Liet. matem. rink, **46***, spec. nr.*, 2006, 412–416

Tvarkaraščių sudarymo algoritmo analizė

Julius SIMONAVIČIUS, Narimantas LISTOPADSKIS (KTU)
el. paštas: simojuli@gmail.com, narlis@ktu.lt

Gamybos tvarkaraščių sudarymo uždavinys

Gamybos tvarkaraščių sudarymo uždavinys (toliau bus vadinamas tvarkaraščių sudarymu, *angl. schedule – tvarkaraštis*) – paskirstyti ribotus resursus užduotims diskrečioje sistemoje. Toliau kalbėsime apie gamybos tvarkaraščių sudarymą, nors iš esmės šis uždavinys ir nesiskiria nuo kitų tvarkaraščių sudarymų, skiriasi tik tai, kaip apibrėžiama mašina. Gamybos tvarkaraščių sudarymo atveju mašina suprantama kaip tam tikra gamybos linijos dalis. Tvarkaraščių sudaryme dažniausiai susiduriama su dviem apribojimais:

a) mašinų pajėgumų ribos,

b) technologiniai eiliškumo apribojimai.

Bet koks rezultatas tenkinantis šias dvi sąlygas galėtų būti vadinamas tvarkaraščių sudarymo uždavinio sprendiniu. Rastas sprendinys turėtų atsakyti į du klausimus:

a) kokie resursai bus panaudoti įvykdyti konkrečią užduotį,

b) kada kiekviena užduotis bus įvykdyta?

Tvarkaraščių sudarymas nagrinėja uždavinius, susijusius su resursų paskirstymu laike, norint atlikti tam tikrą rinkinį užduočių. Kiekvienas tvarkaraščių sudarymo uždavinys yra apibūdinamas trimis komponentais: mašinų skaičius M, darbų skaičius J ir atlikimo laikas kiekvienam darbui kiekvienoje mašinoje matricoje A (kitaip tariant, operacijų atlikimo laikai). Apribojimų rinkinį C privalo tenkinti kiekvienas tvarkaraštis.

Tvarkaraščių sudarymo uždaviniai yra įdomūs kartu ir iš praktinės, ir iš teorinės pusės. Priklausomai nuo pasirinkto sprendimo metodo, reikia pasirinkti tokį uždavinį, kuris būtų kiek įmanoma lengviau apibūdinamas. Reikia suformuluoti labai abstraktų modelį, atitinkantį realaus gyvenimo situaciją.

Paprastumo dėlei tarkime, kad kiekvienas darbas gali turėti tik vieną operaciją su bet kuria mašina. Apdirbimo laikai bus užduodami kaip dydžio $J * M$ matrica A, kur $A_{j,m}$ yra apdirbimo laikas darbui j su mašina m. Kompiuterinių resursų taupymo ir paprastumo sumetimais, visi apdirbimo laikai yra griežtai sveikieji teigiami skaičiai. Tuomet skaičius vienetas atitinka mažiausią nedalomą laiko vienetą, kitaip tariant jis turėtų būti pasirinktas kiekvienam uždaviniui individualiai. Tvarkaraščių sudarymo uždavinio tikslas – surasti tvarkaraštį. Toks tvarkaraštis yra pilnai ir vienareikšmiškai apibrėžiamas matricos S pagalba, kurios dydis $J * M$, kur $S_{j,m}$ yra operacijos atlikimo pradžios laikas darbui j ir m mašinai. Tinkamame tvarkaraštyje darbai arba mašinos

negali būti naudojami dvejose vietose tuo pačiu metu, todėl būtinai turi būti tenkinami šie apribojimai:

$$\forall i,\ mS_{i,m} \geqslant 0;$$
$$\forall i, j, n\ \ (i \neq j \wedge S_{i,m} \leqslant S_{j,m}) \Rightarrow S_{i,m} + A_{i,m} \leqslant S_{j,m};$$
$$\forall i, m, n\ \ (m \neq n \wedge S_i,\ m \leqslant S_{j,n}) \Rightarrow S_{i,m} + A_{i,m} \leqslant S_{j,n}.$$

