrimas $\rho_C = 0.3$



Pav. 3.11 Rekombinacijos parametrų tyrimas $\rho_C = 0.7$

Kaip matome iš 3.10 ir 3.11 paveikslėlių, šiuo atveju egzistuoja ryški tiesinė priklausomybė tarp mutavimo lygmens ir sprendinio kokybės garantijos. Dėl to galime teigti, kad optimalus mutacijos lygis yra $r = 1$. Panagrinėkime mutacijos ir kryžminimo tikimybių įtaką sprendinio kokybei, kai turimas vienetinis mutacijos lygis.



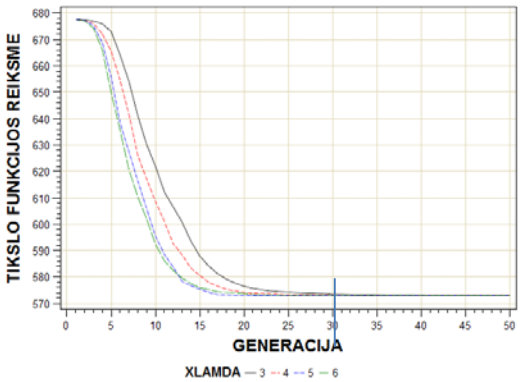
Pav. 3.12 Rekombinacijos parametrų tyrimas : paviršius $\rho_M \times \rho_C = pop_{average}$

Kaip matome iš 3.12 grafiko, ryškių parametrų įtakos tikslo funkcijai tendencijų įžvelgti negalime. Labai tikėtina, kad rasta optimali parametro reikšmė $(\rho_C = 0.4, \rho_M = 0.1)$ yra priklausoma nuo pradinių duomenų matricos, Tačiau pakartojus eksperimentą 50 kartų nusistovėjimas gaunamas būtent šiame taške, kurį ir laikysime optimaliu parametrų rinkiniu $L = 100$ genų ilgio chromosomai.

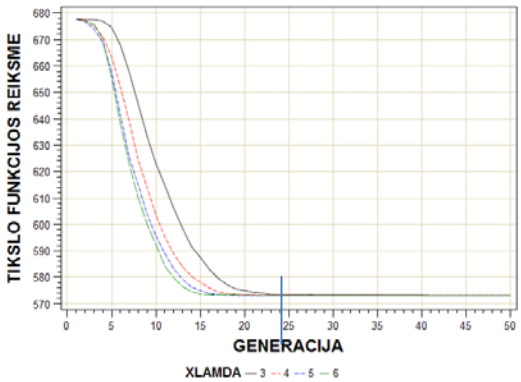
Sekančiame tyrimo etape nagrinėjome, kaip keičiasi genetinių operatorių parametrai kintant pradinės duomenų matricos dydžiui. Šiuo atveju buvo pasirinkti du papildomi chromosomų ilgiai $L_1 = 200$ genų ir $L_2 = 50$ genų. Tiriant aplinkos parametrus buvo nagrinėjamas tik $\mu + \lambda$ atrankos metodas. Tyrimo metu buvo naudojami šie parametrai :

- 200 generacijų ciklas;
- populiacijos dydžiai $pop_{size} \in \{200,300,400,500\}$;
- Atrankos metodo parametrai $\lambda \in \{3\mu, 4\mu, 5\mu, 6\mu, 7\mu\}$;
- Rekombinacijos parametrai $\rho_C = 0.7$, $\rho_M = 0.1$, $r = 1$. Rekombinacijai buvo naudojama sukeitimo mutacija;

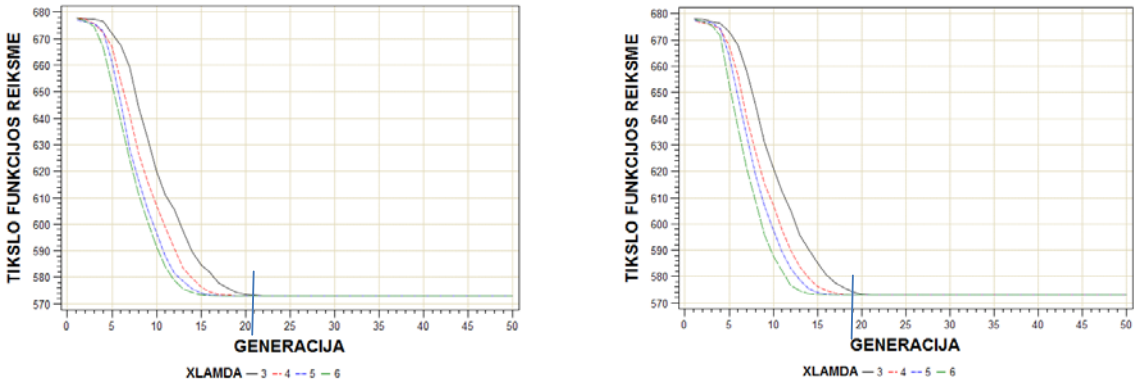
Pirmuoju atveju tyrėme $L = 200$ genų chromosomą.



