



Milda Marija BARAVYKAITĖ

**INVESTIGATION AND DEVELOPMENT OF
PARALLEL ALGORITHM TEMPLATES**

**Summary of Doctoral Dissertation
Technological Sciences, Informatics Engineering (07T)**

1276

VGTU

**LEIDYKLA
TECHNIKA
Vilnius 2006**

VILNIUS GEDIMINAS TECHNICAL UNIVERSITY

Milda Marija BARAVYKAITĖ

**INVESTIGATION AND DEVELOPMENT OF
PARALLEL ALGORITHM TEMPLATES**

Summary of Doctoral Dissertation
Technological Sciences, Informatics Engineering (07T)



Doctoral dissertation was prepared at Vilnius Gediminas Technical University in 2002–2006

Scientific Supervisor

Prof Dr Habil Raimondas ČIEGIS (Vilnius Gediminas Technical University, Technological Sciences, Informatics Engineering – 07T)

The Dissertation is being defended at the Council of Scientific Field of Informatics Engineering at Vilnius Gediminas Technical University:

Chairman

Prof Dr Habil Gintautas DZEMYDA (Institute of Mathematics and Informatics, Technological Sciences, Informatics Engineering – 07T)

Members:

Prof Dr Habil Rimantas BARAUSKAS (Kaunas University of Technology, Physical Sciences, Informatics – 09P)

Assoc Prof Dr Romas BARONAS (Vilnius University, Physical Sciences, Informatics – 09P)

Prof Dr Habil Romualdas BAUŠYS (Vilnius Gediminas Technical University, Technological Sciences, Informatics Engineering – 07T)

Prof Dr Habil Jonas MOCKUS (Institute of Mathematics and Informatics, Technological Sciences, Informatics Engineering – 07T)

Opponents:

Prof Dr Habil Aleksandr DEMENTJEV (Institute of Physics, Physical Sciences, Physics – 02P)

Assoc Prof Dr Olegas VASILECAS (Vilnius Gediminas Technical University, Technological Sciences, Informatics Engineering – 07T)

The dissertation will be defended at the public meeting of the Council of Scientific Field of Informatics Engineering in the Senate Hall of Vilnius Gediminas Technical University at 10 a. m. on 30 June 2006.

Address: Saulėtekio al. 11, LT-10223 Vilnius-40, Lithuania

Tel.: +370 5 274 4952, +370 5 274 4956; fax +370 5 270 0112;

e-mail: doktor@adm.vtu.lt

The summary of the doctoral dissertation was distributed on 30 May 2006.

A copy of the doctoral dissertation is available for review at the Library of Vilnius Gediminas Technical University (Saulėtekio al. 14, Vilnius, Lithuania)

© Milda Baravykaitė, 2006

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS

Milda Marija Baravykaitė

**LYGIAGREČIŲJŲ ALGORITMŲ ŠABLONŲ
TYRIMAS IR KŪRIMAS**

Daktaro disertacijos santrauka
Technologijos mokslai, informatikos inžinerija (07T)



Disertacija rengta 2002–2006 metais Vilniaus Gedimino technikos universitete.

Mokslinis vadovas

prof. habil. dr. Raimondas ČIEGIS (Vilniaus Gedimino technikos universitetas, technologijos mokslai, informatikos inžinerija – 07T).

Disertacija ginama Vilniaus Gedimino technikos universiteto Informatikos inžinerijos mokslo krypties taryboje:

Pirmininkas

prof. habil. dr. Gintautas DZEMYDA (Matematikos ir informatikos institutas, technologijos mokslai, informatikos inžinerija – 07T).

Nariai:

prof. habil. dr. Rimantas BARAUSKAS (Kauno technologijos universitetas, fiziniai mokslai, informatika – 09P),

doc. dr. Romas BARONAS (Vilniaus universitetas, fiziniai mokslai, informatika – 09P),

prof. habil. dr. Romualdas BAUŠYS (Vilniaus Gedimino technikos universitetas, technologijos mokslai, informatikos inžinerija – 07T),

prof. habil. dr. Jonas MOCKUS (Matematikos ir informatikos institutas, technologijos mokslai, informatikos inžinerija – 07T).

Oponentai:

prof. habil. dr. Aleksandr DEMENTJEV (Fizikos institutas, fiziniai mokslai, fizika – 02P),

doc. dr. Olegas VASILECAS (Vilniaus Gedimino technikos universitetas, technologijos mokslai, informatikos inžinerija – 07T).

Disertacija bus ginama viešame Informatikos inžinerijos mokslo krypties tarybos posėdyje 2006 m. birželio 30 d. 10 val. Vilniaus Gedimino technikos universiteto senato posėdžių salėje.

Adresas: Saulėtekio al. 11, LT-10223 Vilnius-40, Lietuva.