Minimizuojama funkcija f (gamybos trukmės kriterijus) arba kitaip tariant laikas, kai baigiama vykdyti paskutinė operacija, išraiška:

$$f(S) = \max_{i,m}(S_{i,m} + A_{i,m}).$$

Pagrindiniai tvarkaraščių sudarymo nagrinėjami uždaviniai

FSSP (*angl. flow shop scheduling problem*) tvarkaraščių sudarymo atveju visos mašinos gamykloje yra sustatytos vienoje linijoje. Tai reiškia, kad darbai pirmajai mašinai gali būti surikiuoti be apribojimų, bet vėliau jie turi būti apdoroti kiekvienos mašinos ta pačia seka, kuria jie buvo apdoroti pirmos mašinos. Tai gali būti formalizuota tokiu būdu:

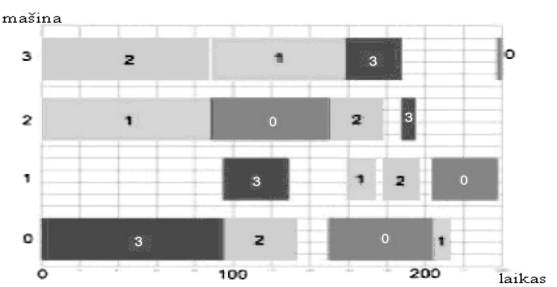
$$S \sim C \Leftrightarrow \forall i, j, m, n\ \ (S_{i,m} \leqslant S_{j,m} \Rightarrow S_{i,n} \leqslant S_{j,n}).$$

JSSP (*angl. job shop scheduling problem*) tvarkaraščių sudarymo atveju, kiekvienas darbas turi pakliūti į mašinas tam tikra iš anksti numatyta tvarka. Uždavinio apibūdinime turi būti duota dydžio $J * M$ matrica B. Kiekviena B eilutė j nusako, kokia tvarka darbas j turi aplankyti mašinas. Tai gali būti formalizuota tokiu būdu:

$$S \sim C \Leftrightarrow \forall i, x, y\ \ (x < y \Rightarrow S_{i,B_{i,x}} \leqslant S_{i,B_{i,y}}).$$

1 pav. pateikiamas JSSP sprendinio pavyzdys. Tokia iliustracija yra vadinama *Ganto diagrama*. Kiekviena eilutė atstovauja vieną mašiną. Skirtingų spalvų stačiakampiai atstovauja skirtingus darbus. Horizontali ašis reprezentuoja laiko tėkmę. Pavyzdinio tvarkaraščio trukmė 240 laiko vienetų.

Ganto diagramos gali būti naudojamos tikrinant tvarkaraštį. Pagal ją galima patikrinti ar tvarkaraštis tenkina apribojimus.



1 pav. Pavyzdinio uždavinio sprendinys.

OSSP (*angl. open shop scheduling problem*) tvarkaraščių sudarymo atveju jokių papildomų apribojimų netaikoma, t.y.

$$C = \phi.$$

Genetinis algoritmas

Genetiniai algoritmai yra prisitaikantys ir gali būti naudojami sprendžiant paieškos ir optimizavimo uždavinius. Šis algoritmas buvo sugalvotas remiantis genetiniais procesais vykstančiais gamtoje. Per daugelį kartų natūrali populiacija evoliucionuoja remiantis natūralios atrankos principu, t.y. stipriausio individo išlikimo principu, kuris pirmą karta buvo aprašytas Čarlzo Darvino knygoje „Rūšių pradmenys“. Mėgdžiojant šį procesą, genetiniai algoritmai yra pajėgūs prisitaikyti prie realių gyvenimiškų situacijų ir rasti jų sprendinius, jei algoritmas buvo tinkamai sudarytas.

Prieš pradedant spręsti uždavinį, pirmiausiai reikia rasti tinkamą kodavimą (arba atstovavimą) formuluojamam uždaviniui. Taip pat reikalinga *patikrinimo funkcija*, su kuria galime įvertinti tam tikrų individų (arba tiesiog sprendinių) gerumą, tam kad būtų įmanoma du individus palyginti tarpusavyje. Genetinio algoritmo realizacijos vykdymo metu pastoviai yra atrenkami du tėvai, kurie kryžminami tarpusavyje, kad gauti palikuonį.

Tariame, kad potencialūs uždavinio sprendiniai gali būti atstovaujami parametrų rinkinio. Šie parametrai (kitaip vadinami genais) yra sujungiami kartu ir tokiu būdu gaunama tam tikra reikšmių seka (chromosoma). Genetinėje terminologijoje, šis parametrų rinkinys atstovaujamas vienos chromosomos yra vadinamas *individu*. Individo tinkamumas, kuris yra nustatomas patikrinimo funkcijos pagalba, priklauso nuo jo chromosomos.