$pop_{size} = 200$



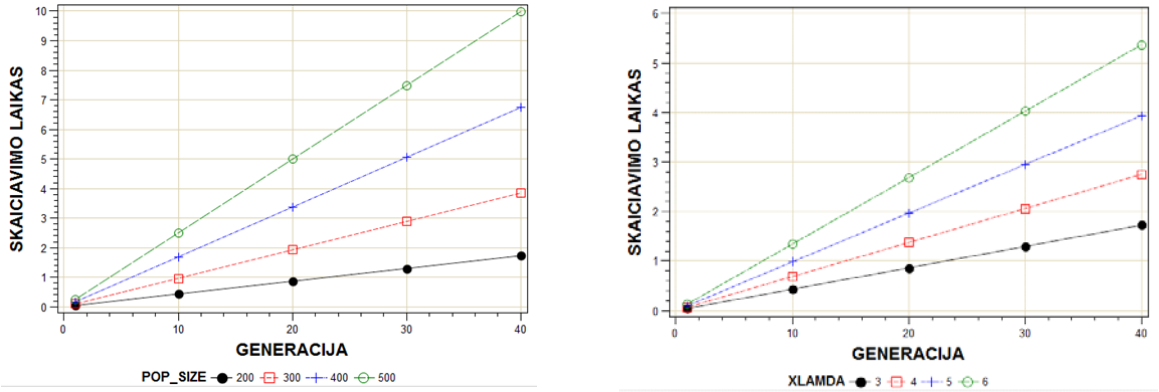
$pop_{size} = 300$



$pop_{size} = 400$ $pop_{size} = 500$

Pav. 3.13 Aplinkos parametrų tyrimas : chromosomos ilgis $L = 200$

Iš 3.13 pav. matome, kad didinant populiacijos dydį, konvergavimo greitis generacijų atžvilgiu didėja. Esant $pop_{size} = 200$ chromosomų populiacijos dydžiui reikia atlikti 30 generacijų, kol algoritmas nusistovi minimume, kai tuo tarpu esant $pop_{size} = 500$ chromosomų dydžiui, tam reikia tik 20 generacijų. Tačiau 10 generacijų sutaupymas šiuo atveju sistemos darbo laiko resursus padidina 3.8 kartus (nuo 1.3 sekundžių esant $pop_{size} = 200$ atveju iki 4.9 sekundžių esant $pop_{size} = 500$). Taigi šiuo atveju pasirinksime $pop_{size} = 200$ populiacijos dydį.



$\lambda = 3\mu$ $pop_{size} = 200$

Pav. 3.14 Algoritmo darbo laikų palyginimas

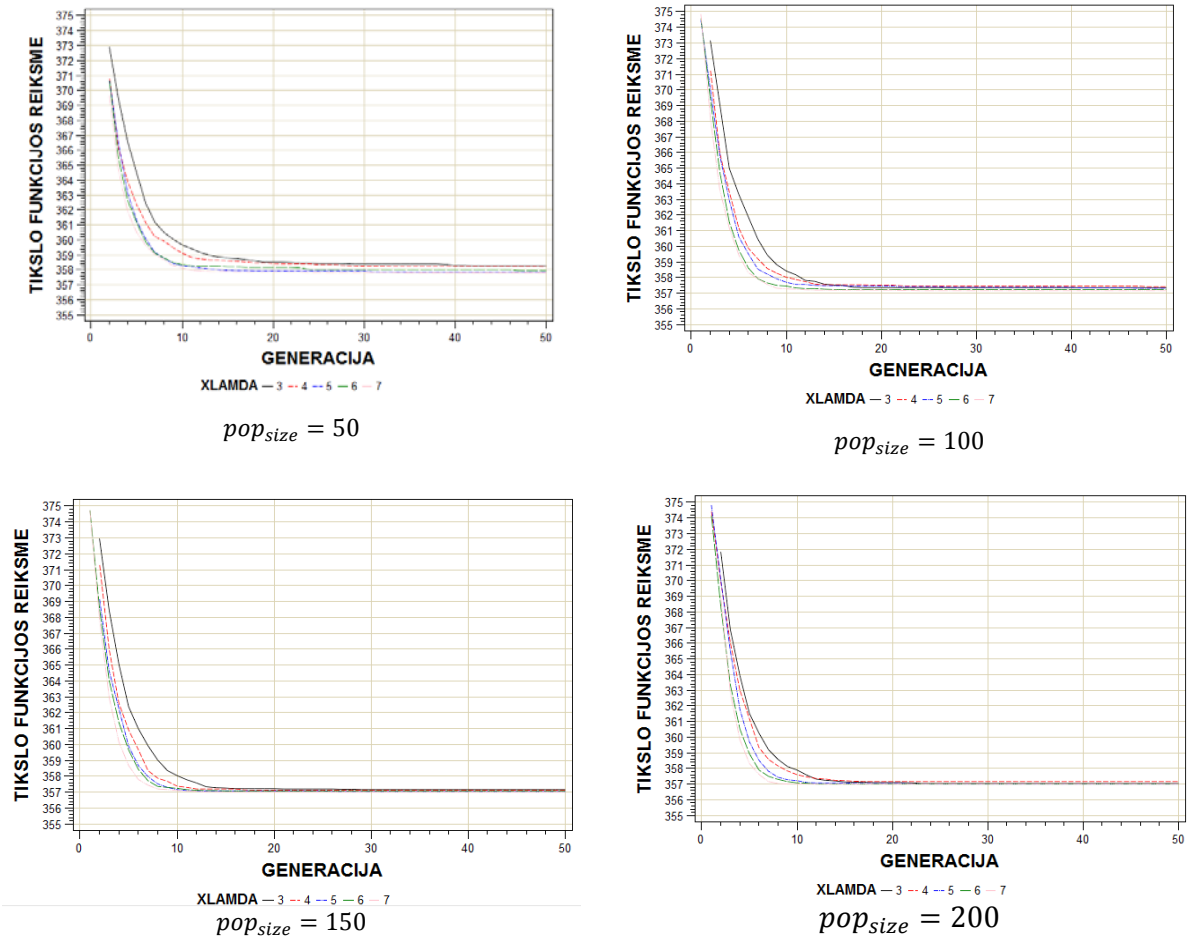
Mažiausiai laiko resursų reikia esant parametrų rinkiniui $pop_{size} = 200$ ir $\lambda = 3\mu$ (3.15 pav). Kokybės garantija šiuo atveju yra labai artima vienetui $\eta = \frac{48}{50} = 0.96$ (6 lentelė).

Pop_{size}	$\lambda = k \cdot \mu$	η^*
200	3	48
200	4	49
200	6	49
300	3	49
300	4	49
200	5	50
300	5	50
300	6	50

Pop_{size}	$\lambda = k \cdot \mu$	η^*
400	3	50
400	4	50
400	5	50
400	6	50
500	3	50
500	4	50
500	5	50
500	6	50

Lentelė 6 Kokybės garantijos įverčiai kiekvienam parametrų rinkiniui (atlikus 40 generacijų)

Šiuo atveju rekombinacijos parametrus tirti yra netikslinga, nes jau esamas parametrų rinkinys duoda labai gerus rezultatus ir parametrų įverčių gerinimas algoritmo efektyvumo beveik nepakeistų. Tačiau dėl sistemos bendrumo ištyrėme parametrų rinkinį $r = 1$, $\rho_C = 0.4$ ir $\rho_M = 0.1$ bei gavome, $\eta = \frac{49}{50} = 0.98$. Sprendinio kokybės garantija padidėjo, todėl ateityje taikysime šį parametrų rinkinį.



