VILNIUS UNIVERSITY

ALGIRDAS LANČINSKAS

PARALLELIZATION OF RANDOM SEARCH GLOBAL OPTIMIZATION ALGORITHMS

Summary of Doctoral Dissertation

Physical Sciences,
Informatics (09 P)

Vilnius, 2013

Doctoral dissertation was prepared at the Institute of Mathematics and informatics of Vilnius University in 2009–2013.

Scientific supervisor

Prof. Dr. Julius Žilinskas (Vilnius University, Physical Sciences, Informatics – 09 P).

The dissertation will be defended at the Council of the Scientific Field of Informatics at Institute of Mathematics and Informatics of Vilnius University:

Chairman

Prof. Dr. Habil. Gintautas Dzemyda (Vilnius University, Physical Sciences, Informatics – 09 P),

Members:

Prof. Dr. Habil. Raimondas Čiegis (Vilnius Gediminas Technical University, Physical Sciences, Informatics – 09 P),
Prof. Dr. Habil. Artūras Kaklauskas (Vilnius Gediminas Technical University, Technological Sciences, Informatics Engineering – 07 T),
Prof. Dr. Habil. Kazys Kazlauskas (Vilnius University, Physical Sciences, Informatics – 09 P),
Assoc. Prof. Dr. Rimantas Vaicekauskas (Vilnius University, Physical Sciences, Informatics – 09 P).

Opponents:

Prof. Dr. Eduardas Bareiša (Kaunas University of Technology, Technological Sciences, Informatics Engineering – 07 T),
Assoc. Prof. Dr. Olga Kurasova (Vilnius University, Physical Sciences, Informatics – 09 P).

The dissertation will be defended at the public meeting of the Council of the Scientific Field of Informatics in the auditorium number 203 at the Institute of Mathematics and Informatics of Vilnius University, at 1 p.m. on June 11 2013.

Address: Akademijos st. 4, LT-08663 Vilnius, Lithuania.

The summary of the doctoral dissertation was distributed on May 10 2013.

A copy of the doctoral dissertation is available for review at the Library of Vilnius University.

VILNIAUS UNIVERSITETAS

ALGIRDAS LANČINSKAS

ATSITIKTINĖS PAIEŠKOS GLOBALIOJO OPTIMIZAVIMO ALGORITMŲ LYGIAGRETINIMAS

Daktaro disertacijos santrauka

Fiziniai mokslai (P 000)
Informatika (09 P)

Vilnius, 2013

Disertacija rengta 2009–2013 metais Vilniaus universiteto Matematikos ir informatikos institute.

Mokslinis vadovas

prof. dr. Julius Žilinskas (Vilniaus universitetas, fiziniai mokslai, Informatika – 09 P).

Disertacija ginama Vilniaus universiteto Informatikos mokslo krypties taryboje:

Pirmininkas:

prof. habil. dr. Gintautas Dzemyda (Vilniaus universitetas, fiziniai mokslai, informatika – 09 P).

Nariai:

prof. habil. dr. Raimondas Čiegis (Vilniaus Gedimino technikos universitetas, fiziniai mokslai, informatika – 09 P),
prof. habil. dr. Artūras Kaklauskas (Vilniaus Gedimino technikos universitetas, technologijos mokslai, informatikos inžinerija – 07 T),
prof. habil. dr. Kazys Kazlauskas (Vilniaus universitetas, fiziniai mokslai, informatika – 09 P),
doc. dr. Rimantas Vaicekauskas (Vilniaus universitetas, fiziniai mokslai, informatika – 09 P).

Oponentai:

prof. dr. Eduardas Bareiša (Kauno technologijos universitetas, technologijos mokslai, informatikos inžinerija – 07 T),
doc. dr. Olga Kurasova (Vilniaus universitetas, fiziniai mokslai, informatika – 09 P).

Disertacija bus ginama Vilniaus universiteto viešame Informatikos mokslo krypties tarybos posėdyje 2013 m. birželio 11 d. 13 val. Vilniaus universiteto Matematikos ir informatikos instituto 203 auditorijoje.

Adresas: Akademijos g. 4, LT-08663 Vilnius, Lietuva.

Disertacijos santrauka išsiuntinėta 2013 m. gegužės 10 d.

Disertaciją galima peržiūrėti Vilniaus universiteto bibliotekoje.

Introduction

Research Area and Topicality of the work

Optimization problems in research and practice are known from ancient times. Nowadays a lot of various optimization methods help us solve optimization problems in various fields such as chemistry, biology, biomedicine, operational research, etc.

Normally it is easier to solve optimization problems having some specific properties of objective function such as linearity, convexity, differentiability, etc. However, there are a lot of practical problems that do not satisfy such properties or even cannot be expressed in an adequate mathematical form. Moreover non-convex optimization problems may have several local solutions, and finding the best one is a hard task. Therefore, it is popular to use random search optimization methods in solving such optimization problems.

Modern parallel computing technologies allow us to solve optimization problems requiring a lot of computational recourses. The relevant problem is decomposition of the optimization problem into independent tasks and their optimal distribution among processors with respect to minimize costs of communication between processors.

