KAUNO TECHNOLOGIJOS UNIVERSITETAS
MATEMATIKOS IR GAMTOS MOKSLŲ FAKULTETAS

Karolis Dunderis

GENETINIO ALGORITMO, NAUDOJAMO GAMYBOS TVARKARAŠČIO SUDARYMUI, TYRIMAS

Baigiamasis magistro projektas

Vadovas
Doc. dr. Narimantas Listopadskis

KAUNAS, 2015

KAUNO TECHNOLOGIJOS UNIVERSITETAS
MATEMATIKOS IR GAMTOS MOKSLŲ FAKULTETAS

GENETINIO ALGORITMO, NAUDOJAMO GAMYBOS TVARKARAŠČIO SUDARYMUI, TYRIMAS

Baigiamasis magistro projektas
Taikomoji matematika (kodas 621G10003**)**

Vadovas
(parašas) Doc. dr. Narimantas Listopadskis
(data)

Recenzentas
(parašas)
(data)

Projektą atliko
(parašas) Karolis Dunderis
(data)

KAUNAS, 2015

KAUNO TECHNOLOGIJOS UNIVERSITETAS
MATEMATIKOS IR GAMTOS MOKSLŲ FAKULTETAS
Karolis Dunderis
Taikomoji matematika (621G10003)

Baigiamojo projekto „Genetinio algoritmo, naudojamo gamybos tvarkaraščio sudarymui, tyrimas“

AKADEMINIO SĄŽININGUMO DEKLARACIJA

2015 m. birželio mėn. 4 d.
Kaunas

Patvirtinu, kad mano, **Karolio Dunderio,** baigiamasis darbas tema „Genetinio algoritmo, naudojamo gamybos tvarkaraščio sudarymui, tyrimas“ yra parašytas visiškai savarankiškai, o visi pateikti duomenys ar tyrimų rezultatai yra teisingi ir gauti sąžiningai. Šiame darbe nei viena darbo dalis nėra plagijuota nuo jokių spausdintinių ar internetinių šaltinių, visos kitų šaltinių tiesioginės ir netiesioginės citatos nurodytos literatūros nuorodose. Įstatymu nenumatytų piniginių sumų už šį darbą niekam nesu mokėjęs.

Aš suprantu, kad išaiškėjus nesąžiningumo faktui, man bus taikomos nuobaudos, remiantis Kauno technologijos universitete galiojančia tvarka.

_____ _____
(studento vardas ir pavardė, įrašyti ranka) (parašas)

TURINYS

Dunderis, K. The research of genetic algorithm used in production scheduling: Master's work in applied mathematics / supervisor dr. assoc. prof. N.Listopadskis; Mathematical Modeling department, Faculty of Mathematics and Natural Sciences, Kaunas University of Technology. Kaunas, 2015. - 74 p.

SUMMARY

Genetic algorithms are one of the best ways to solve a problem for which little is known. They are very general algorithms and will work well in any search space. All you need to know is what you need for the solution to be able to do well, and the genetic algorithm will be able to create a high quality solution. Genetic algorithms use the principles of selection and evolution to produce several solutions to a given problem.

In this research, we used genetic algorithms to solve production scheduling problems. There were three problems in total, each with a different difficulty, which were approached by using different kinds of mutation (exchange, inversion, insertion) and crossover (cycle, order, position based) combinations, and different population sizes (10, 25, 50, 75). Each mutation and crossover pairs were generated ten times and then the best solution was registered. Each generation was timed, so that we could compare time required to solve each problem using each population size and mutation / crossover pair.

The results have shown, that position based crossover and insertion mutation pair gives the best solutions while solving all three of these problems. Also, the results have shown that insertion mutation gives the best solutions among other mutations. Although the largest population (size 75) was expected to produce the best solutions, it failed to find a better solution then the population of size 50.

In conclusion, although the results have shown what type of mutation and crossover pair gave the best results for these production scheduling problems, it does not mean, that we would get the same results while solving completely different scheduling problems.

LENTELIŲ SĄRAŠAS

PAVEIKSLŲ SĄRAŠAS

ĮVADAS

Darbo planavimas (angl. open shop scheduling) yra optimizavimo informatikos ir operacinių tyrimų uždavinys. Uždavinys paprasčiausiai yra apibėžiamas taip:

Turime n skirtingo dydžio darbų $J_1, J_2, \dots, J_n$, kurie turi būti suplanuoti ant m identiškų mašinų, minimizuojant darbo trukmę. Darbo trukme laikomas plano ilgis (t.y. kai visi darbai yra užbaigti). Šiais laikais ši problema yra pateikiama kaip prijungimo problema (dinaminis planavimas), t.y. kiekvienas darbas yra pristatomas ir prijungimo algoritmas turi atlikti sprendimą ką su tuo darbu daryti. Tik priėmus sprendimą yra pristatomas naujas darbas.

Šis uždavinys yra vienas geriausiai žinomų prijungimo uždavinių ir buvo pirmasis uždavinys, kuriam buvo pristatyta konkurencinė analizė (R.Graham 1966 m.).

Egzistuoja įvairių šio uždavinio variantų, tokių kaip:

- Mašinos gali būti susijusios, nepriklausomos, lygios
- Mašinoms gali reikti pertraukų tarp darbų, arba jos gali dirbti be pertraukų
- Mašinos gali priklausyti nuo tam tikros sekos nustatymų
- Uždavinio tikslas gali būti minimizuoti darbo trukmę, vėlumą, maksimalų vėlavimą ir t.t. Taip pat gali būti naudojami keli tikslai
- Darbai gali turėti tam tikrus apribojimus, pavyzdžiui darbas i turi būti pabaigtas prieš darbą j
- Darbai ir mašinos turi bendrus apribojimus, pavyzdžiui kai kurie darbai gali būti atliekami tik ant tam tikrų mašinų
- Tam tikra darbų aibė gali būti susieta su tam tikra mašinų aibe
- Gali būti naudojami fiksuoti arba tikimybiniai apdorojimo laikai
- Gali būti ir kitokių apribojimų

Matematiškai šis uždavinys gali būti užrašomas taip:

Tegu $M = \{M_1, M_2, \dots, M_m\}$ ir $J = \{J_1, J_2, \dots, J_n\}$ yra atitinkamai baigtinės mašinų ir darbų aibės. Tegu X žymi visų nuoseklių darbų priskyrimų mašinoms aibę, kur kiekvienas darbas yra atliekamas ant kiekvienos mašinos tik vieną kartą; elementai $x \in X$ gali būti užrašomi kaip $n \times m$ matricos, kuriose i stulpelis nurodo kokius darbus ir kokia eilės tvarka juos atliks M_i mašina. Pavyzdžiui matrica

$$x = \begin{pmatrix} 1 & 2 \\ 2 & 3 \\ 3 & 1 \end{pmatrix}$$

parodo, kad mašina M_1 atliks tris darbus J_1, J_2, J_3 tokia eilės tvarka J_1, J_2, J_3, o mašina M_2 darbus atliks tokia eilės tvarka J_2, J_3, J_1.

Tarkime turi tam tikrą tikslo funkciją $C : X \to [0, +\infty]$. Tikslo funkcija gali būti interpretuota kaip bendras vykdymo laikas ir gali būti užrašoma atsižvelgiant į laikus $C_{ij} : M \times J \to [0, +\infty]$, t.y. kiek laiko mašina M_i vykdys darbą J_j.

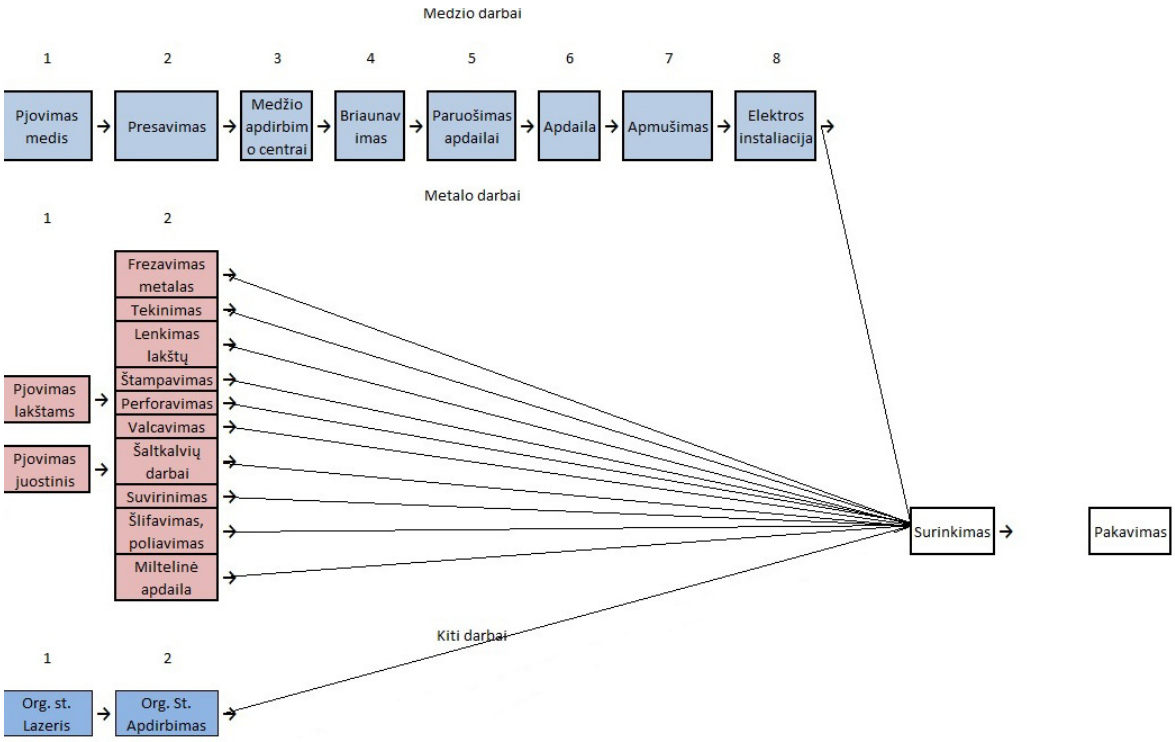
Šio uždavinio tikslas yra rasti tokį darbų $x \in X$ priskyrimą, kad $C(x)$ būtų minimali, t.y. neegzistuotų tokio $y \in X$, kur $C(x) > C(y)$.

1. TYRIMO OBJEKTAS

Šiame skyriuje aprašomas nagrinėtas uždavinys, taip pat aprašoma uždaviniui spręsti naudotas programinis paketas.

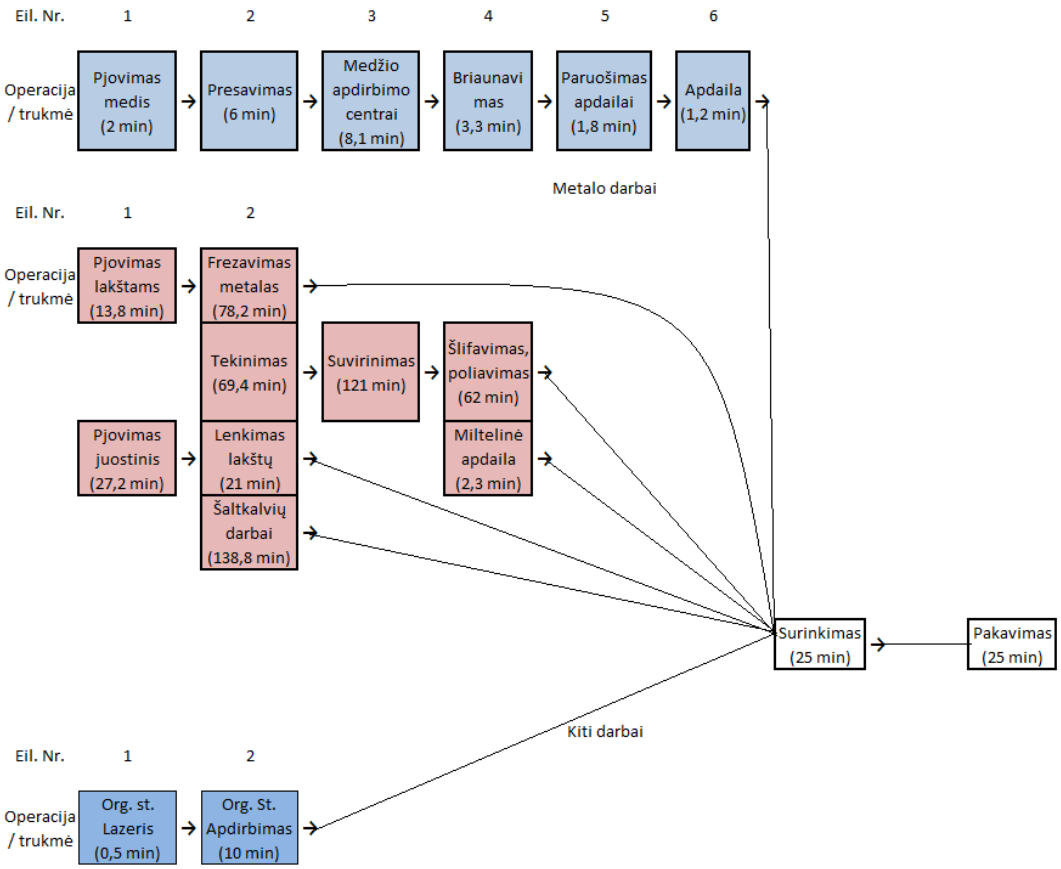
1.1. NAGRINĖTAS UŽDAVINYS

Šiame darbe buvo nagrinėtas vienos įmonės prekių gamybos tvarkaraštis. Bendras įmonės veiklos tvarkaraštis pateiktas 1.1.1 pav.

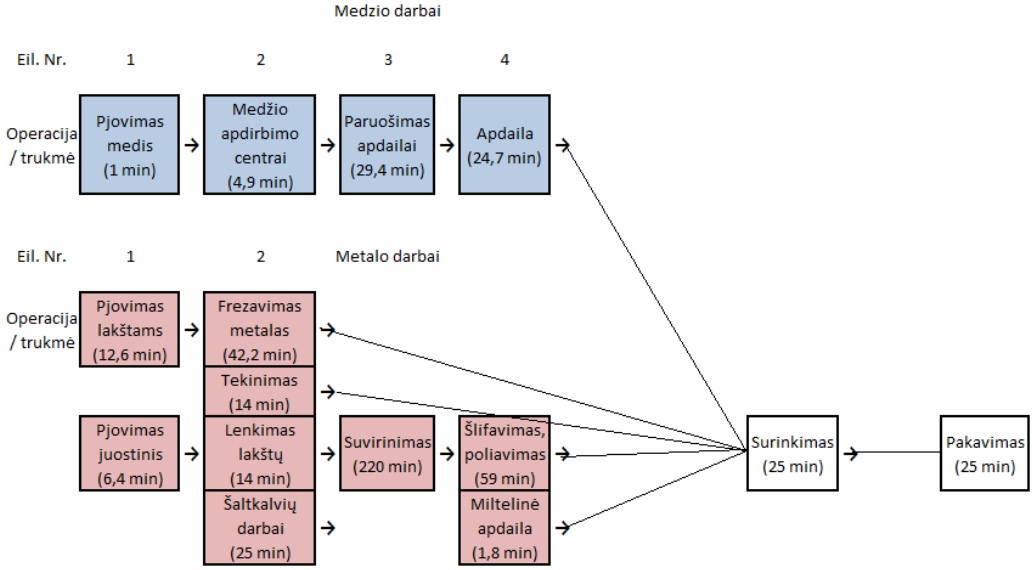


1.1.1 pav. Bendras įmonės veiklos tvarkaraštis

Iš šios įmonės buvo gauti duomenys apie dviejų prekių sudėtinių dalių (medis, metalas, organinis stiklas) apdorojimo trukmes, kurios buvo naudotos tiramojoje dalyje.
Nagrinėtų prekių gamybos tvarkaraščiai pateikti 1.1.2 – 1.1.3 pav.



1.1.2 pav. Pirmos prekės gamybos tvarkaraštis



1.1.3 pav. Antros prekės gamybos tvarkaraštis

1.2. PROGRAMINĖS ĮRANGOS APŽVALGA

MATLAB (iš žodžių *MATrix LABoratory*) yra daugiaplatformė MathWorks programinė įranga, skirta įvairių mokslo šakų problemoms spręsti, ypač matematinėms. Kaip galima spręsti iš pavadinimo, turi puikias galimybes manipuliacijoms su matricomis – būtent toks buvo pirminis šios programos tikslas. Dabar tai didžiulis galingas paketas, turintis savitą lengvai perprantamą programavimo kalbą.

Nors simbolinėms matematinėms manipuliacijoms geriau tinka Maple ir Mathematica produktai, su MathCad pradedantis vartotojas gali greičiau gauti pirmuosius rezultatus, MATLAB kaip programavimo kalba turi didesnį lankstumą, kuris naudingas įvairiems fizikos, biologijos bei kitiems matematiniams modeliams kurti ir tirti. Svarbus MATLAB trūkumas (palyginus su tradicinėmis programavimo kalbomis) yra reikalavimas visiems naudojantiems turėti įkeltas mokamas MATLAB bei visų reikalingų jo papildomų modulių kopijas. Tradicinės programavimo kalbos šiuo metu turi atviro kodo ar nemokamus kompiliatorius bei pasiekia didesnį vykdymo greitį, tačiau programavimas jomis ilgiau užtrunka. Bioinformatikiniuose tyrimuos MATLAB neretai naudojamas kaip „žvalgybinis“ paketas: pasiteisinę matematiniai modeliai vėliau perrašomi C (neretai generuojant žymias kodo dalis su Maple) lygiagrečiam vykdymui superkompiuteriuose.

2. NP SUNKUMO UŽDAVINIAI

Sunkūs uždaviniai, kurie negali būti išspręsti „tradiciniu“ būdu yra vadinami NP uždaviniais. Yra daug uždavinių kuriems žinome greitų (polinominių) algoritmų. Egzistuoja tokių uždavinių, kurių neįmanoma išspręsti algoritmiškai. Kai kuriems uždaviniams buvo įrodyta, kad jie yra neišsprendžiami per polinominį laiką.

Tačiau egzistuoja daugybė svarbių uždavinių, kuriems labai sunku rasti sprendinį, bet jį radus yra labai lengva patikrinti atsakymą. Šis faktas privedė prie NP - pilnų uždavinių (angl. *NP-complete*). NP reiškia nedeterministinis polinomas (angl. *nondeterministic polynomial*) ir reiškia, kad yra įmanoma „spėti“ sprendinį (naudojant tam tikrą nedeterministinį algoritmą) ir tada jį patikrinti. Jei turėtume mašiną, kuri galėtų spėlioti, tai galėtume rasti sprendinį per priimtiną laiką. NP – pilnų uždavinių studijavimas padeda suprastinti uždavinius juos apribojant, kai atsakymas gali būti „taip“ arba „ne“. Kadangi yra uždavinių su sunkesniais sprendiniais, buvo sugalvota NP – sunkių (angl. *NP – hard*) uždavinių klasė. Ši klasė nėra apribojama, kaip NP – pilnų uždavinių klasė.

NP uždaviniams išspręsti gali būti naudojamas pilno perrinkimo (angl. *brute force*) algoritmas. Tačiau šis algoritmas yra labai lėtas (dažniausiai $O(2^n)$ eilės) ir net šiek tiek didesniem uždavinaim yra nenaudojamas. Šiais laikais, nėra žinoma, ar egzistuoja greitesnis algoritmas. Daugelis galvoja, kad šitoks algoritmas neegzistuoja, todėl yra ieškoma alternatyvių metodų – tokių metodų pavyzdys yra genetiniai algoritmai.

3. GENETINIAI ALGORITMAI

Genetiniai algoritmai yra evoliucinio skaičiavimo dalis, kuri yra sparčiai auganti dirbtinio intelekto sritis. Genetiniai algoritmai yra paremti Darvino evoliucijos teorija.

Evoliucinio skaičiavimo idėja buvo paskelbta I. Rechenbergo 1960 metais jo projekte pavadinimu „Evoliucijos strategijos“ (org. kalb. „*Evolutionsstrategie*“). Jo įdėja buvo išplėtota kitų tyrėjų. Genetiniai algoritmai (angl. *Genetic Algorithms* (GAs)) buvo sugalvoti Johno Hollando ir išplėtota jo paties kartu jo studentais bei kolegomis. Taip 1975 metais buvo išleista Hollando knyga „Naturalių ir dirbtinių sistemų pritaikymas“ (angl. „*Adaption in Natural and Artificial Systems*“).

1992 metais Johnas Koza panaudojo genetinį algoritmą evoliucionuoti programas, kurios atliktų tam tikras užduotis. Jis savo metodą pavadino „genetiniu programavimu“ (angl. „*Genetic Programming*“(GP)). Buvo naudojamos LISP programos, nes programos parašytos ta kalba galėjo būti išreikštos „išnagrinėto medžio“ (angl. „*parse tree*“) forma, kuri yra genetinių algoritmų tyrimo objektas.

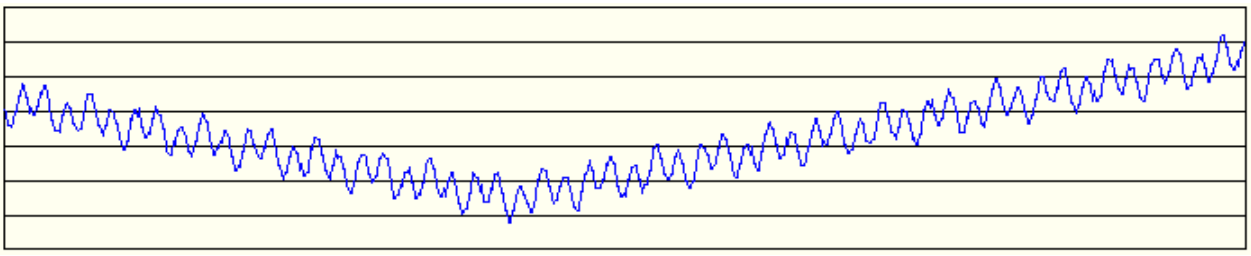
3.1 CHROMOSOMOS IR REPRODUKCIJA

Visi gyvi organizmai susideda iš ląstelių. Kiekvienoje ląstelėje yra vienodas chromosomų rinkinys. Chromosomos yra DNR grandinės, kurios tarnauja kaip modelis visam organizmui. Chromosomoje yra genų, DNR blokų. Kiekvienas genas koduoja tam tikrą baltymą. Iš esmės, kiekvienas genas koduoja bruožą, pavyzdžiui akių spalvą. Galimi bruožo nustatymai (pvz. mėlyna, ruda) yra vadinami aleliai (angl. *alleles*). Kiekvienas genas turi savo vietą chromosomoje. Ši vieta yra vadinama lokusu (angl. *locus*). Visas genetinių medžiagų (visų chromosomų) rinkinys yra vadinamas genomu (angl. *genome*). Tam tikras genomo genų rinkinys yra vadinamas genotipu (angl. *genotype*). Genotipas nulemia organizmo fenotipo (angl. *phenotype*) vystymasį, jo fizines bei protines charakteristikas, tokias kaip akių spalva, intelektas ir t.t.

Reprodukcijos metu, pirmiausia įvyksta rekombinacija (angl. *recombination*). Genai iš tėvų suformuoja naują chromosomą. Naujai sukurtas palikuonis tada gali mutuoti. Mutacija reiškia, kad kai kurie DNR elementai yra pakeičiami. Šie pakitimai daugiausia atsiranda dėl klaidų atsiradusių kopijuojant genus iš tėvų. Organizmo tinkamumas (angl. *fitness*) yra matuojamas jo sėkme gyvenime.

3.2 PAIEŠKOS ERDVĖ

Jei sprendžiame kažkokią problemą, mes dažniausiai ieškome kažkokio sprendinio, kuris būtų geriausias iš kitų sprendinių. Visų galimų atsakimų erdvė yra vadinama paieškos erdve (angl. *search space*). Kiekvienas taškas šioje erdvėje atitinka galimą sprendinį. Kiekvienas galimas sprendinys gali būti „pažymėtas“ savo reikšme arba tinkamumu sprendžiamai problemai. Mes ieškome savo sprendinio, kuris yra vienas taškas (arba daugiau) tarp galimų sprendinių – t.y. vienas taškas paieškos erdvėje. Tada sprendinio paieška yra ekvivalenti tam tikro ekstremumo (minimumo ar maksimumo) paieškai erdvėje. Paieškos erdvė gali būti visiškai išnagrinėta kol bus rastas tinkamas sprendinys, bet dažniausiai, mes žinome vos kelis jos taškus ir generuojame kitus taškus tęsiantis paieškos procesui. Paieškos erdvės pavyzdys pateiktas 3.2.1 paveiksle.



3.2.1 pav. Paieškos erdvės pavyzdys

Problema yra ta, kad paieška gali būti labai sudėtinga. Niekas nežino kur ieškoti sprendinio ir kur pradėti paiešką. Yra daugybė metodų rasti tinkamą sprendinį (nebūtinai geriausią), pavyzdžiui kopimo kalnu metodas (angl. *hill climbing*), tabu paieška (angl. *tabu search*), atkaitinimo modeliavimas (angl. *simulated annealing*) ir genetinis algoritmas. Sprendinys rastas vienu iš šių metodų yra skaitomas kaip geras sprendinys, nes retai yra įmanoma įrodyti, kas yra tikrasis optimumas.

3.3 GA VEIKIMO ALGORITMAS

1. Sugeneruoti atsitiktinę populiaciją iš *n* chromosomų (tinkami sprendiniai uždaviniui).
2. Įvertinti tinkamumo funkciją *f(x)* kiekvienai *x* chromosomai populiacijoje.
3. Sukurti naują populiaciją remiantis šiais žingsniais:
    - Parinkti dvi pirmines chromosomas iš populiacijos remiantis jų tinkamumo funkcija (kuo geresnė tinkamumo funkcija, tuo didesnis šansas būti parinktam).
    - Sukryžminti dvi pirmines chromosomas (su tam tikra kryžminimo tikimybe), taip sukuriant palikuonį. Jei kryžminimas neįvyko, palikuonis yra visiška priminių chromosomų kopija.
    - Mutuoti palikuonį kiekviename lokuse (su tam tikra mutavimo tikimybe).
    - Įdėti naują palikuonį į naują populiaciją.
4. Naudoti naujai sugeneruotą populiaciją tolimesniam algoritmo veikimui.
5. Jei galutinė sąlyga yra tenkinama, sustoti ir gražinti geriausią sprendinį į dabartinę populiaciją.
6. Eiti į „2“ žingsnį.

Iš algoritmo veikimo matome, kad kryžminimas ir mutacija yra svarbiausia genetinio algoritmo dalis. Atlikimas bus iš esmės nulemtas šių dviejų operatorių.

3.4 GA OPERATORIAI

3.4.1 CHROMOSOMOS KODAVIMAS

Kiekvienoje chromosomoje kažkokiu būdu turėtų būti informacija apie sprendinį. Labiausiai naudojamas kodavimas yra dvejetainis (angl. *binary*) kodavimas. Tada chromosoma atrodys taip, kaip pavaizduota 3.4.1.1 pav:

Chromosoma 1	1101100100110110
Chromosoma 2	1101111000011110

3.4.1.1 pav. Chromosomos pavyzdys

Kiekviena chromosoma turi vieną dvejetainę eilutę. Kiekviena tos eilutės dalis gali simbolizuoti kažkokią sprendinio charakteristiką. Žinoma yra daug daugiau kodavimo būdų. Tai

daugiausia priklauso nuo uždavinio. Pavyzdžiui, galima koduoti sveikus skaičius arba realius skaičius, kartais geriau yra koduoti tam tikras kombinacijas ir pan.

3.4.2 KRYŽMINIMAS

Kai yra nuspręsta kokį kodavimą naudoti, galime keliauti prie kryžminimo. Kryžminimo metu yra parenkami genai iš pagrindinių chromosomų ir sukuriamas palikuonis. Paprasčiausias būdas tai padaryti yra atsitiktinai pasirinkti kryžminimo tašką ir viską, kas yra prieš tą tašką nukopijuoti iš pirmos pagrindinės chromosomos, o viską, kas yra už to taško - iš antrosios pagrindinės chromosomos. Tada kryžminimas gali atrodyti taip, kaip pavaizduota 3.4.2.1 pav:

Chromosoma 1	11011 | 00100110110
Chromosoma 2	11011 | 11000011110
Palikuonis 1	11011 | 11000011110
Palikuonis 2	11011 | 00100110110

3.4.2.1 pav. Kryžminimo pavyzdys

Yra ir daugiau būdų kaip atlikti kryžminimą, pavyzdžiui, galime paimti daugiau kryžminimo taškų. Kryžminimas gali būti sudėtingas ir labai priklauso nuo chomosomos kodavimo. Tam tikram uždaviniui sukurtas specifinis kryžminimas gali pagerinti genetinio algoritmo darbą.

3.4.2.1 CIKLINIS KRYŽMINIMAS

Ciklinis kryžminimas (angl. *cycle crossover*) sudaro palikuonį iš išrikiuotų chromosomų nustatant ryšius tarp dviejų pagrindinių (angl. *parent*) chromosomų. Tam kad sukurti palikuonį, ciklai yra nukopijuojami iš atitinkamų pagrindinių (tėviškųjų) chromosomų. Ciklinio kryžminimo veikimo algoritmas:

1) Pradėti pirmosios tėviškosios chromosomos pirmajame gene.
2) Nustatyti antrosios tėviškosios chromosomos geną, kuris yra toje pačioje pozicijoje.
3) Eiti į poziciją pirmose tėviškoje chromosomoje, kurioje yra toks pat genas.
4) Pridėti šį geną į ciklą.
5) 2-4 žingsnius kartoti tol, kol grįšime į pradinį geną.

Indeksai, kurie sudaro ciklą paskui yra naudojami sukuriant palikuonį tokiu būdu: pirmo ciklo genai iš pirmosios tėviškosios chromosomos nukopijuojami į pirmąjį palikuonį, antro ciklo genai iš pirmosios tėviškosios chromosomos nukopijuojami į antrąjį palikuonį ir taip toliau.

Algoritmo veikimo pavyzdys:

Tėviškoji chromosoma 1: 8 4 7 3 6 2 5 1 9 0

Tėviškoji chromosoma 2: 0 1 2 3 4 5 6 7 8 9

Pirmo ciklo reikšmės: 8 9 0. Jos bus paryškintos orandžine spalva.

Pirmas ciklas: pradeda pirmoje pirmosios tėviškosios chromosomos reikšmėje ir einame į tą pačią poziciją antrojoje tėviškoje chromosomoje. 8 eina į 0. Tada ieškome 0 pirmojoje tėviškoje chromosomoje, kurią randame dešimtoje pozicijoje. Einame į antrąją tėviškąją chromosomą toje pačioje pozicijoje ir randame ten 9. Tada ieškome devintos pozicijos pirmojoje tėviškoje chromosomoje, kurioje yra 8. Kandangi pradėjome su 8, tai ciklas pabaigtas.

Tėviškoji chromosoma 1: ~~8~~ 4 7 3 6 2 5 1 ~~9 0~~

Tėviškoji chromosoma 2: ~~0~~ 1 2 3 4 5 6 7 ~~8 9~~

Antro ciklo reikšmės: 4 1 7 2 5 6. Jos bus pažymėtos raudonai.

Antras ciklas: pradedame ties 4 ir einame į 1. 1 randame aštuntoje pirmosios tėviškosios chromosomos vietoje. Einame į 7. Iš 7 į 2, tada iš 2 į 5, iš 5 į 6, o iš 6 į 4, kur ciklas pasibaigia.

Tėviškoji chromosoma 1: ~~8~~ ~~4 7~~ 3 ~~6 2 5 1~~ ~~9 0~~

Tėviškoji chromosoma 2: ~~0~~ ~~1 2~~ 3 ~~4 5 6 7~~ ~~8 9~~

Trečio ciklo reikšmės: 3.

Trečias ciklas: vienintelis likęs ciklas yra vientinio ilgio ir yra reikšmė 3.

Palikuonių sudarymas:

Tėviškoji chromosoma 1: 8 4 7 3 6 2 5 1 9 0

Tėviškoji chromosoma 2: 0 1 2 3 4 5 6 7 8 9

Palikuonis 1: 8 1 2 3 4 5 6 7 9 0

Palikuonis 2: 0 4 7 3 6 2 5 1 8 9

Kopijavimo ciklas 1: Pirmo ciklo reikšmės iš pirmosios tėviškosios chromosomos nukopijuojamos į pirmą palikuonį, o reikšmės iš antrosios tėviškosios chromosomos nukopijuojamos į antrą palikuonį.

Kopijavimo ciklas 2: Antro ciklo reikšmės iš pirmosios tėviškosios chromosomos nukopijuojamos į antrą palikuonį, o reikšmės iš antrosios tėviškosios chromosomos nukopijuojamos į pirmą palikuonį.

Kopijavimo ciklas 3: Pirmo ciklo reikšmės iš pirmosios tėviškosios chromosomos nukopijuojamos į pirmą palikuonį, o reikšmės iš antrosios tėviškosios chromosomos nukopijuojamos į antrą palikuonį.

3.4.2.2 TVARKOS KRYŽMINIMAS

Tvarkos kryžminimas (angl. *order crossover*) yra gan paprastas perstatymo kryžminimas. Iš esmės, iš pirmosios tėviškosios chromosomos paimamas nuoseklus alelių ruožas kuris bus toje pačioje vietoje palikuonyje. Likusios vietos užpildomos antrosios tėviškosios chromosomos reikšmėmis iš eilės, kaip yra išsidėstę chromosomoje. Tvarkos kryžminimo veikimo algoritmas:

1) Atsitiktinai pasirinkti nuoseklų alelių ruožą iš pirmosios tėviškosios chromosomos.
2) Ruožą įstatyti į pirmąjį palikuonį.
3) Pradedant nuo dešinės alelių ruožo pusės, palikuonis užpildomas reikšmėmis iš antrosios tėviškosios chromosomos.
4) Jei norima sudaryti antrą palikuonį, sukeisti tėviškąsias chromosomas vietomis ir grįžti į pirmą žingsnį.