$pop_{size} = 50$ $pop_{size} = 100$

$pop_{size} = 150$ $pop_{size} = 200$

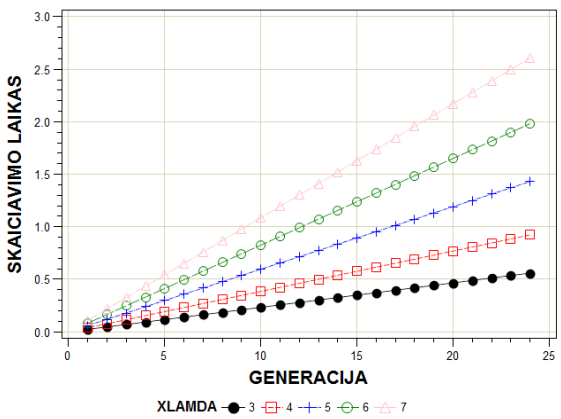
Pav. 3.15 Aplinkos parametrų tyrimas : chromosomos ilgis $L = 50$

Antruoju atveju tirsime chromosomos ilgį $L = 50$ genų. Kaip matome iš 3.15 pav. visos iki šiol gautos tendencijos yra patvirtinamos, t.y. Didinant populiacijos skaičių ir parametrą λ konvergavimas generacijų atžvilgiu vyksta greičiau, tačiau laiko atžvilgiu, reikia daugiau sistemos resursų. Lentelėje nr. 6 matome, kad geriausi rezultatai (sprendinio kokybės garantijos atžvilgiu) pasiekiami esant 150 arba 200 chromosomų populiacijos dydžiams. Esant šių dydžių populiacijoms algoritmas minimume nusistovi apytiksliai po 20 generacijų.

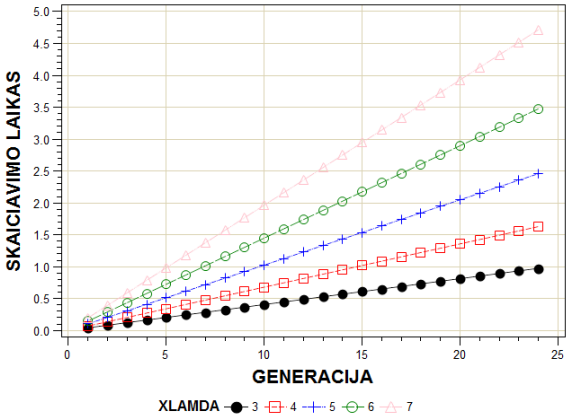
Pop_{size}	$\lambda = k \cdot \mu$	η^*
50	3	23
50	4	24
50	6	27
50	5	31
50	7	32
100	4	38
100	3	39
100	5	41
100	6	44
100	7	45

Pop_{size}	$\lambda = k \cdot \mu$	η^*
150	3	45
200	4	46
150	4	47
150	6	48
150	5	48
200	3	48
150	7	49
200	5	50
200	6	50
200	7	50

Lentelė 7 Kokybės garantijos įverčiai kiekvienam parametrų rinkiniui (atlikus 20 generacijų)



$pop_{size} = 150$



$pop_{size} = 200$

Pav. 3.16 Algoritmo darbo laikų palyginimas

Parinkus rekombinacijos parametrus $r = 1$, $\rho_C = 0.4$ ir $\rho_M = 0.1$, sprendinio kokybės garantija išauga iki $\eta = \frac{50}{50} = 1$. Lentelėje nr. 7 pateikiama šio tyrimo rezultatų suvestinė.

Bendra gautų rezultatų suvestinė pateikiama 8 lentelėje. Atlikus tyrimą, paaiškėjo, jog genetinio algoritmo parametrai labai nedaug priklauso nuo tirtų chromosomų ilgių – pradinių duomenų matricų dydžio. Yra nemaža tikimybė, kad ryškesni parametrų pokyčiai gali pasireikšti stebint didesnių ilgių chromosomas (tvarkaraščiuose atsiranda daugiau darbų ir išilgėja jų vykdymo laikas), tačiau kaip jau minėjome anksčiau, dėl gamybinio plano patikimumo, nerekomenduojama tvarkaraščius sudaryti ilgiem periodam.

Chromosomos ilgis	**50**	**100**	**200**
Populiacijos dydis	200	200	200
Generacijų skaičius	20	40	40
Genetines atrankos operatorius	$\mu + \lambda$		
Genetines atrankos operatoriaus parametras	$\lambda = 3\mu$	$\lambda = 4\mu$	$\lambda = 3\mu$
Kryžminimo tikimybė	0.4		
Mutavimo metodas	sukeičiantis mutavimas		
Mutavimo lygis	1		
Mutavimo tikimybė	0.1		

Lentelė 8 Genetinio algoritmo parametrų apibendrinimas

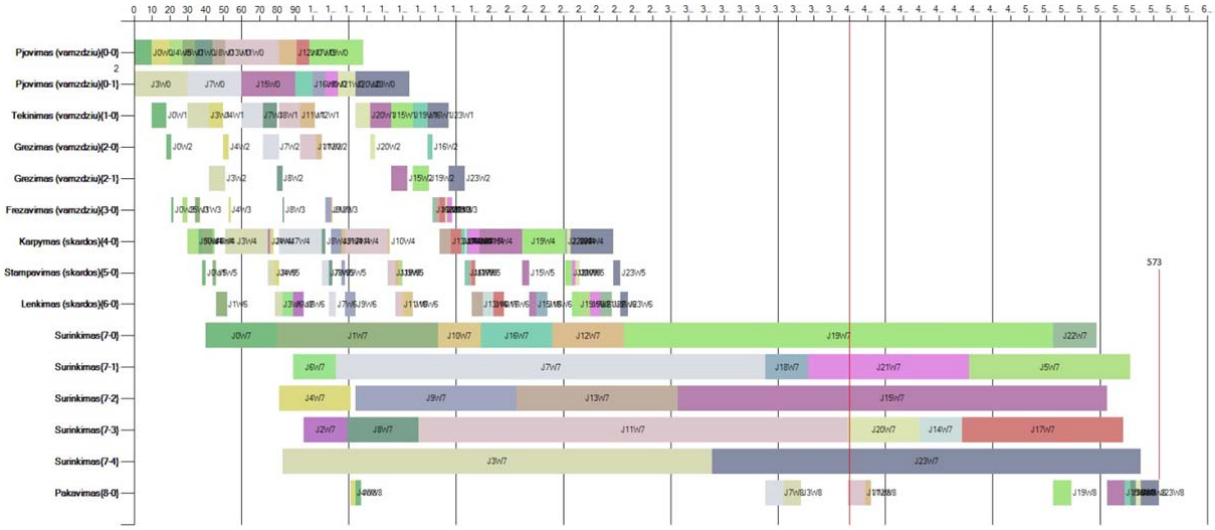
3.3. Genetinio algoritmo ir TOC euristikos palyginimas