Tel.: +370 5 274 4952, +370 5 274 4956; faksas +370 5 270 0112;

el. paštas doktor@adm.vtu.lt

Disertacijos santrauka išsiuntinėta 2006 m. gegužės 30 d.

Disertaciją galima peržiūrėti Vilniaus Gedimino technikos universiteto bibliotekoje (Saulėtekio al. 14, Vilnius, Lietuva)

VG TU leidyklos „Technika“ 1276 mokslo literatūros knyga

GENERAL DESCRIPTION

Topicality of the problem. Parallel computations are used in many areas where large computation amounts are required and/or faster response is critical. The development of parallel algorithms and parallel programming are complicated and require additional knowledge. To ease the parallel programming and algorithm parallelization many tools are developed. One class of such tools is algorithm templates. The idea of algorithm templates is to separate the problem specific and general parts of the algorithm. The general part can be implemented as a reusable module. The problem specific part of the algorithm should be implemented by the user. When these ideas are used for parallel programming, the template has to specify the main parts of the parallel algorithm: partitioning, communications, mapping, synchronization.

Master – slave algorithms are very popular and widely used in parallel programming. It is efficient both for homogeneous and heterogeneous parallel computers. Parallelization tools of master – slave algorithms using popular communication libraries are useful.

Many problems of engineering, economics and other fields are formulated as optimization problems. Optimization problems are often complicated and computation intensive. Therefore the creation of automatic parallelization tools of popular branch and bound optimization algorithms is relevant.

Grillage type foundations are popular in case of weak grounds. It consists of beams that are supported by piles or reside on other beams. In order to reduce the number of piles preserving the required safety the computation intense optimization is required and parallel calculations are useful.

The main aims and tasks of the work are:

1. To extend the functionality of the master – slave algorithm template by including the possibility to use it with different communication libraries. For this purpose the template layer structure is formed and it is adjusted to use PVM and MPI communication libraries.
2. To create the branch and bound algorithm template that would allow:
 - a) to test different variants of sequential branch and bound algorithm,
 - b) automatic parallelization of branch and bound algorithms,
 - c) to test new algorithms based on branch and bound algorithm logic.
3. To use parallel global optimization algorithms for optimization of grillage type foundations.

Scientific novelty. The master – slave algorithm template is an automatic parallelization tool of the popular algorithm. The layer structure of the template

allows to use the tool with different communication libraries and to extend its functionality.

The tool for branch and bound algorithm that can be used for sequential algorithm testing, automatic creation of parallel branch and bound algorithm programs and new branch and bound based algorithms testing has a wide functionality. Modified black box global optimization algorithm based on branch and bound logic is introduced and tested. Parallel global optimization algorithms are used for optimization of grillage type foundations and allow us to investigate complicated load and geometry cases.

Methodology of research includes algorithm templates, algorithm theory, parallel algorithm investigation, experimental investigations. C, C++ programming languages, object oriented technology, PVM and MPI communication libraries.

Practical value. Automatic parallelization tools allow easy development of parallel master – slave and branch and bound algorithm based programs. Master – slave parallelization tool is effective for homogeneous and heterogeneous parallel computers with both MPI and PVM libraries. Branch and bound algorithm template has wide applications: sequential testing, parallelization and new algorithm development. The introduced black box global optimization algorithm does not require any additional information about the objective function and is useful for practical applications. Parallel global optimization algorithm application for engineering problem allows us to investigate wide range of problems.

The scope of the scientific work. The doctoral dissertation consists of the introduction, two chapters, general results and conclusions and the list of references. The total scope of the doctoral dissertation is 84 pages, 32 pictures, 7 tables. The language of doctoral dissertation is Lithuanian.

CONTENT

In the introduction the topicality of the problem is defined, the goals and tasks of the research are formulated, the scientific novelty of the dissertation and the practical value of the work are presented. The analysis of parallel algorithm design and template programming is given. The main steps of parallel algorithm design are: partitioning, communication, agglomeration and mapping. Levels of parallelization vary from ones done from scratch by the user to semi automatic and automatic. Processor load balance is very important for efficient execution of parallel algorithms. The main types of processor load balancing are static and dynamic. Dynamic load balancing is a complicated process that consists of load measurement, load information exchange initiation, load

balancing initiation, load balancing operation and termination detection. Algorithm templates can be used to implement semi automatic and automatic parallelization tools. They have to cover main steps of parallel algorithm development. Algorithm templates have to separate problem dependent and independent parts of the algorithm, have to enable hierarchical usage of templates, to be platform independent and efficient.