Algoritmo veikimo pavyzdys:

Tėviškoji chromosoma 1: 8 4 7 3 6 2 5 1 9 0

Tėviškoji chromosoma 2: 0 ~~1~~ ~~2~~ ~~3~~ 4 ~~5~~ ~~6~~ 7 8 9

Palikuonis 1: 0 4 7 3 6 2 5 1 8 9

1 žingsnis: parinktas nuoseklus alelių ruožas (pabrauktas)

2 žingsnis: ruožas įdedamas į palikuonį toje pačioje pozicijoje.

3 žingsnis: pradedant dešiniaja ruožo puse užpildome likusias palikuonio chromosomos vietas.

Pastebime, kad 1, 2 ir 3 yra praleidžiami, nes jie yra išbraukti, todėl 4 yra įdėtas į antrąją pirmojo palikuonio poziciją.

3.4.2.3 POZICIJA PAREMTAS KRYŽMINIMAS

Pozicija paremtas kryžminimas (angl. *position based crossover*) tai dar vienas kryžminimo metodas. Šis metodas yra panašus į tvarkos kryžminimą, tačiau šiame metode pradžioje pasirinkti skaičiai neturi eiti vienas po kito (neturi būti viename ruože). Pozicija paremto kryžminimo veikimo algoritmas:

1) Atsitiktiniu būdu iš vieno iš tėviškųjų chromosomų parinkti tam tikrą skaičių reikšmių su atitinkamomis jų pozicijomis.
2) Sudaryti palikuonio prototipą, kuriame įrašyti prieš tai pasirinktas reikšmes į tokias pat pozicijas.
3) Iš kitos tėviškosios chromosomos ištrinti jau pasirinktas reikšmes. To pasekoje gauname reikšmes, kurios reikalingos užbaigti palikuoniui.
4) Įrašyti likusias reikšmes iš kairės į dešinę remiantis 3 žingsnyje gauta seka. Tokiu būdu gaunamas palikuonis.

Algoritmo veikimo pavyzdys:

Tėviškoji chromosoma 1: 1 2 3 4 5 6 7 8 9

Palikuonis: 4 2 3 1 5 6 7 8 9

Tėviškoji chromosoma 2: ~~5~~ 4 ~~6~~ 3 1 ~~9~~ ~~2~~ 7 8

Iš pirmosios tėviškosios chromosomos atsitiktiniu būdu buvo parinktos reikšmės 2, 5, 6, 9 (pabrauktos). Jos su atitinkamomis pozicijomis yra nukopijuojamos į palikuonį. Iš antrosios tėviškosios chromosomos pašalinamos reikšmės, kuriomis katik buvo užpildytas palikuonis. Likusios reikšmės, į palikuonį yra įrašomos tokia tvarka, kokia yra išsidėsčiusios reikšmės kai dalis reikšmių iš antrosios tėviškosios chromosomos buvo pašalinta.

3.4.3 MUTACIJA

Po kryžminimo proceso įvyksta mutacija. Taip būna todėl, kad išvengti įvykio, kai visi populiacijoje esantys sprendiniai atsiduria išspręsto uždavinio lokaliame optimume. Mutacija atsitiktinai pakeičia pirmagimį. Dvejetainiame kodavime atsitiktinai parinktos dalys gali būti pakeistos iš *0* į *1*, arba iš *1* į *0*. Mutacija gali atrodyti taip, kaip pavaizduota 3.4.3.1 pav:

Prad. palikuonis 1	1101111000011110
Prad. palikuonis 2	1101100100110110
Mut. palikuonis 1	1100111000011110
Mut. palikuonis 2	1101101100110110

3.4.3.1 pav. Mutacijos pavyzdys

Mutacija priklauso tiek nuo kodavimo, tiek nuo kryžminimo. Pavyzdžiui, kai koduojame kombinacijas, mutacija gali pakeisti du genus.

3.4.3.1 SUKEITIMO MUTACIJA

Sukeitimo mutacijos (angl. *exchange mutation*) metu palikuonyje atsitiktiniu būdu yra parenkamos dvi pozicijos. Šios dvi pozicijos yra sukeičiamos vietomis ir taip gaunamas mutavęs palikuonis. Pavyzdys:

Turime pirmagimį: 1 2 3 4 5 6 7 8 9.

Atsitiktiniu būdu pasirenkame dvi pozicijas: 1 2 3 4 5 6 7 8 9.

Šios dvi pozicijos yra sukeičiamos vietomis: 1 2 6 4 5 3 7 8 9.

3.4.3.2 PERSTATYMO MUTACIJA

Perstatymo mutacijos (angl. *inversion mutation*) metu palikuonyje atsitiktiniu būdu yra parenkamas ruožas. Šis ruožas yra apverčiamas ir taip gaunamas mutavęs palikuonis

Pavyzdys:

Turime pirmagimį: 1 2 3 4 5 6 7 8 9.

Atsitiktiniu būdu pasirenkame ruožą: 1 2 3 4 5 6 7 8 9.

Šis ruožas yra apverčiamas: 1 2 6 5 4 3 7 8 9.

3.4.3.3 ĮTERPIMO MUTACIJA

Įterpimo mutacijos (angl. *insertion mutation*) metu palikuonyje atsitiktiniu būdu yra parenkama pozicija. Ši pozicija yra įterpiama į atsitiktinai parenkamą vietą ir taip gaunamas mutavęs palikuonis.

Pavyzdys:

Turime pirmagimį: 1 2 3 4 5 6 7 8 9.

Atsitiktiniu būdu pasirenkame poziciją: 1 2 3 4 5 6 7 8 9.

Poziciją įterpiame į atsitiktinai parinktą vietą: 1 2 6 3 4 5 7 8 9.

3.4.4 KRYŽMINIMO IR MUTACIJOS TIKIMYBĖS

Kryžminimo bei mutacijos tikimybės yra du pagrindiniai genetinių algoritmų parametrai. Kryžminimo tikimybė nusako, kaip dažnai įvyks kryžminimo procesas. Jei kryžminimas neįvyks, palikuonis bus visišką pradinių chromosomų kopija. Jei kryžminimo tikimybė yra 1, tada visi palikuonys yra gauti iš kryžminimo. Jei tikimybė yra 0, tai visa nauja karta yra gauta iš visiškų senos kartos chromosomų kopijų (tai nereiškia, kad nauja karta bus visiškai identiška senajai). Kryžminimas atliekamas tikintis, kad naujos chromosomos turės geras senų chromosomų savybes ir naujosios chromosomos bus geresnės. Tačiau yra gera mintis leisti daliai populiacijos išlikti iki kitos kartos.

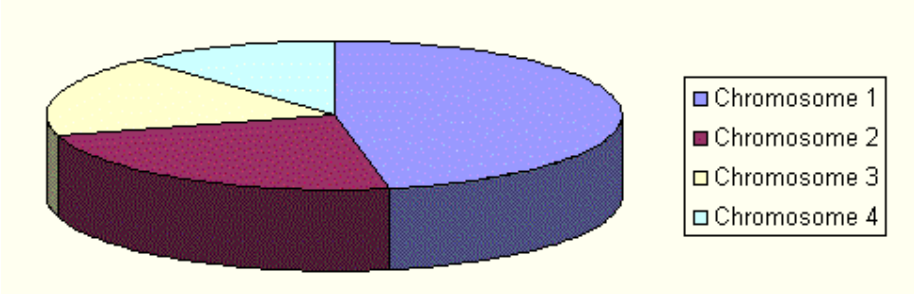
Mutacijos tikimybė nusako kaip dažnai chromosomos dalys mutuos. Jei mutacija neįvyksta, palikuonis imamas po kryžminimo (arba kopija) be jokios žalos. Jei mutacija įvyksta, dalis chromosomos pasikeičia. Jei mutacijos tikimybė yra 1, tai visa chromosoma pasikeičia, jei tikimybė 0, tai nepasikeičia niekas. Mutacija naudojama, kad išvengti įvykio, kai genetinis algoritmas atsiduria lokaliame ekstremume, tačiau tai turėtų įvykti nedažnai, nes tokiu atveju genetinis algoritmas virstu atsitiktine paieška.

Yra ir kitų genetinių algoritmų parametrų. Vienas jų, taip pat svarbus, yra populiacijos dydis. Populiacijos dydis nusako, kiek chromosomų yra vienoje populiacijoje (vienoje kartoje). Jei yra per mažai chromosomų, genetinis algoritmas turės mažą galimybę atlikti kryžminimą ir tokiu būdu bus išnagrinėta nedidelė paieškos erdvės dalis. Kita vertus, jei yra per daug chromosomų, genetinis algoritmas suletėja. Tyrimai rodo, kad po tam tikro kiekio (kuris priklauso nuo kodavimo ir pačio uždavinio) nėra naudinga didinti populiacijos dydį, nes tai nebepagreitina sprendimo proceso.

3.5 ATRANKA

Kaip žinome iš genetinių algoritmų veikimo, iš populiacijos yra parenkamos chromosomos, kurios bus naudojamos kryžminimui. Problema yra tame, kad nežinoma, kaip šias chromosomas parinkti. Pagal Darvino evoliucijos teoriją, stipriausi turėtų išlikti ir sukurti naują palikuonį. Yra daugybė metodų, kurie padeda parinkti geriausias chromosomas. Kelis iš jų paminėsime.

Ruletės rato atranka. Pagrindinės chromosomos yra parenkamos pagal jų tinkamumo funkciją. Kuo geresnė chromosoma, tuo didesnis šansas, kad ji bus išrinkta. Įsivaizduokime ruletės ratą ant kurio sudėtos visos chromosomos esančios populiacijoje, kur kiekviena turi savo vietą pagal tinkamumo funkcijos reikšmę (3.5.1 pav):



3.5.1 pav. Ruletės rato atranka

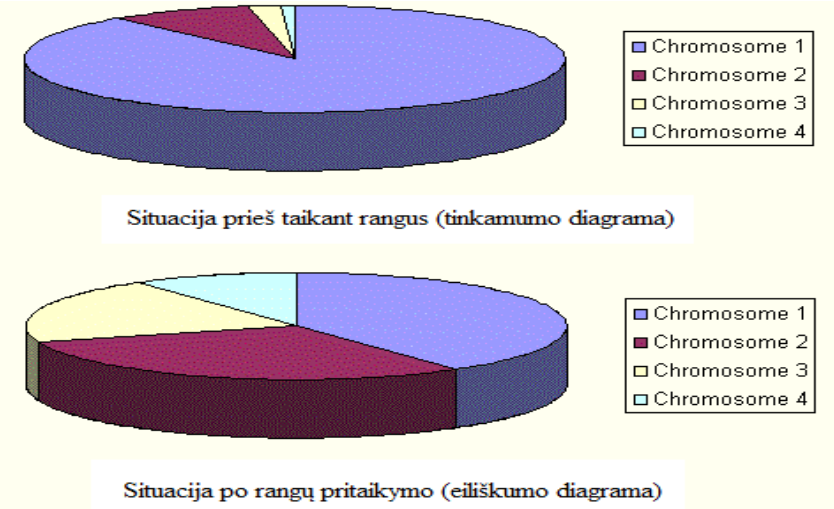
Tada metamas ruletės rutuliukas ir taip parenkama chromosoma. Chromosoma su didžiausia tinkamumo funkcijos reikšme bus parinkta dažniausiai.

Tai gali būti pavaizduota naudojant šį algoritmą:

1. Suskaičiuoti visų chromosomų tinkamumo funkcijų reikšmių sumą *S*.
2. Sugeneruoti atsitiktinį skaičių *r* iš intervalo *(0,S)*.
3. Eiti per populiaciją ir sumuoti tinkamumo funkcijų reikšmę nuo 0 iki *s*. Kai suma *s* yra didesnė už *r*, sustoti ir gražinti chromosomą ant kurios algoritmas sustojo.

Žinoma, pirmas žingsnis yra atliekamas tik vieną kartą kiekvienai populiacijai.

Rango atranka. Prieš tai paminėta atranka turi sunkumų, kai tinkamumo funkcijos labai viena nuo kitos skiriasi. Pavyzdžiui, jeigu geriausios chromosomos tinkamumo funkcija užima 90% visos ruletės, tai likusios chromosomos turės labai mažą tikimybę būti išrinktomis. Rango atranka pirmiausia populiacijai priskiria rangus, o tada kiekvienai chromosomai priskiriama tinkamumo funkcijos reikšmė pagal priskirtą rangą. Prasčiausia turės tinkamumo funkcijos reikšmę 1, antra pagal prastumą – 2 ir t.t., o geriausia įgys reikšmę *N* (chromosomų skaičius populiacijoje). 3.5.2 paveikslėlyje parodyta, kaip pasikeičia situacija, kai tinkamumas pakeičiamas eiliškumu.



3.5.2 pav. Rango atrankos taikymo pavyzdys

Po šio pakeitimo visos chromosomos turi šansą būti parinktomis. Tačiau šis metodas gali lemti lėtą konvergavimą, nes geriausios chromosomos beveik nesiskiria nuo kitų.

4. TIRIAMOJI DALIS

Tiriamojoje dalyje buvo nagrinėjama genetinių algoritmų tikslumo priklausomybė nuo kryžminimo bei mutacijos metodų, sprendžiant gamybos tvarkaraščio sudarymo uždavinį. Darbe buvo naudojami trijų rūšių kyžminimai, trijų rūšių mutacijos, bei trijų skirtingų sunkumų uždaviniai, kurie buvo sprendžiami naudojant skirtingo dydžio populiacijas. Tyrimo metu taip pat buvo atsižvelgiama į algoritmo veikimo trukmę.

4.1. TRIJŲ RŪŠIŲ GAMINIAI ANT TRIJŲ MAŠINŲ

Šiame skyriuje buvo nagrinėjamas uždavinys, kuriame trijų rūšių darbai turėjo būti sudėlioti ant trijų mašinų. Uždavinys pateiktas 4.1.1 lentelėje, kurioje kiekvieno darbo trukmė yra pateikta minutėmis.

4.1.1 lentelė. Trijų rūšių darbų ir trijų mašinų uždavinys.

Gaminio rūšis / Gaminio nr.	1	2	3
1	12.6	14	14
2	6.4	14	14
3	2	6	8.1

Darbai turėjo būti išdeliojami ant mašinų taip, kad kiekvienos rūšies darbai tarpusavyje nepersidengtų, t.y. jei šiuo metu yra vykdomas i – tos rūšies darbas ant j – tos mašinos, tai kitas tos pačios rūšies darbas ant ant k – tos mašinos (kai $k \neq j$) negalės būti uždedamas kol pirmasis darbas nebus pabaigtas. Uždavinys buvo sprendžiamas naudojant visas trijų kryžminimo metodų (ciklinio, tvarkos bei pozicija paremto) ir trijų mutacijos metodų (sukeitimo, perstatymo bei įterpimo) kombinacijas. Uždaviniui spręsti buvo nustatyta kryžinimo tikimybė 0.1, mutacijos tikimybė 0.1, algoritmo žingsnių skaičius 30. Uždavinys buvo spręstas algoritmo populiacijas parenkant 10, 25, 50 ir 75. Algoritmas su kiekviena kombinacija buvo paleistas po 10 kartų ir užfiksuojamas geriausias gautas rezultatas. Gauti rezultatai pateikti 4.1.2 - 4.1.5 lentelėse, kuriose pateiktas geriausias bendras gamybos laikas minutėmis su kiekviena kombinacija.

4.1.2 lentelė. Trijų rūšių darbų ir trijų mašinų uždavinio rezultatai, kai populiacija yra 10

Kryžminimas / Mutacija	Ciklinis	Tvarkos	Pozicija paremtas
Sukeitimo	102.2	102.2	102.2
Perstatymo	102.2	102.2	102.2
Įterpimo	102.2	102.2	102.2

4.1.3 lentelė. Trijų rūšių darbų ir trijų mašinų uždavinio rezultatai, kai populiacija yra 25

Kryžminimas / Mutacija	Ciklinis	Tvarkos	Pozicija paremtas
Sukeitimo	102.2	102.2	102.2
Perstatymo	102.2	102.2	102.2
Įterpimo	102.2	102.2	102.2

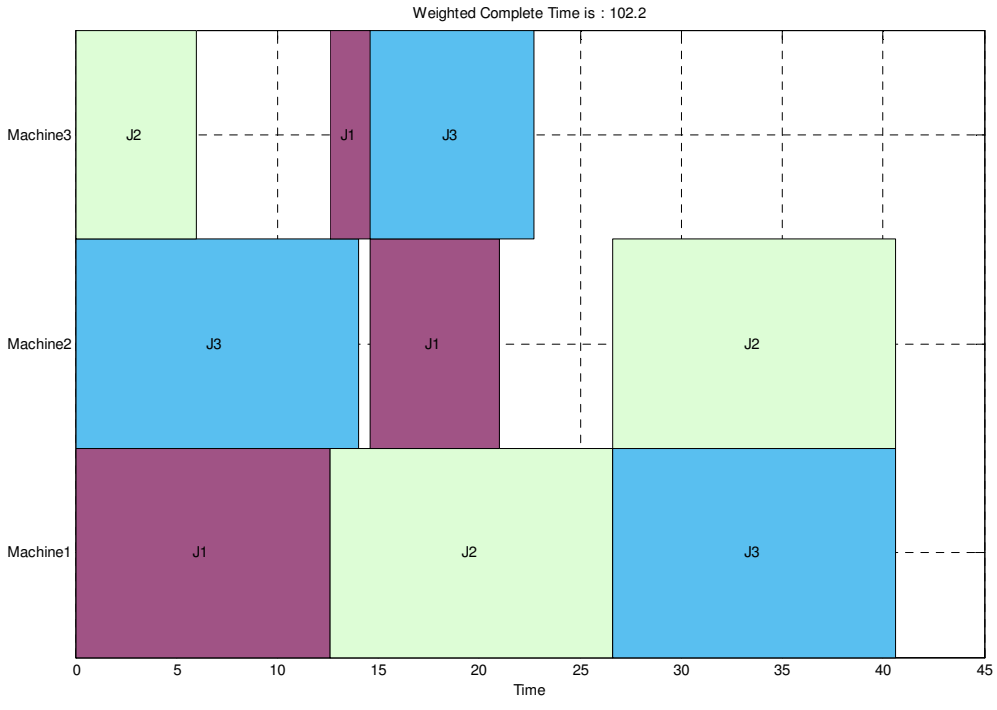
4.1.4 lentelė. Trijų rūšių darbų ir trijų mašinų uždavinio rezultatai, kai populiacija yra 50

Kryžminimas / Mutacija	Ciklinis	Tvarkos	Pozicija paremtas
Sukeitimo	102.2	102.2	102.2
Perstatymo	102.2	102.2	102.2
Įterpimo	102.2	102.2	102.2

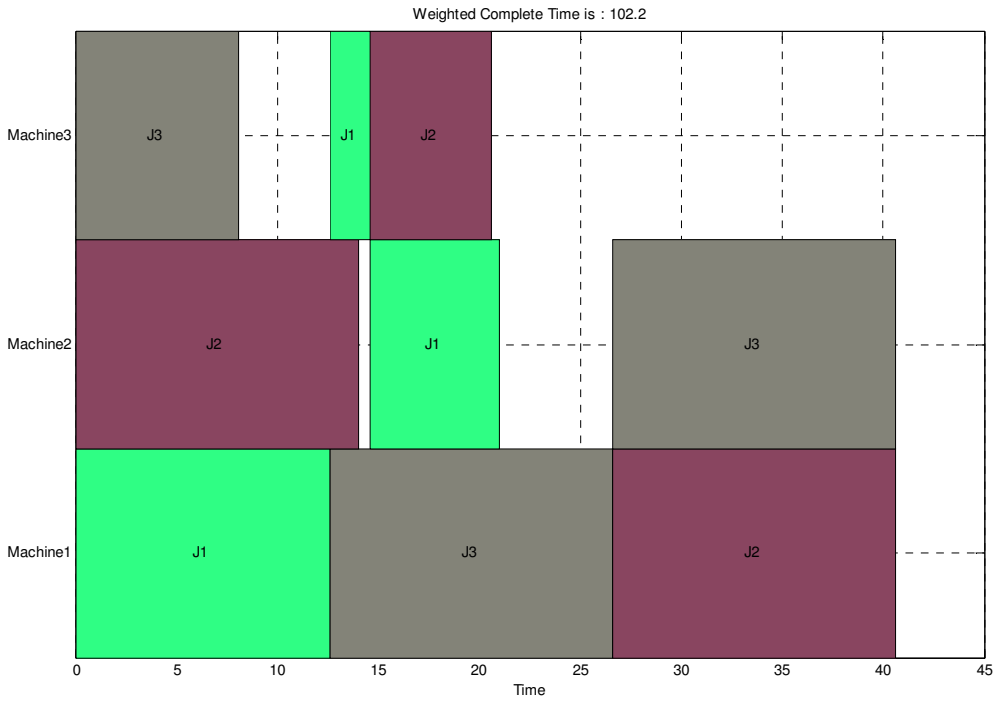
4.1.5 lentelė. Trijų rūšių darbų ir trijų mašinų uždavinio rezultatai, kai populiacija yra 75

Kryžminimas / Mutacija	Ciklinis	Tvarkos	Pozicija paremtas
Sukeitimo	102.2	102.2	102.2
Perstatymo	102.2	102.2	102.2
Įterpimo	102.2	102.2	102.2

Kaip matome iš 4.1.2 – 4.1.5 lentelių, sprendžiant trijų rūšių darbų ir trijų mašinų uždavinį visi gauti rezultatai yra vienodi, tačiau darbų tvarkaraščiai tarpusavyje skyrėsi. Šie tvarkaraščiai pateikti 4.1.1 – 4.1.2 pav.

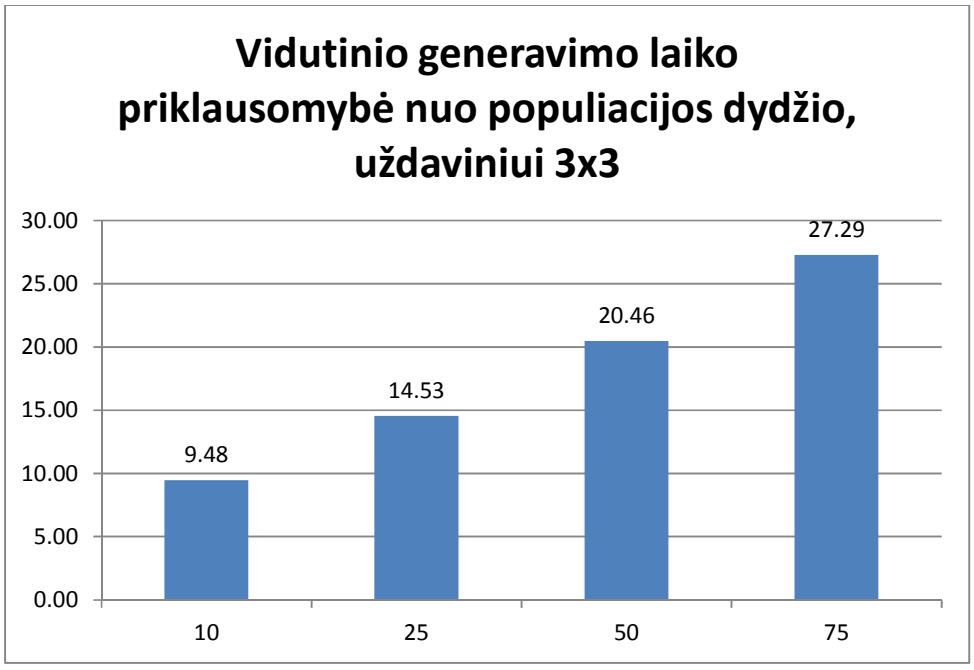


4.1.1 pav. Geriausias gautas trijų rūšių darbų ir trijų mašinų uždavinio tvarkaraštis (1)



4.1.2 pav. Geriausias gautas trijų rūšių darbų ir trijų mašinų uždavinio tvarkaraštis (2)

Sprendžiant šį uždavinį buvo fiksuojama kiekvieno generavimo trukmė. Vidutinės generavimo trukmės, sprendžiant šį uždavinį, pateiktos 4.1.3 pav.



4.1.3 pav. Trijų rūšių darbų ir trijų mašinų uždavinio vidutinės generavimo trukmės

Kaip matome iš 4.1.3 pav. sprendžiant trijų rūšių darbų ir trijų mašinų uždavinį, didinant populiacijos dydį, generavimo trukmė kinta tiesiškai. Lyginant mažiausią bei didžiausią nagrinėtą populiaciją, generavimo trukmė tarp jų skiriasi beveik tris kartus. Nepaisant didelio skirtumo tarp populiacijų, gauti rezultatai buvo vienodi, todėl šiam uždaviniui spręsti pakanka naudoti populiaciją, kurios dydis yra 10.

4.2. PENKIŲ RŪŠIŲ GAMINIAI ANT PENKIŲ MAŠINŲ

Šiame skyriuje buvo nagrinėjamas uždavinys, kuriame penkių rūšių darbai turėjo būti sudėlioti ant penkių mašinų. Uždavinys pateiktas 4.2.1 lentelėje, kurioje kiekvieno darbo trukmė yra pateikta minutėmis.

4.2.1 lentelė. Penkių rūšių darbų ir penkių mašinų uždavinys.

Gaminio rūšis / Gaminio nr.	1	2	3	4	5
1	12.6	14	14	25	1.8
2	6.4	14	14	4.9	15
3	2	6	8.1	3	13.8
4	6	1.8	15	2.3	10
5	2	3.3	1.8	15	21

Darbai turėjo būti išdeliojami ant mašinų taip, kad kiekvienos rūšies darbai tarpusavyje nepersidengtų, t.y. jei šiuo metu yra vykdomas i – tos rūšies darbas ant j – tos mašinos, tai kitas tos pačios rūšies darbas ant ant k – tos mašinos (kai $k \neq j$) negalės būti uždedamas kol pirmasis darbas nebus pabaigtas. Uždavinys buvo sprendžiamas naudojant visas trijų kryžminimo metodų (ciklinio, tvarkos bei pozicija paremto) ir trijų mutacijos metodų (sukeitimo, perstatymo bei įterpimo) kombinacijas. Uždaviniui spręsti buvo nustatyta kryžinimo tikimybė 0.1, mutacijos tikimybė 0.1, algoritmo žingsnių skaičius 30. Uždavinys buvo spręstas algoritmo populiacijas parenkant 10, 25, 50 ir 75. Algoritmas su kiekviena kombinacija buvo paleistas po 10 kartų ir užfiksuojamas geriausias gautas rezultatas. Gauti rezultatai pateikti 4.2.2 – 4.2.5 lentelėse, kuriose pateiktas geriausias bendras gamybos laikas minutėmis su kiekviena kombinacija.

4.2.2 lentelė. Penkių rūšių darbų ir penkių mašinų uždavinio rezultatai, kai populiacija yra 10

Kryžminimas / Mutacija	Ciklinis	Tvarkos	Pozicija paremtas
Sukeitimo	346.8	353	327.6
Perstatymo	329.4	346.2	335
Įterpimo	328.6	342	327

4.2.3 lentelė. Penkių rūšių darbų ir penkių mašinų uždavinio rezultatai, kai populiacija yra 25

Kryžminimas / Mutacija	Ciklinis	Tvarkos	Pozicija paremtas
Sukeitimo	327.1	318.7	314.8
Perstatymo	319.6	315.1	312.6
Įterpimo	314.5	314.7	308.8

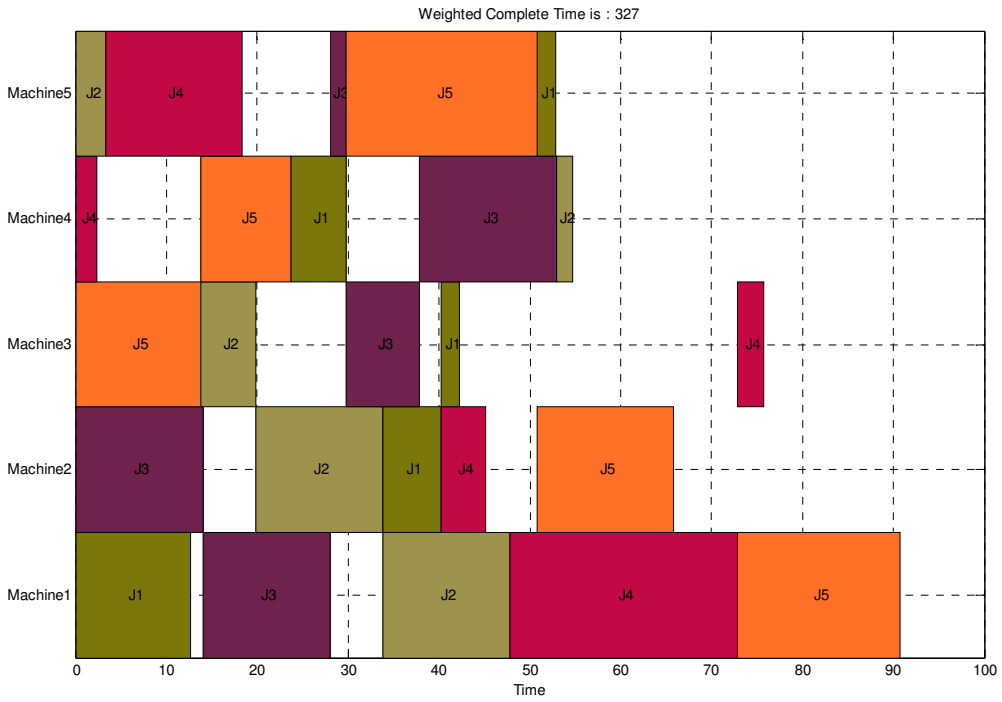
4.2.4 lentelė. Penkių rūšių darbų ir penkių mašinų uždavinio rezultatai, kai populiacija yra 50

Kryžminimas / Mutacija	Ciklinis	Tvarkos	Pozicija paremtas
Sukeitimo	320.9	314.1	311.2
Perstatymo	310.2	312.1	300.7
Įterpimo	308.6	305.4	298.6

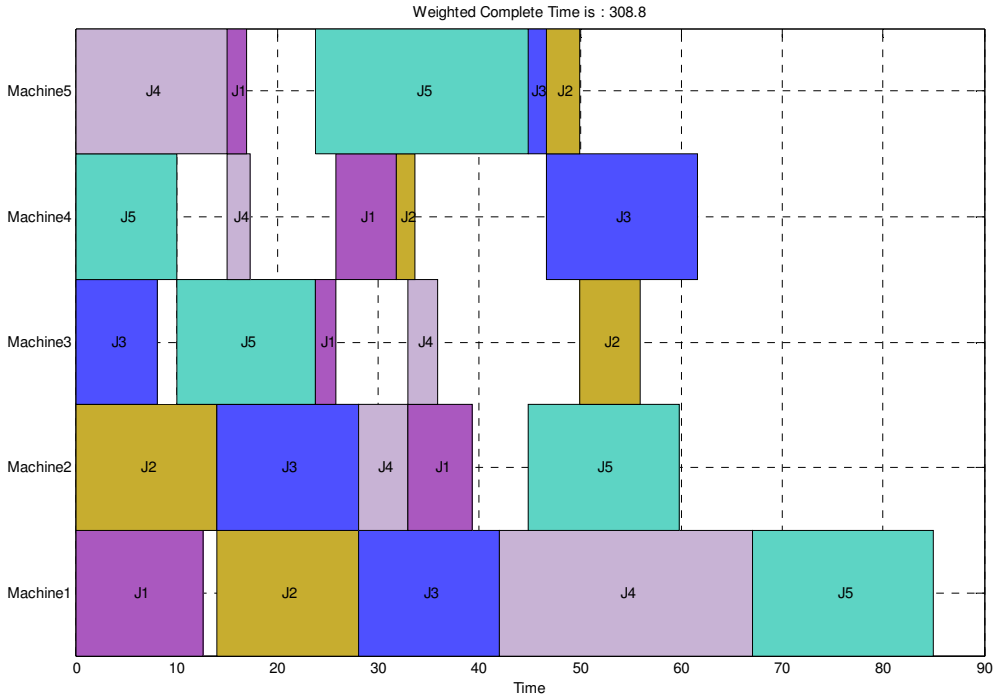
4.2.5 lentelė. Penkių rūšių darbų ir penkių mašinų uždavinio rezultatai, kai populiacija yra 75

Kryžminimas / Mutacija	Ciklinis	Tvarkos	Pozicija paremtas
Sukeitimo	334	326	312.9
Perstatymo	316.2	312.3	305.9
Įterpimo	315.7	306.3	302

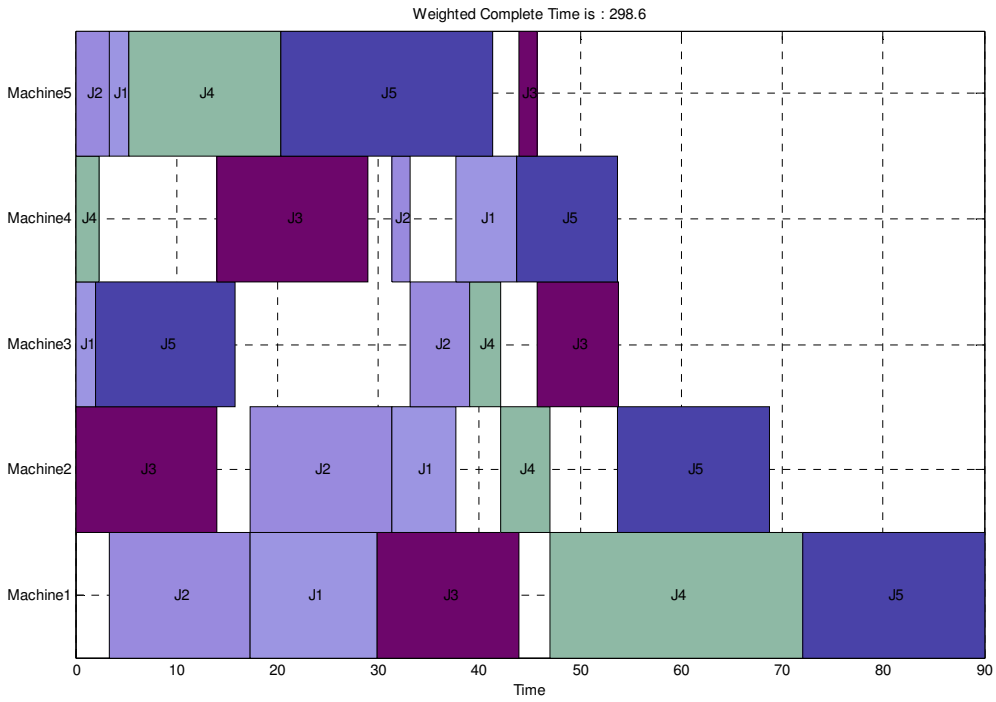
Kaip matome iš 4.2.2 – 4.2.5 lentelių, visi gauti rezultatai gerokai skiriasi vienas nuo kito. Geriausią rezultatą davė kombinacija pozicija paremtas kryžminimas ir įterpimo mutacija. Taip pat pastebime, kad įterpimo mutacija davė geriausius rezultatus prie kiekvieno nagrinėto kryžminimo metodo, bei su visais populiacijų dydžiais. Geriausi tvarkaraščiai prie kiekvieno populiacijos dydžio pateikti pateikti 4.2.1 – 4.2.4 pav.