Šiame skyrelyje palyginsime rezultatus, gautus naudojant genetinį algoritmą su optimaliu parametrų rinkiniu bei TOC euristikos metodu gautą sprendinį. Tyrimui naudosime 3 ilgių chromosomas su jau tirtais duomenimis (8 lentelė).

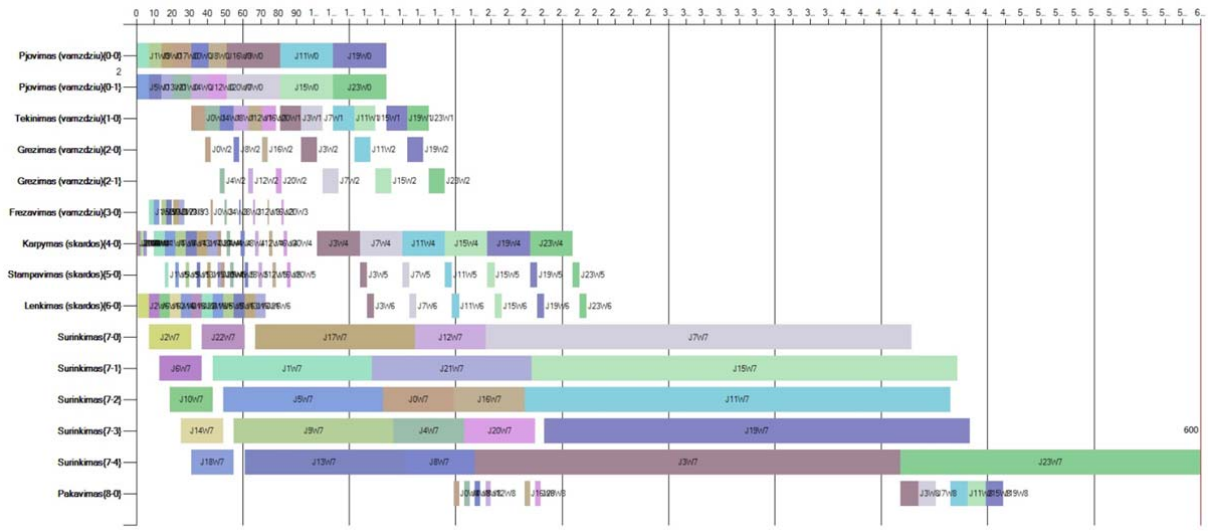
Chromosomos ilgis	GA sprendinys	TOC sprendinys
50	357	374
100	381	405
200	573	600

Lentelė 9 GA ir TOC algoritmų palyginimas

Pastebėkime, kad visais atvejais geresni sprendiniai gaunami naudojant genetinį algoritmą (9 lentelė). Panagrinėkime genetinio algoritmo pranašumo prieš TOC euristiką priežastis.



Pav. 3.17 Genetinio algoritmo sprendinys, kai pateikta $L = 200$ genų ilgio chromosoma



Pav. 3.18 TOC euristikos algoritmo sprendinys, kai pateikta $L = 200$ genų ilgio chromosoma

Nagrinėjant duomenis nustatėme, kad apribojimas visą laiką buvo surinkimo darbo centras su 5 surinkimo linijom (įrengimais) (3 lentelė). Genetinio algoritmo pranašumą šiuo atveju lėmė gebėjimas niveliuoti darbo laikus - kaip matome iš 3.17 pav., naudojant genetinį algoritmą, didelė dalis trumpiausių darbų buvo atliekami gamybos proceso viduryje, dėl to buvo galima lengvai išlyginti darbo centro įrengimų apkrautumą. Be to, priešingai nei 3.18 paveiksle, čia visi darbo centrai užduotis baigia vykdyti beveik panašiu metu. Deja, bet ši taisyklė galioja tik tiriamiems duomenims, kurie pasižymi tuo, kad apribojimas yra priešpaskutinis darbo centras, be to paskutiniame darbo centre – „pakavimas“ visos užduotys yra labai trumpos ir pabaigiamos panašiu metu kaip ir paskutinė užduotis, išeinanti iš apribojimo darbo centro. Situacijai, kurioje ne visi gaminiai yra pakuojami, tačiau kai kurie yra pakuojami labai ilgai (pvz. jiem gaminamos specialios pakuotės, gaminiai ruošiami eksportui į ne ES šalis ir t.t.), šis dėsningumas netiks. Šiuo atveju suradus geriausią sprendinį greičiausiai gausime, kad kažkurie apribojimo įrengimai bus kur kas labiau apkrauti, nei tie, kurie gamina sudėtingo pakavimo gaminius. Taigi apibendrinus, galime teigti, kad universalių taisyklių, kuriomis vadovaujantis visais atvejais būtų sudaromas optimalus tvarkaraštis, surašyti negalime. Dėl to taikome genetinį algoritmą, kuris tiria įvairius tvarkaraščius su įvairiai pasiskirsčiusiais darbo laikais.

IŠVADOS

Šios išvados gali būti taikomos dirbant su lankstaus srauto gamybos modeliais, kuomet gaminių partijos ir užduočių trukmės nėra didelės ($t < 8\,val$).Išvados tinkamos sudarant tik statinius tvarkaraščius, kuriuose pagalbiniai laikai (gaminių transportavimas, įrengimų derinimas) yra pastovūs ir labai trumpi ($t < 20\text{min}$).