Chapter 1. Master – slave algorithm template

Master – slave algorithm is very popular parallel algorithm often used as centralized dynamic load balancing algorithm. It implements MIMD computation strategy and defines two types of processes (master and slaves) performing different functions over different data. Master process generates the job pool and distributes jobs among the slave processes. Slaves perform calculations of received jobs and send the results back to the master. Generally jobs can be of different complexity, there can be several dynamically formed job pools, if necessary calculations of jobs can be repeated. The partitioning is performed by forming the job pool. Only communications between master and slave are allowed. If necessary, agglomeration can be performed while forming the job pool. Mapping is performed by the master process. The general master – slave algorithm is presented:

Master's part:

```

Formulate the job pool  $P_i, i = 1, 2, \dots, M$ .
Send the task from  $P_i$  to every slave;
while (not all  $M$  tasks are returned) do
    receive the result from slave  $s_j$ ;
    add the received result to the totals;
    if (the job pool is not empty) do
        Send not processed task  $P_k$  to the  $j$ th slave;
    else send the termination signal to the  $j$ th slave.
    end if
end do

```

Slave's part

```

while (termination is not received) do
    receive task  $P_k$  from the master process;
    solve  $P_k$ ;
    send the result to the master.
end do

```

The aim of template programming is to save the user from doing parallel programming from scratch. To achieve this the master – slave algorithm template is organized into layers, presented in figure 1.

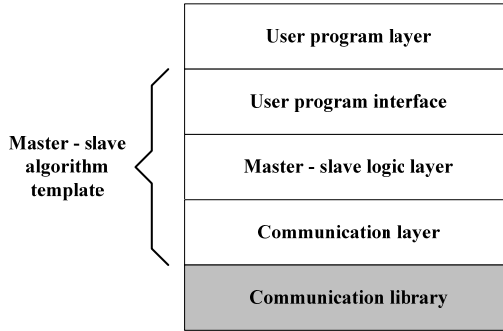


Fig 1. Scheme of the master – slave algorithm template layer structure

The user program layer consists of the user program that has to be formulated according to the requirements of the template. The structure of the main part of the program is the following:

```

struct t_init_data id;
struct t_data      t;
struct t_result    res;

M_prepare_job_pool(&id)
if ( ! S_initialize(id) ) return 0;
while ( M_take_piece_from_pool(&t))
{
    S_compute(t, &res);
    M_add_to_result(res);
}
M_print_result();

```

Here `M_prepare_job_pool(&id)` is a master procedure that initializes master process and forms the `t_init_data` structure that is passed to the slave processes. `S_initialize(id)` is the slave procedure that receives `t_init_data` and initializes slave process. `M_take_piece_from_pool(&t)` is a master procedure that takes one job from the job pool and forms `t_data` structure to describes the taken job. The structure sends it to the slave. `S_compute(t, &res)` performs calculations of the job and forms the result structure `t_result`. `M_add_to_result(res)` is a master procedure that summarizes the received results and `M_print_result()` prints out the totals of the calculations.

User program interface layer includes the parser that transforms data structures used by the user into ones used by the template and links user procedures with master – slave logic procedures. Master – slave logic layer implements main steps of master – slave algorithm logic. Here additional features of the algorithm like multiple job pool formulation and job recalculation are allowed. Communication layer links chosen communication library with the template. Because of the layer structure the template can be used with different communication libraries, it was tested using PVM and MPI.

The efficiency of obtained parallel programs was tested calculating prime numbers using the sieve of Erathosten. Calculations were performed on RS/6000 cluster and VGTU cluster vilkas.vtu.lt, using PVM and MPI. The efficiency higher than 0.97 was obtained in all cases.

Global optimization algorithm parallelized using master – slave algorithm template was used for optimization of grillage type foundations. Grillage consists of separate beams, which may be supported by piles, or may reside on other beams, or a mixed scheme may be the case. The optimal scheme of grillage should posses the minimum possible number of piles. Theoretically, reactive forces in all piles should approach the limit magnitudes of reactions for those piles. This goal can be achieved by choosing appropriate pile positions. The random search method was used: a number of points were generated in the feasible region. Then the local minimization was performed for different initial conditions. Since all problems can be solved independently, the master – slave parallel algorithm was suited for parallelization of the whole job. The parallel master – slave algorithm distributes jobs dynamically and good efficiency of parallel execution was obtained. The algorithm was tested for a single beam with trapezoidal distributed loading. The obtained pile placement scheme gives nearly equal reaction forces for all supports.

Chapter 2. Branch and bound algorithm template

Many problems in engineering, physics, economic and other subjects may be formulated as optimization problems, where the minimum value of an objective function should be found. Mathematically the problem is formulated as follows:

$$f^* = \min_{X \in D} f(X), \quad (1)$$

where $f(X)$ is an objective function, X are decision variables, and D is a search space. Besides of the minimum f^* , one or all minimizers $X^* : f(X^*) = f^*$ should be found. Type of optimization is defined by the search space and the objective function of the problem. In combinatorial optimization the search space is discrete. In global optimization the objective function is a nonlinear function of continuous variables $f : \mathfrak{R}^n \rightarrow \mathfrak{R}$, where n is number of variables, and a continuous search space $D \subseteq \mathfrak{R}^n$ is called feasible region.