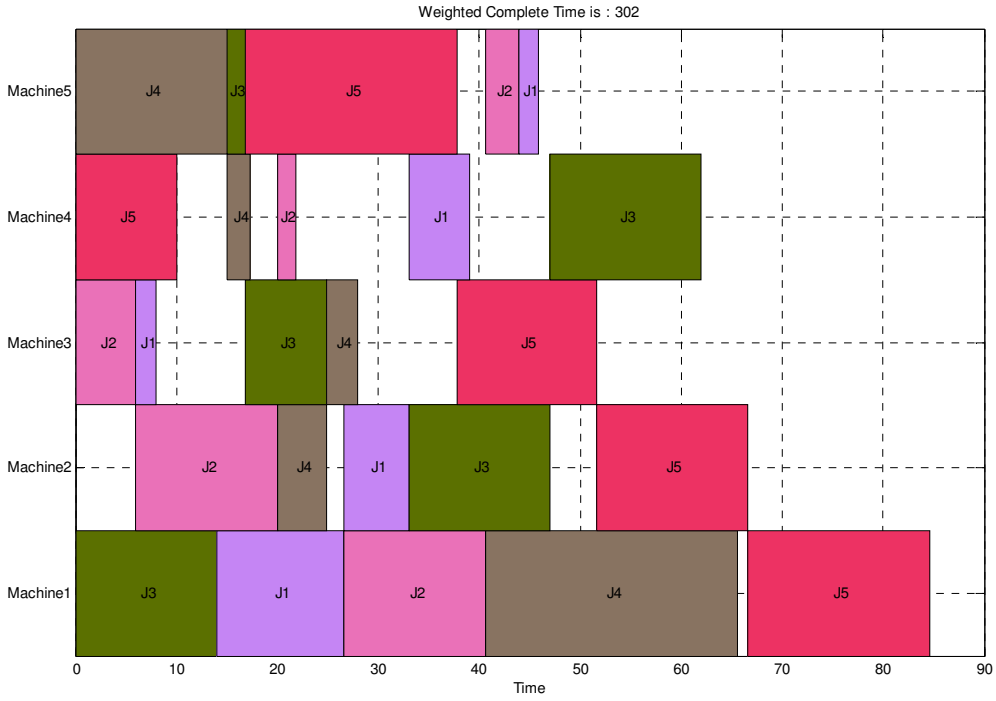
4.2.1 pav. Geriausias gautas penkių rūšių darbų ir penkių mašinų uždavinio tvarkaraštis kai populiacija yra 10



4.2.2 pav. Geriausias gautas penkių rūšių darbų ir penkių mašinų uždavinio tvarkaraštis kai populiacija yra 25

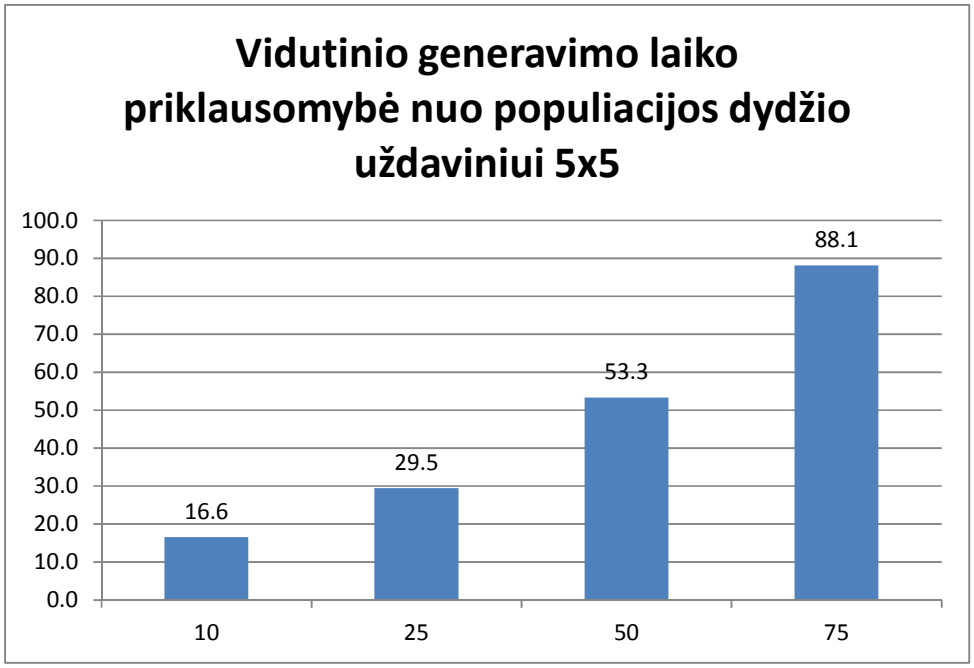


4.2.3 pav. Geriausias gautas penkių rūšių darbų ir penkių mašinų uždavinio tvarkaraštis kai populiacija yra 50



4.2.4 pav. Geriausias gautas penkių rūšių darbų ir penkių mašinų uždavinio tvarkaraštis kai populiacija yra 75

Sprendžiant šį uždavinį buvo fiksuojama kiekvieno generavimo trukmė. Vidutinės generavimo trukmės, sprendžiant šį uždavinį, pateiktos 4.2.5 pav.



4.2.5 pav. Penkių rūšių darbų ir penkių mašinų uždavinio vidutinės generavimo trukmės

Kaip matome iš 4.2.5 pav. sprendžiant penkių rūšių darbų ir penkių mašinų uždavinį, didinant populiacijos dydį, generavimo trukmė kinta eksponentiškai. Lyginant mažiausią bei didžiausią nagrinėtą populiaciją, generavimo trukmė tarp jų skiriasi daugiau nei penki kartus. Taip pat pastebime, kad sprendžiant šį uždavinį naudojant populiaciją lygią 75, nors ir kai kurioms kombinacijoms ir buvo rastas geresnis rezultatas, tačiau bendras geriausias rezultatas nepranoko to, kuris buvo gautas naudojant populiaciją lygią 50. Šiam uždaviniui spręsti pakanka naudoti populiaciją, kurios dydis yra 50, dėl gaunamo tikslumo ir laiko sąnaudų.

4.3. SEPTYNIŲ RŪŠIŲ GAMINIAI ANT SEPTYNIŲ MAŠINŲ

Šiame skyriuje buvo nagrinėjamas uždavinys, kuriame penkių rūšių darbai turėjo būti sudelioti ant penkių mašinų. Uždavinys pateiktas 4.3.1 lentelėje, kurioje kiekvieno darbo trukmė yra pateikta minutėmis.

4.3.1 lentelė. Septynių rūšių darbų ir Septynių mašinų uždavinys.

Gaminio rūšis / Gaminio nr.	1	2	3	4	5	6	7
1	12.6	14	14	25	1.8	6.4	12.6
2	6.4	14	14	4.9	15	14	14
3	2	6	8.1	3	13.8	14	25
4	6	1.8	15	2.3	10	1.8	4.9
5	2	3.3	1.8	15	21	4.9	15
6	15	15	21	24.7	2.3	15	24.7
7	25	25	25	25	25	25	25

Darbai turėjo būti išdeliojami ant mašinų taip, kad kiekvienos rūšies darbai tarpusavyje nepersidengtų, t.y. jei šiuo metu yra vykdomas i – tos rūšies darbas ant j – tos mašinos, tai kitas tos pačios rūšies darbas ant ant k – tos mašinos (kai $k \neq j$) negalės būti uždedamas kol pirmasis darbas nebus pabaigtas. Uždavinys buvo sprendžiamas naudojant visas trijų kryžminimo metodų (ciklinio, tvarkos bei pozicija paremto) ir trijų mutacijos metodų (sukeitimo, perstatymo bei įterpimo) kombinacijas. Uždaviniui spręsti buvo nustatyta kryžinimo tikimybė 0.1, mutacijos tikimybė 0.1, algoritmo žingsnių skaičius 30. Uždavinys buvo spręstas algoritmo populiacijas parenkant 10, 25, 50 ir 75. Algoritmas su kiekviena kombinacija buvo paleistas po 10 kartų ir užfiksuojamas geriausias gautas rezultatas. Gauti rezultatai pateikti 4.3.2 – 4.3.5 lentelėse, kuriose pateiktas geriausias bendras gamybos laikas minutėmis su kiekviena kombinacija.

4.3.2 lentelė. Septynių rūšių darbų ir septynių mašinų uždavinio rezultatai, kai populiacija yra 10

Kryžminimas / Mutacija	Ciklinis	Tvarkos	Pozicija paremtas
Sukeitimo	996	996	977.7
Perstatymo	1028	1012.7	986
Įterpimo	981.7	994.4	940

4.3.3 lentelė. Septynių rūšių darbų ir septynių mašinų uždavinio rezultatai, kai populiacija yra 25

Kryžminimas / Mutacija	Ciklinis	Tvarkos	Pozicija paremtas
Sukeitimo	979	977.6	965.6
Perstatymo	975	975.7	961.5
Įterpimo	974.2	962.2	937.6

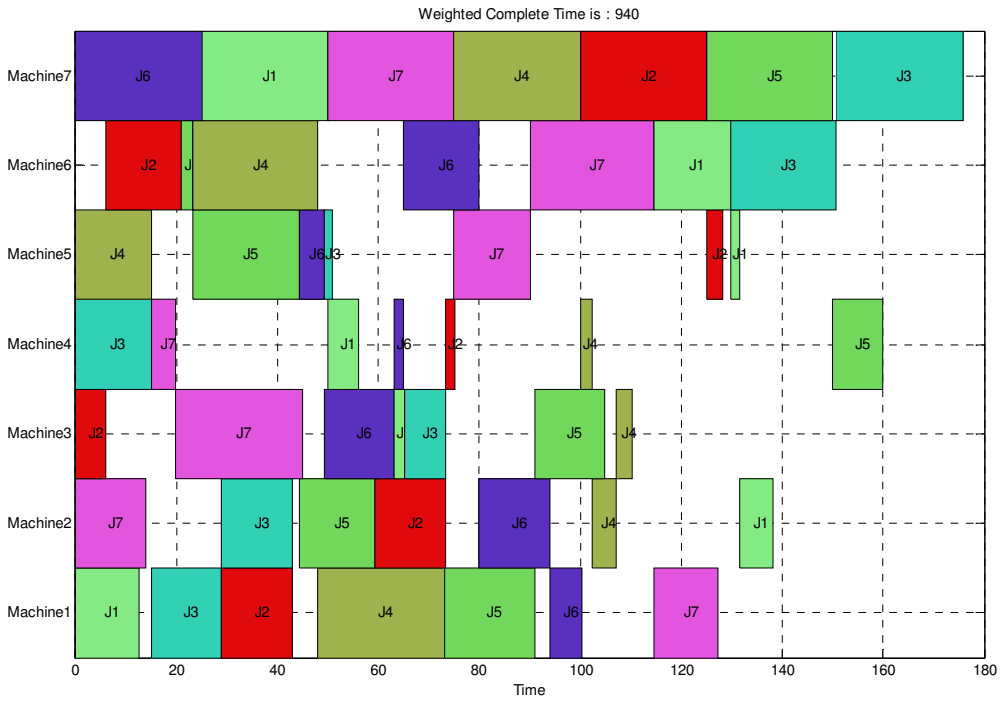
4.3.4 lentelė. Septynių rūšių darbų ir septynių mašinų uždavinio rezultatai, kai populiacija yra 50

Kryžminimas / Mutacija	Ciklinis	Tvarkos	Pozicija paremtas
Sukeitimo	962.3	947.8	943.8
Perstatymo	959.2	940.8	940.4
Įterpimo	948.3	940.2	912.9

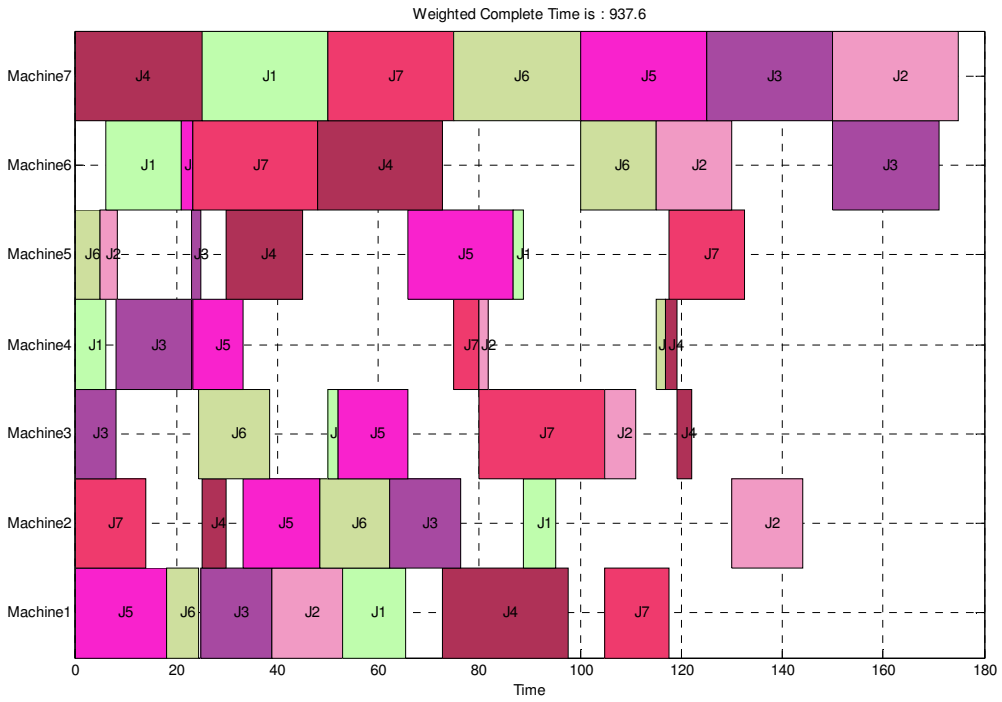
4.3.5 lentelė. Septynių rūšių darbų ir septynių mašinų uždavinio rezultatai, kai populiacija yra 75

Kryžminimas / Mutacija	Ciklinis	Tvarkos	Pozicija paremtas
Sukeitimo	966.1	959.2	939.9
Perstatymo	953.6	940.2	924.6
Įterpimo	948.2	932.7	923.9

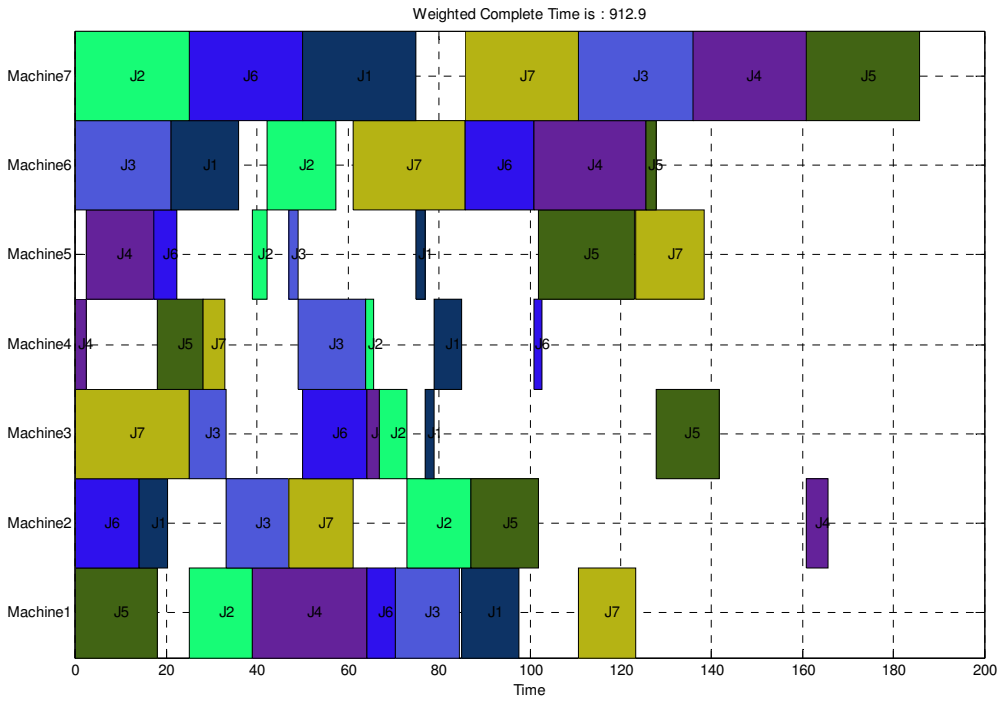
Kaip matome iš 4.3.2 – 4.3.5 lentelių, visi gauti rezultatai gerokai skiriasi vienas nuo kito. Geriausią rezultatą davė kombinacija pozicija paremtas kryžminimas ir įterpimo mutacija. Taip pat pastebime, kad įterpimo mutacija davė geriausius rezultatus prie kiekvieno nagrinėto kryžminimo metodo, bei su visais populiacijų dydžiais Geriausi tvarkaraščiai prie kiekvieno populiacijos dydžio pateikti pateikti 4.3.1 – 4.3.4 pav.



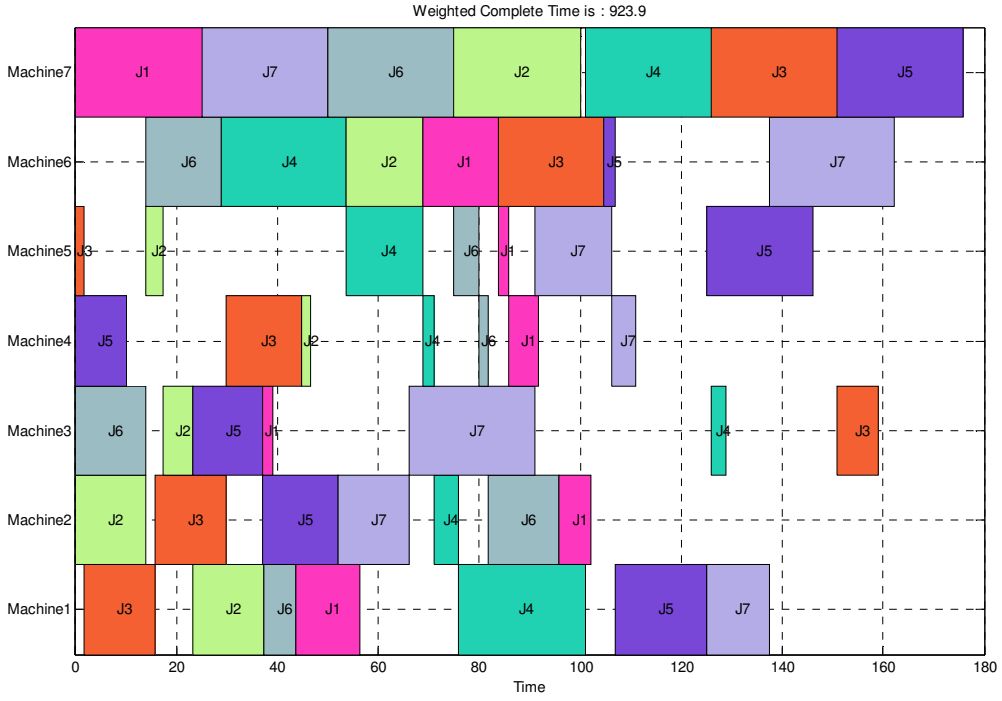
4.3.1 pav. Geriausias gautas septynių rūšių darbų ir septynių mašinų uždavinio tvarkaraštis kai populiacija yra 10



4.3.2 pav. Geriausias gautas septynių rūšių darbų ir septynių mašinų uždavinio tvarkaraštis kai populiacija yra 25

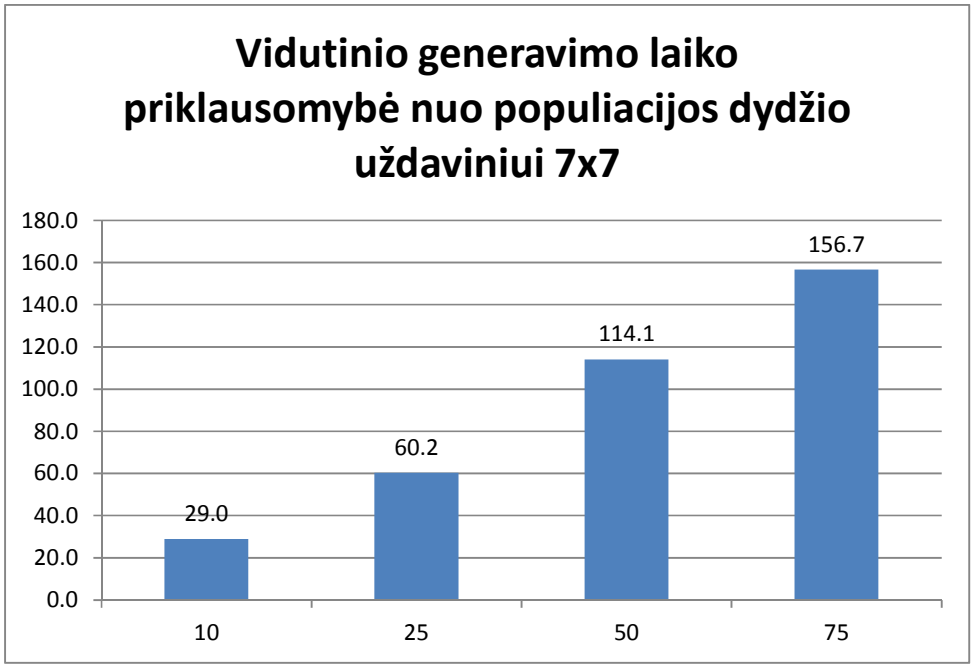


4.3.3 pav. Geriausias gautas septynių rūšių darbų ir septynių mašinų uždavinio tvarkaraštis kai populiacija yra 50



4.3.4 pav. Geriausias gautas septynių rūšių darbų ir septynių mašinų uždavinio tvarkaraštis kai populiacija yra 75

Sprendžiant šį uždavinį buvo fiksuojama kiekvieno generavimo trukmė. Vidutinės generavimo trukmės, sprendžiant šį uždavinį, pateiktos 4.3.5 pav.



4.3.5 pav. Septynių rūšių darbų ir septynių mašinų uždavinio vidutinės generavimo trukmės

Kaip matome iš 4.3.5 pav. sprendžiant septynių rūšių darbų ir septynių mašinų uždavinį, didinant populiacijos dydį, generavimo trukmė kinta tiesiškai. Lyginant mažiausią bei didžiausią nagrinėtą populiaciją, generavimo trukmė tarp jų skiriasi daugiau nei penki kartus. Taip pat pastebime, kad sprendžiant šį uždavinį naudojant populiaciją lygią 75, nors ir kai kurioms kombinacijoms ir buvo rastas geresnis rezultatas, tačiau bendras geriausias rezultatas nepranoko to, kuris buvo gautas naudojant populiaciją lygią 50. Šiam uždaviniui spręsti pakanka naudoti populiaciją, kurios dydis yra 50, dėl gaunamo tikslumo ir laiko sąnaudų.

DISKUSIJA

Norint rasti kiek įmanoma geresnį gamybos tvarkaraštį, trijų skirtingų sunkumų uždaviniams spręsti buvo naudojamos trijų rūšių mutacijos (sukeitimo, perstatymo, įterpimo) bei trijų rūšių kryžminimai (ciklinis, tvarkos, pozicija paremtas). Su kiekviena mutacijos ir kryžminimo pora buvo atliekama po dešimt generavimų, ko pasekoje užfiksuojamas geriausias rezultatas. Kiekvienas uždavinys buvo sprendžiamas naudojant keturių dydžių.

Sprendžiant uždavinį su trimis darbų rūšimis ir trimis mašinomis, gauti rezultatai sutapo naudojant visus populiacijų dydžius. Kadangi didėjant populiacijos dydžiui, auga ir generavimo laikas, tai šiam uždaviniui užtenka naudoti populiacijos dydį lygų 10. Šiam uždaviniui spręsti visos mutacijų bei kryžminimo poros ya lygiavertės.

Sprendžiant uždavinį su penkiomis darbų rūšimis ir penkiomis mašinomis visos kombinacijos davė skitingus rezultatus. Geriausiai su šiuo uždaviniu susidorojo pozicija paremto kryžminimo ir įterpimo mutacijos pora. Nors ir tikėtasi, kad naudojant didžiausią populiaciją bus gautas geriausias rezultatas, tačiau geresnio rezultato, nei gauto su populiacija 50 gauti nepavyko.

Sprendžiant uždavinį su septyniomis darbų rūšimis ir septyniomis mašinomis visos kombinacijos davė skirtingus rezultatus. Kaip ir sprendžiant uždavinį su penkiomis darbų rūšimis ir penkiomis mašinomis, geriausią rezultatą parodė pozicija paremto kryžminimo ir įterpimo mutacijos pora. Taip pat, kaip ir pries tai spręstame uždavinyje, naudojant dydžiausią popuiaciją nebuvo gauti patys geriausi rezultatai.

Pagal rezultatus galima teigti, kad šiam uždaviniui spręsti labiausiai tinka pozicija paremto kryžminimo ir įterpimo mutacijos pora. Optimalus populiacijos dydis yra 50, nes populiacija lygi 75 nedavė geresnių rezultatų ir užtruko daug ilgiau atliekant skaičiavimus.

SPRENDIMO POBŪDŽIO LYGINIMAS SU PANAŠAUS TIPO MOKSLINIAIS DARBAIS

Šiame darbe nagrinėtas uždavinys buvo sprendžiamas taikant devynias mutacijų bei kryžminimų poras, naudojant keturis populiacijų dydžius. Darbo tikslas buvo rasti geriausią gamybos tvarkaraštį minimizuojant bendrą mašinų darbo laiką.

A.Ranjini, B.S.E.Zoraida savo darbe [8] tyrė kaip kinta GA rezultatai, kai yra taikomi skirtingi kryžminimo metodai. Jų darbe geriausius rezultatus davė nerūšiuotas pasekmės sukeitimo kryžminimo metodas (angl. *unordered subsequence exchange*). Šis kryžminimo metodas yra labai panašus į metodaą, naudotą šiame darbe – tvarkos. Nors šie metodai panašūs, tačiau šiame magistro darbe tvarkos kryžminimo metodas nedavė pačių geriausių rezultatų.

Liang Sun, Xiaochun Cheng, Yanchun Liang [9] uždavinio sprendimui panaudojo baudos (angl. *penalty*) funkciją bei kloninę (angl. *clonal*) atranką. Ši atranka yra paremta B limfocitų (angl. *B lymphocytes*) sugebėjimą atpažinti antigenus su tam tikru stiprumu. Šiame magistro darbe buvo naudota ruletės rato principu paremta atranka.

Mahanim Omar, Adam Baharum, Yahya Abu Hasan [10] gamyklos tvarkaraščio sudarymo uždaviniui spręsti sugalvojo panaudoti atstumus (angl. *distance*) tarp tėviškųjų chromosomų kryžminimo bei mutacijų metoduose ir kaimynystės (angl. *neighbourhood*) savoką. Atstumai buvo naudojami atrankai, kur atstumas parodydavo chromosomos tinkamumą. Šis atrankos metodas yra panašus į ruletės rato principu paremta atranką.

Yra daugybė būdų kaip spręsti gamybos tvarkaraščio sudarymo uždavinį naudojant genetinius algoritmus, tačiau universalaus sprendimo būdo, kuris tiktų visokio pobūdžio uždaviniams, dar nėra.

IŠVADOS

Šiame darbe buvo nagrinėjamas gamybos tvarkaraščio sudarymo uždavinys naudojant genetinius algoritmus. Buvo tiriama kaip pagrindinės genetinių algoritmų dalys, t.y. kryžminimas bei mutacija įtakoja uždavinio rezultatus. Tam patikrinti buvo naudojami trys kryžminimo metodai (ciklinis, tvarkos bei pozicija paremtas) ir trys mutacijos metodai (sukeitimo, perstatymo bei įterpimo). Taip pat darbe buvo naudojami skirtingi populiacijų dydžiai. Buvo stebimi rezultatai, gauti su visomis kryžminimų bei mutacijų kombinacijomis, bei buvo fiksuojamas generavimo laikas. Tam, kad patikrinti kaip šios kombinacijos reaguoja į uždavinio sunkumą, buvo naudojami trijų skirtingų sunkumų uždaviniai.

- Sprendžiant uždavinį su trimis darbų rūšimis ir trimis mašinomis visos kombinacijos gavo tokį pat bendrą gamybos laiką – 102.2, su visais populiacijų dydžiai. Iš šio rezultato galima teigti, kad esant pakankamai lengvam uždaviniui, visos kryžminimų ir mutacijų kombinacijos su uždaviniu susidoroja vienodai, ko pasekoje galima naudoti mažiausią populiaciją, tam, kad sutaupyti laiko.
- Sprendžiant uždavinį su penkiomis darbų rūšimis ir penkiomis mašinomis visos kombinacijos davė skitingus rezultatus. Geriausią rezultatą davė kombinacija pozicija paremtas kryžminimas ir įterpimo mutacija – 298.6, kai buvo naudojama populiacija lygi 50. Taip pat buvo pastebėta, kad įterpimo mutacija davė geriausius rezultatus su kiekvienu nagrinėtu kryžminimo metodu. Blogiausius rezultatus davė sukeitimo mutacija su visomis trimis kryžminimo kombinacijomis, su visais populiacijų dydžiais.
- Sprendžiant uždavinį su septyniomis darbų rūšimis ir septyniomis mašinomis visos kombinacijos davė skirtingus rezultatus. Geriausią rezultatą davė kombinacija pozicija paremtas kryžminimas ir įterpimo mutacija – 912.9, kai buvo naudojama populiacija lygi 50. Taip pat buvo pastebėta, kad įterpimo mutacija davė geriausius rezultatus su kiekvienu nagrinėtu kryžminimo metodu. Blogiausius rezultatus davė sukeitimo mutacija su visomis trimis kryžminimo kombinacijomis.

Iš šio tyrimo galime daryti išvadą, kad didėjant uždavinio sunkumui tam tikros kryžminimo bei mutacijų kombinacijos duoda prastus rezultatus. Geriausiai su uždavinio sunkumo didėjimu susidorojo kombinacija pozicija paremtas kryžminimas ir įterpimo mutacija. Naudojant šią porą, kai buvo sprendžiama visų trijų sunkumų uždaviniai, naudojant visus keturis populiacijų dydžius, buvo gauti geriausi rezultatai. Optimaliausias populiacijos dydis buvo 50, kuris pranoko mažesnes populiacijas, bei davė geresnius rezultatus nei didesnė (75) populiacija.