Individai vienos iteracijos (kartos) metu yra atrenkami ir kryžminami tarpusavyje ir tokiu būdu gaunami palikuonys, kurie patenka į sekančia populiaciją. Tėvai yra atsitiktinai atrenkami iš populiacijos, pagal schemą, kuri teikia pirmenybę geresniems individams. Išsirinkus du tėvus, jų chromosomos yra apjungiamos naudojantis *kryžminimo* ir *mutacijos* procedūromis. Mutacija taikoma keletui individų tam, kad išlaikyti populiacijos įvairovę. Pateikiu standartinio genetinio algoritmo pseudokodą:

```
Genetinis Algoritmas
{
  Generuoti pradinę populiaciją Pt
  Patikrinti populiaciją Pt
  Kartoti kol sustojimo kriterijus netenkinamas
  {
   Išrinkti elementus iš Pt kurie bus kopijuojami į Pt+1
   Kryžminti elementusiš Pt ir įdėti į Pt+1
   Mutuoti elementus iš Pt ir įdėti į Pt+1
   Patikrinti naują populiaciją Pt+1
   Pt = Pt+1
  }
}
```

Kad gauti gerus palikuonis, genetiniu algoritmu yra apdorojama atsitiktinai sugeneruota pradinė populiacija. Gali būti daugybė genetinio algoritmo variantų, keičiant

mutacijos, kryžminimo ir kitus reprodukcijos operatorius. Atrinkimo ir kryžminimo operatoriai nusako, kurie individai taps tėvais ir kurie tėvų genai bus naudojami formuojant jų palikuonį. Mutacijos operatorius suteikia atsitiktinius pokyčius populiacijoje. Mutacijos operatorius neleidžia metodui konvertuoti per greitai (t.y. jei tinkamai parinktas, neleidžia konvertuoti į lokalų minimumą), taip pat įveda naujos informacijos į populiaciją.

Aptarsiu mano naudota evoliucinę strategija. Reprodukcijos metu pirmiausiai išrenkami keletas pačių geriausių individų kurie pateks ir į sekančią populiaciją (tai vadinama *elitine strategija*). Tokios reprodukcijos privalumas prieš paprastą tikimybinę reprodukciją yra tai, kad geriausias populiacijos individas monotoniškai mažėja kiekvienos iteracijos metu. Tačiau jei parametrai netinkamai parinkti, gali pasireikšti ir elitinės strategijos trūkumas, t.y. konvergavimas į lokalų minimumą. Didesnės mutacijos parametro reikšmės dažniausiai leidžia šio trūkumo išvengti.

Geriausi sprendiniai įtraukiami į mutuoti negalinčių individų sąrašą. Tai padeda užtikrinti nuoseklesnį konvergavimą. Tam, kad paspartinti konvergavimą blogiausi sprendiniai pakeičiami atsitiktinai sugeneruotais. Visi likę sprendiniai gali mutuoti.

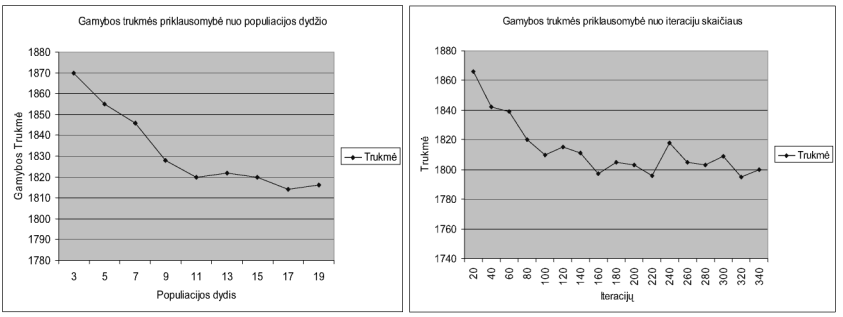
Genetinio algoritmo tyrimas

Kadangi genetinio algoritmo gaunamiems rezultatams didelės įtakos turi jo parametrų parinkimas, atlikome jų įtakos tyrimą. Šio tyrimo pagalba norėjome išsiaiškinti kokiu būdu skirtingi parametrai įtakoja sprendinius ir kaip juos parinkti kiek įmanoma optimaliau. 2 ir 3 pav. pateikiami populiacijos dydžio, iteracijų skaičiaus, kryžminimo ir mutavimo tyrimo grafiniai rezultatai.