1. Tyrimo metu nustatyta kad $\mu + \lambda$ genetinės atrankos metodas yra efektyviausias. Jo pagalba buvo gaunami geriausi sprendiniai tikslo funkcijos atžvilgiu. Be to šių sprendinių kokybės garantija η svyravo nuo 0.8 iki 1. Žinoma, šio metodo integravimas į genetinį algoritmą labai smarkiai padidina pačio algoritmo intensifikaciją – paiešką kiekvieno sprendinio aplinkoje, dėl to vyksta pakankamai greitas konvergavimas, kurio eigoje visos populiacijos sprendinių dispersija artėja į nulį. Šis faktas padidina tikimybę, kad surastas sprendinys tėra tik tam tikros aplinkos lokalus optimumas. Deja, praktinių metodų šiam faktui patvirtinti nėra. Tačiau apibendrinus tyrimą, galime teigti, kad šio metodo pagalba buvo gaunamas geriausias sprendinys lyginant su kitais genetinės atrankos metodais.

2. Ryškaus skirtumo tarp sukeitimo mutacijos ir inversijos mutacijos pastebėti nepavyko. Tiek su vienu, tiek su kitu metodu sudaryti sprendiniai, tikslo funkcijos atžvilgiu buvo pakankamai panašūs. Taip pat pastebėta tendencija, kad inversijos mutavimas puikiai dirba su įvairiais mutavimo lygiais, kai tuo tarpu sukeitimo mutacija geriausiai dirba tik esant vienetiniam mutavimo lygiui.

3. Buvo pastebėta, kad didinant populiacijos dydį ir atrankos parametrą λ spartėja konvergavimas (generacijų atžvilgiu), tačiau eksponentiškai išauga laiko resursų poreikis. Kadangi šiuo atveju didžiausią įtaką skaičiavimo laikui daro populiacijos dydis bei parametras λ, to dėl atrenkant optimalius parametrus buvo vertinama ir sprendinio kokybės garantija ir algoritmo darbo laikas. Iš kitos pusės, per daug greitas konvergavimas (ypač esant didesnės dimensijos duomenų matricai) smarkiai padidina tikimybę, kad gautas sprendinys bus tik tam tikras lokalus optimumas. Apibendrinus tyrimo rezultatus sudarėme aplinkos parametrų įverčių lentelę (8 lentelė).

4. Tiriant rekombinacijos parametrus buvo pastebėta, kad šių parametrų rinkinys yra labiau priklausomas nuo pačių duomenų specifikos. Jokio parametrų kombinacijų dėsningumo įžvelgti nepavyko (39 pav.). Dėl to buvo pasirinktas parametrų rinkinys, su kuriuo buvo gaunama didžiausia sprendinio kokybės garantija.

5. Palyginus tvarkaraščius, sudarytus su genetiniu algoritmu bei TOC euristika, gavome, kad visais atvejais geresni rezultatai pasiekiami naudojant genetinį algoritmą. Tiriamiems duomenims genetinio algoritmo pranašumas pasireiškė gebėjimu geriau skirstyti darbus pagal jų trukmes gamybinio proceso eigoje. Tačiau tai būdinga tik tirtiem duomenim.

Rekomendacijos

Sukurta programinė įrangą, ją patobulinus, bus naudojama sudarinėjant nagrinėtos įmonės gamybinius tvarkaraščius. Pagrindinis dabartinės programos trūkumas yra rezultatų pateikimas. Kaip jau minėjome anksčiau, Ganto diagrama yra labiau planuotojo įrankis, tačiau užduočių pateikimui ir kontrolei gamyboje ji netinka. Dėl to planuojama šią programinę įrangą taikyti kaip rekomendacinio pobūdžio tvarkaraščių sudarymo sistemą, kurios sprendiniai būtų atvaizduojami ne tik Ganto diagramoje, bet ir kiekių valdymo lentelėje (2 pav.).

Sudaryta sistema ir atrinkti metodai su parametrais tinkami naudoti srautinio tipo gamybos modeliuose. Sistemoje nevertinami įvairūs atsitiktiniai įvykiai, kurie gali įtakoti tvarkaraščio vykdymą, dėl to rekomenduojama tvarkaraštį sudaryti ne ilgesniam negu vienos dienos laikotarpiui. Tai padidina ne tik tvarkaraščio tikslumą, bet kartu ir įvykdymo tikimybę.

Literatūra

1. Goncalves, J.F., Mendes, J.J.M, Resende, M.G.C., : A Hybrid Genetic Algorithm For Job Shop Scheduling Problem. *AT&T Labs Research Technical Report TD-5EAL6J, 2002*.
2. Karmakar, S., Mahanty, B. : Minimizing Makespan For Flexible Flow Shop Scheduling Problem In A Paint Company. *Department of Industrial Engeneering and Management. Indian Institute of Technology*. Prieiga per internetą : http://www.academia.edu/940409/Minimizing_Makespan_for_a_Flexible_Flow_Shop_Scheduling_Problem_in_a_Paint_Company [žiūrėta 2013.06.07].
3. Wirth, N. : ALGORITHMS + DATA STRUCTURES =PROGRAMS, *Prentice-Hall, INC. 1976.*
4. Peter J.B. Hancock, An Empirical Comparison Of Selection Methods In Evolutionary Algorithms. *Departament of Psychology, University of Stirling. Scotland. FK9 4LA, 1994* Prieiga per intenretą : http://aimm02.cse.ttu.edu.tw/class_2006_1/op/References/An%20empirical%20comparison%20of%20selection%20methods%20in%20evolutionary%20algorithms.pdf [žiūrėtą 2013.06.07].
5. Hans-Georg Beyer, Hans-Paul Schwefel, Evolution Strategies, *Departament of computer science XI, University of Dortmund, Germany, Kluwer Academic Publishers 2002* Prieiga per internetą : http://www.cs.bham.ac.uk/~pxt/NIL/es.pdf [žiūrėta 2013.06.07].
6. G.Felinskas, Euristinių metodų tyrimas ir taikymas ribotų išteklių tvarkaraščiams optimizuoti, *Daktaro disertacija, VDU, Vilnius 2007*. Prieiga per internetą : http://www.mii.lt/files/mii_dis_07_felinskas.pdf [žiūrėta 2013.06.07].
7. N.M. Razali, J.Geraghty. Genetic Algorythm Performance with Different Selection Strategies in Solving TSP. *Proceedings of the World Congress on Engeneering 2011, Vol II Prieiga per internetą :* http://www.iaeng.org/publication/WCE2011/WCE2011_pp1134-1139.pdf [žiūrėta 2013.06.07].
8. M.Affenzeller, S.Wagner. A self-Adaptive Model for Selective Pressure Handling with Theory of Genetic Algorithms. *Institute of Systems Science Systems Theory and Information Technology, Hohannes Kepler University. Lecture notes in computer Science Vol 2809, 2003*. Prieiga per internetą : http://link.springer.com/chapter/10.1007%2F978-3-540-45210-2_35# [žiūrėta 2013.06.07].
9. Mitsuo Gen, Runwei Cheng, Genetic Algorithms & Engeneering Optimisation. *John Wiley & Sons, Inc 2000.*
10. S.Sarmady. An Investigation on Genetic Algorithm Parameters. *School of Computer Sciences, University of Sains Malaysia.* Prieiga per internetą :

http://sarmady.com/siamak/papers/genetic-algorithm.pdf [žiūrėta 2013.06.07].

