Liet. matem. rink, **45***, spec. nr.*, 2005, 119–123

Procesorių apkrovos balansavimas lygiagrečiuose šakų ir rėžių algoritmuose*

Milda BARAVYKAITĖ (VGTU)
el. paštas: mmb@fm.vtu.lt

1. Įvadas

Darbe aprašomas šakų ir rėžių (ŠR) algoritmų lygiagretinimo įrankio kūrimas. Siekiama apjungti įvairius ŠR algoritmus ir palengvinti vartotojui eksperimentavimą su nuosekliaisiais ir lygiagrečiaisiais ŠR algoritmais. Įdiegus paprastus duomenų srities padalinimo tarp procesorių algoritmus, vienas iš lygiagrečiųjų programų efektyvumą mažinančių veiksnių yra išaugantis procesorių apkrovos disbalansas [1]. Dėl to į šabloną įdiegtas papildomas dinaminio apkrovos balansavimo modulis.

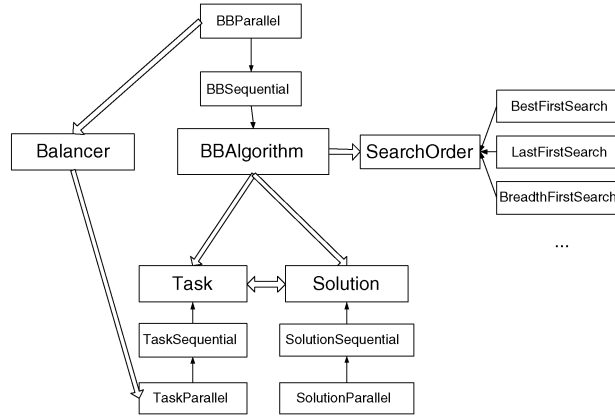
2 skyriuje aprašytas ŠR algoritmo šablono struktūra, 3 skyriuje aptartas dinaminis apkrovos balansavimas ir 4 skyriuje pateikti skaičiavimo eksperimentų rezultatai.

2. Šakų ir rėžių algoritmo šablonas

Nuoseklieji ir lygiagretieji (ŠR) algoritmai yra plačiai analizuoti. ŠR algoritmai gali būti pritaikyti įvairiems uždaviniams spręsti [6,2]. Atskyrus bendrąją visiems uždaviniams ŠR algoritmo dalį galima formuoti algoritmo šabloną, kuris atlieka veiksmus su objektais, priklausančiais nuo sprendžiamo uždavinio. Šablone siekiama apjungti įvairius ŠR algoritmų variantus ir leisti vartotojui nesunkiai eksperimentuoti tiek su nuosekliaisiais tiek su lygiagrečiaisiais algoritmais. Naudojant šabloną, lygiagrečiosios programos yra gaunamos automatiškai. Į šabloną siekiama įtraukti standartinius metodus, paliekant galimybę šabloną praplėsti. 1 pav. pateikiama ŠR algoritmo šablono C++ klasių schema.

BBAlgorithm – tai apibendrinta šakų ir rėžių algoritmų klasė. Naudojant objektinio programavimo paveldimumo technologiją, gaunami įvairūs nuoseklieji algoritmai, o iš jų – lygiagretieji algoritmai. *Task* – užduoties klasė, kurioje turi būti apibrėžti užduoties arba tyrimų srities inicializavimas, srities dalinimas (šakos žingsnis), sprendinių toje srityje rėžio skaičiavimas. Naudojant lygiagrečiuosius ŠR algoritmus, papildomai reikia paruošti užduotį perdavimui tarp procesorių (*Pack*) ir paėmimui po persiuntimų (*Unpack*) metodus. *Solution* klasėje apibrėžiamas sprendžiamo uždavinio sprendinys. Naudojant lygiagrečiuosius algoritmus su geriausio sprendinio perdavimu,

*Darbe pateikti rezultatai gauti vizito IDRIS superkompiuterių centre metu, remiant HPC_Europe programai.



1 pav. Šakų ir rėžių algoritmų lygiagretinimo įrankio klasių schema.

reikia apibrėžti sprendinio paruošimo persiuntimui ir paruošimo naudojimui po persiuntimo metodus. *Task* ir *Solution* priklauso nuo sprendžiamo uždavinio, todėl jie turi būti pateikti šablono vartotojo. *SearchOrder* – užduočių tyrimo tvarkos klasė. Užduočių tyrimo tvarka gali stipriai įtakoti algoritmo vykdymo efektyvumą, tačiau tvarkos aprašymas nesiskiria nuosekliam ar lygiagrečiam algoritmui. *Balancer* dinaminio apkrovos balansavimo klasė.