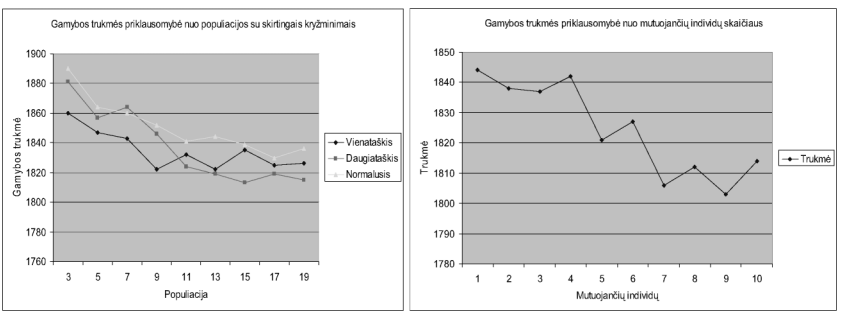
Tyrimo rezultatai buvo gauti nagrinėjant tuos pačius pradinius duomenis keičiant minėtų parametrų reikšmes. Iš gautų rezultatų galima daryti išvadą, kad algoritmo gaunamiems rezultatams populiacijos dydis turi panašų poveikį kaip ir iteracijų skaičius. Abiejų šių parametrų didinimas leidžia tikėtis geresnio rezultato padidėjusio resursų sunaudojimo kaina, tačiau didinant parametrų reikšmes sprendinio trukmė konverguoja ir pernelyg didelių parametrų reikšmių parinkimas būtų nepraktiškas. Taigi kiekvienam konkrečiam uždaviniui atlikus panašų tyrimą galima rasti optimalias šių parametrų reikšmes. Analogiškos išvados tinka ir mutuojančių individų skaičiui. Taip pat algoritmo efektyvumas gali priklausyti ir nuo pasirinktos kryžminimo schemos. Kiekvienam uždaviniui efektyviausias kryžminimas gali būti rastas eksperimentiškai.

Skruzdžių kolonijos optimizavimo algoritmas

Tolimesniame darbe genetinį algoritmą lyginsime su dažnai naudojamu skruzdžių kolonijos optimizavimo algoritmu (toliau ACO). Kaip ir genetinis, ACO tinkamas spęsti šio tipo uždaviniam dėl savo universalumo ir lankstumo naujiems apribojimams. ACO algoritmo principas paremtas natūraliu skruzdžių elgesiu gamtoje: skruzdės keliauja nuo lizdo iki maisto šaltinio, trumpiausias kelias randamas feromonų pėdsakų pagalba. Kiekviena skruzdė juda atsitiktinai ir savo kelyje palieka feromonų pėdsakus. Pėdsakai sekančias skruzdes skatina taip pat rinktis šį kelią o daugiau feromonų kelyje didina tikimybę, kad šiuo keliu seks dar daugiau skruzdžių. ACO pranašumas prieš standartinį genetinį algoritmą yra tai, kad jis išlaiko atmintį apie visos kolonijos veiklą,



2 pav. Vienos kartos populiacijos dydžio ir iteracijų skaičiaus analizė.



3 pav. Kryžminimo ir mutavimo analizė.

ne tik paskutiniosios generacijos. Taip pat ACO mažiau jautrus prastiems pradiniams sprendiniams dėl atsitiktinės kelio pasirinkimo strategijos ir kolonijos atminties.

Literatūra

1. S. Van Otterloo, *Evolutionary Algorithms and Scheduling Problems* (2002).
2. J.F. Gonçalves, J.J. de Magalhães Mendes, M.G.C. Resende, *A Hybrid Genetic Algorithm for the Job Shop Scheduling Problem* (2002).
3. O. Kämäräinen, V. Ek, K. Nieminen, S. Ruuth, *Large Scale Generalized Resource Constrained Scheduling Problems: A Genetic Algorithm Approach* (2000).
4. B. Carter, K. Park, *How Good are Genetic Algorithms at Finding Large Cliques: an Experimental Study* (1993).
5. S.-C. Lin, E.D. Goodman, W.F. Punch III, *Investigating Parallel Genetic Algorithms on Job Shop Scheduling Problems* (1996).
6. M. Ventresca, B.M. Ombuki, *Ant Colony Optimization for Job Shop Scheduling Problem* (2004).

SUMMARY

J. Simonavičius, N. Listopadskis. The analysis of machine scheduling problem algorithm

The analysis of machine scheduling problem and possible algorithms for solving it are presented in this paper.

Keywords: combinatorial optimization, job shop scheduling problem, genetic algorithm, elitist strategy.