Branch and bound algorithm defines the technique for managing the list of sub-spaces, the process of discarding the sub-spaces and their partitioning. It can be used both for combinatorial and global optimization. An iteration of the classical branch and bound algorithm processes a node in the search tree representing a not yet explored sub-space of the search space. Iteration has three main components: selection of the node to process, branching of the search tree by dividing the selected sub-space and bounding of the branches by discarding not promising sub-spaces. Bounding rule implements the calculation of the lower and upper bounds of function value over the sub-space. Before the cycle of iterations the list of candidate sub-spaces should be initialized by covering search space by one or more sub-spaces.

Main strategies of selection of the node to process are the following:

- Best first. Select a candidate with minimal lower bound.
- Depth first. Select the youngest candidate.
- Breadth first. Select the oldest candidate.
- Improved selection. It is based on heuristic or probabilistic criteria.

In combinatorial optimization ‘leaf’ of search tree means that the node is a solution, in global optimization it means that it is a small sub-region bracketing a solution with predefined accuracy.

Branch and bound algorithms have general scheme and differ only by rules of covering, selection, branching and bounding. Algorithm and sub-space selection rules can be separated from the problem specific branching and bounding. Generalized branch and bound algorithm can be implemented as the template for development of sequential branch and bound algorithms. Only particular rules should be implemented by the template user.

The general branch and bound algorithm is presented, where L denotes candidate set, S the solution, $UB(D_i)$ and $LB(D_i)$ denote upper and lower bounds for minimum value of the objective function over sub-space D_i .

Cover solution space D by $L = \{L_j \mid D \subseteq \bigcup L_j, j = \overline{1, m}\}$ using *covering rule*.

$S = \emptyset, UB(D) = \infty$.

while (sub-space list is not empty $L \neq \emptyset$) **do**

 Choose $I \in L$ using *selection rule*, exclude I from L .

if ($LB(I) < UB(D) + \varepsilon$) **then**

 Branch I into p subsets I_j using *branching rule*.

for all ($I_j, j = \overline{1, p}$) **do**

 Find $UB(I_j \cap D)$ and $LB(I_j)$ using *bounding rules*.

$UB(D) = \min(UB(D), UB(I_j \cap D))$

if ($LB(I_j) < UB(D) + \varepsilon$) **then**

if (I_j is a leaf) **then** $S = I_j$.

else $L = L \cup \{I_j\}$

end if

end if

end for

end if

end while

Parallel branch and bound algorithms. Several parallel branch and bound algorithms were implemented in the parallel branch and bound algorithm template. Considering a parallel execution, any subspace of the search space can be searched independently and in any order. Implemented algorithms divide initial space into several sub-spaces that are mapped to the processes and branch and bound algorithm is performed. Improved algorithms with the best known function value exchange are implemented.

Parallel optimization algorithms have unpredictably varying unstructured search space. Because of the domain decomposition the order of search can differ for parallel and sequential branch and bound algorithm even using the same sub-space selection rule. The number of nodes in the generated search tree can be used as a unit to measure the complexity of branch and bound algorithm. Then the growth of number of sub-spaces searched using parallel algorithm can be measured by the search overhead factor:

$$SOF = \frac{W_p}{W_0}, \quad (2)$$

where W_p is the sum of processed sub-spaces by parallel algorithm by all processors, and W_o is the sum of sub-spaces processed by the sequential algorithm.

The parallel branch and bound algorithm template consists of C++ classes for implementation of the branch and bound algorithm. A problem to be solved and the solution of the problem should be described by the user. MPI library is used for underlying communications. General structure of the template is given in figure 2.

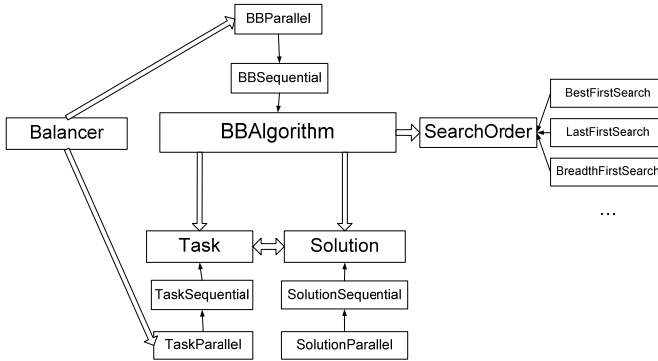


Fig 2. Branch and bound algorithm template class scheme

The template allows to test sequential branch and bound algorithm programs, to obtain parallel programs automatically and to test new algorithms based on branch and bound algorithm logic.