3. Dinaminis procesorių apkrovos balansavimas

ŠR šablone įdiegti lygiagretieji algoritmai remiasi pradinės tyrimo srities padalinimu tarp procesorių. Vienas šių algoritmų efektyvumą mažinančių veiksnių yra procesorių apkrovos disbalansas [1]. ŠR algoritme užduotys generuojamos dinamiškai algoritmo vykdymo metu. Dėl to, sprendimo pradžioj padalinant užduotis nėra žinoma, kiek sričių bus generuota ir galimas atvejis, kad kai kurie procesoriai generuos daugiau užduočių už kitus, dėl to dirbs ilgiau ir lygiagretusis algoritmas bus neefektyvus.

Duomenų lygiagretumo atveju, kai procesoriaus apkrova tiesiogiai susijusi su gautu duomenų skaičiumi, apkrovai balansuoti naudojamas statinis balansavimas. Tačiau ŠR algoritmams statinis balansavimas neduoda gerų rezultatų. Norint gerinti šablone įdiegtų algoritmų efektyvumą, siekiama, kad pabaigęs tirti algoritmo pradžioje gautas užduotis, procesorius gautų užduočių iš labiau apkrautų procesorių, jei tokių yra.

Dinaminiam apkrovos balansavimui būtini šie žingsniai [5]:

- apkrovos matavimas;
- apkrovos informacijos pasikeitimo inicijavimas;
- balansavimo inicijavimas;
- balansavimo operacija:
  - balansavimo partnerio pasirinkimas,
  - užduočių perskirstymo taisyklė,
  - perskirstomų užduočių pasirinkimo taisyklė,
  - užduočių perdavimas;

- skaičiavimų pabaigos taisyklė.

Kuriant balansavimo modulį, siekiama leisti išbandyti įvairius balansavimo algoritmus, įdiegti dažniausiai naudojamus algoritmus, paliekant galimybę praplėsti šabloną.

Šablone naudojamas paprasčiausias *apkrovos matas* ŠR algoritmams – dar neištirtų užduočių skaičius [3]. Tačiau šis matas nėra tikslus. Reikėtų taip pat bandyti įvertinti, kiek perspektyvi yra užduotis. Tokiu atveju naudojamas ne tik užduočių skaičius, bet ir užduočių rėžis ir užduočių vieta tyrimo eilėje [5]. Sudėtingesnis apkrovos mato vertinimas gali priklausyti nuo sprendžiamo uždavinio ir turi būti įdiegtas šablono vartotojo.

Apkrovos informacijos apsikeitimo inicijavimas gali priklausyti nuo procesoriaus apkrovos būsenos, t.y. inicijuojamas, kai procesorius yra labai apkrautas, arba neapkrautas, arba periodinis. Šablone įdiegtas algoritmas, kad apkrovos informacijos pasikeitimą inicijuotų procesorius, kurs nebeturi neištirtų užduočių.

Balansavimo operacijos inicijavimo žingsnis apima gautos apkrovos informacijos apdorojimą ir balansavimo partnerių pasirinkimą. Dinaminis apkrovos balansavimas – sudėtinga procedūra reikalaujanti papildomų skaičiavimų ir duomenų perdavimų, dėl to ne visada verta ją atlikti. Tegul T_1 – laikas, kurį užtruktų labiausiai apkrautas procesorius vykdydamas užduotis, nenaudojant balansavimo; T_2 – laikas, kurį užtruktų labiausiai apkrautas procesorius vykdydamas užduotis, naudojant balansavimą; T_3 – užduočių paruošimo perdavimui laikas, T_4 – užduočių perdavimo laikas. Balansavimą verta vykdyti, kai $T_1 \geqslant T_2 + T_3 + T_4$. Įdiegta taisyklė, kad balansavimas inicijuojamas, jei bent iš vieno procesoriaus gauta informacija rodo, kad jo apkrova didesnė už nurodytą užduočių skaičių. Šis skaičius priklauso nuo sprendžiamo uždavinio. Apkrovos informacijos pasikeitimas vykdomas asinchroniškai, balansavimo inicijavimo sprendimą priima balansavimo informaciją surinkęs procesorius. Įvertinama, kad siunčiama apkrovos informacija gali ateiti pavėluotai, ir gali keistis laikui bėgant. Nepavykus susisiekti su labiausiai į partnerius tinkančiu (stipriausiai apkrautu) procesoriumi, bandoma vykdyti balansavimą su kitais tinkamais partneriais.