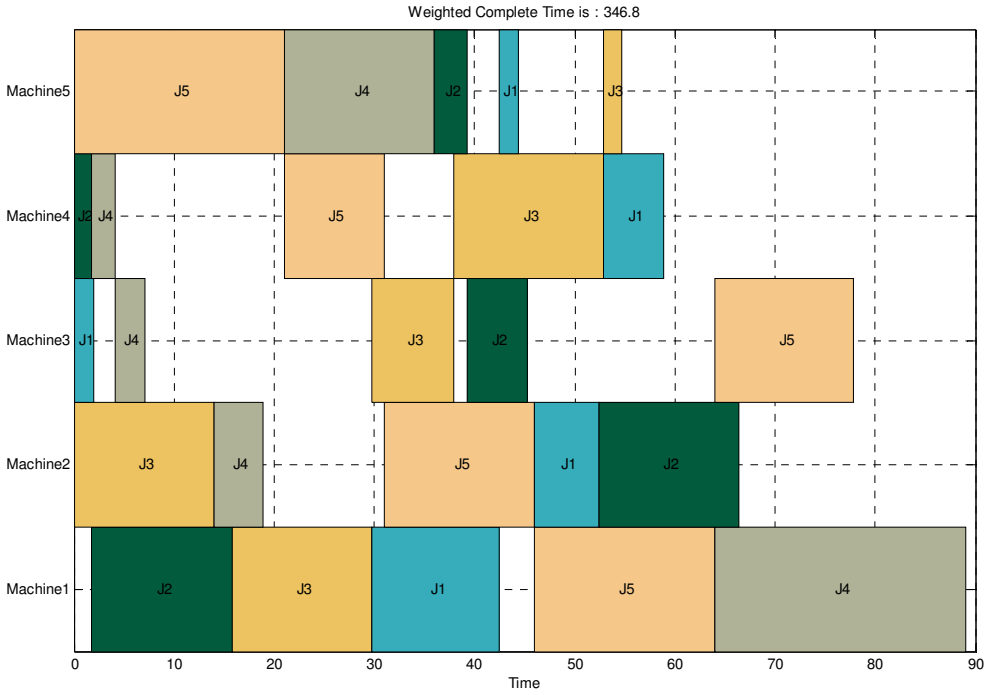
LITERATŪROS SĄRAŠAS

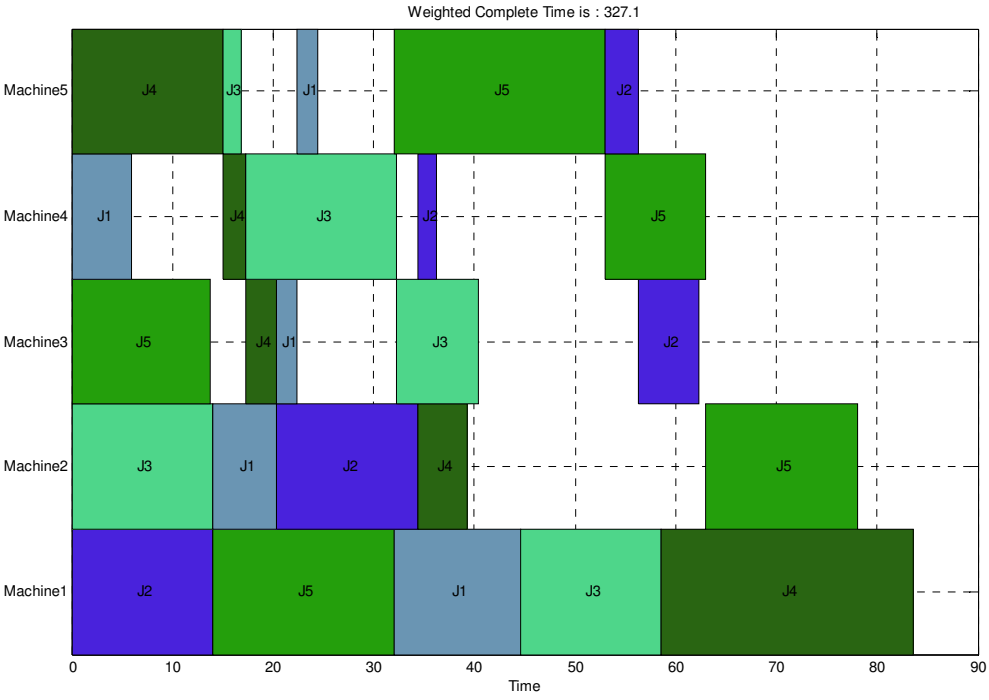
1. Jorge Puente, Helio R. Díez, Ramiro Varela, Camino R. Vela, Luis P. Hidalgo: „Heuristic Rules and Genetic Algorithms for Open Shop Scheduling Problem“.
2. Pierre Lopez , François Roubellat: „Open Shop Scheduling“.
3. Melanie Mitchell, „An Introduction to Genetic Algorithms“.
4. David E. Goldberg: „Genetic Algorithms“.
5. Uday K. Chakraborty: „Computational Intelligence in Flow Shop and Job Shop Scheduling“.
6. Edward Grady Coffman, John L. Bruno „Computer and job-shop scheduling theory“.
7. Michael L. Pinedo: „Planning and Scheduling in Manufacturing and Services“.
8. A.Ranjini, B.S.E.Zoraida: „Analysis of selection schemes for solving job shop scheduling problem using genetic algorithm“.
9. Liang Sun, Xiaochun Cheng, Yanchun Liang: „Solving Job Shop Scheduling Problem Using Genetic Algorithm with Penalty Function“.
10. Mahanim Omar, Adam Baharum, Yahya Abu Hasan: „A job-shop scheduling problem (JSSP) using genetic algorithm (GA)“.
11. http://www.cs.ubc.ca/labs/beta/Courses/CPSC532D-05/Slides/scheduling-chris.pdf
12. http://perso.ens-lyon.fr/frederic.vivien/EPIT/Slides/Schwiegelshohn.pdf
13. http://se.mathworks.com/discovery/genetic-algorithm.html
14. http://lancet.mit.edu/mbwall/presentations/IntroToGAs/

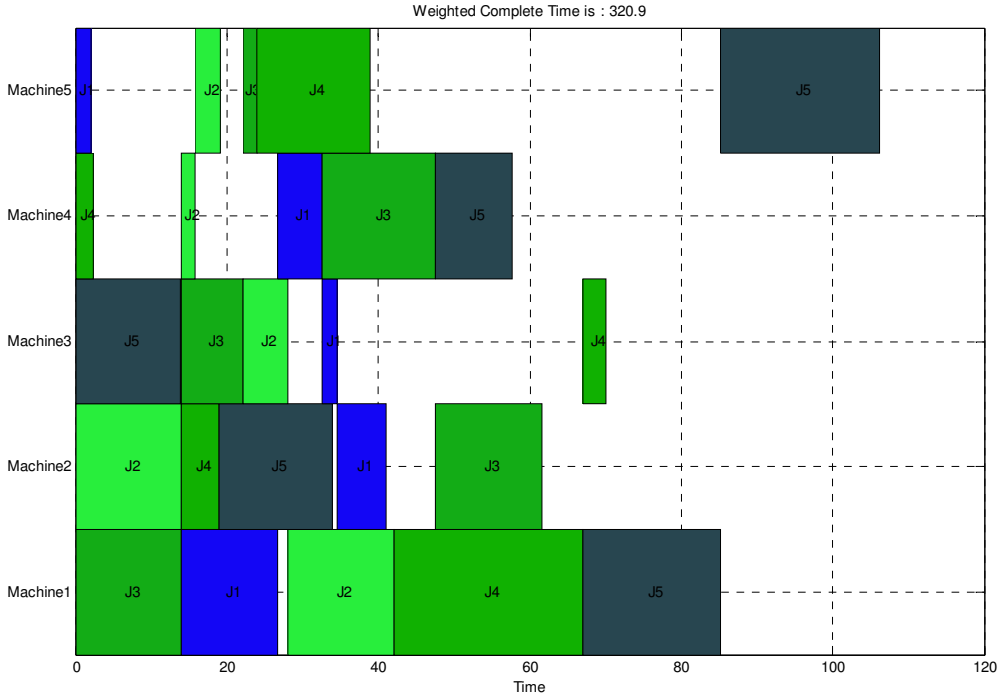
PRIEDAI

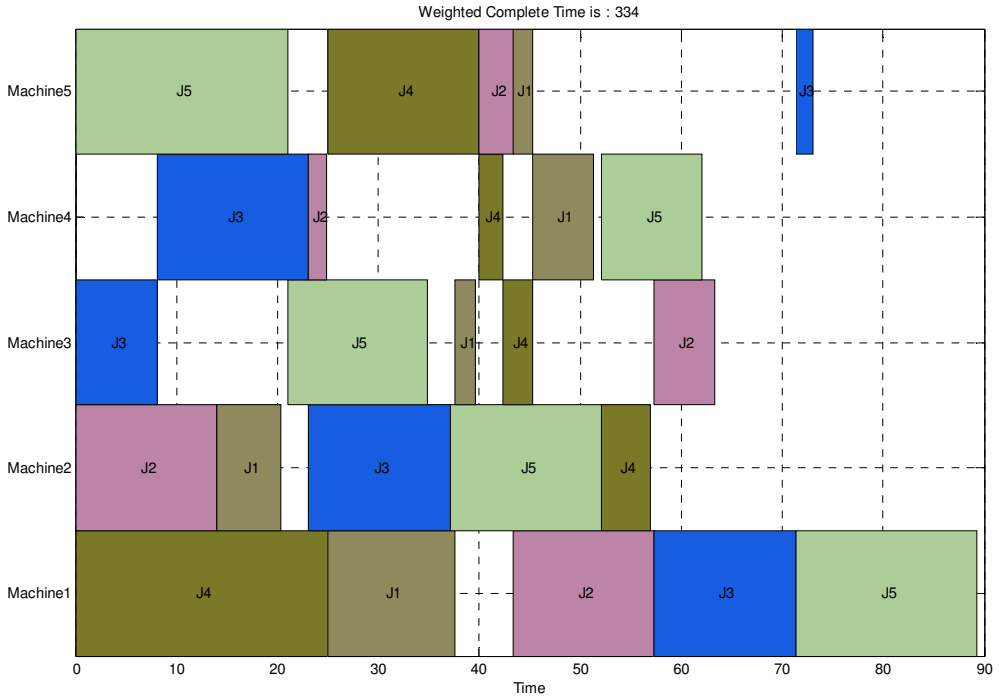
Tvarkaraščiai, gauti nagrinėjant uždavinį su penkiomis darbų rūšimis ir penkiomis mašinomis, naudojant:

Sukeitimo mutaciją ir ciklinį kryžminimą (kai populiacija yra 10, 25, 50, 75)

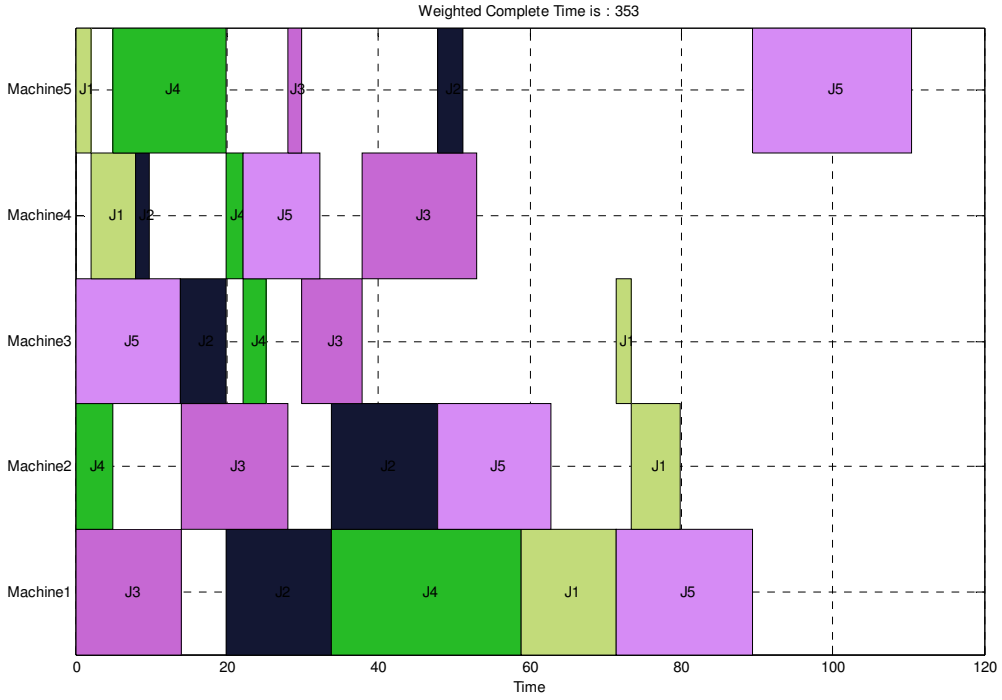


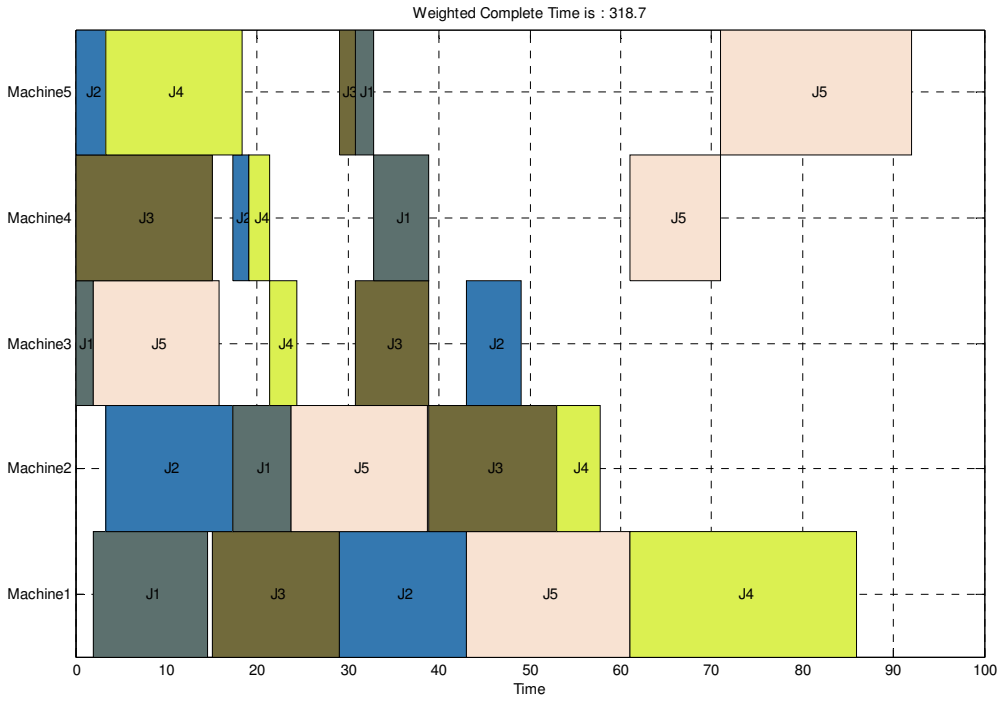


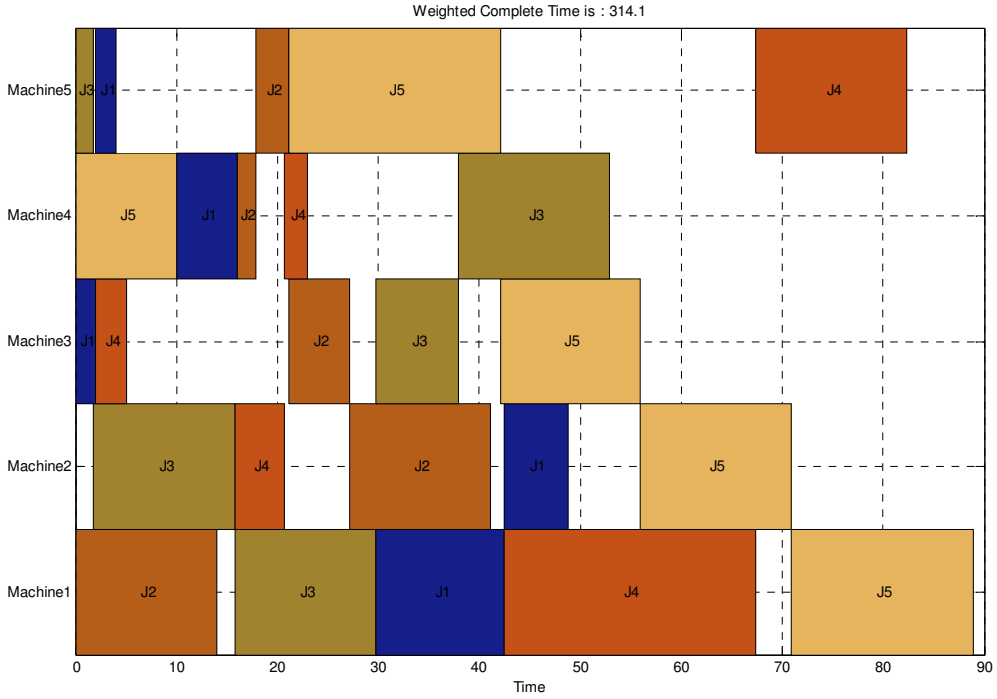

Weighted Complete Time is : 320.9
Machine5
Machine4
Machine3
Machine2
Machine1
J1
J2
J3
J4
J5
0
20
40
60
80
100
120
Time

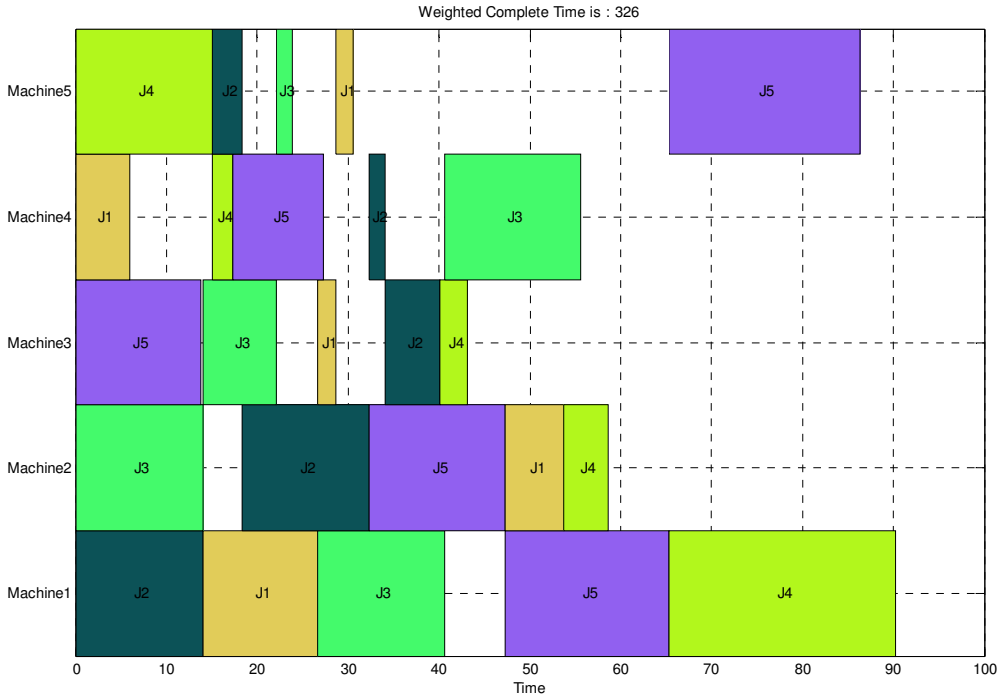

Weighted Complete Time is : 334
Machine5
Machine4
Machine3
Machine2
Machine1
J1
J2
J3
J4
J5
0
10
20
30
40
50
60
70
80
90
Time

Sukeitimo mutaciją ir tvarkos kryžminimą (kai populiacija yra 10, 25, 50, 75)

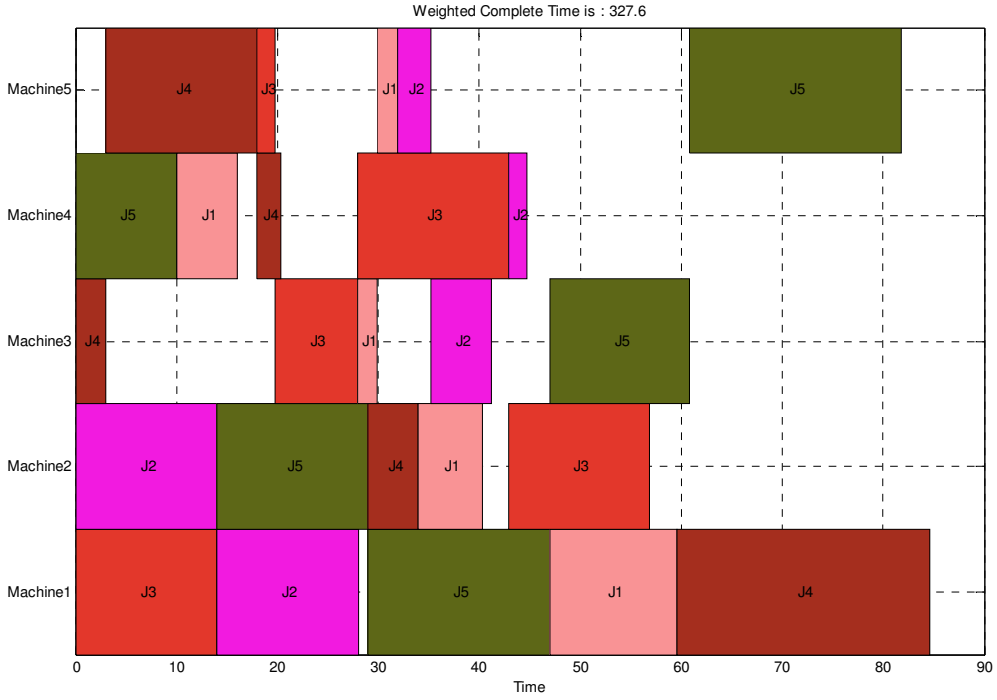


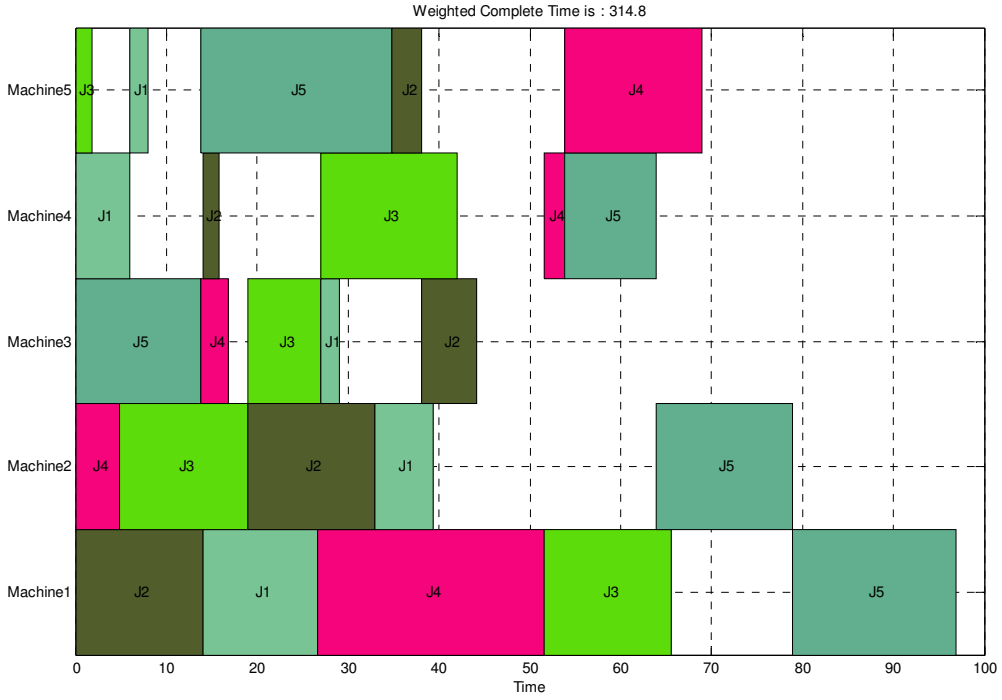


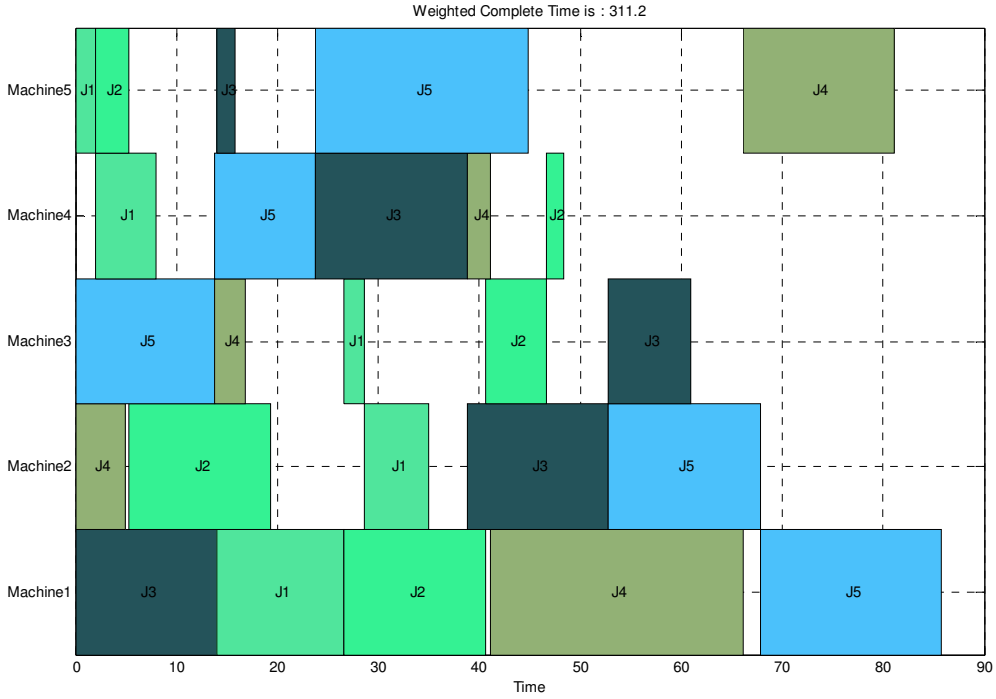

Weighted Complete Time is : 314.1
Machine5
Machine4
Machine3
Machine2
Machine1
J3
J1
J2
J5
J4
J5
J1
J2
J4
J3
J1
J4
J2
J3
J5
J3
J4
J2
J1
J5
J2
J3
J1
J4
J5
0
10
20
30
40
50
60
70
80
90
Time

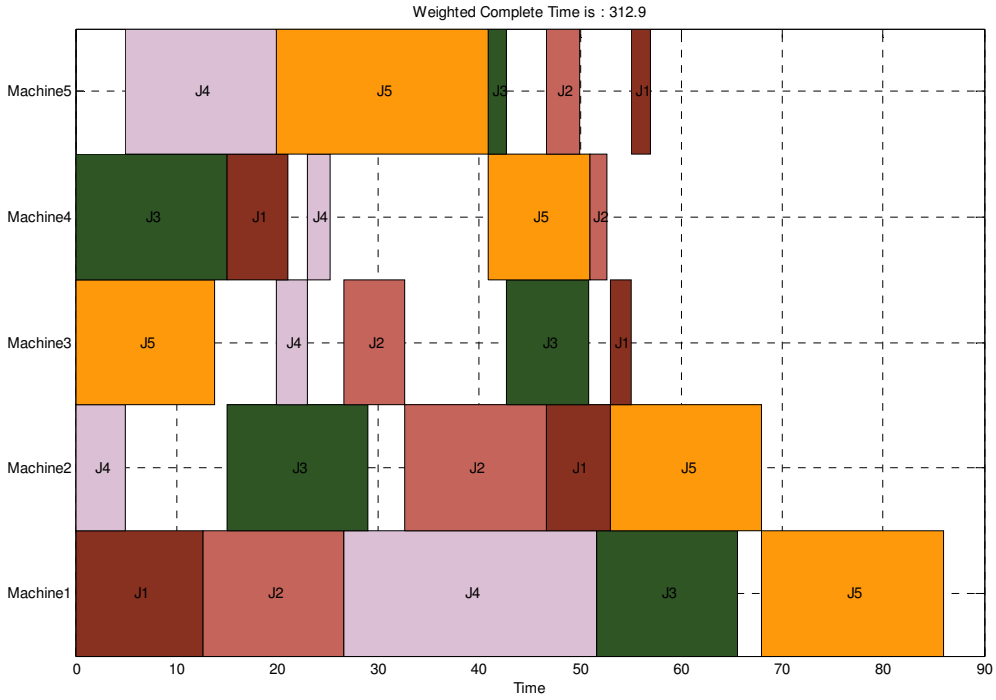

Weighted Complete Time is : 326
Machine5
Machine4
Machine3
Machine2
Machine1
J4
J2
J3
J1
J5
J1
J4
J5
J2
J3
J5
J3
J1
J2
J4
J3
J2
J5
J1
J4
J2
J1
J3
J5
J4
0
10
20
30
40
50
60
70
80
90
100
Time

Sukeitimo mutaciją ir pozicija paremtą kryžminimą (kai populiacija yra 10, 25, 50, 75)

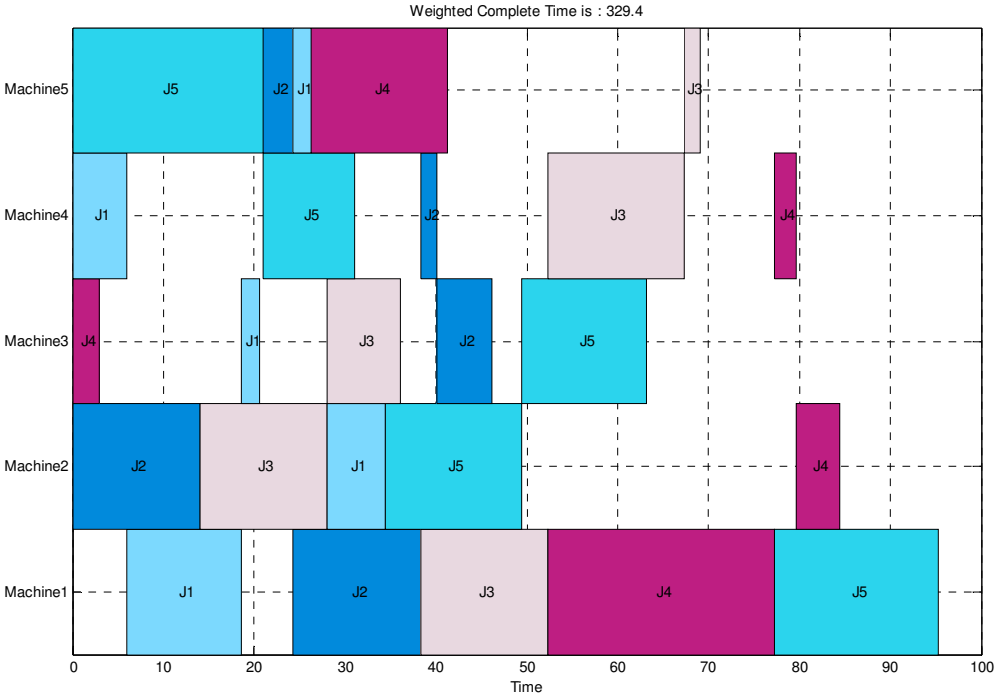


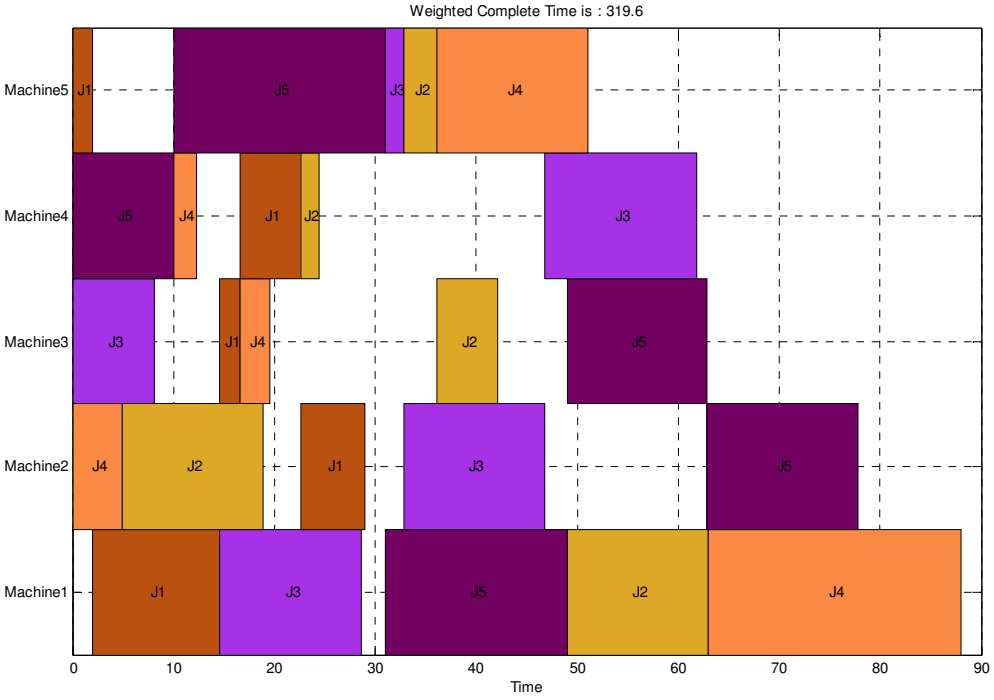


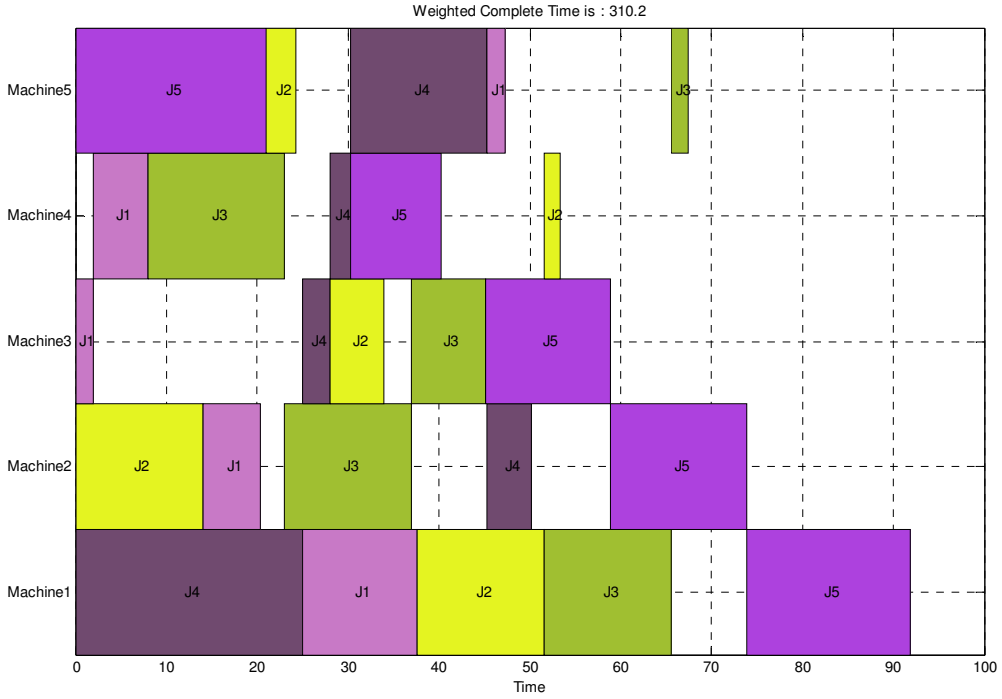

Weighted Complete Time is : 311.2
Machine5
Machine4
Machine3
Machine2
Machine1
J1
J2
J3
J4
J5
0
10
20
30
40
50
60
70
80
90
Time