The template was tested solving Traveling salesman problem (TSP) for combinatorial optimization and minimization of different Lipshitz functions. The experiments performed on *vilkas.vtu.lt* and IDRIS supercomputer centre show that the efficiency of implemented parallel algorithms is reduced because of the growth of SOF factor and processor load disbalance. The SOF factor can be reduced using algorithms with best known function value exchange. The static load balancing does not control load disbalance enough. Distributed dynamic load balancing algorithms based on diffusion algorithms are implemented in the branch and bound algorithm template. The experiments show that implemented balancing algorithms reduce processor load disbalance.

The branch and bound algorithm template can be used for branch and bound based algorithm testing. The black box global optimization algorithm was used to optimize grillage type foundations. Computations were performed on *vilkas.vtu.lt* and the optimized pile placement scheme was obtained.

The black box global optimization algorithm was modified to increase the probability of rejection of subspaces that does not contain optimal solution. It is formulated following the main steps of branch and bound algorithm. For calculation of lower and upper bounds M points are chosen in D_i as starting points of Simplex method for local minimum search. If one local minimum is found then the best known function value is updated and the subspace is rejected from further search. If no local minimum is found then the big bound $LB(D_i)$ is assigned and the subspace is rejected from the list L . If several local minima are found then the lower bound is calculated as:

$$MinLM - C(MaxLM - MinLM), \quad (3)$$

where $MinLM$ is the smallest minimum value, $MaxLM$ is the highest minimum value and C is the algorithm parameter. The solution is formed after the branching step is performed fixed number of times over the initial search space. The algorithm was tested with several test functions: with one, several and many local minima and one and several global minima. Algorithm performs fast and correctly for functions with several local minima, in case of several global minima, only one is found. In case of many local minima, algorithm performs slower.

General results and conclusions

1. The master – slave algorithm template was modified by forming the layer structure. This structure allows us to use the tool with different communication libraries. The tool was tested using PVM and MPI communication libraries and efficient parallel programs were obtained (Efficiency higher than 0.9 for test problems).
2. Master – slave algorithm was used for parallelization of the algorithm for optimization of grillage type foundations. The generated parallel program performed efficiently, the obtained pile placement was good and similar to ones obtained using engineering methods.
3. The generalized branch and bound algorithm template was analyzed. The algorithm template was formed by separating the problem dependent parts of the algorithm from the constant algorithm steps. The template allows us to test different variants of branch and bound algorithm and study the relevance of algorithm for particular problem.
4. The parallel branch and bound algorithm template based on the sequential one is introduced. Parallel algorithms based on initial search space decomposition are implemented in the template. The parallel algorithm template allows automatic parallelization of branch and bound algorithm

programs and experimental selection of suitable algorithms for particular problem. The efficiency of parallel branch and bound algorithms based on initial search space decomposition is reduced due to the increased SOF factor and processor load disbalance. SOF factor can be reduced using algorithm with best known function value exchange. Dynamic load balancing module is implemented to reduce the processor load disbalance.

5. The parallel black box global optimization algorithm is applied for optimization of grillage type foundations, it allow us to investigate the cases of complicated geometry and loadings. The modified black box global optimization algorithm is formulated and investigated. It is formulated according to the rules of branch and bound algorithm.

Published works on the topic of the dissertation

In the acknowledged editions

1. Baravykaitė, M.; Belevičius, R.; Čiegis, R. One application of the parallelization tool of master–slave algorithms. *Informatica*, 13 (4), 2002, p. 1–12. ISSN 0868-4952 (ISI Master List).
2. Čiegis, R.; Baravykaitė, M.; Belevičius, R. Parallel global optimization of foundation schemes in civil engineering. *Lecture Notes in Computer Science*, Vol 3732, 2006, Applied Parallel Computing: 7th International Conference, PARA 2004, Lyngby, Denmark, June 20–23, 2004. Revised Selected Papers. Editors: Jack Dongarra, Kaj Madsen, Jerzy Wasniewski, 2006, p. 305–313. ISSN 0302-9743 (ISI Master List).
3. Baravykaitė, M.; Šablinskas, R. The template programming of parallel algorithms. *Mathematical Modelling and Analysis*, 7(1), 2002, p. 11–20. ISSN 1392-6292.
4. Baravykaitė, M.; Čiegis, R.; Žilinskas, J. Template realization of generalized branch and bound algorithm. *Mathematical Modelling and Analysis*, 10(3), 2005, p. 217–236. ISSN 1392-6292.
5. Baravykaitė, M. Lygiagrečiųjų šakų ir režijų algoritmų programų kūrimas. *Lietuvos matematikos rinkinys*, 44, spec. nr., 2004, p. 220–225. ISSN 0132-2818 (in Lithuanian).
6. Baravykaitė, M. Procesorių apkrovos balansavimas lygiagrečiuose šakų ir režijų algoritmuose. *Lietuvos matematikos rinkinys*, 45, spec. nr., 2005, p. 119–123. ISSN 0132-2818 (in Lithuanian).