Balansavimo operacija vykdoma sinchronizuotai tarp dviejų procesorių: užduočių siuntėjo ir užduočių gavėjo. Užduočių gavėjas praneša užduočių siuntėjui, kad šis persiųstų užduotis. Užduočių siuntėjas laikinai sustabdo ŠR algoritmo vykdymą ir pagal *užduočių paskirstymo* ir *užduočių pasirinkimo* taisykles ruošia užduotis persiuntimui. Šios taisyklės gali priklausyti nuo apkrovos mato ir naudojamos užduočių tyrimo eilės. *Užduočių paskirstymo* taisykle nustatoma, kokiu santykiu padalinti užduotis, esančias eilėje, o *pasirinkimo* taisykle – kurias užduotis siųsti. Tyrimai teigia, kad pusės užduočių perdavimas nėra efektyvus, o nusiuntus netinkamas užduotis, balansavimo operacija gali neduoti norimo rezultato [5]. Įvykdžius šias taisykles, užduotys yra paruošiamos perdavimui, o po perdavimo abiejuose procesoriuose toliau vykdomas ŠR algoritmas.

Įdiegus dinaminį apkrovos balansavimo algoritmą tapo sudėtinga nustatyti skaičiavimų pabaigą. Yra nemažai siūlymų, kaip greičiausiai nustatyti, kada nebeieškoti balansavimo partnerių. Šiuo metu įdiegtas paprasčiausias algoritmas, kai procesoriai vykdo balansavimo operacija tik nustatytą skaičių kartų. Šablonas gali būti praplėstas kitomis pabaigos nustatymo taisyklėmis, aprašytomis [3].

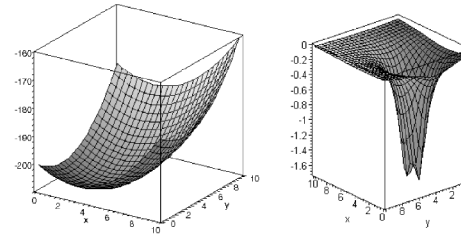
4. Skaičiavimo eksperimentų rezultatai

Skaičiavimo eksperimentai su lygiagrečiosiomis programomis, gautomis naudojant ŠR šabloną, sprendžiant įvairius Lipšico funkcijos minimizavimo ir keliaujančio pirklio uždavinius aprašyti [1]. Šiame straipsnyje aprašytais eksperimentais siekta išbandyti balansavimo modulio tinkamumą ŠR šablonui. Pasirinktas Lipšico funkcijos minimizavimo dvimatėje srityje uždavinys [4]. 2 pav. parodyti tiriamų funkcijų grafikai. Skaičiavimai buvo atlikti IDRIS superkompiuterių centre (`www.idris.fr`).

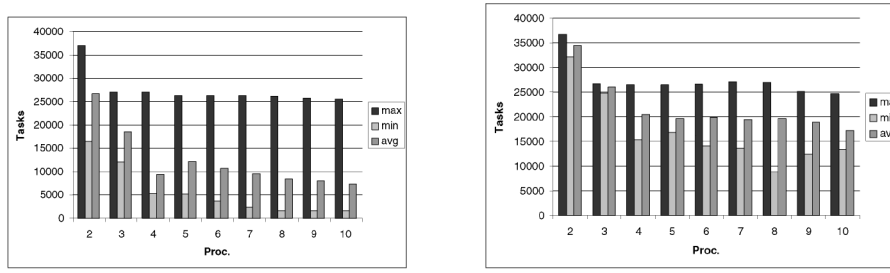
Skaičiavimo eksperimentuose tirta, kiek skiriasi daugiausiai ir mažiausiai apkrautų procesorių ištirtų užduočių skaičiai. Vidutinis procesoriaus ištirtų užduočių skaičius parodo, ar daug procesorių, spręsdami uždavinį ištiria po mažai užduočių. 3 pav. *a* dalyje parodyti $f_1(x)$ minimalus, maksimalus ir vidutinis ištirtų užduočių skaičius, nenaudojant dinaminio apkrovos balansavimo, o *b* dalyje – panaudojus. Iš rezultatų matyti, kad nors labiausiai apkrauto procesoriaus ištirtų užduočių skaičius sumažėjo nedaug, tačiau kiti procesoriai tuo metu vykdė naudingą darbą.