The area of this research work is random search global optimization methods, their parallelization and application in solving practical global optimization problems.

Research Object

The research object of the work is as follows:

- random search algorithms for global optimization;
- parallel computing systems;
- methods for generation of random numbers' sequences.

The Aim and Tasks

The aim of the dissertation is to modify existing random search algorithms for global optimization and propose new parallel algorithms in order to solve global optimization problems more efficiently. The following tasks were formulated in order to reach the aim:

1. to review the existing random search algorithms for global optimization and define the group of algorithms to be investigated;
2. to investigate opportunities and existing methods for generation of random numbers in parallel computing systems;
3. to modify and parallelize relevant algorithms in order to solve global optimization problems more efficiently with respect to software and hardware being used and the problem being solved;
4. to investigate experimentally the efficiency of proposed algorithms;

5. to compare the achieved results with other well known random search algorithms for global optimization;

6. to investigate the opportunities of application of the proposed algorithms for solution of practical global optimization problems.

Scientific Novelty

Modification of the Particle Swarm Optimization algorithm for Multiple Gravity Assist problem, based on reduction of the search area has been proposed. Several strategies to exchange data between processors in parallel version of Particle Swarm Optimization algorithm have been investigated.

The algorithm for local multi-objective optimization, based on single agent stochastic search strategy, has been proposed and incorporated into Non-dominated Sorting Genetic Algorithm, thus developing a hybrid algorithm for global multi-objective optimization.

The strategy to parallelize Pareto ranking of solutions in multi-objective optimization algorithms has been proposed and applied to develop a parallel version of the algorithm, suited for computations in both distributed and shared memory parallel computing systems. The efficiency of parallel algorithm has been experimentally investigated by solving multi-objective optimization problems on up to 2048 processors.

Defending Propositions

1. Appropriate reduction of the search space can significantly increase the performance of Particle Swarm Optimization algorithm when solving Multiple Gravity Assist problem.

2. The proposed hybrid algorithm for global multi-objective optimization efficiently solves various multi-objective optimization problems used in the experimental investigation.

3. The proposed strategies for modification and parallelization of Non-dominated Sorting Genetic Algorithm solves efficiently multi-objective optimization problems on high performance computing systems.

4. Proposed random search global optimization algorithms solve efficiently Multiple Gravity Assist and multi-objective Competitive Facility Location problems.

Approbation and Research Publications

Results of the research have been published in 6 scientific papers: 4 articles in the periodical scientific publications; 2 articles in the proceedings of scientific conferences. The main results have been presented in 10 national and international scientific conferences and workshops.

The Contents of the Work

The dissertation consists of introduction, 3 main parts and the general conclusions. The dissertation also includes a list of figures, list of notations and abbreviations, and the list of references. The scope of the work is 106 pages that include 37 figures, 10 tables and 4 algorithms. The list of references consists of 77 sources.

1 Random Search Methods for Global Optimization

Global optimization problems can be found in various fields of science and industry, i.e. mechanics, economics, operational research, control engineering, project management, etc. In general, global optimization is a branch of applied mathematics that deals with finding "the best available" (usually minimum or maximum) values of a given objective function. Without reducing the generality we will focus on the case of minimization, since any maximization problem can be easily transformed to a minimization one.

Mathematically, a global optimization problem with d variables is to find the value

$$f^* = \min_{\mathbf{x} \in D} f(\mathbf{x}) \tag{1}$$

and a decision vector $\mathbf{x}^* \in D$ such that

$$f(\mathbf{x}^*) = f^*, \tag{2}$$

where $f(\mathbf{x})$ is an *objective function* which is subject to minimization while variable vector $\mathbf{x} = (x_1, x_2, \ldots, x_d)$ varies in a *search space* $D \in \mathbb{R}^d$, and d defines the number of variables.

Relevant application of global optimization is solution of Multiple Gravity Assist (MGA) problem. This problem is very important in analysis and planning of mission of launching a spacecraft along a trajectory which leads to some astronomical body. The aim of the mission is to land on or to put the spacecraft into orbit of the body. Global optimization methods are used to help trajectory planners in choosing the best decision on the starting date and other relevant parameters subject to minimization of costs of the mission.

There are a lot of methods and algorithms proposed to solve various optimization problems, including MGA, however dissertation is focused on random search algorithms. One of them is Single Agent Stochastic Search (SASS).

SASS is based on selection of new solution within neighborhood of the best solution found so far. A neighbor solution is calculated by adding random vector ξ to the solution $\mathbf{x}$ which has the best fitness value. The vector ξ is generated utilizing Gaussian perturbations. If the new candidate solution $\mathbf{x}' = \mathbf{x} + \xi$ does not improve fitness value of solution $\mathbf{x}$ then the opposite candidate solution $\mathbf{x}'' = \mathbf{x} - \xi$ is evaluated. The values of bias and standard deviation of the random perturbation are adjusted dynamically depending on successes and failures in selection of neighbor solutions.