In the other editions

1. Baravykaitė, M. Parallel branch and bound algorithm template. In: Proceedings of 10th international conference „Mathematical Modelling and Analysis“, 2nd International Conference „Computational Methods in

Applied Mathematics“, June 1–5, 2005, Trakai, Lithuania, Editor: R. Čiegis, p. 165–170. ISBN 9986-05-924-0.

2. Baravykaitė, M. Lygiagrečiųjų programų automatinio generavimo priemonė šeimininkas – darbininkai algoritams. Iš: 5-osios Lietuvos jaunųjų mokslininkų konferencijos „Lietuva be mokslo – Lietuva be ateities“ medžiaga: Matematika ir informatika, 2002, p. 32–38. ISBN 9986-05-528-8 (in Lithuanian).
3. Baravykaitė, M. Lygiagretieji šakų ir režijų algoritmai. Iš: 7-osios Lietuvos jaunųjų mokslininkų konferencijos „Lietuva be mokslo – Lietuva be ateities“ medžiaga: Matematika ir informatika, 2004, p. 51–57. ISBN 996-05-724-8 (in Lithuanian).

LYGIAGREČIŲJŲ ALGORITMŲ ŠABLONŲ TYRIMAS IR KŪRIMAS

Mokslo problemos aktualumas. Daugelyje mokslo sričių susiduriama su uždaviniais, kuriuos išspręsti esamais kompiuteriais užtruktų nepriimtina ilgai arba neįmanoma dėl tų uždavinių dydžio ir skaičiavimų apimtys. Lygiagrečiais skaičiavimais siekiame spręsti uždavinius greičiau, arba spręsti didesnius uždavinius. Tačiau tiek lygiagrečiųjų algoritmų sudarymas, tiek jų programinis realizavimas yra sudėtingi procesai, reikalaujantys daug žinių ir įgūdžių. Taupant laiką ir pastangas, yra kuriami lygiagretinimo įrankiai, lengvinantys įvairių lygiagrečiųjų programų ar metodikų naudojimą.

Šeimininkas – darbininkai algoritmai yra dažnai naudojami ir paplitę įvairiose lygiagrečiųjų skaičiavimų srityse. Lygiagrečiųjų *šeimininkas – darbininkai* algoritmų programų kūrimo automatizavimas palengvina daugelio uždavinių sprendimą. *Šeimininkas – darbininkai* algoritmas efektyvus vykdant tiek homogeniniuose, tiek heterogeniniuose lygiagrečiuosiuose kompiuteriuose. Duomenims tarp procesorių perduoti naudojamos duomenų perdavimo bibliotekos sparčiai vystomos. Nemažai anksčiau kurtų lygiagretinimo įrankių naudoja PVM duomenų perdavimo biblioteką. Išpopuliarėjus MPI duomenų perdavimo bibliotekai aktualu pritaikyti įrankius jos naudojimui.

Daug praktinių uždavinių gali būti formuluojami kaip optimizavimo uždaviniai. Daugelis optimizavimo uždavinių sudėtingi ir reikalaujantys daug skaičiavimų. *Šakų ir režijų* algoritmo principai taikomi kuriant tikslus ir euristinius algoritmus optimizavimo uždavinių sprendimui. Optimizavimo algoritmų lygiagretinimas yra sudėtingas uždavinys. Aktualu ruošti įrankius tiek lengvinančius optimizavimo algoritmų taikymą įvairiems uždaviniams, tiek algoritmų lygiagretinimui, tiek naujų algoritmų kūrimui ir testavimui.

Polinių statinių pamatai naudojami vietose, kur namai statomi ant silpno grunto. Praktiniuose taikymuose aktualu pamatus suprojektuoti taupiai ir nepažeidžiant saugumo reikalavimų. Todėl polinių pamatų optimizavimo uždavinys yra aktualus, tačiau esant sudėtingoms apkrovoms sunkiai išsprendžiamas. Šio uždavinio sprendimui taikant lygiagretinimo priemonės galima gauti sprendinius, artimus optimaliems.