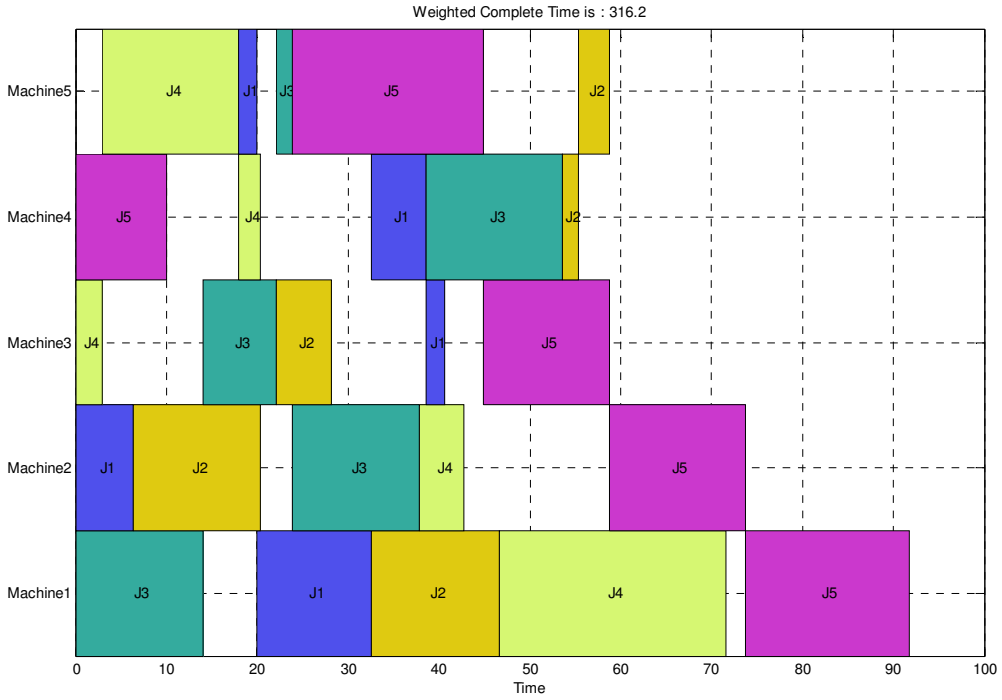

Weighted Complete Time is : 312.9
Machine5
Machine4
Machine3
Machine2
Machine1
J1
J2
J3
J4
J5
0
10
20
30
40
50
60
70
80
90
Time

Perstatymo mutaciją ir ciklinį kryžminimą (kai populiacija yra 10, 25, 50, 75)

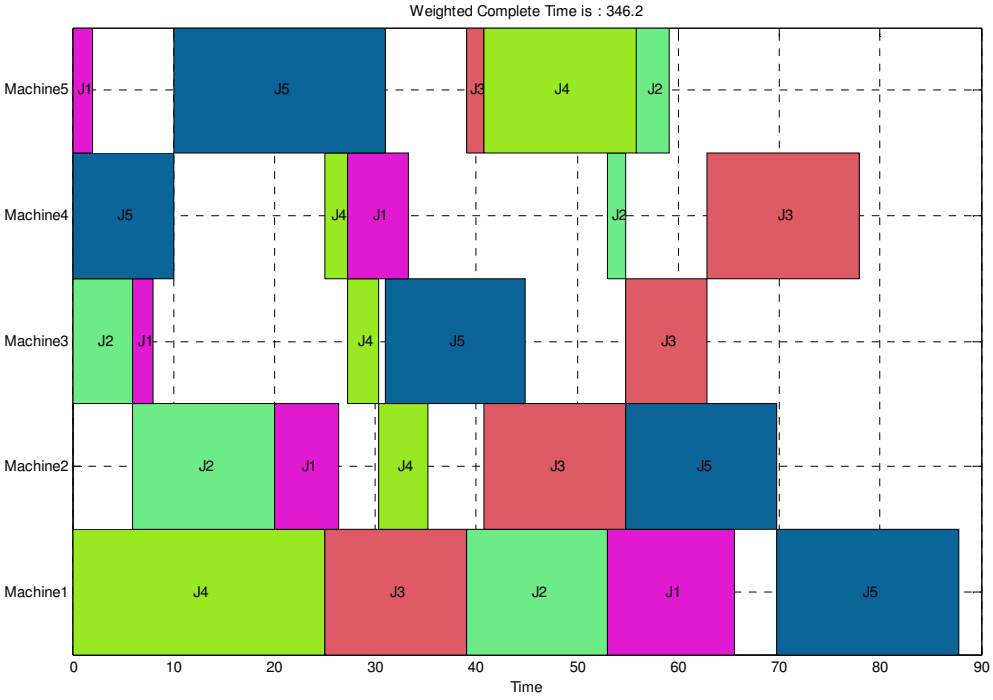


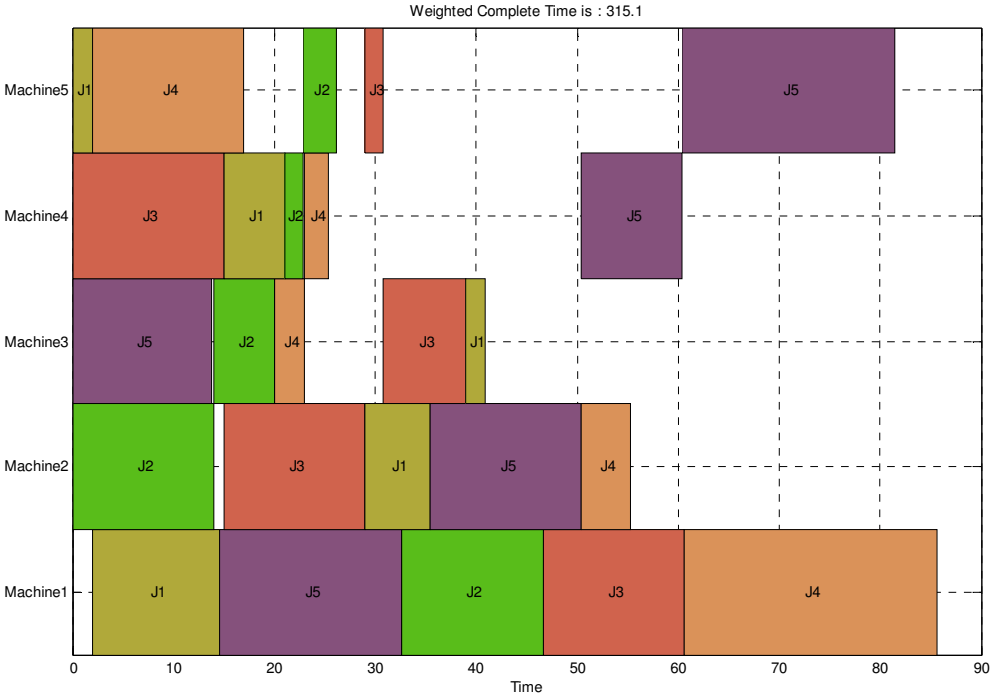


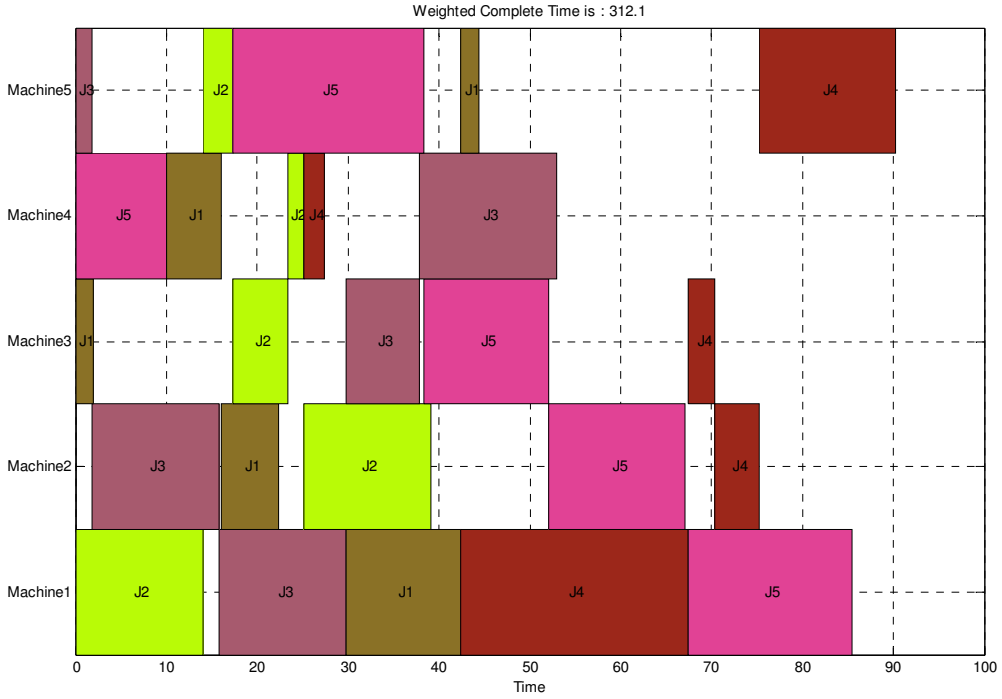

Weighted Complete Time is : 310.2
Machine5
Machine4
Machine3
Machine2
Machine1
J5
J2
J4
J1
J3
J1
J3
J4
J5
J2
J1
J4
J2
J3
J5
J2
J1
J3
J4
J5
J4
J1
J2
J3
J5
0
10
20
30
40
50
60
70
80
90
100
Time

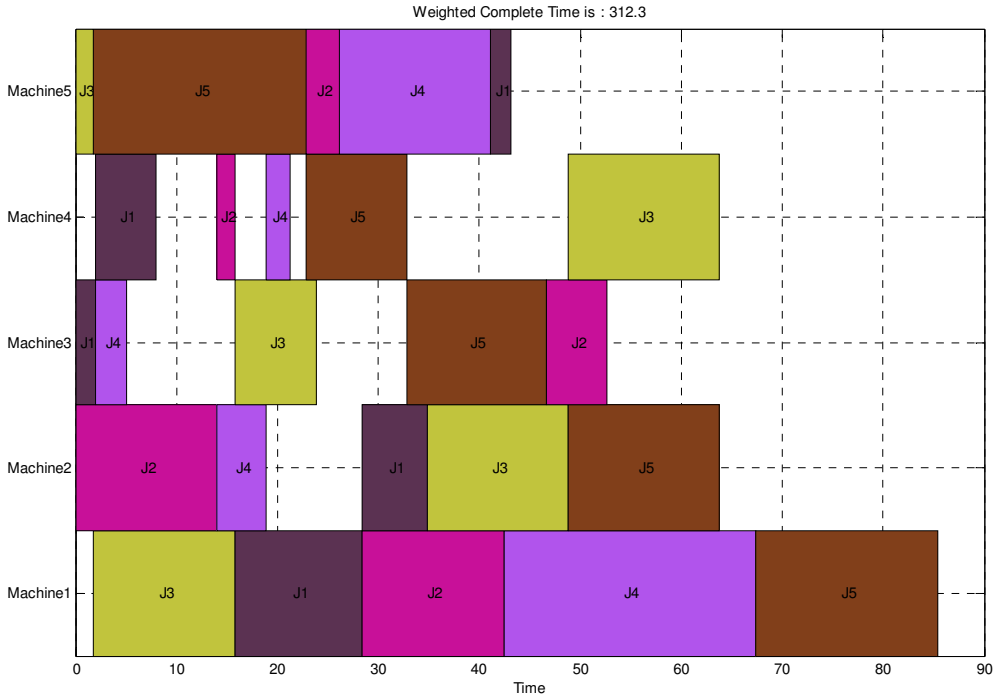

Weighted Complete Time is : 316.2
Machine5
Machine4
Machine3
Machine2
Machine1
J4
J1
J3
J5
J2
J5
J4
J1
J3
J2
J4
J3
J2
J1
J5
J1
J2
J3
J4
J5
J3
J1
J2
J4
J5
0
10
20
30
40
50
60
70
80
90
100
Time

Perstatymo mutaciją ir tvarkos kryžminimą (kai populiacija yra 10, 25, 50, 75)

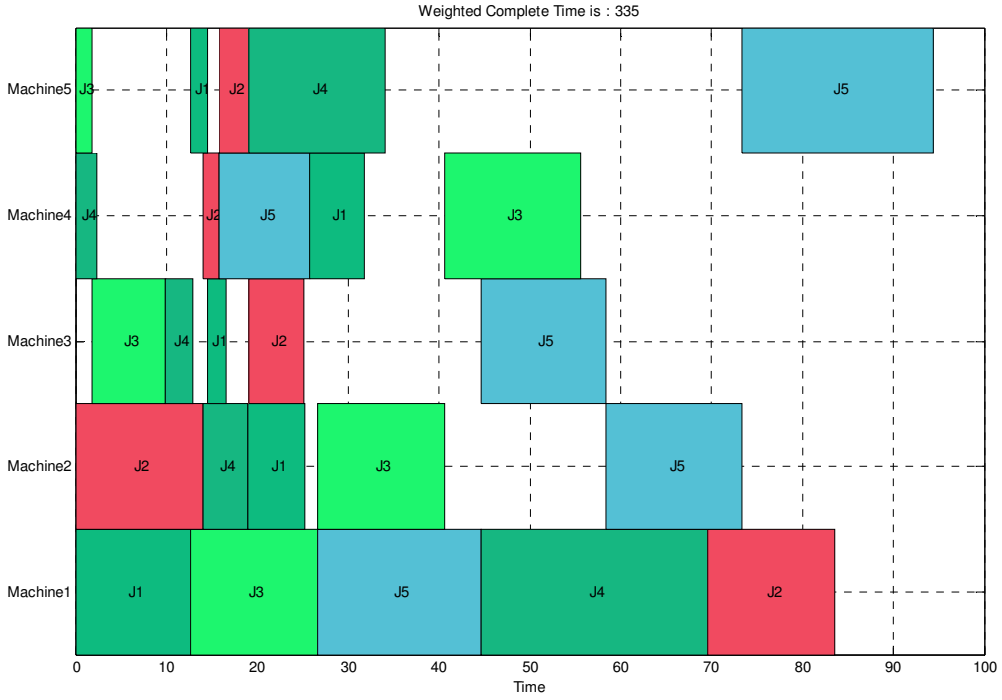


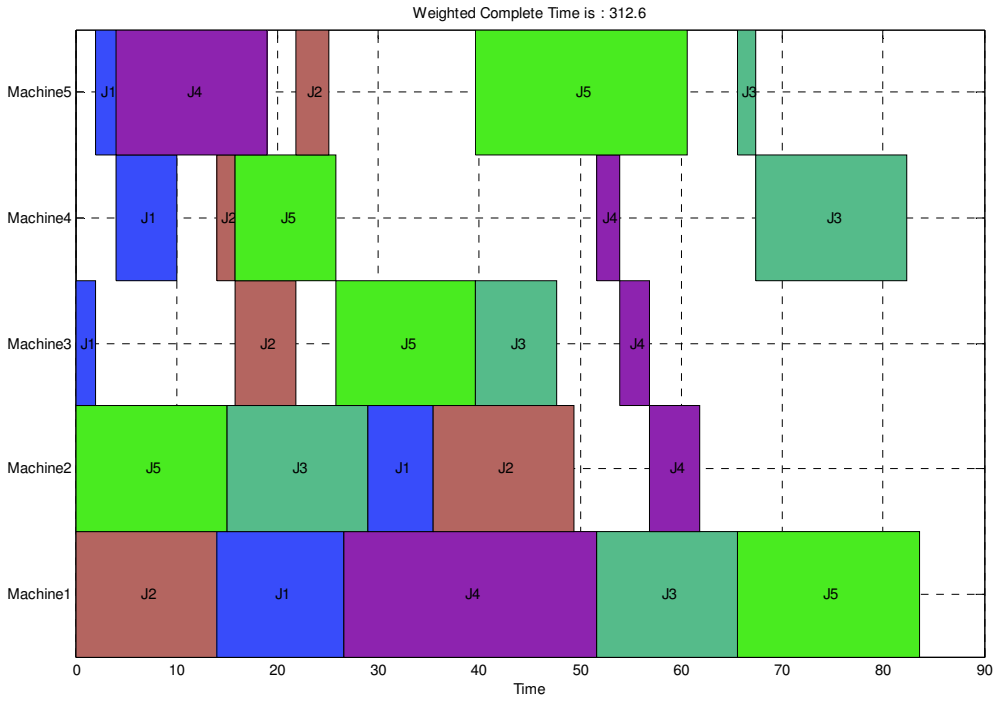


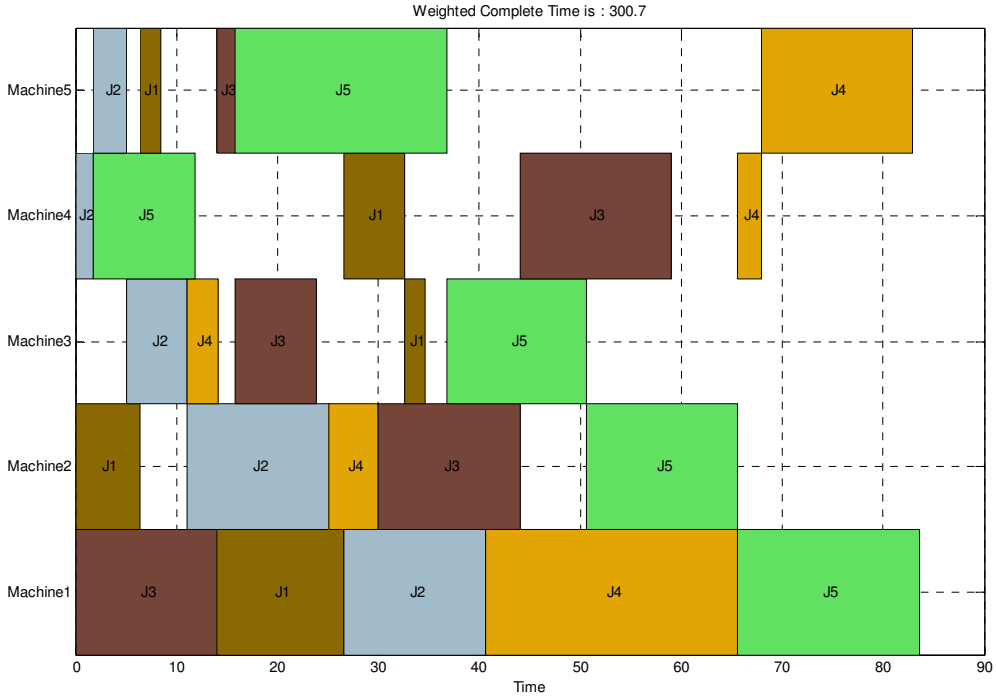

Weighted Complete Time is : 312.1
Machine5
Machine4
Machine3
Machine2
Machine1
J3
J2
J5
J1
J4
J5
J1
J2
J4
J3
J1
J2
J3
J5
J4
J3
J1
J2
J5
J4
J2
J3
J1
J4
J5
0
10
20
30
40
50
60
70
80
90
100
Time

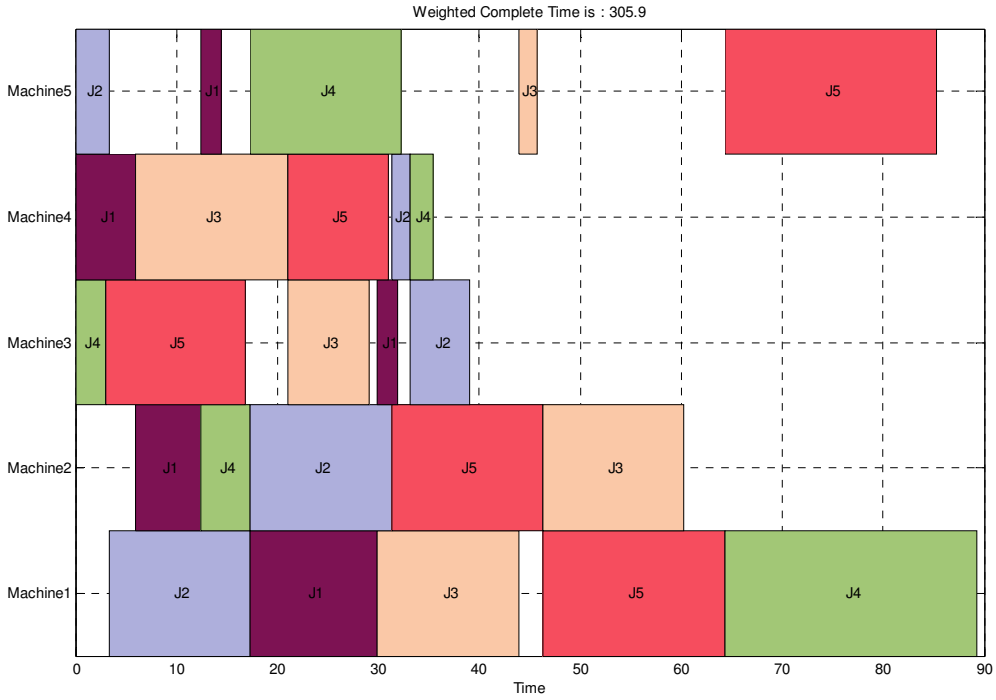

Weighted Complete Time is : 312.3
Machine5
Machine4
Machine3
Machine2
Machine1
J3
J5
J2
J4
J1
J1
J2
J4
J5
J3
J1
J4
J3
J5
J2
J2
J4
J1
J3
J5
J3
J1
J2
J4
J5
0
10
20
30
40
50
60
70
80
90
Time

Perstatymo mutaciją ir pozicija paremtą kryžminimą (kai populiacija yra 10, 25, 50, 75)

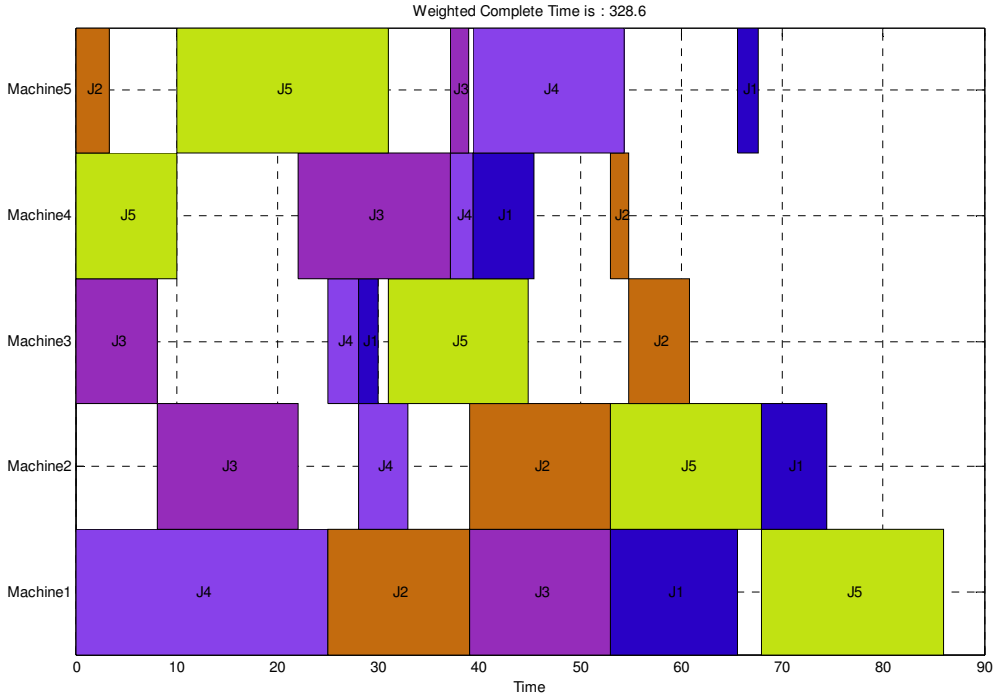


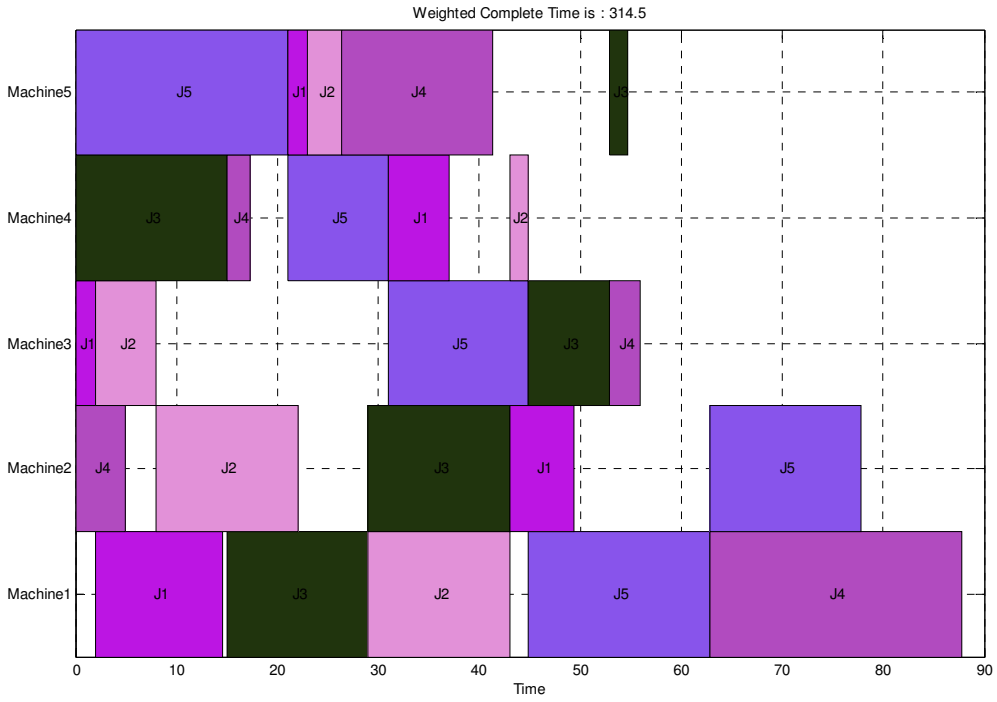


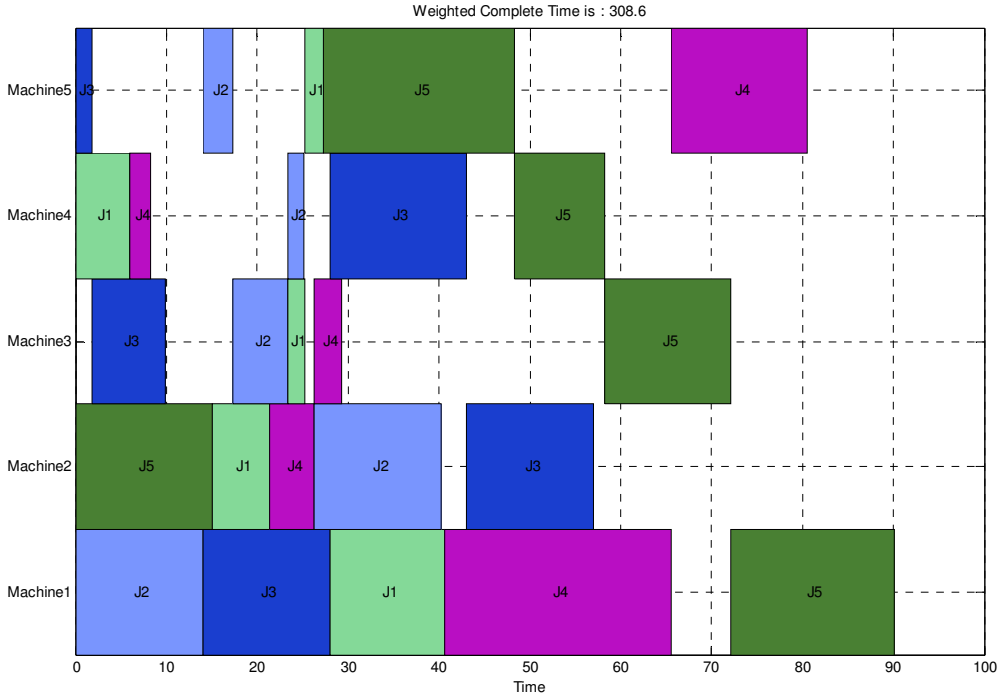

Weighted Complete Time is : 300.7
Machine5
Machine4
Machine3
Machine2
Machine1
J2
J1
J3
J5
J4
0
10
20
30
40
50
60
70
80
90
Time

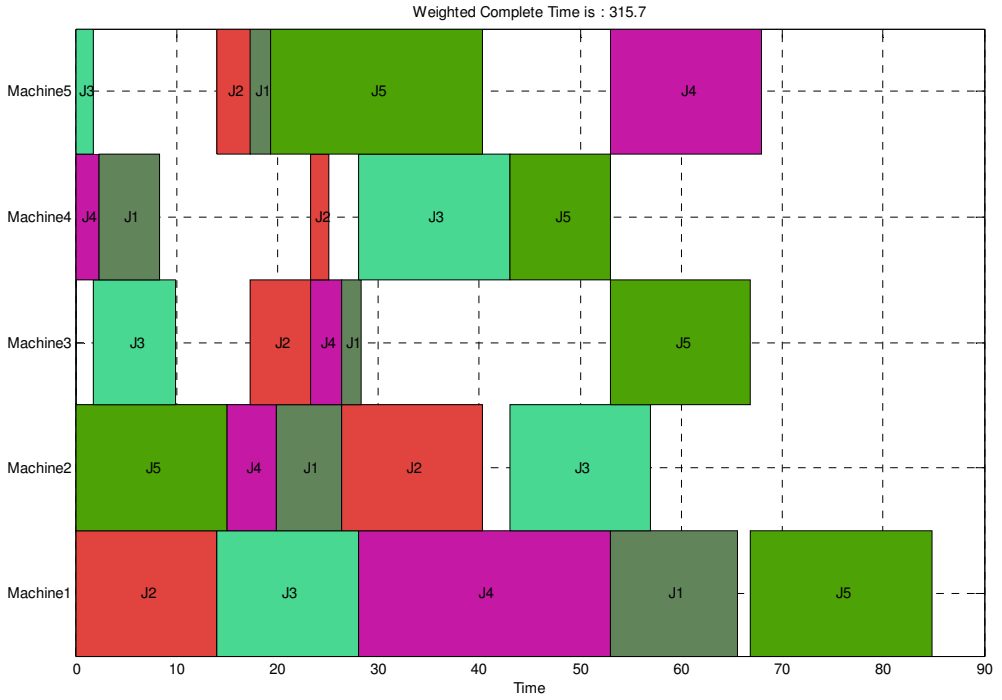

Weighted Complete Time is : 305.9
Machine5
Machine4
Machine3
Machine2
Machine1
J2
J1
J4
J3
J5
0
10
20
30
40
50
60
70
80
90
Time

Įterpimo mutaciją ir ciklinį kryžminimą (kai populiacija yra 10, 25, 50, 75)