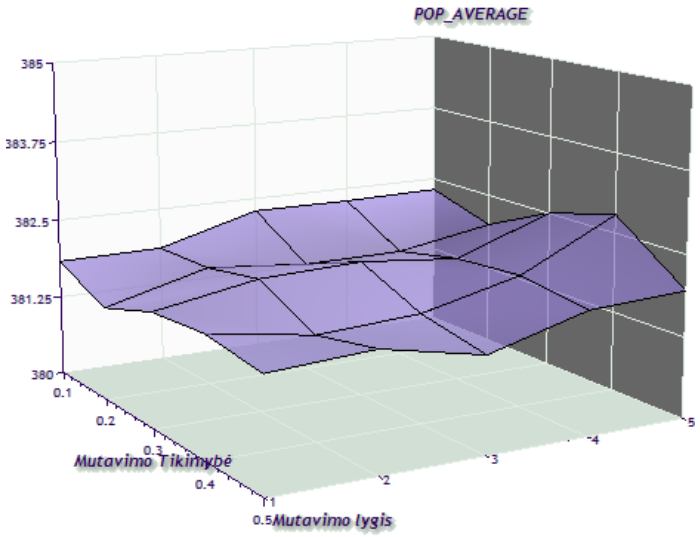
11. Ono, I., Yamamura, M., Kobayashi, S. : A Genetic Algorithm for Job-Shop Scheduling Problems Using Job-based Order Crossover. *Proc of ICEC'96, 1996.* Prieiga per internetą : http://members.multimania.co.uk/ozgurkelemci/30_nisan_indirilenler/icec96.pdf [žiūrėta 2013.06.07].
12. Omar, M., Baharum, A., Abu Hasan, Y. : A Job-Shop Scheduling Problem (JSSP) Using Genetic Algorithm (GA). *School of Mathematical Sciences, Univeristy of Sains Malaysia. Proceedings of 2nd IMT-GT Regional Conference on Mathematics, Statistics and Applications. 2006*. Prieiga per internetą: http://math.usm.my/research/onlineproc/CS13.pdf [žiūrėta 2013.06.07].
13. Šeda, M. : Mathematical Models of Flow Shop and Job Shop Scheduling Problems. *International Journal of Applied Mathematics and Computer Sciences Volume 4, Number 4* Prieiga per internetą : http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.118.1024 [žiūrėta 2013.06.07].
14. Ayad, A.R., Awad, H.A., Yassin, A.A. :Parametric analysis for genetic algorithms handling parameters. *Alexandria University, Alexandria Engeneering Journal, 2013*. Prieiga per internetą : http://www.sciencedirect.com/science/article/pii/S111001681200097X [žiūrėta 2013.06.07].
15. Blickle, T., Thiele, L. : A Comparison of Selection Schemes used in Genetic Algorithms, *Swiss Federal Institute of Technology (ETH), TIK-Report Nr. 11, 2nd Edition, 1995, Switzerland*. Prieiga per internet : http://www4.hcmut.edu.vn/~hthoang/dktm/Selection.pdf [žiūrėta 2013.06.07].
16. Gen, M., Cheng, R., Lin, L. : Network Models and Optimisation : Multiopbejctive Genetic Algorithm Approach.
17. Woeppel Mark J. Gamybininko vadovas: Kaip Įgyvendini Apribojimų Valdymo pokyčius ?, *Lietuvių kalba, RGrupė, 2007*
18. Takeda H. Sinchroninė Gamyba „Pačiu laiku" – būdas visai įmonei. *Rgrupė, 2005*
19. Blum, C., Roli, A. : Methaeuristics in Combinatorial Optimisation : Overview and Conceptual Comparison. *ACM Computing Surveys (CSUR) journal, Vol35, Issue 3, 2003, New York.*
20. Wolpert, David H., and William G. MACREADY, 1997. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, **1**(1), 67–82.

Priedas Nr1. Rekombinacijos parametrų įtakos rezultatai (inversinė mutacija)

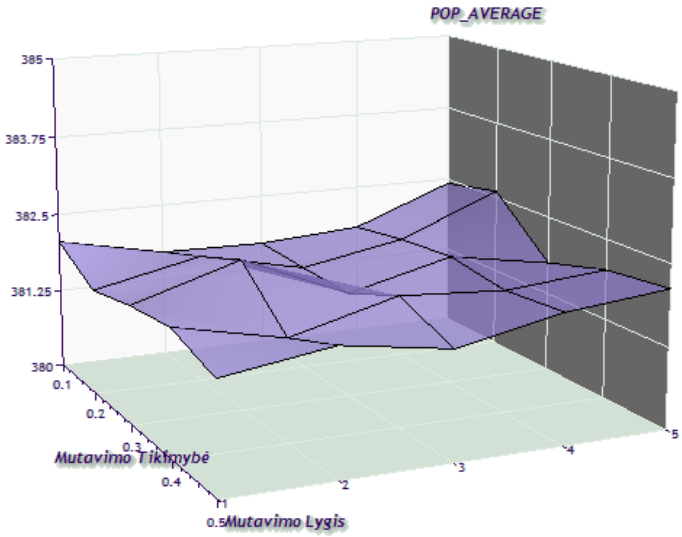
r	ρ_M	ρ_C	η^*
5	0.2	0.5	31
3	0.5	0.6	32
1	0.4	0.7	33
3	0.1	0.4	33
4	0.4	0.3	33
5	0.4	0.4	33
1	0.5	0.6	34
2	0.2	0.3	34
2	0.3	0.4	34
4	0.3	0.3	34
1	0.1	0.5	35
1	0.1	0.7	35
2	0.3	0.5	35
2	0.5	0.5	35
3	0.1	0.7	35
3	0.3	0.3	35
3	0.4	0.7	35
4	0.1	0.7	35
4	0.5	0.6	35
5	0.1	0.7	35
5	0.4	0.3	35
1	0.5	0.3	36
3	0.4	0.3	36
3	0.4	0.5	36
4	0.1	0.4	36
4	0.2	0.6	36
4	0.4	0.5	36
4	0.5	0.7	36
5	0.3	0.7	36
5	0.5	0.3	36
1	0.1	0.4	37
1	0.2	0.3	37
1	0.4	0.4	37
1	0.4	0.5	37
1	0.5	0.4	37
2	0.2	0.5	37
2	0.2	0.6	37
3	0.3	0.4	37
3	0.4	0.6	37
4	0.3	0.5	37
4	0.4	0.4	37
4	0.4	0.6	37