Darbo tikslai ir uždaviniai

- Praplėsti *šeimininkas – darbininkai* algoritmo šablono funkcionalumą, pritaikant naudoti su įvairiomis duomenų perdavimo bibliotekomis.
- Suformuoti *šakų ir rėžių* algoritmo įrankį, lengvinantį eksperimentavimą su nuosekliaisiais algoritmais ir lygiagrečiųjų *šakų ir rėžių* algoritmų programų kūrimą.
- Lygiagrečiuosius optimizavimo algoritmus taikyti statybos technologijos uždaviniams.

Spręsti uždaviniai

- Suformuoti *šeimininkas – darbininkai* algoritmo šablono lygmenų struktūrą. Pritaikyti šabloną naudoti su MPI duomenų perdavimo biblioteka.
- Sukurti *šakų ir rėžių* algoritmo šabloną, pritaikytą:
 - eksperimentavimui su nuosekliaisiais algoritmais;
 - automatiniam lygiagrečiųjų programų kūrimui;
 - naujų algoritmų, paremtų *šakų ir rėžių* algoritmo logika, testavimui.
- Polinių statinių pamatų optimizavimui taikyti lygiagrečiuosius optimizavimo algoritmus, generuotus naudojant *šeimininkas – darbininkai* algoritmo šabloną ir lygiagrečiuosius algoritmus, paremtus *šakų ir rėžių* algoritmo logika.

Mokslinis naujumas

- *Šeimininkas – darbininkai* lygiagretinimo priemonė – algoritmo šablonas, suformuotas lygmenų struktūra, paruoštas naudoti su įvairiomis duomenų perdavimo bibliotekomis.
- Sukurtas *šakų ir rėžių* algoritmo šablonas, leidžiantis eksperimentuoti su nuosekliaisiais algoritmais, generuoti lygiagrečiųjų algoritmų programas bei kurti naujus algoritmus, paremtus *šakų ir rėžių* algoritmo logika. Pasiūlytas modifikuotas juodosios dėžės globaliosios optimizacijos algoritmas.
- Globaliosios optimizacijos lygiagretieji algoritmai taikyti polinių pamatų optimizavimo uždaviniams.

Tyrimų metodika apima algoritmų teoriją, lygiagrečiųjų algoritmų efektyvumo ir sudėtingumo analizę, eksperimentinį tyrimų metodą. C, C++ programavimo kalbas, objektinio programavimo technologiją, algoritmų šablonus, PVM ir MPI duomenų perdavimo bibliotekas.

Gautų rezultatų reikšmė. *Šeimininkas – darbininkai* algoritmo šablonas pritaikytas darbui su kitomis duomenų perdavimo bibliotekomis. Tai praplečia įrankio naudojimo lygiagretinimo automatizavimui galimybes. Įrankis testuotas homogeninėje ir heterogeninėje aplinkose, gautos efektyviai vykdomos lygiagrečiosios programos.

Globaliosios optimizacijos lygiagrečiųjų algoritmų efektyvumas įtakojamas daugelio veiksnių, automatiniai algoritmų lygiagretinimo įrankiai leidžia nesudėtingai eksperimentuoti ir parinkti tinkamus algoritmų variantus. Šablono rėmuose testuotas modifikuotas juodosios dėžės globaliosios optimizacijos algoritmas.

Polinių pamatų optimizavimo uždaviniais taikyti lygiagretieji globaliosios optimizacijos algoritmai ir gauti tinkami sprendiniai, kuriuos charakterizuoja beveik lygios reakcijos jėgos atramose.

Darbe teikiama

- *Šeimininkas – darbininkai* algoritmo šablonas, pritaikytas naudoti su PVM ir MPI duomenų perdavimo bibliotekomis.
- Nuoseklusis ir lygiagretusis *šakų ir rėžių* algoritmų šablonas.
- Juodosios dėžės modifikuotas globaliosios optimizacijos algoritmas.

Darbo apimtis. Disertaciją sudaro įvadas, 2 skyriai, darbo rezultatų apibendrinimas ir išvados, literatūros sąrašas. Bendra disertacijos apimtis – 84 puslapiai, 32 iliustracijos, 7 lentelės. Disertacijos rezultatai skelbti 9 publikacijose, šia tema skaityti 12 pranešimų mokslinėse konferencijose.

Darbo rezultatai ir išvados

1. Modifikuotas *šeimininkas – darbininkai* algoritmo šablonas. Suformuota lygmenų struktūra:
 - Vartotojo programos;
 - Vartotojo programos sąsajos;
 - *Šeimininkas – darbininkai* algoritmo logikos;
 - Duomenų perdavimo.Šablono lygmenų struktūra leidžia įrankį naudoti su įvairiomis duomenų perdavimo bibliotekomis. Šablonas paruoštas naudoti su PVM ir MPI duomenų perdavimo bibliotekomis. Atlikus eksperimentus, parodyta, kad

naudojant abi bibliotekas automatiškai generuojamos efektyviai vykdomos lygiagrečiosios programos. Eksperimentuose gautas efektyvumas didesnis už 0,9.