Eksperimentų su $f_2(x)$ funkcija rezultatai pateikti 4 pav. grafikuose. Naudojant pasirinktas balansavimo taisykles, gauti nepriimtini balansavimo rezultatai. Nors pavyko sumažinti procesorių apkrovos disbalansą, tiek bendras, tiek maksimalus ištirtų užduočių skaičius išaugo. Tai rodo pasirinktų taisyklių netinkamumą duotam atvejui.

Iš skaičiavimo rezultatų matyti, kad procesoriai, pabaigę savo užduotis, toliau tyrė užduotis, gautas iš kitų procesorių. Dėl to sumažėjo skirtumas tarp labiausiai ir mažiausiai apkrautų procesorių ištirtų užduočių skaičiaus.

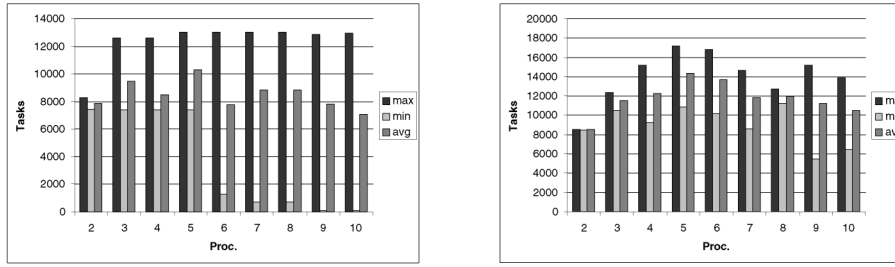


2 pav. Skaičiavimo eksperimentuose tirtų funkcijų $f_1(x)$ ir $f_2(x)$ grafikai.



a) *b)*

3 pav. Funkcijos $f_1(x)$ apkrovos disbalansas, nenaudojant ir panaudojus apkrovos balansavimą.



a) *b*)

4 pav. Funkcijos $f_2(x)$ apkrovos disbalansas, nenaudojant ir panaudojus apkrovos balansavimą.

5. Išvados

Vienas iš lygiagrečiųjų ŠR programų efektyvumą mažinančių veiksnių yra išaugantis procesorių apkrovos disbalansas. Papildžius ŠR šabloną dinaminio apkrovos balansavimo moduliu, siekiama pagerinti lygiagrečiųjų algoritmų efektyvumą. Dinaminis apkrovos balansavimas yra sudėtingas procesas, kai kurie proceso žingsniai gali priklausyti nuo sprendžiamo uždavinio ir turi būti patikslinti įrankio vartotojo. Panaudojus labai paprastus dinaminio apkrovos balansavimo žingsnius, pavyko sumažinti procesorių apkrovos disbalansą $f_1(x)$ funkcijai. Tačiau funkcijos $f_2(x)$ rezultatai rodo, kad įrankį reikia papildyti įvairiasniais dinaminio balansavimo algoritmais.

Literatūra

1. M. Baravykaitė, Lygiagrečiųjų šakų ir rėžių algoritmų programų kūrimas, *Liet. matem. rink.*, **44** (spec. nr.) (2004).
2. J. Clausen, Parallel search-based methods in optimization, in: *Applied Parallel Computing – Industrial Computation and Optimization*, *Proceedings of PARA96*, *Lecture Notes in Computer Science*, vol. 1184, Springer (1996), pp. 176–185.
3. A. Grama, A. Gupta, G. Karypis, V. Kumar, *Introduction to Parallel Computing*, second edition, Addison Wesley (2003).
4. R. Horst, P.M. Pardalos, N.V. Thoai, Introduction to global optimization, in: *Nonconvex Optimization and its Applications*, vol. 48, Kluwer Academic Publishers (2000), pp. 237–265.
5. C. Xu, F. Lau, *Load Balancing in Parallel Computers. Theory and Practice*, Kluwer Academic Publishers (1997).
6. J. Žilinskas, Parallel algorithms for Lipschitz global optimization with simplicial partitioning, *Information Technology and Control*, **4**(25), 32–36 (2002).

SUMMARY

M. Baravykaitė. Processor load balancing for parallel branch and bound algorithms

In this atricle the development of parallel branch and bound algorithm template is presented. Attention is focused on the dynamic load balancing module of the template. The structure of dynamic load balancing is disscussed and some results of calculation experiments are presented.

Keywords: dynamic load balancing, parallel template, branch and bound algorithms.