r	ρ_M	ρ_C	η^*
4	0.5	0.5	37
5	0.1	0.3	37
5	0.1	0.4	37
5	0.1	0.5	37
5	0.2	0.6	37
5	0.3	0.4	37
5	0.4	0.5	37
5	0.4	0.6	37
5	0.5	0.7	37
1	0.1	0.6	38
1	0.2	0.6	38
1	0.2	0.7	38
1	0.3	0.4	38
2	0.1	0.5	38
2	0.1	0.6	38
2	0.3	0.3	38
2	0.3	0.6	38
2	0.5	0.3	38
2	0.5	0.4	38
3	0.2	0.3	38
3	0.2	0.7	38
3	0.3	0.6	38
3	0.5	0.7	38
4	0.1	0.3	38
4	0.1	0.6	38
4	0.3	0.4	38
4	0.3	0.6	38
5	0.3	0.3	38
5	0.3	0.6	38
5	0.5	0.4	38
5	0.5	0.6	38
1	0.1	0.3	39
1	0.3	0.3	39
1	0.3	0.5	39
1	0.5	0.5	39
2	0.1	0.4	39
2	0.2	0.4	39
2	0.2	0.7	39
2	0.4	0.5	39
2	0.4	0.6	39
2	0.5	0.6	39
3	0.1	0.3	39

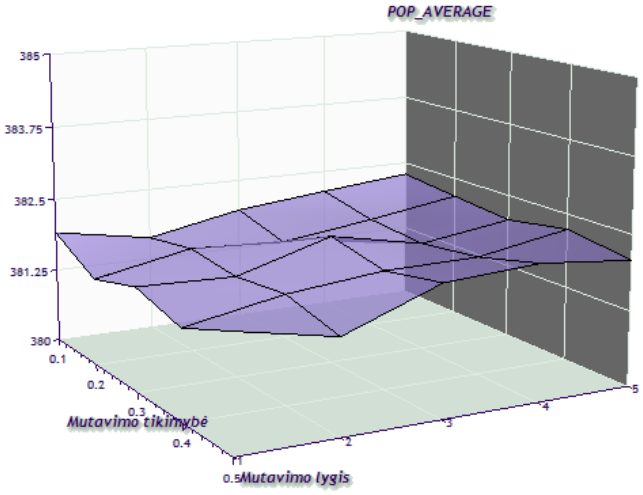
r	ρ_M	ρ_C	η^*
3	0.1	0.5	39
3	0.2	0.4	39
3	0.5	0.5	39
4	0.2	0.5	39
4	0.5	0.4	39
5	0.1	0.6	39
5	0.2	0.3	39
5	0.2	0.4	39
5	0.2	0.7	39
5	0.5	0.5	39
1	0.2	0.5	40
1	0.3	0.6	40
1	0.4	0.3	40
2	0.1	0.3	40
2	0.1	0.7	40
2	0.4	0.3	40
2	0.4	0.4	40
3	0.1	0.6	40
3	0.2	0.5	40
3	0.5	0.3	40
4	0.2	0.4	40
4	0.5	0.3	40
5	0.4	0.7	40
1	0.3	0.7	41
3	0.3	0.7	41
4	0.1	0.5	41
4	0.2	0.3	41
4	0.2	0.7	41
4	0.3	0.7	41
4	0.4	0.7	41
5	0.3	0.5	41
2	0.3	0.7	42
2	0.4	0.7	42
2	0.5	0.7	42
3	0.2	0.6	42
3	0.3	0.5	42
3	0.4	0.4	42
1	0.2	0.4	43
1	0.4	0.6	43
1	0.5	0.7	43
3	0.5	0.4	44



Kryžminimo tikimybė PCX=0.4



Kryžminimo tikimybė PCX=0.5



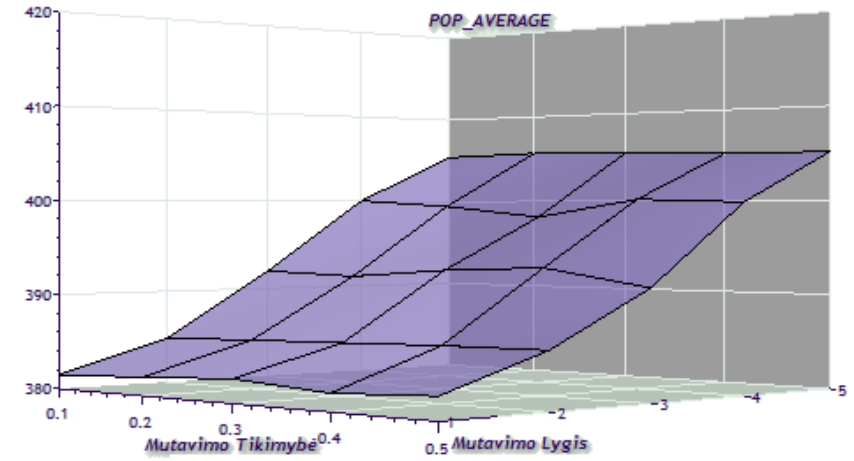
Kryžminimo tikimybė PCX=0.6

Priedas Nr2. Rekombinacijos parametrų įtakos rezultatai (sukeitimo mutacija)