2. Polinių statinių pamatų optimizavimo uždaviniui taikytas *šeimininkas – darbininkai* algoritmas. Nustatyta, kad šis paprastas lygiagretusis algoritmas yra efektyvus. Nesudėtingos geometrijos uždaviniui gauti geri rezultatai – polių išdėstymo schema. Tačiau panašūs rezultatai buvo gauti ir ekspertiniais metodais. Taikytas lygiagretusis algoritmas leidžia efektyviai išnaudoti lygiagrečiųjų skaičiavimų galimybes.
3. Išnagrinėtas apibendrinto *šakų ir rėžių* algoritmo šablono sudarymas. Atlikta *šakų ir rėžių* algoritmų lygiagretinimo analizė. Atskyrus kintamas, nuo sprendžiamo uždavinio priklausančias, dalis nuo nekintamų algoritmo dalių, sukurtas nuoseklusis *šakų ir rėžių* algoritmo šablonas, pritaikytas kombinatorikos ir globaliosios optimizacijos uždaviniams spręsti. Naudojant šablono, vartotojas gali nesunkiai eksperimentuoti su įvairiais nuosekliais *šakų ir rėžių* algoritmais ir eksperimentiškai tirti algoritmo tinkamumą sprendžiamam uždaviniui.
4. Remiantis nuosekliau *šakų ir rėžių* algoritmo šablonu, sukurtas lygiagretusis *šakų ir rėžių* algoritmų šablonas. Šablono struktūra leidžia automatiškai generuoti įvairių lygiagrečiųjų *šakų ir rėžių* algoritmų programas, naudojantis nuosekliomis vartotojo programomis. Šablone įdiegti lygiagretieji algoritmai, paremti tyrimų srities padalinimu tarp procesorių. Šablono struktūra pritaikyta eksperimentiškai parinkti tinkamus algoritmus ar jų modifikacijas sprendžiamiems uždaviniams. Nustatyta, kad šablone įdiegtų lygiagrečiųjų *šakų ir rėžių* algoritmų efektyvumą mažina išaugantis SOF faktorius ir procesorių apkrovos disbalansas. Siūloma SOF faktorių mažinti procesoriams keičiantis informacija apie geriausius žinomas lokaliuosius sprendinius. Procesorių apkrovos balansavimui sukurtas dinaminio apkrovos balansavimo modulis.
5. Polinių statinių pamatų optimizavimo uždaviniams taikytas juodosios dėžės globaliosios optimizacijos algoritmas. Sudėtingiems uždaviniams gautas geras sprendinys. *Šakų ir rėžių* algoritmo šablono rėmuose pasiūlytas ir ištirtas modifikuotas juodosios dėžės globaliosios optimizacijos algoritmas, nereikalaujantis papildomos informacijos apie tiriamas funkcijas. Tai leidžia *šakų ir rėžių* algoritmo šablono plačiau naudoti praktiniams uždaviniams spręsti.

Trumpos žinios apie autorių

Milda Marija Baravykaitė gimė 1978 m. birželio 14 d. Vilniuje. 2000 m. įgijo inžinerinės informatikos bakalauro laipsnį Vilniaus Gedimino technikos universiteto Fundamentinių mokslų fakultete. 2002 m. apgynė magistro baigiamąjį darbą tema „Lygiagrečiųjų programų generavimo įrankiai“ ir įgijo magistro laipsnį VGTU Fundamentinių mokslų fakultete. 2002–2006 m. studijavo VGTU informatikos inžinerijos krypties doktorantūroje Matematinio modeliavimo katedroje. 2005 m. stažavosi IDRIS superkompiuterių centre ir PRiSM laboratorijoje Prancūzijoje. 2002–2006 m. dirbo asistente VGTU Fundamentinių mokslų fakulteto Matematinio modeliavimo katedroje.

Milda Marija Baravykaitė

**INVESTIGATION AND DEVELOPMENT OF PARALLEL
ALGORITHM TEMPLATES**

**Summary of Doctoral Dissertation
Technological Sciences, Informatics Engineering (07T)**

Milda Marija Baravykaitė

LYGIAGREČIŲJŲ ALGORITMŲ ŠABLONŲ TYRIMAS IR KŪRIMAS

**Daktaro disertacijos santrauka
Technologijos mokslai, informatikos inžinerija (07T)**

2006 05 19. 1,25 apsk. leid. 1. Tiražas 100 egz.

Leido Vilniaus Gedimino technikos universiteto

leidykla „Technika“, Saulėtekio al. 11, LT-10223 Vilnius-40

Spausdino UAB „Biznio mašinų kompanija“, Gedimino pr. 60, LT-01110
Vilnius