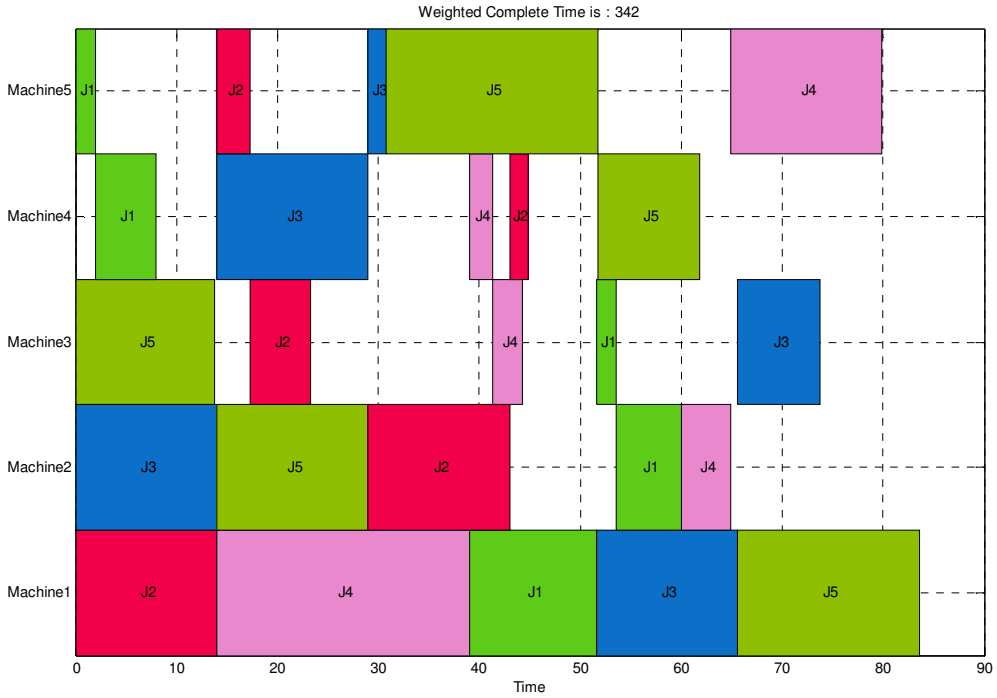


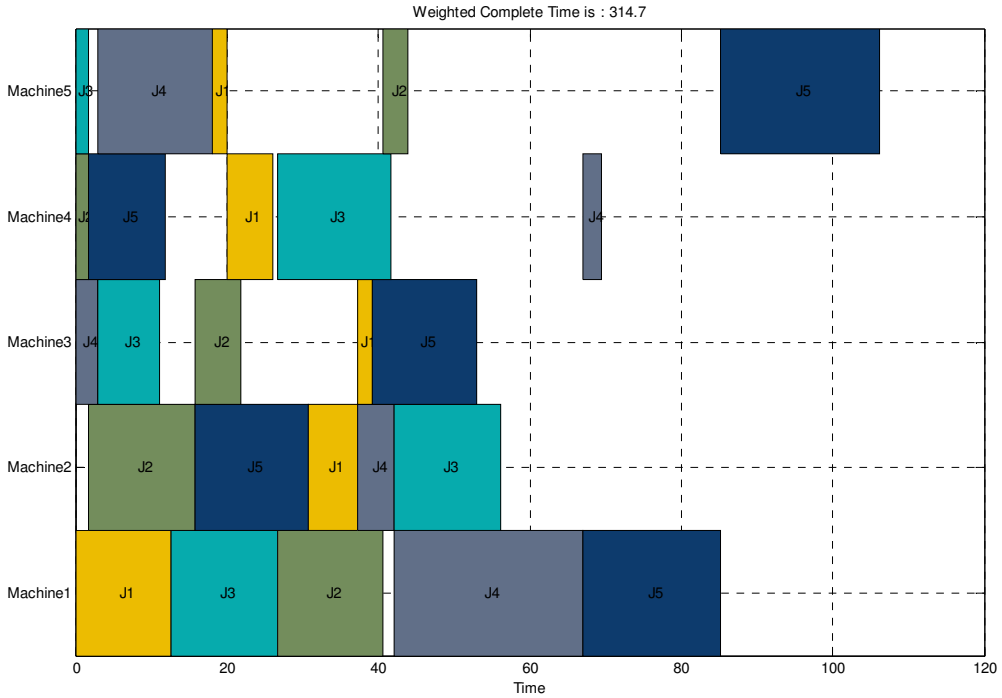


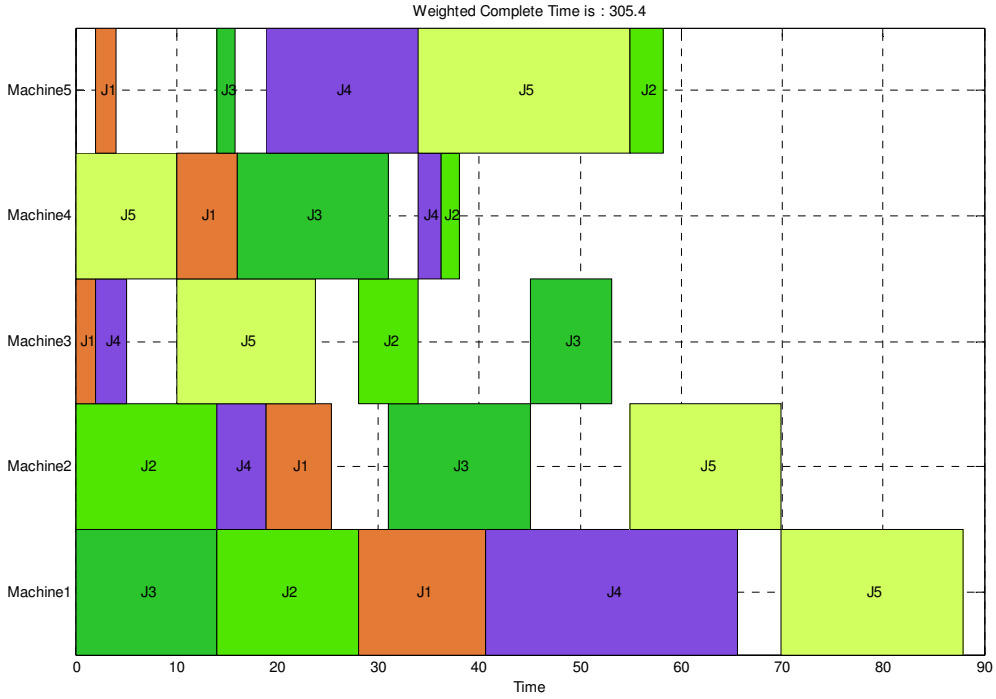

Weighted Complete Time is : 308.6
Machine5
Machine4
Machine3
Machine2
Machine1
J1
J2
J3
J4
J5
0
10
20
30
40
50
60
70
80
90
100
Time

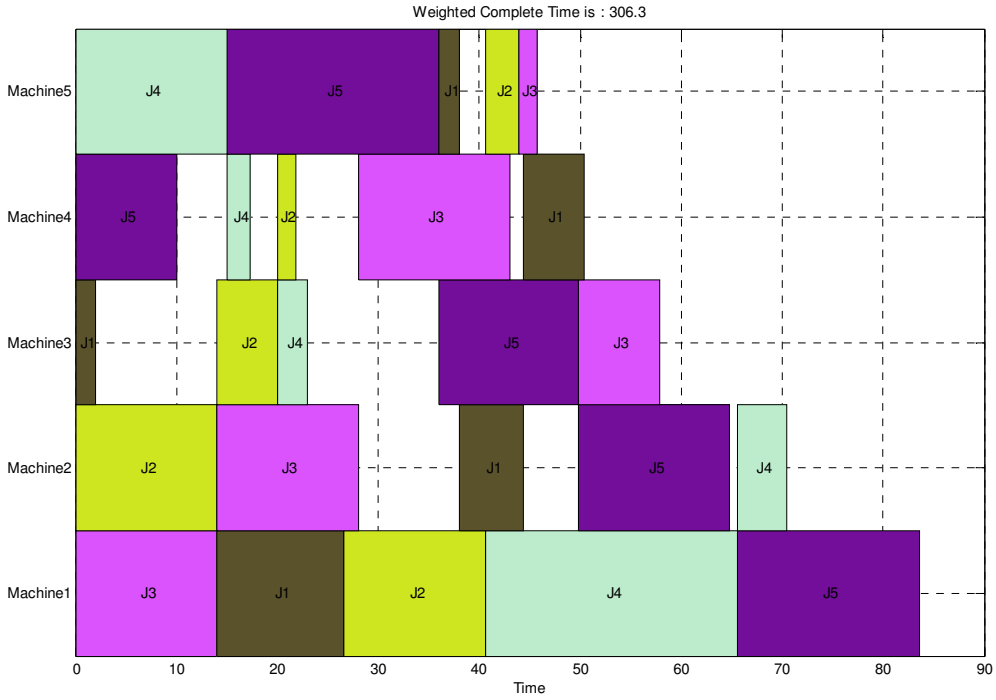

Weighted Complete Time is : 315.7
Machine5
Machine4
Machine3
Machine2
Machine1
J1
J2
J3
J4
J5
0
10
20
30
40
50
60
70
80
90
Time

Įterpimo mutaciją ir tvarkos kryžminimą (kai populiacija yra 10, 25, 50, 75)

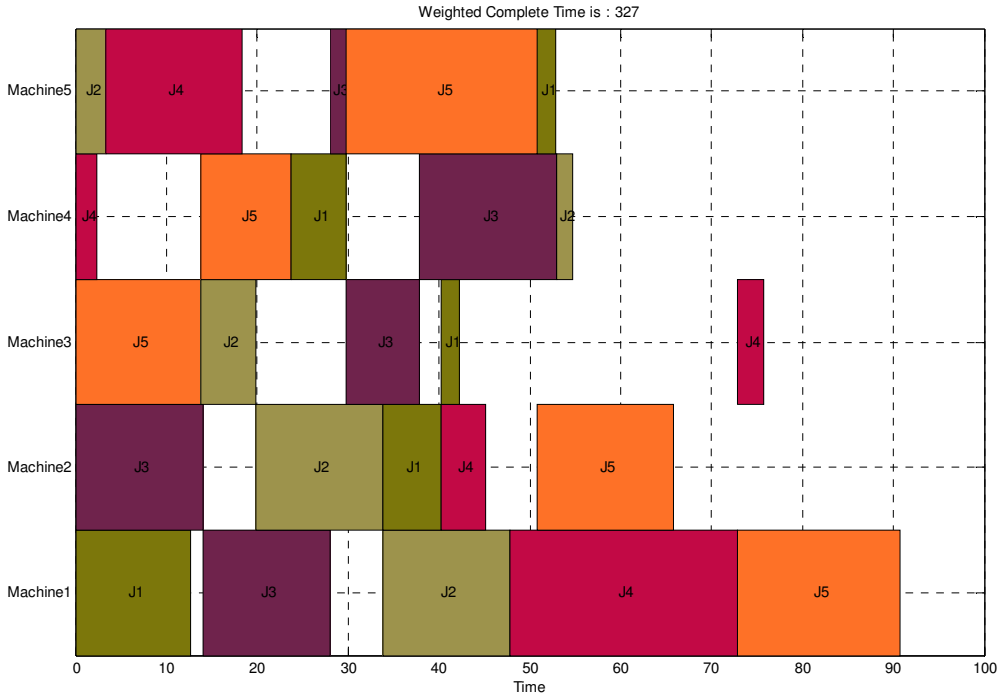


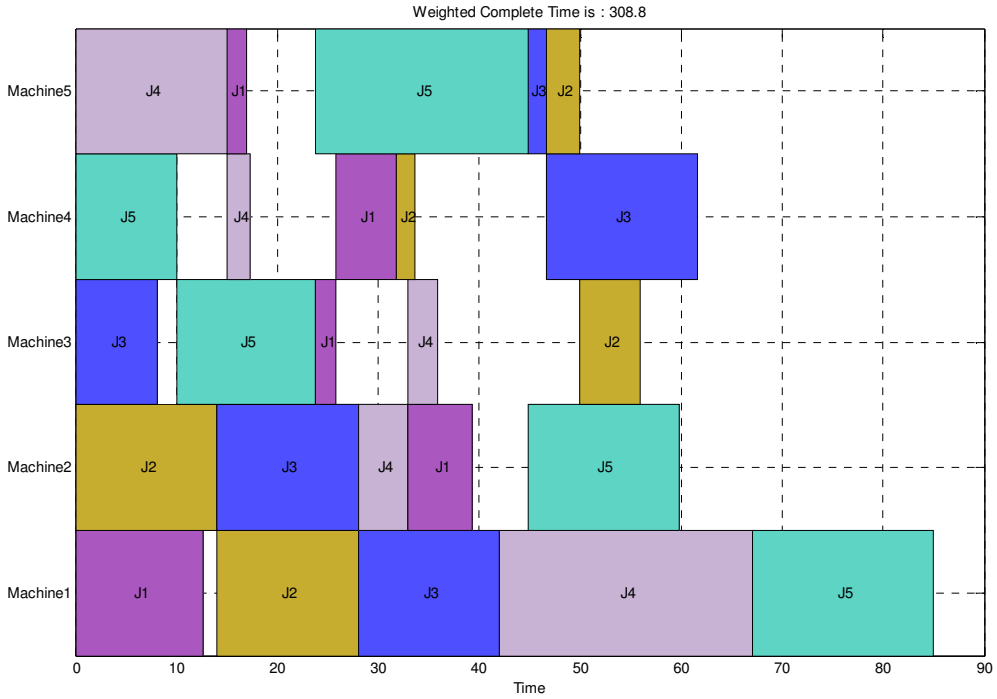


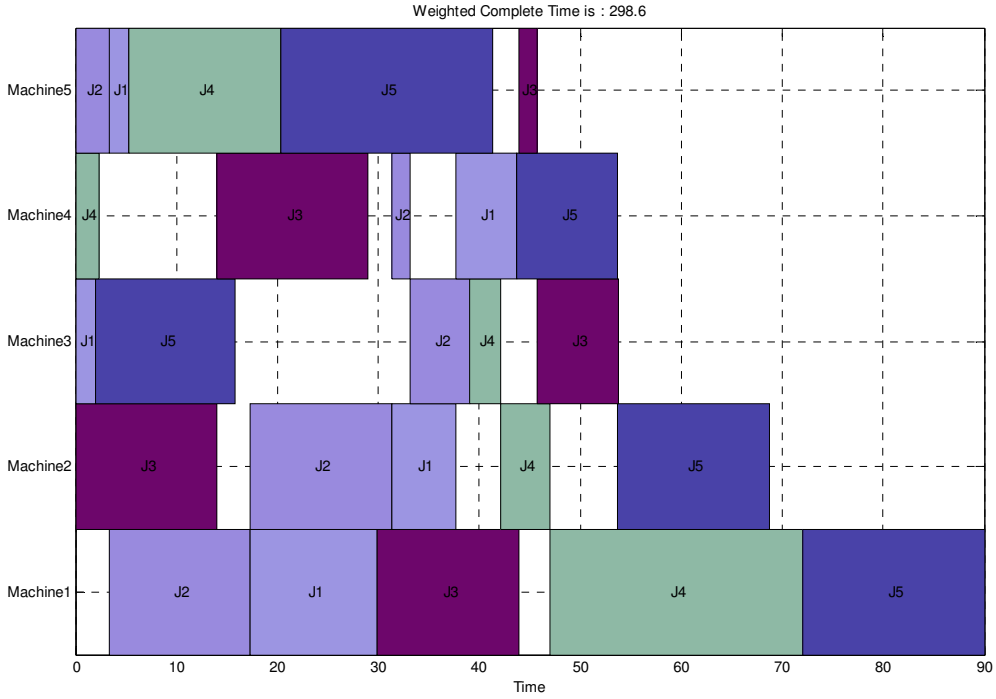

Weighted Complete Time is : 305.4
Machine5
Machine4
Machine3
Machine2
Machine1
J1
J3
J4
J5
J2
J5
J1
J3
J4
J2
J1
J4
J5
J2
J3
J2
J4
J1
J3
J5
J3
J2
J1
J4
J5
0
10
20
30
40
50
60
70
80
90
Time

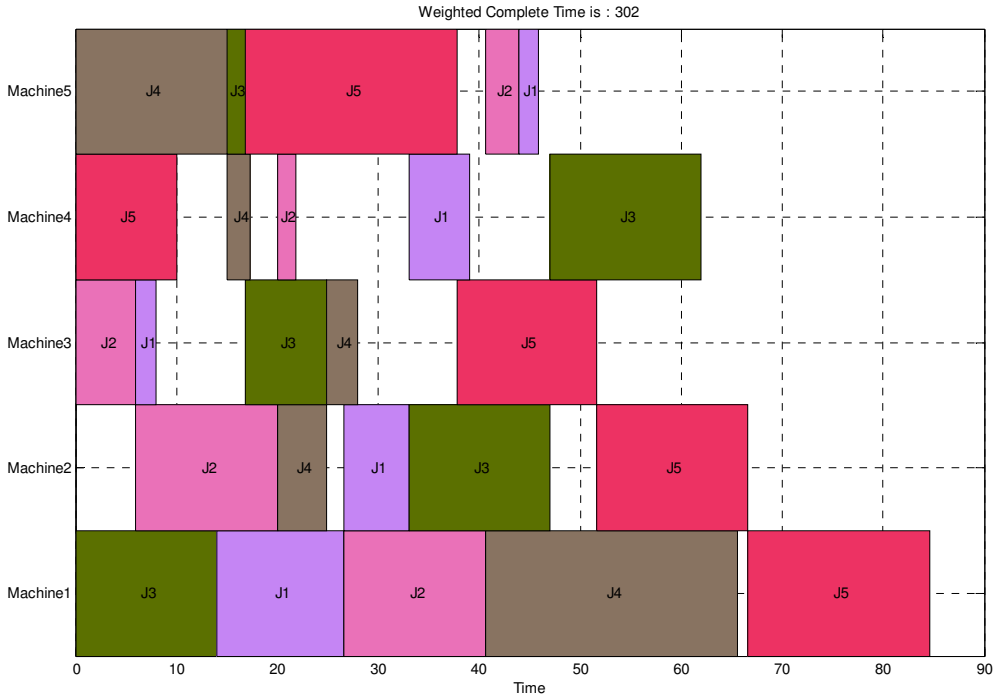

Weighted Complete Time is : 306.3
Machine5
Machine4
Machine3
Machine2
Machine1
J4
J5
J1
J2
J3
J5
J4
J2
J3
J1
J1
J2
J4
J5
J3
J2
J3
J1
J5
J4
J3
J1
J2
J4
J5
0
10
20
30
40
50
60
70
80
90
Time

Įterpimo mutaciją ir pozija paremtą kryžminimą (kai populiacija yra 10, 25, 50, 75)



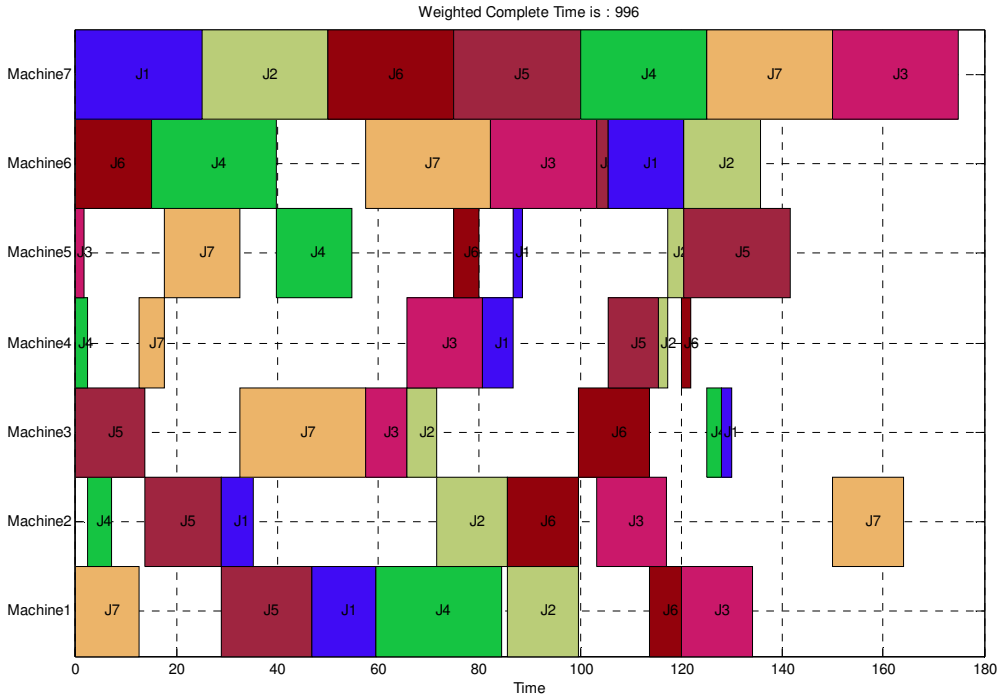


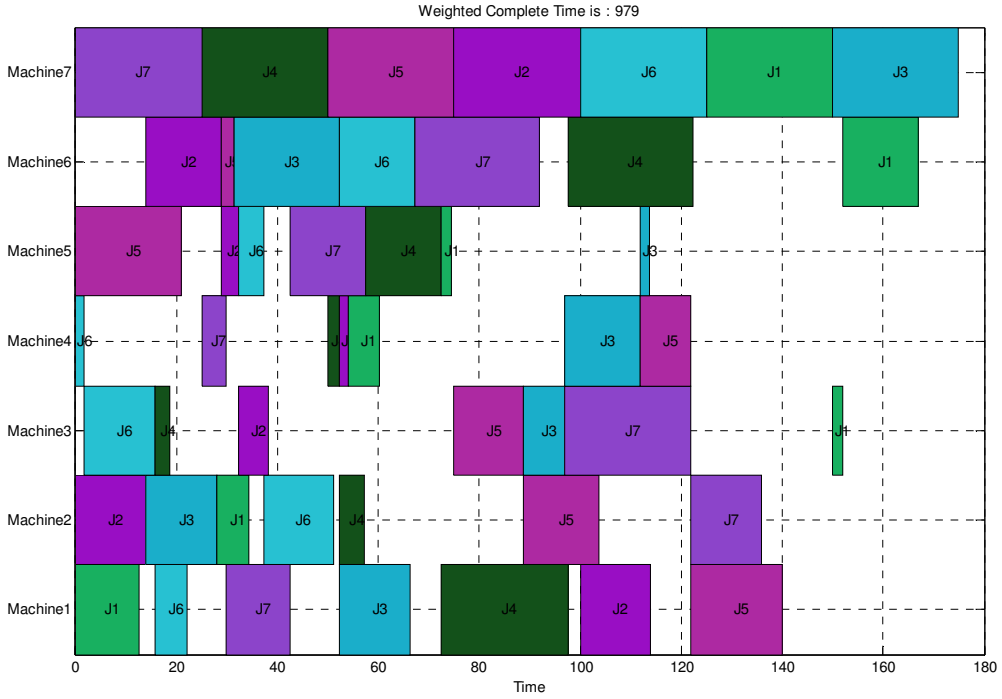

Weighted Complete Time is : 298.6
Machine5
Machine4
Machine3
Machine2
Machine1
J2
J1
J4
J5
J3
J4
J3
J2
J1
J5
J1
J5
J2
J4
J3
J3
J2
J1
J4
J5
J2
J1
J3
J4
J5
0
10
20
30
40
50
60
70
80
90
Time

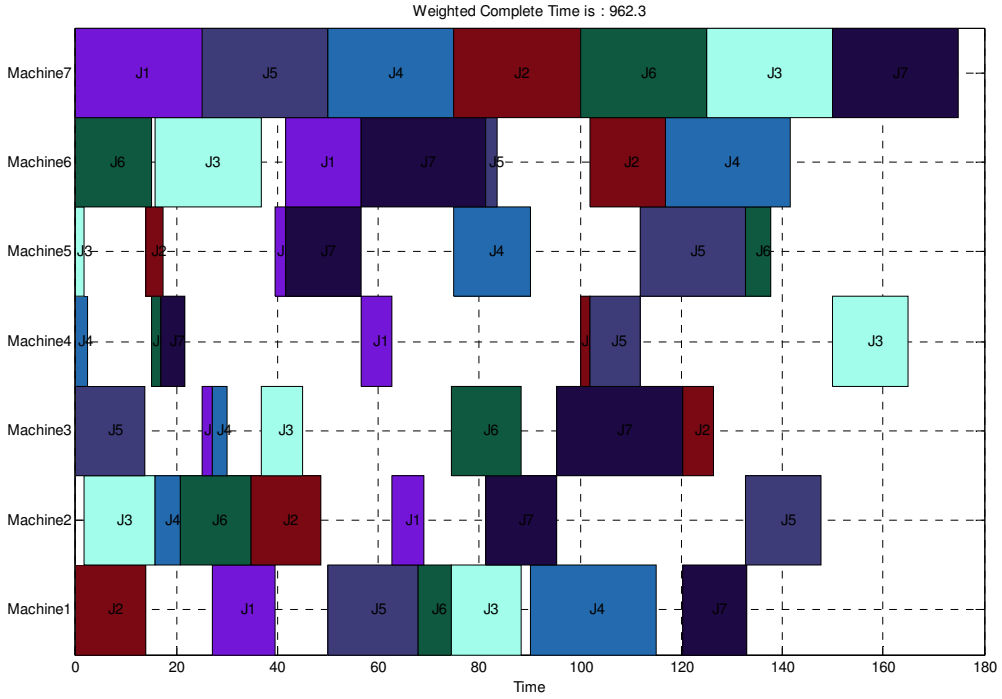

Weighted Complete Time is : 302
Machine5
Machine4
Machine3
Machine2
Machine1
J4
J3
J5
J2
J1
J5
J4
J2
J1
J3
J2
J1
J3
J4
J5
J2
J4
J1
J3
J5
J3
J1
J2
J4
J5
0
10
20
30
40
50
60
70
80
90
Time

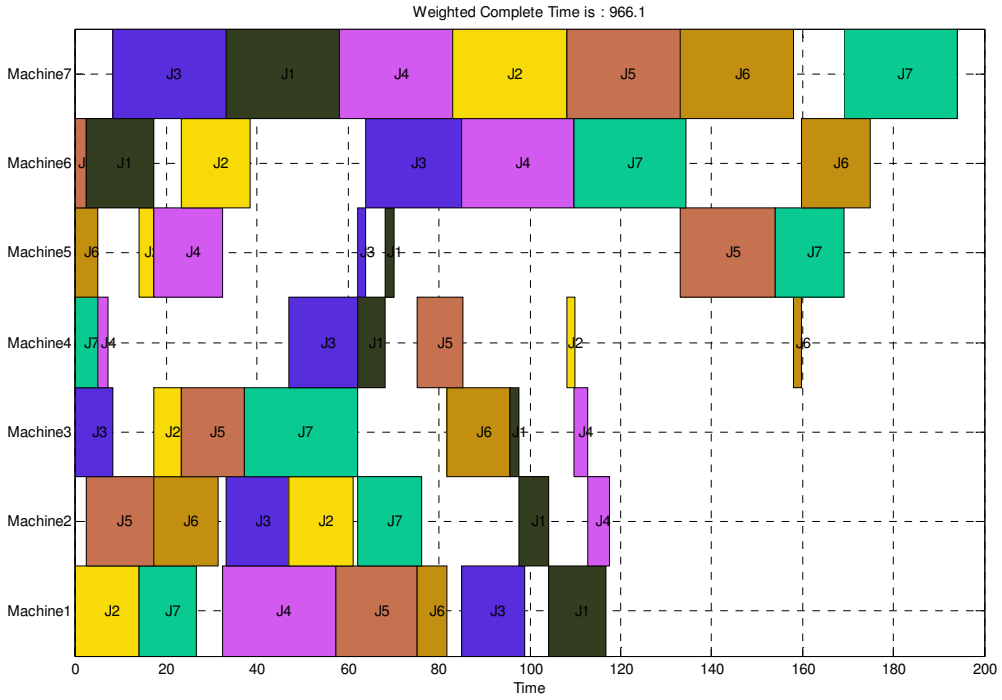
Tvarkaraščiai, gauti nagrinėjant uždavinį su septyniomis darbų rūšimis ir septyniomis mašinomis, naudojant:

Sukeitimo mutaciją ir ciklinį kryžminimą (kai populiacija yra 10, 25, 50, 75)

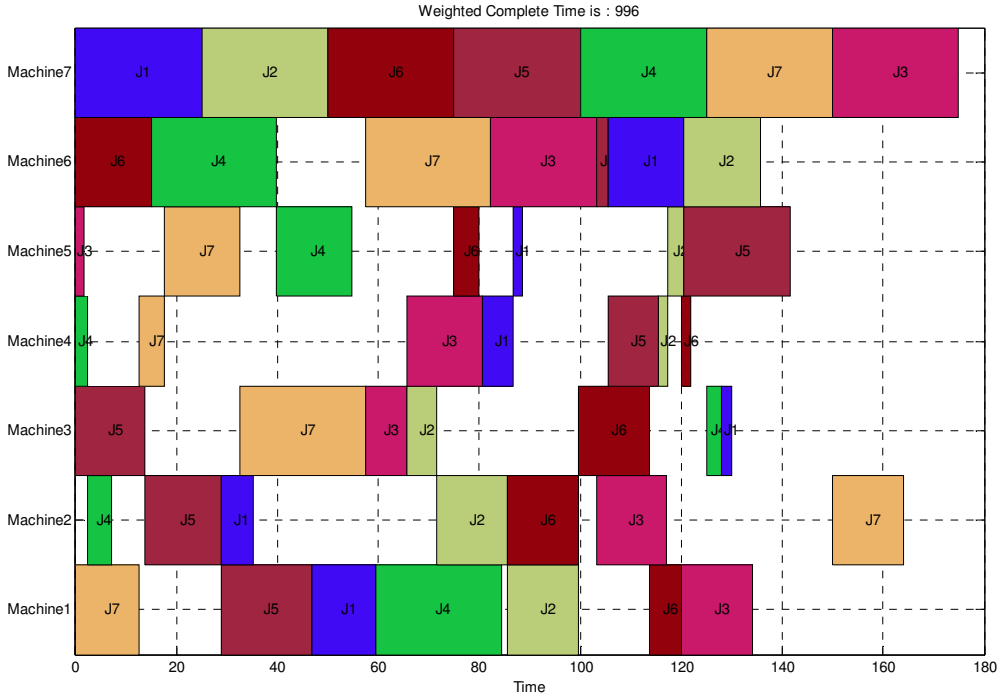


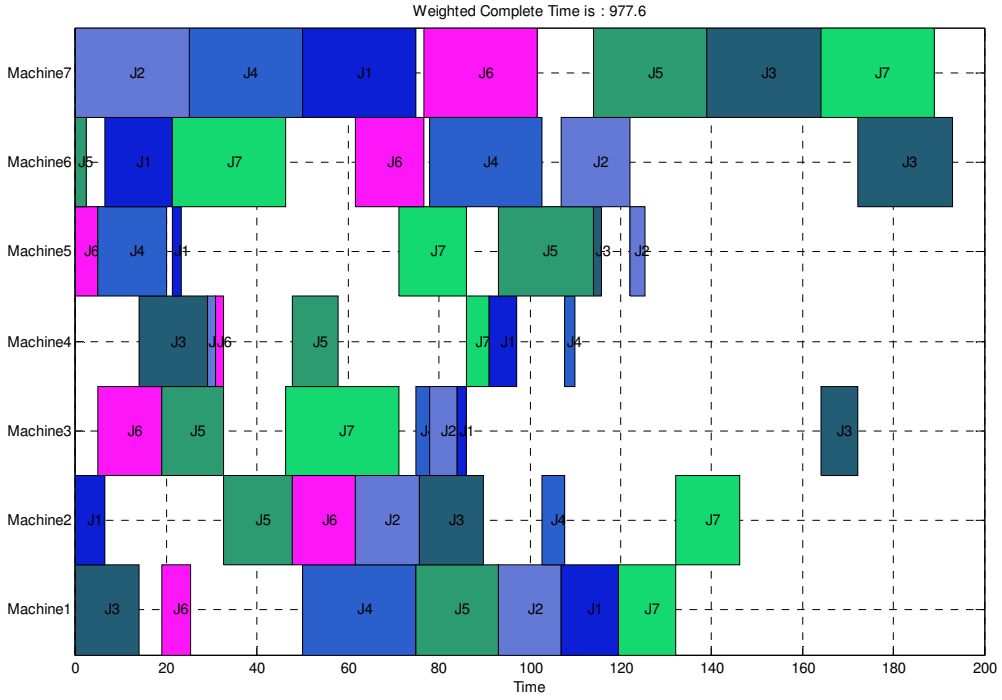


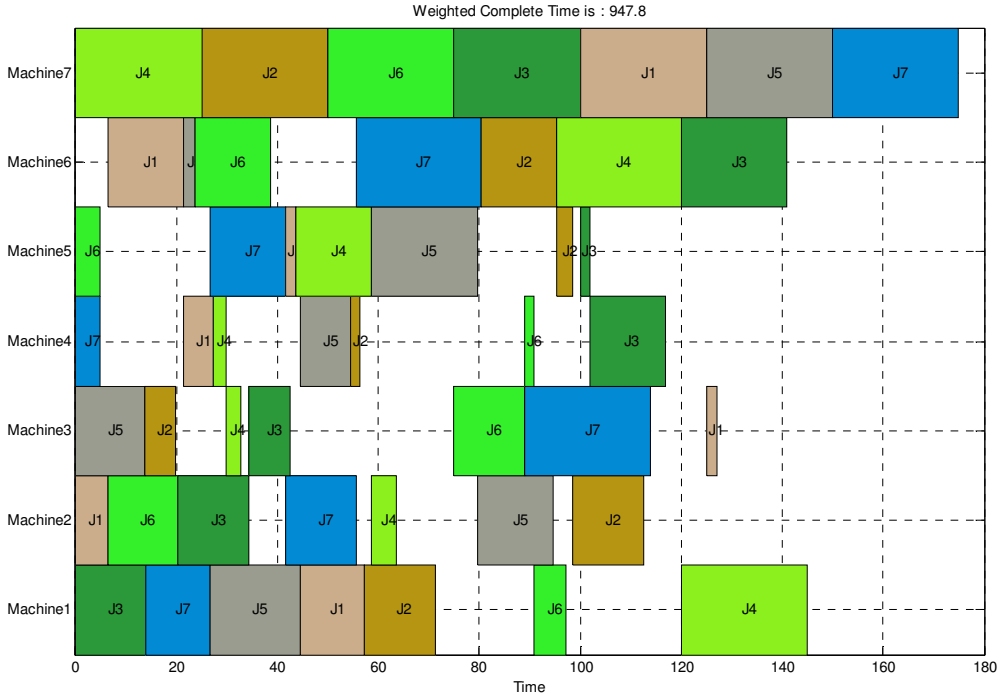

Weighted Complete Time is : 962.3
Machine7
Machine6
Machine5
Machine4
Machine3
Machine2
Machine1
0
20
40
60
80
100
120
140
160
180
Time