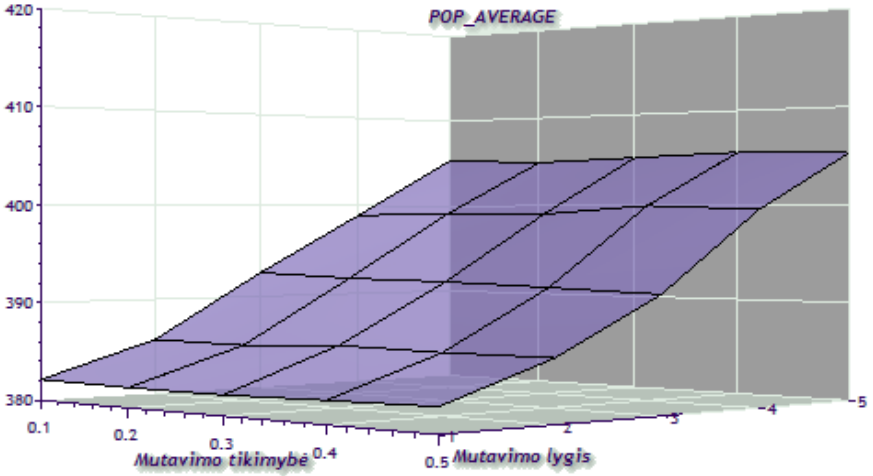
r	ρ_M	ρ_C	η^*
4	0.1	0.7	1
4	0.3	0.7	1
4	0.4	0.4	1
4	0.4	0.5	1
4	0.4	0.7	1
4	0.5	0.5	1
5	0.1	0.5	1
5	0.2	0.3	1
5	0.2	0.5	1
5	0.3	0.6	1
5	0.4	0.3	1
5	0.4	0.4	1
5	0.4	0.6	1
5	0.5	0.5	1
4	0.1	0.5	2
4	0.1	0.6	2
4	0.2	0.3	2
4	0.2	0.5	2
4	0.2	0.7	2
4	0.3	0.3	2
4	0.3	0.4	2
4	0.4	0.3	2
4	0.4	0.6	2
4	0.5	0.3	2
4	0.5	0.6	2
4	0.5	0.7	2
4	0.2	0.6	3
3	0.1	0.7	4
4	0.1	0.3	4
4	0.3	0.6	4
3	0.2	0.6	5
3	0.4	0.7	5
3	0.5	0.7	5
3	0.3	0.4	6
3	0.4	0.6	6

r	ρ_M	ρ_C	η^*
3	0.1	0.4	7
3	0.1	0.5	8
3	0.1	0.6	8
3	0.2	0.5	8
3	0.4	0.4	8
3	0.5	0.6	8
3	0.5	0.3	9
3	0.2	0.7	10
3	0.3	0.3	10
3	0.3	0.5	10
3	0.2	0.3	11
3	0.2	0.4	12
3	0.3	0.6	12
3	0.3	0.7	12
3	0.4	0.3	12
3	0.4	0.5	12
3	0.5	0.5	12
3	0.1	0.3	14
3	0.5	0.4	15
2	0.1	0.5	20
2	0.3	0.5	20
2	0.5	0.5	20
2	0.2	0.6	21
2	0.4	0.5	21
2	0.1	0.7	22
2	0.3	0.6	22
2	0.3	0.7	22
2	0.2	0.3	23
2	0.5	0.7	23
2	0.1	0.3	24
2	0.1	0.6	24
2	0.4	0.4	24
2	0.5	0.3	24
2	0.5	0.4	24
2	0.4	0.7	25

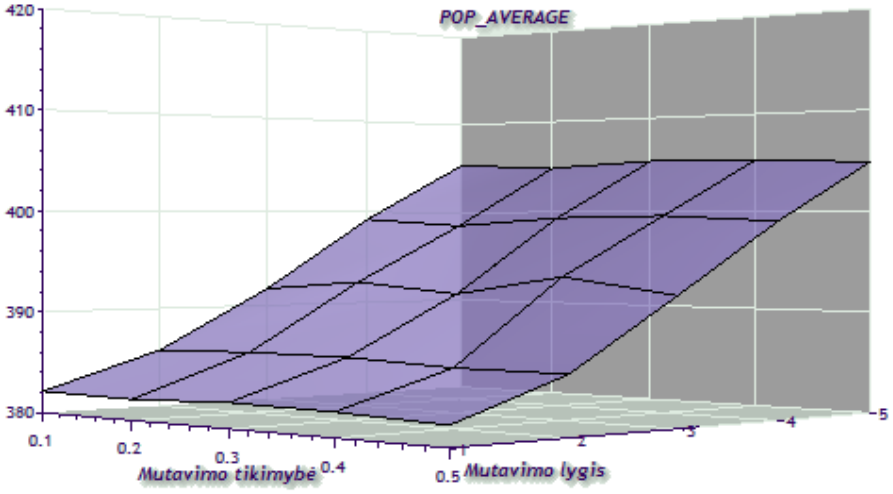
r	ρ_M	ρ_C	η^*
2	0.2	0.5	26
2	0.2	0.7	26
2	0.3	0.3	26
2	0.5	0.6	26
2	0.4	0.6	27
2	0.2	0.4	28
2	0.3	0.4	28
2	0.4	0.3	28
2	0.1	0.4	31
1	0.3	0.7	33
1	0.5	0.5	33
1	0.2	0.3	34
1	0.2	0.7	35
1	0.3	0.6	35
1	0.4	0.6	35
1	0.3	0.4	37
1	0.4	0.3	37
1	0.1	0.5	38
1	0.1	0.6	38
1	0.4	0.7	38
1	0.5	0.7	38
1	0.2	0.4	39
1	0.3	0.5	39
1	0.4	0.5	39
1	0.5	0.4	39
1	0.1	0.3	40
1	0.3	0.3	40
1	0.5	0.3	40
1	0.2	0.5	41
1	0.2	0.6	41
1	0.5	0.6	41
1	0.1	0.7	42
1	0.4	0.4	43
1	0.1	0.4	46



Kryžminimo tikimybė PCX=0.4



Kryžminimo tikimybė PCX=0.5



Kryžminimo tikimybė PCX=0.6

Priedas Nr3 Kompaktinis diskas