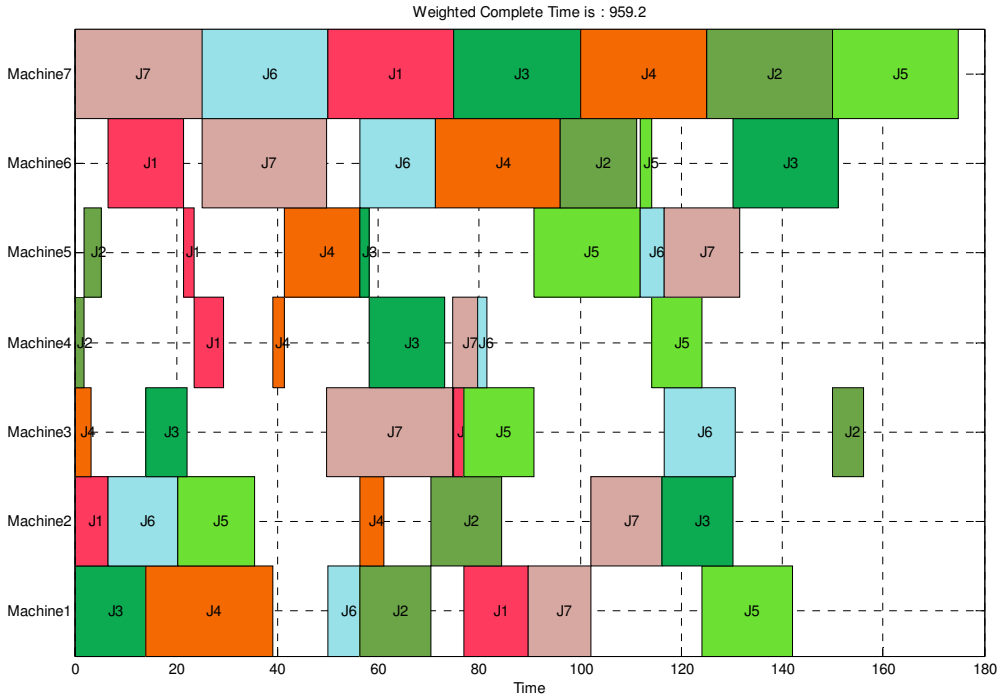

Weighted Complete Time is : 966.1
Machine7
Machine6
Machine5
Machine4
Machine3
Machine2
Machine1
0
20
40
60
80
100
120
140
160
180
200
Time

Sukeitimo mutaciją ir tvarkos kryžminimą (kai populiacija yra 10, 25, 50, 75)

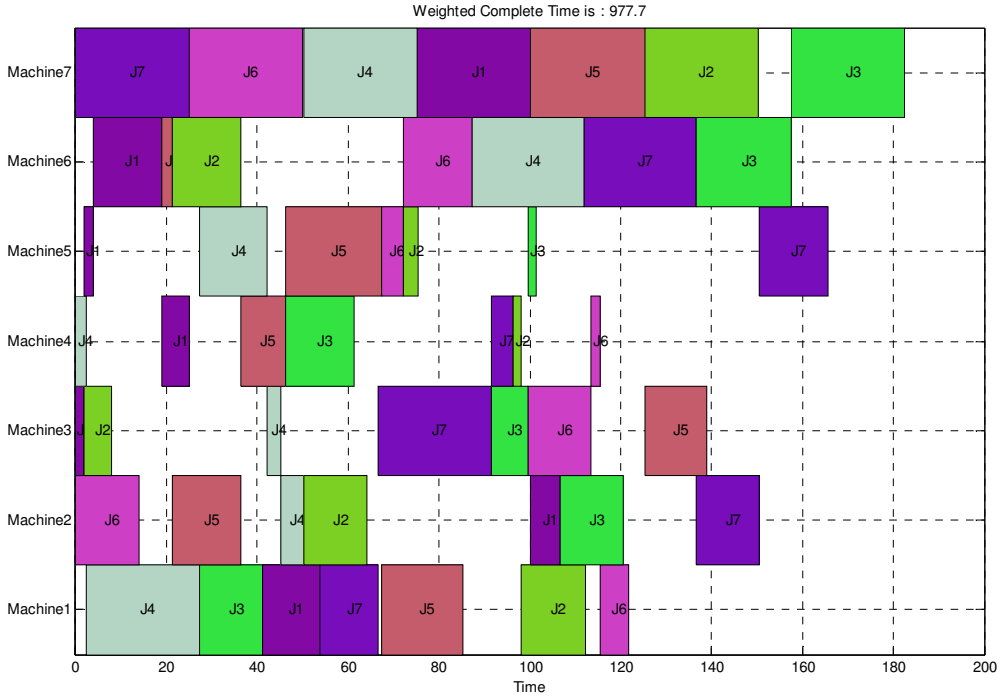


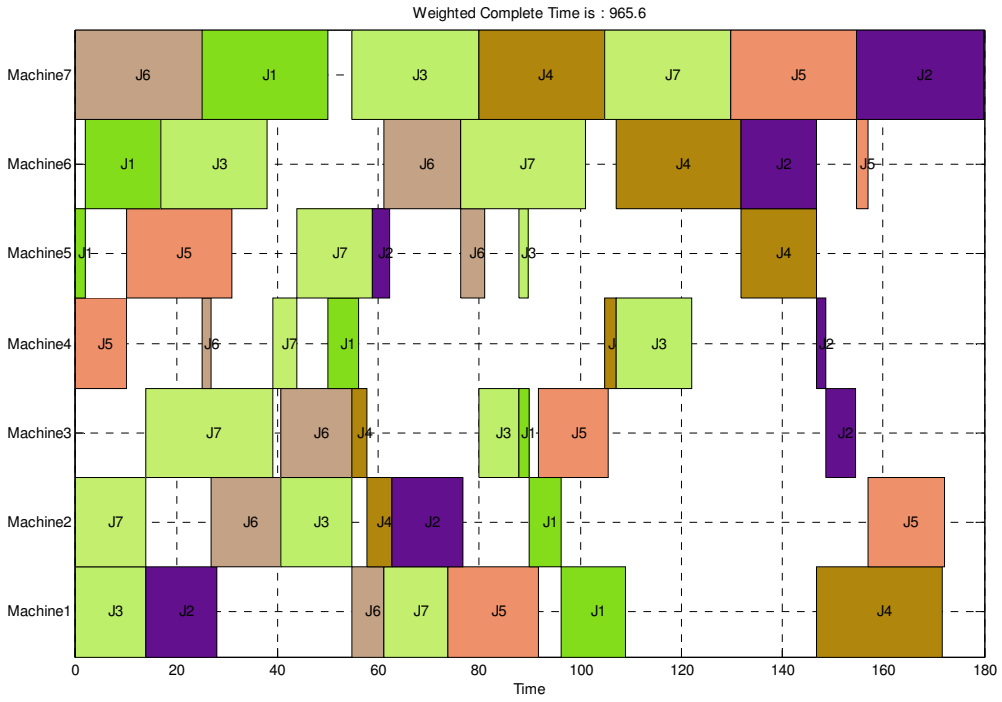


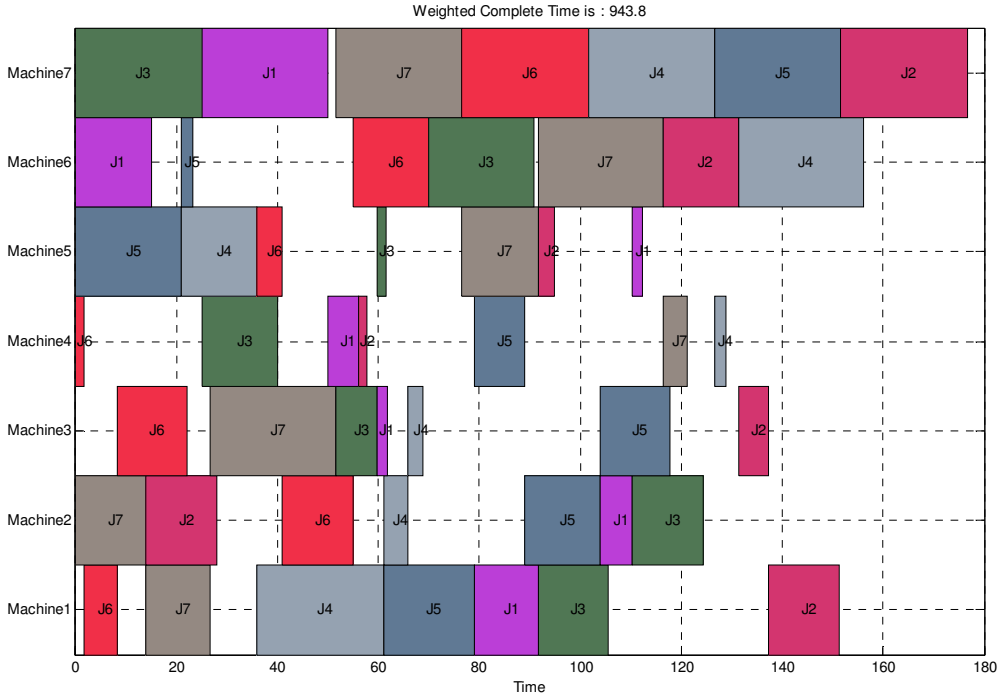

Weighted Complete Time is : 947.8
Machine7
Machine6
Machine5
Machine4
Machine3
Machine2
Machine1
0
20
40
60
80
100
120
140
160
180
Time

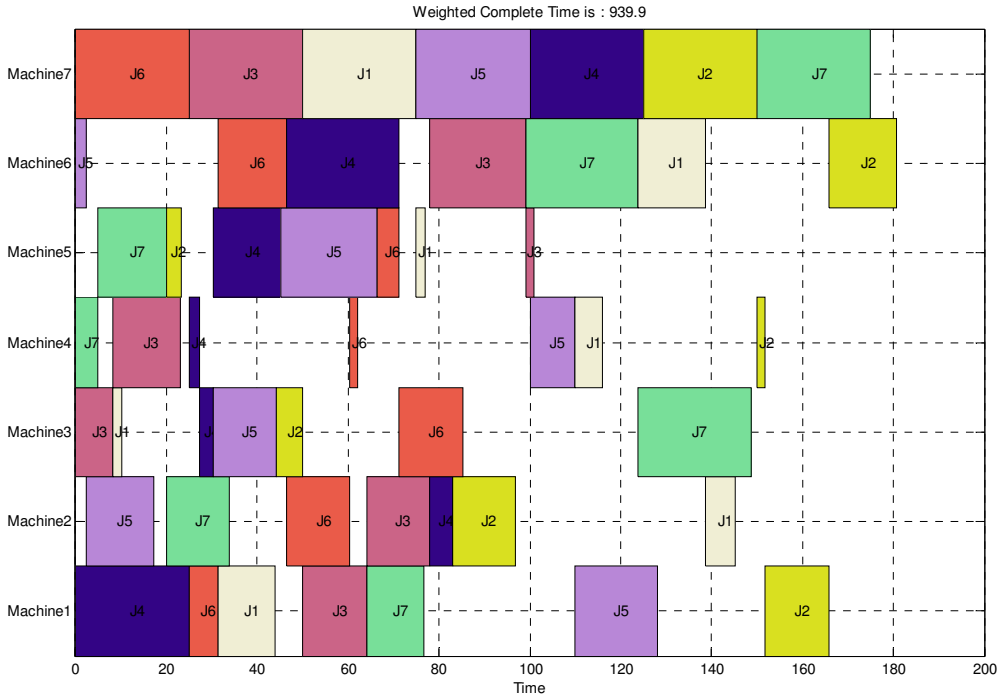

Weighted Complete Time is : 959.2
Machine7
Machine6
Machine5
Machine4
Machine3
Machine2
Machine1
0
20
40
60
80
100
120
140
160
180
Time

Sukeitimo mutaciją ir pozicija paremtą kryžminimą (kai populiacija yra 10, 25, 50, 75)

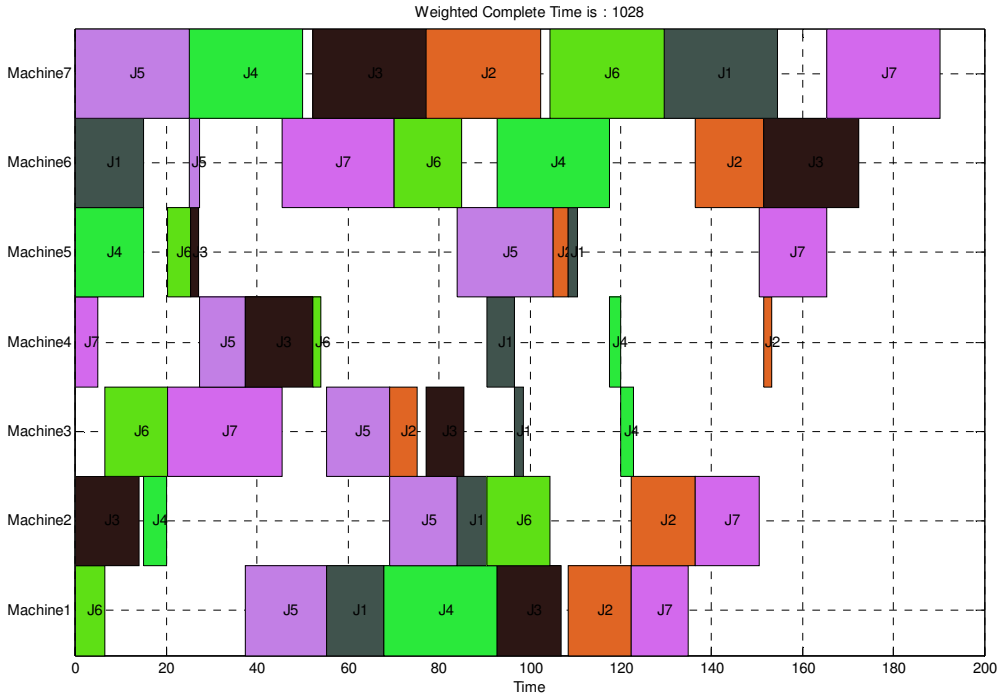


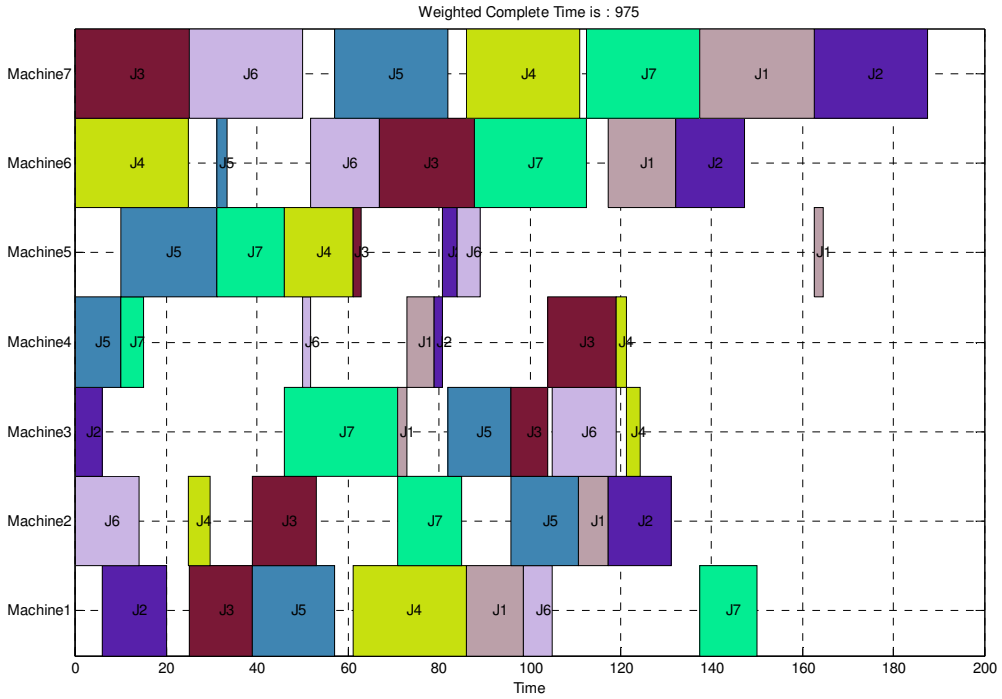


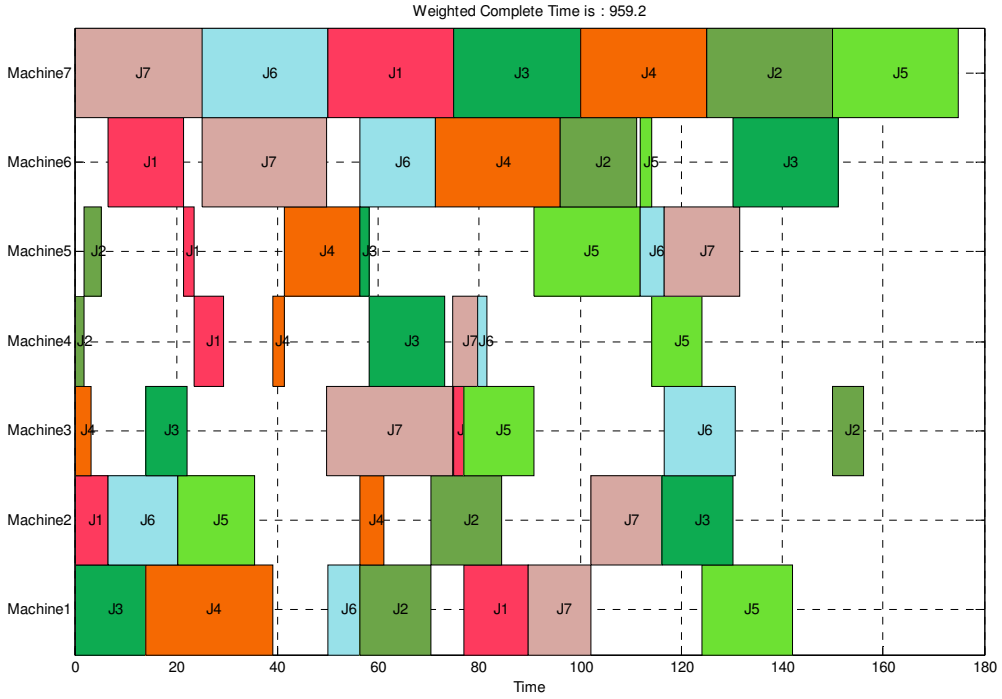

Weighted Complete Time is : 943.8
Machine7
Machine6
Machine5
Machine4
Machine3
Machine2
Machine1
0
20
40
60
80
100
120
140
160
180
Time

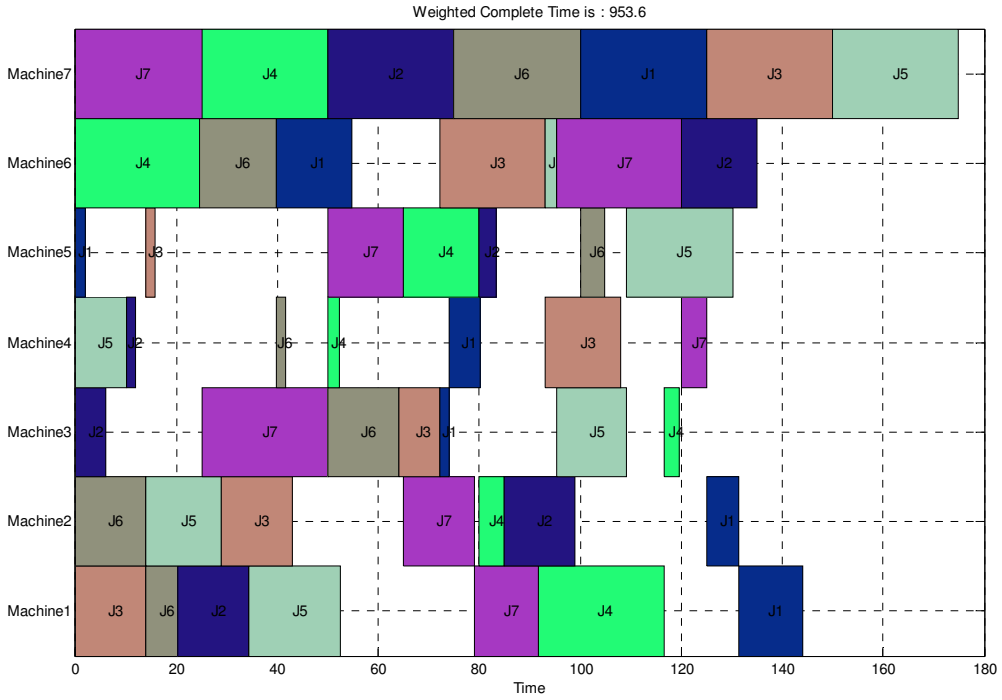

Weighted Complete Time is : 939.9
Machine7
Machine6
Machine5
Machine4
Machine3
Machine2
Machine1
0
20
40
60
80
100
120
140
160
180
200
Time

Perstatymo mutaciją ir ciklinį kryžminimą (kai populiacija yra 10, 25, 50, 75)

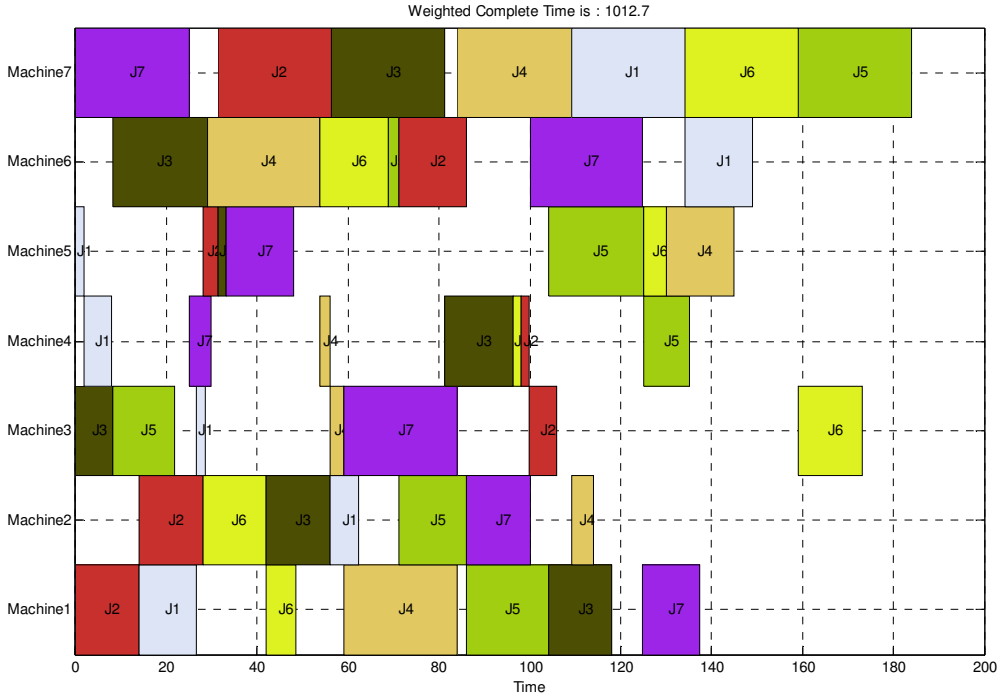


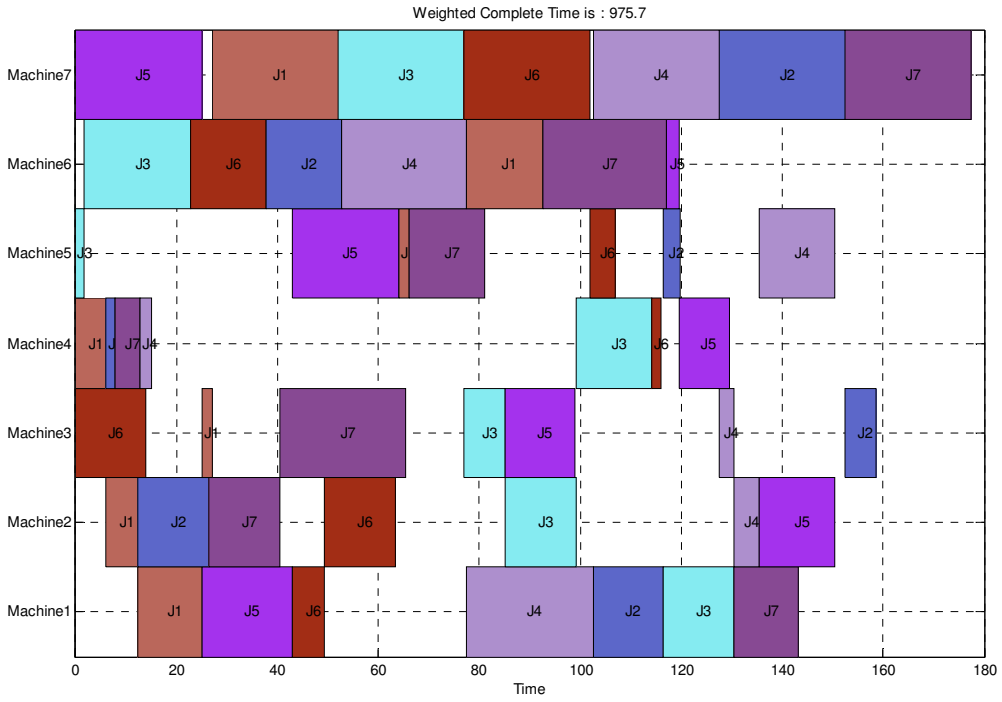


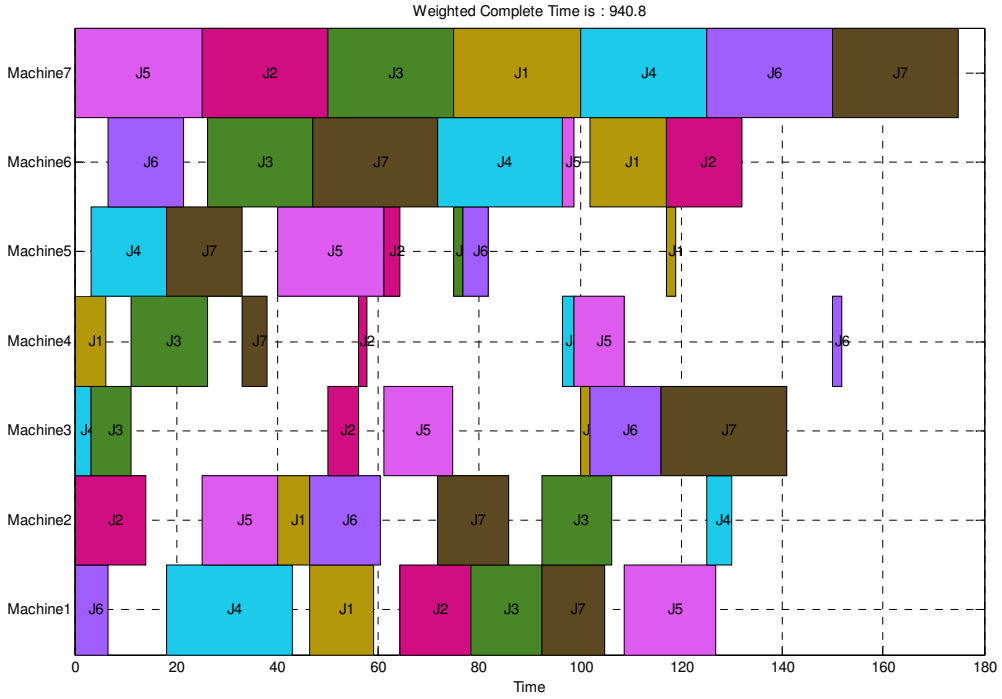

Weighted Complete Time is : 959.2
Machine7
Machine6
Machine5
Machine4
Machine3
Machine2
Machine1
0
20
40
60
80
100
120
140
160
180
Time

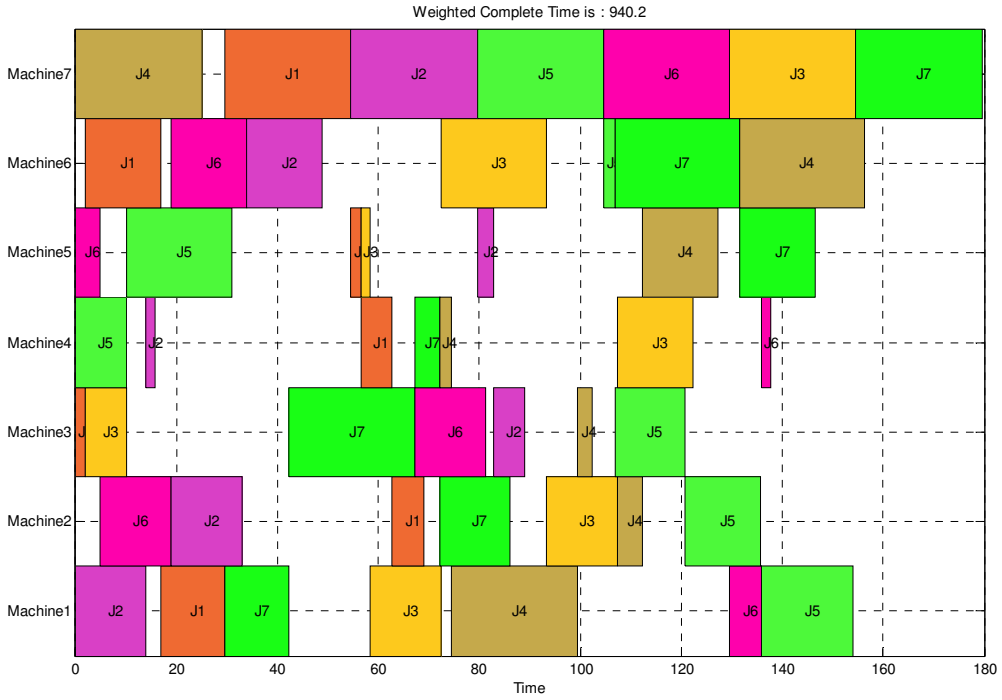

Weighted Complete Time is : 953.6
Machine7
Machine6
Machine5
Machine4
Machine3
Machine2
Machine1
0
20
40
60
80
100
120
140
160
180
Time

Perstatymo mutaciją ir tvarkos kryžminimą (kai populiacija yra 10, 25, 50, 75)

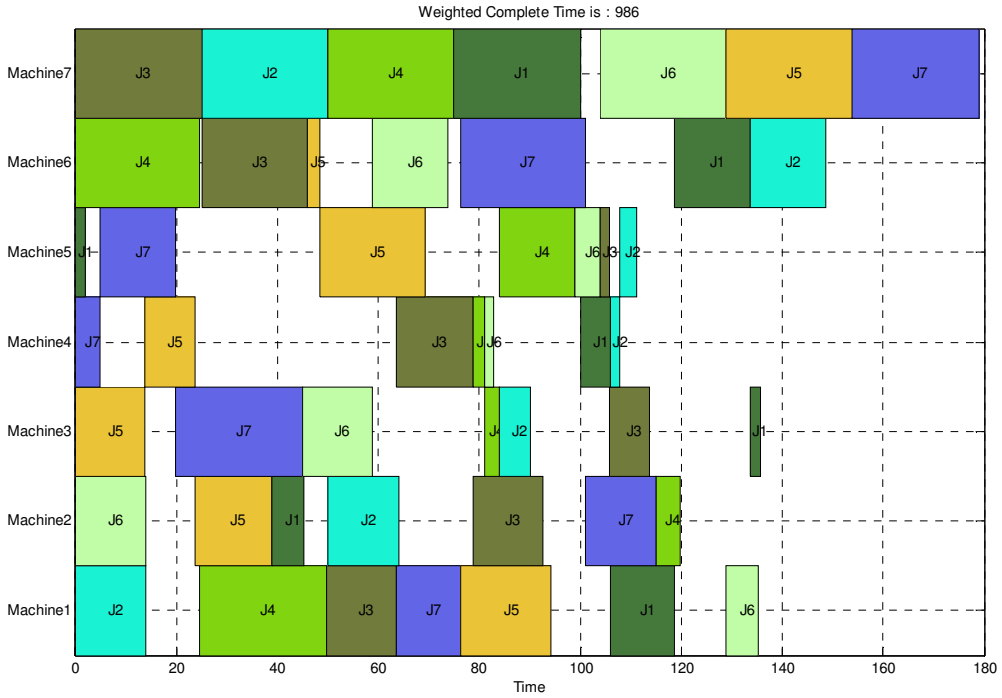


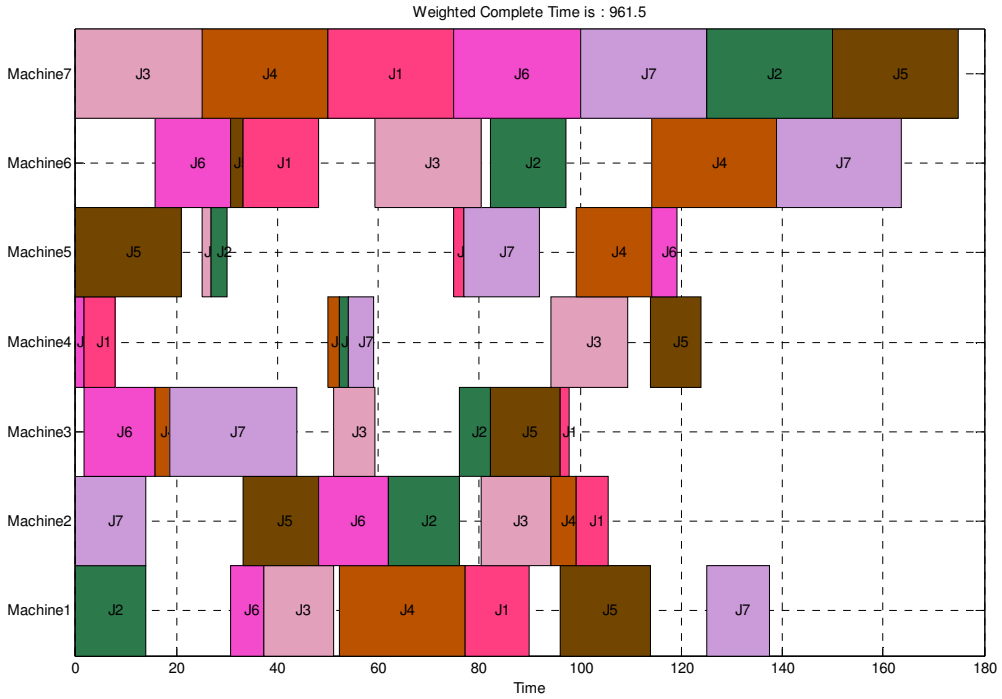


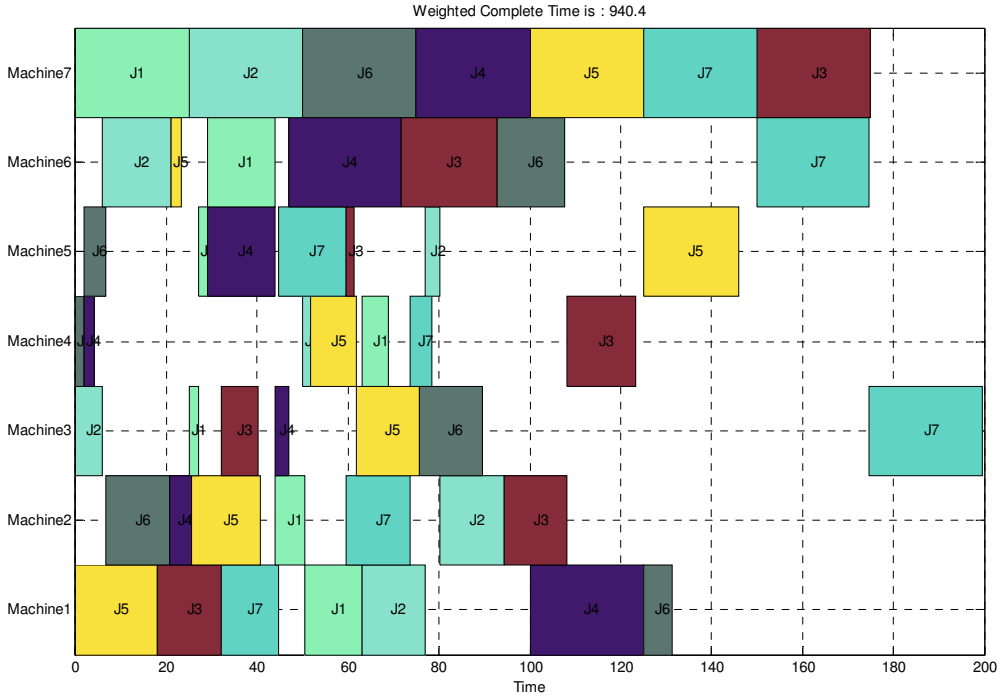

Weighted Complete Time is : 940.8
Machine7
Machine6
Machine5
Machine4
Machine3
Machine2
Machine1
0
20
40
60
80
100
120
140
160
180
Time

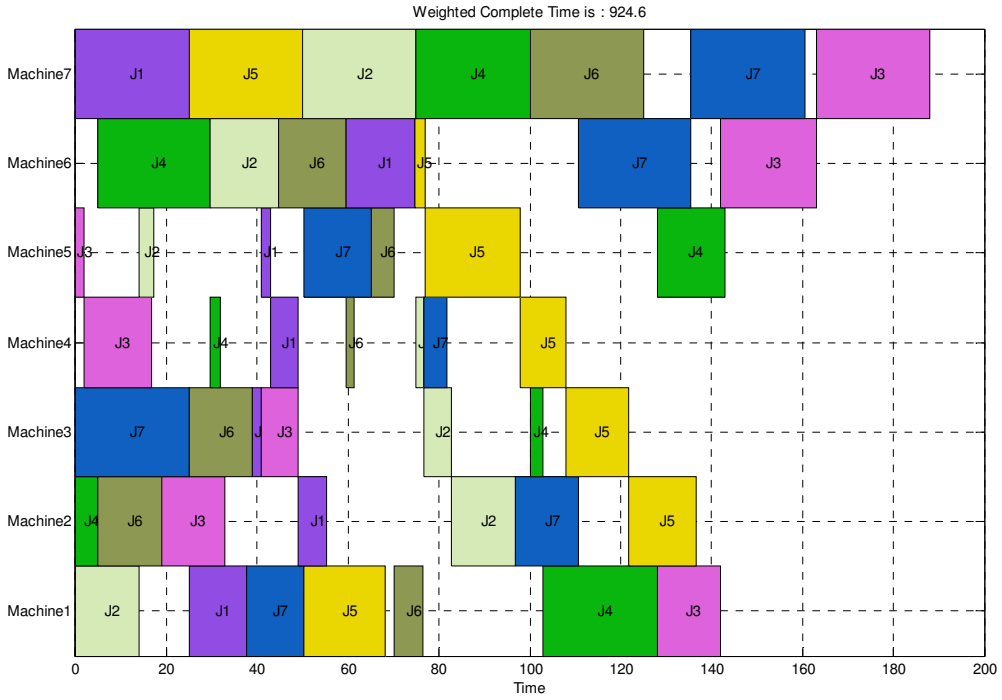

Weighted Complete Time is : 940.2
Machine7
Machine6
Machine5
Machine4
Machine3
Machine2
Machine1
0
20
40
60
80
100
120
140
160
180
Time

Perstatymo mutaciją ir pozicija paremtą kryžminimą (kai populiacija yra 10, 25, 50, 75)

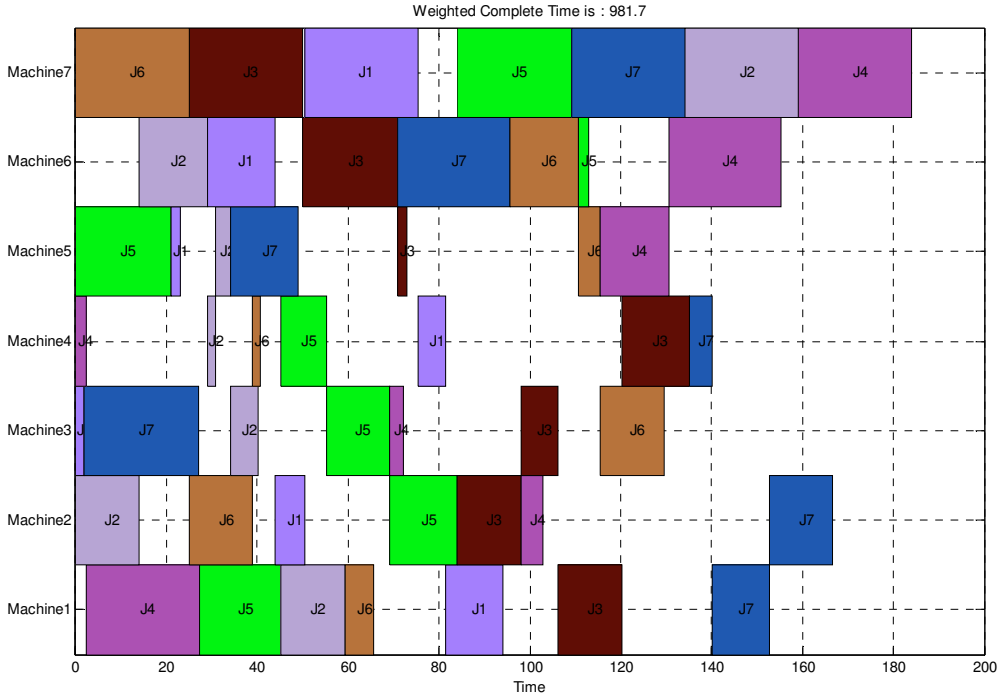


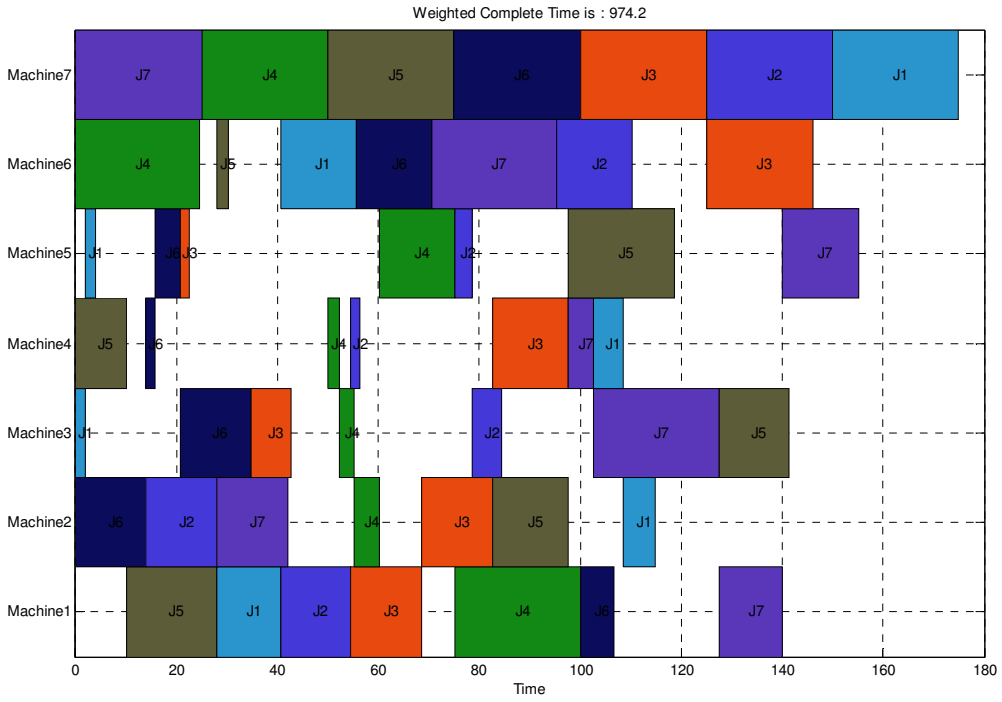


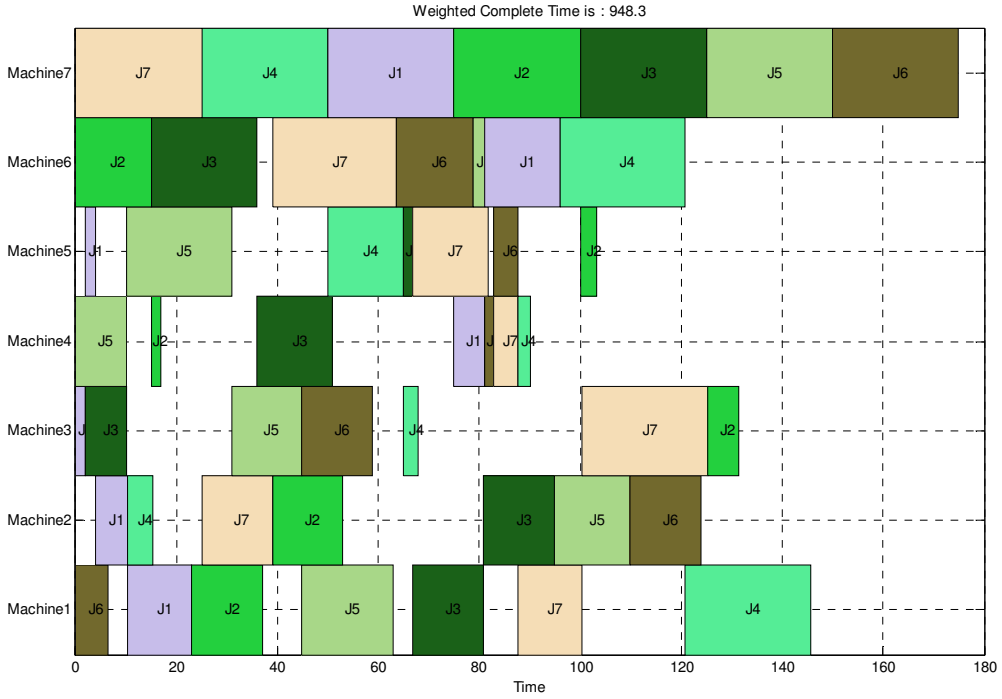

Weighted Complete Time is : 940.4
Machine7
Machine6
Machine5
Machine4
Machine3
Machine2
Machine1
0
20
40
60
80
100
120
140
160
180
200
Time

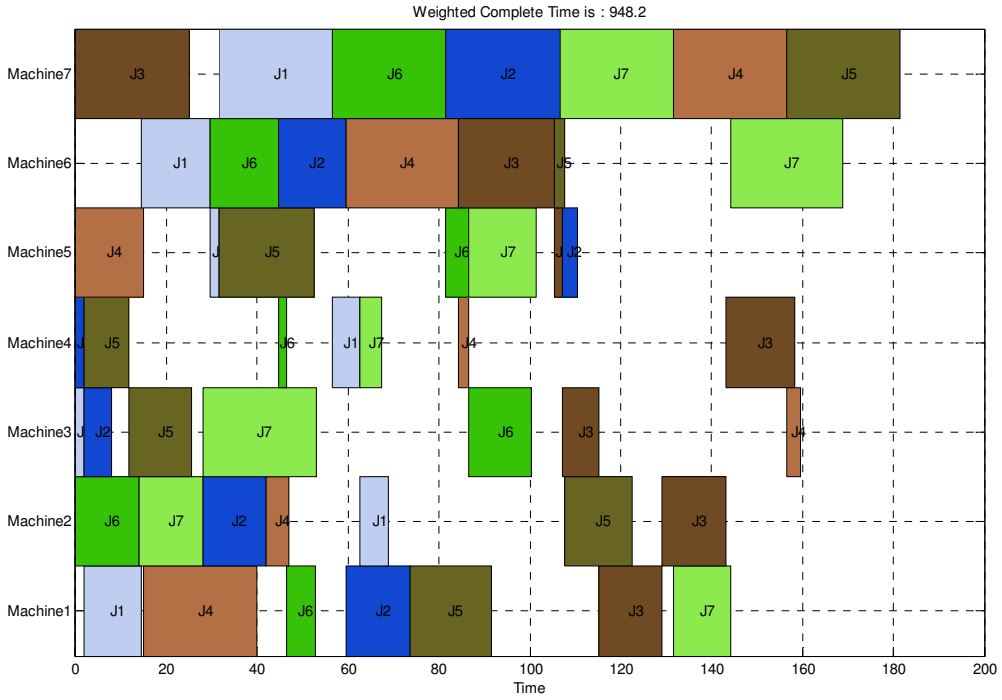

Weighted Complete Time is : 924.6
Machine7
Machine6
Machine5
Machine4
Machine3
Machine2
Machine1
0
20
40
60
80
100
120
140
160
180
200
Time

Įterpimo mutaciją ir ciklinį kryžminimą (kai populiacija yra 10, 25, 50, 75)

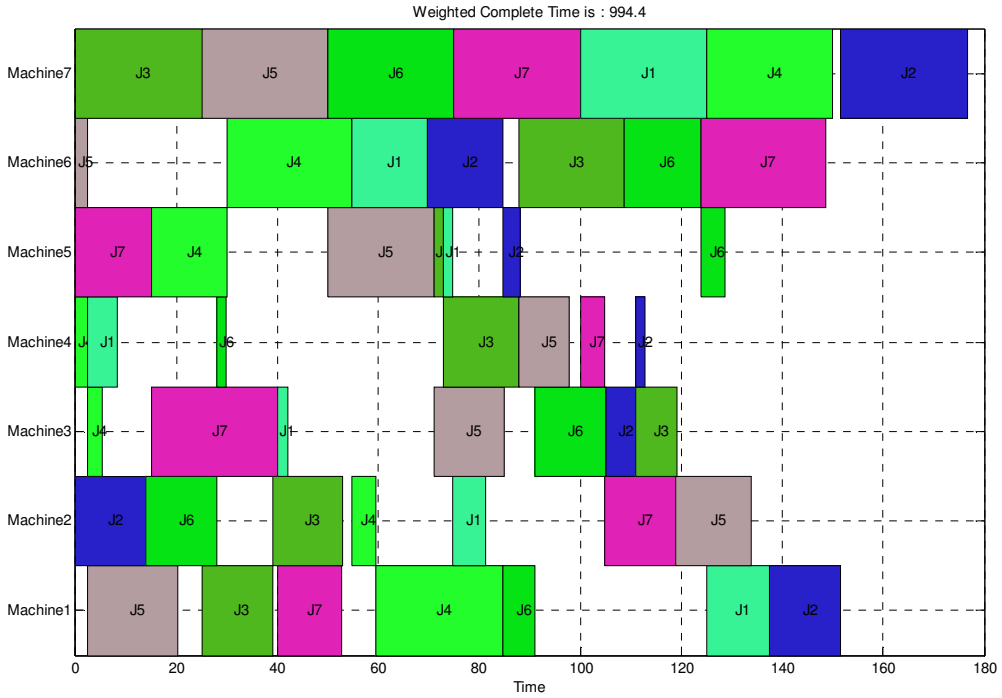


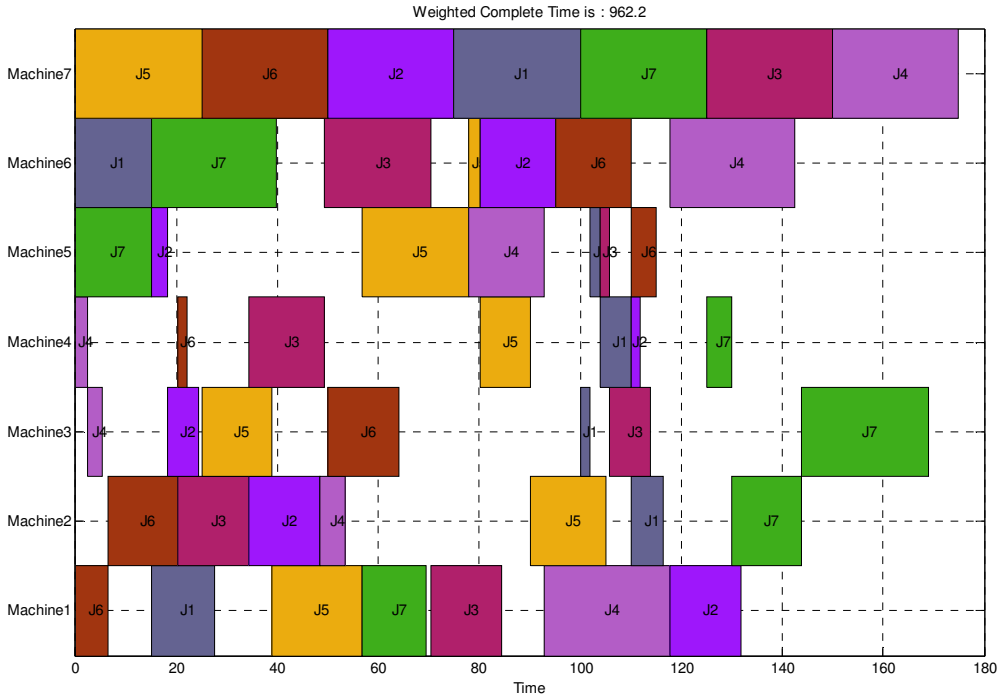


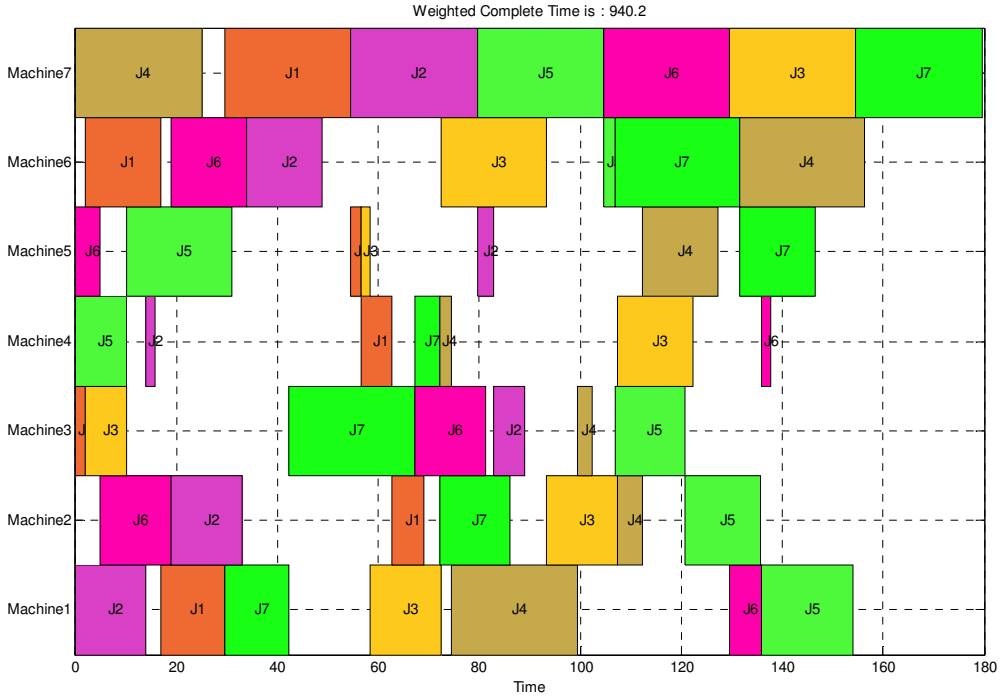

Weighted Complete Time is : 948.3
Machine7
Machine6
Machine5
Machine4
Machine3
Machine2
Machine1
0
20
40
60
80
100
120
140
160
180
Time

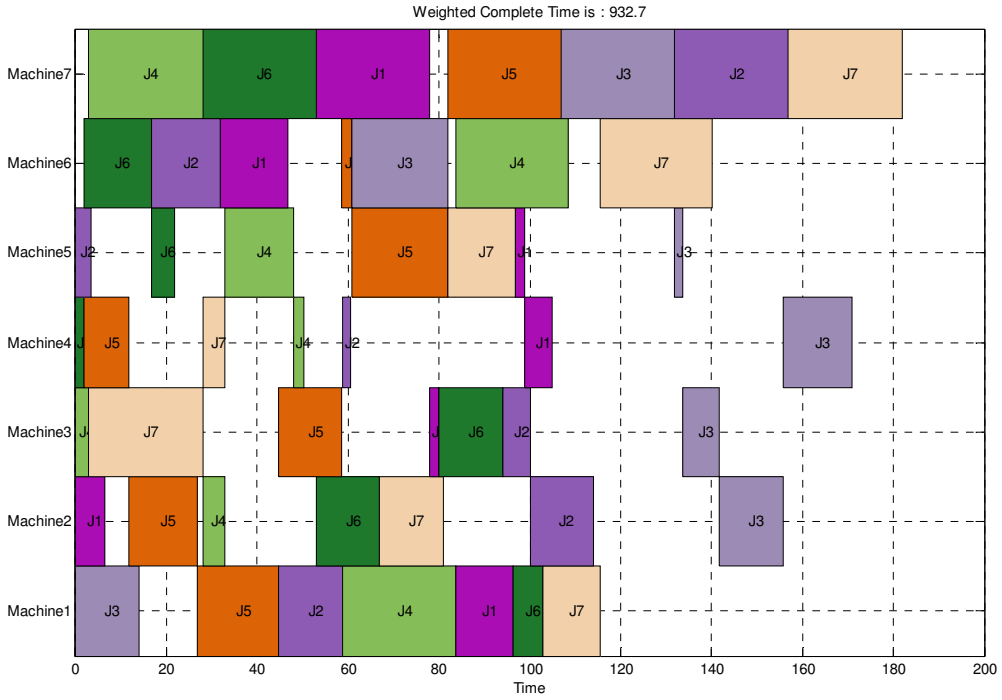

Weighted Complete Time is : 948.2
Machine7
Machine6
Machine5
Machine4
Machine3
Machine2
Machine1
0
20
40
60
80
100
120
140
160
180
200
Time

Įterpimo mutaciją ir tvarkos kryžminimą (kai populiacija yra 10, 25, 50, 75)

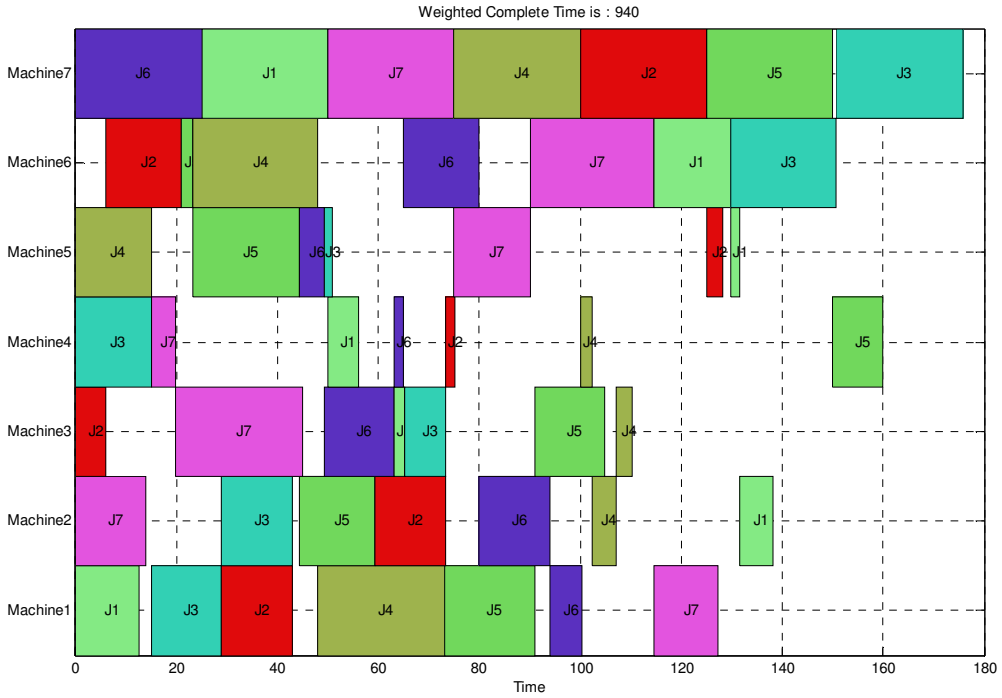


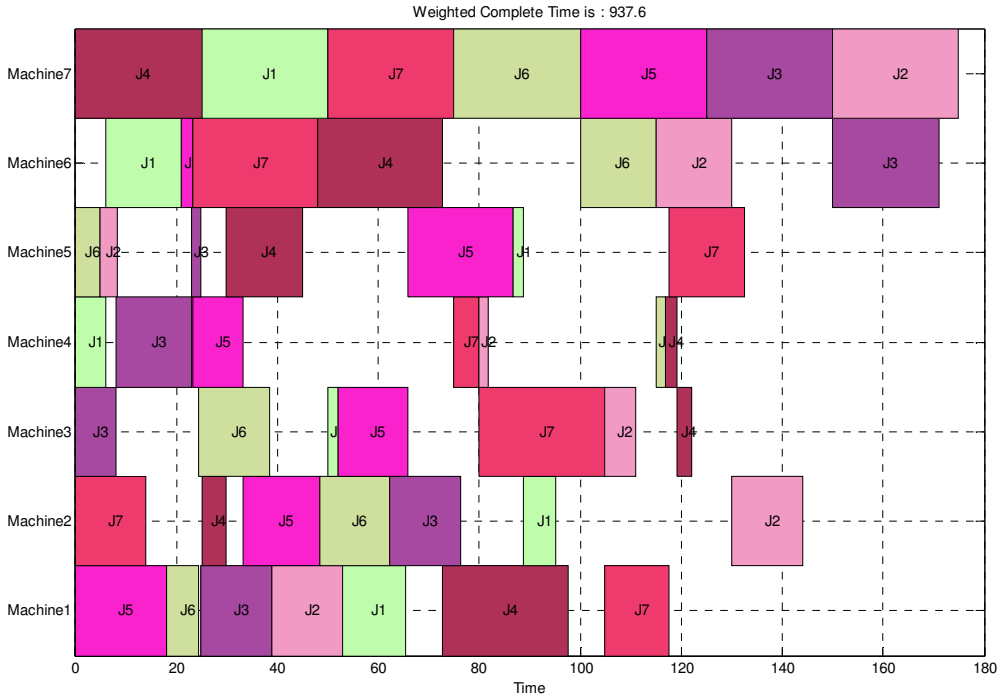


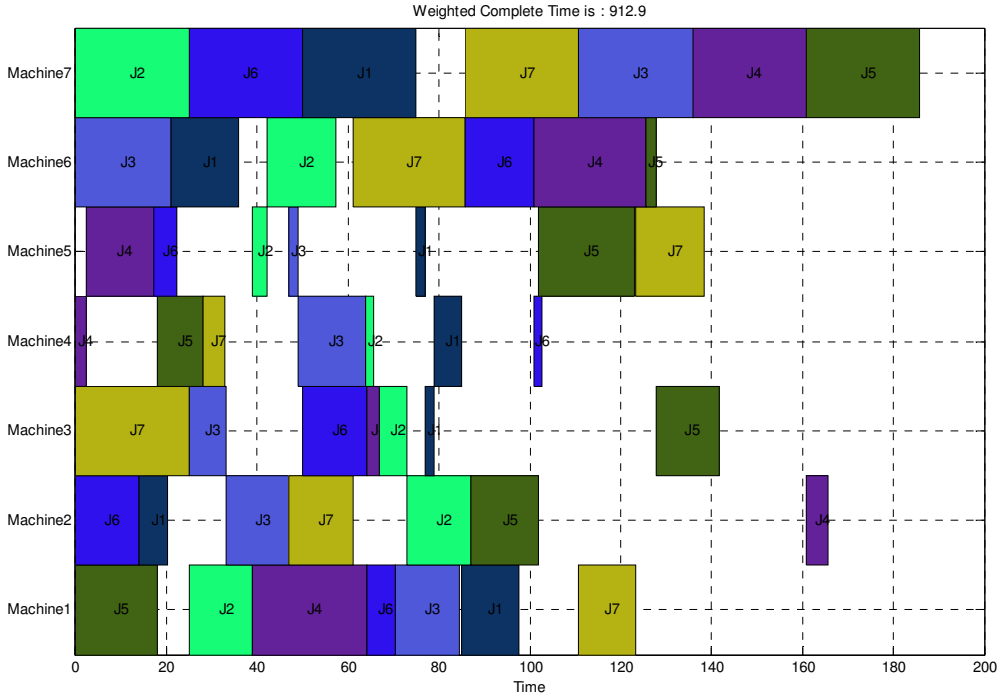

Weighted Complete Time is : 940.2
Machine7
Machine6
Machine5
Machine4
Machine3
Machine2
Machine1
0
20
40
60
80
100
120
140
160
180
Time

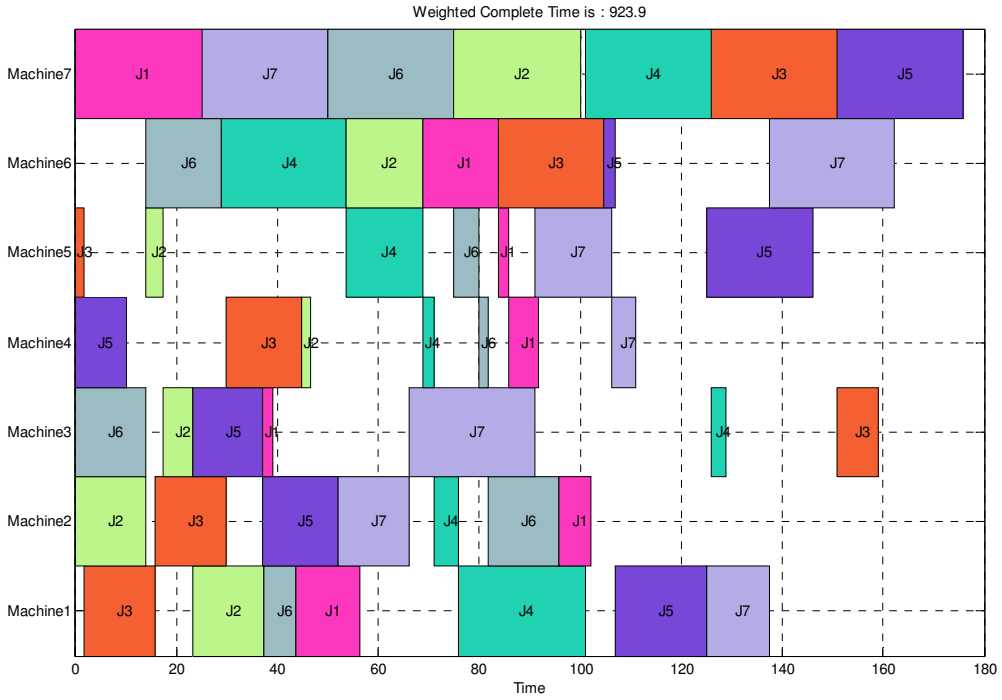

Weighted Complete Time is : 932.7
Machine7
Machine6
Machine5
Machine4
Machine3
Machine2
Machine1
0
20
40
60
80
100
120
140
160
180
200
Time

Įterpimo mutaciją ir pozicija paremtą kryžminimą (kai populiacija yra 10, 25, 50, 75)






Weighted Complete Time is : 912.9
Machine7
Machine6
Machine5
Machine4
Machine3
Machine2
Machine1
0
20
40
60
80
100
120
140
160
180
200
Time


Weighted Complete Time is : 923.9
Machine7
Machine6
Machine5
Machine4
Machine3
Machine2
Machine1
0
20
40
60
80
100
120
140
160
180
Time