SASS has been designed for local single-objective optimization. One of the most popular random search algorithm Another relevant random search algorithm is Particle Swarm Optimization (PSO). PSO is population-based algorithm, and begins with a population (swarm) of randomly generated candidate solutions (particles). Particles "fly" through the search space with velocities which are dynamically adjusted

according to historical behaviors – the best position that is found by a particular particle and the best position which is found by the whole swarm.

Another relevant population-based optimization algorithm is Genetic Algorithm (GA). Algorithm begins with population of randomly generated candidate solutions (parents). New candidate solutions (offsprings) are generated by applying crossover operator to the randomly selected parent solutions. Obtained offsprings are mutated by making small changes of values of variables.

Real-world optimization problems often deal with more than one objective functions that are conflicting to each other – improvement of one objective can lead to deterioration of another. Such type of optimization problems are known as Multi-objective Optimization Problems.

Mathematically, a multi-objective optimization problem with d variables and m objectives in the objective vector

$$F(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_m(\mathbf{x})) \tag{3}$$

is to simultaneously optimize all objectives:

$$F^* = \min_{\mathbf{x} \in D} F(\mathbf{x}). \tag{4}$$

Since there exist conflicts among objectives, it is natural that a single best solution according to all objectives does not exists. However, a set of Pareto optimal solutions, which cannot be improved by any objective without reducing the quality of any another, may be found.

In multi-objective optimization two different solutions can be compared by their *dominance relation*. They can be related to each other in a couple of ways: either one dominates the other or none of them is dominated by the other.

It is said that solution $\mathbf{x}_1$ *dominates* solution $\mathbf{x}_2$ if

(1) solution $\mathbf{x}_1$ is not worse than $\mathbf{x}_2$ by all objectives and

(2) solution $\mathbf{x}_1$ is strictly better than $\mathbf{x}_2$ by at least one objective.

The relation is denoted by $\mathbf{x}_1 \succ \mathbf{x}_2$ and mathematically can be described as

$$\mathbf{x}_1 \succ \mathbf{x}_2 \Leftrightarrow \begin{cases} \forall i \in (1, \ldots, m) & f_i(\mathbf{x}_1) \leq f_i(\mathbf{x}_2) \\ \exists j \in (1, \ldots, m) & f_j(\mathbf{x}_1) < f_j(\mathbf{x}_2) \end{cases} \tag{5}$$

If (5) is satisfied then solution $\mathbf{x}_1$ is *dominator* of solution $\mathbf{x}_2$. The solution which has no dominators is called *non-dominated* or *Optimal in Pareto sense*. If neither $\mathbf{x}_1 \succ \mathbf{x}_2$ nor $\mathbf{x}_2 \succ \mathbf{x}_1$ is not satisfied then the solutions x_1 and x_2 are called indifferent and denoted by $\mathbf{x}_1 \sim \mathbf{x}_2$.

The set of all non-dominated solutions are called *Pareto Set*, and the corresponding set of objective vectors – *Pareto Front*.

Global multi-objective optimization methods are relevant in solving Competitive Facility Location (CFL) problems. Consider several firms which provide some goods or service to customers in a certain geographical area. One of the firms, called the expanding firm (denoted by A) wants to locate new facilities in order to increase its market share in that area. The new facilities may capture customers that were served by the other firms as well as customers which were served by the preexisting

facilities of the expanding firm. Therefore, the expanding firm is also interested in minimizing the lost of market share of its preexisting facilities. Thus firm A faces a bi-objective optimization problem.

Usually it is hard and time consuming to find the true Pareto front therefore a lot of multi-objective optimization algorithms are try to approximate the Pareto-optimal front.

One well known random search algorithm for multi-objective global optimization is Non-dominated Sorting Genetic Algorithm (NSGA-II). It is based on obtaining a new offsprings' population Q by applying genetic operators to the parent solutions from population P. Both populations P and Q are then merged into one population R, which individuals are sorted according to Pareto ranks and reduced by removing half most dominated solutions. The reduced population R is used as a parent population P in the next generation.

2 Modification and Parallelization of the Algorithms

Modification and Parallelization of PSO algorithm

PSO algorithm has been modified by involving the reduction of the search area. Suppose the primary search area is

$$D = [x_1^{LB}, x_1^{UB}] \times [x_2^{LB}, x_2^{UB}] \times \ldots \times [x_d^{LB}, x_d^{UB}], \tag{6}$$

where $x_1^{LB}, x_2^{LB}, \ldots, x_d^{LB}$ and $x_1^{UB}, x_2^{UB}, \ldots, x_d^{UB}$ are respectively lower and upper bounds for values of variables $x_1, x_2, \ldots, x_d$, d – the number of variables. The new search area

$$\tilde{D} = [\tilde{x}_1^{LB}, \tilde{x}_1^{UB}] \times [\tilde{x}_2^{LB}, \tilde{x}_2^{UB}] \times \ldots \times [\tilde{x}_d^{LB}, \tilde{x}_d^{UB}] \tag{7}$$

is calculated by

$$\begin{aligned} \tilde{x}_i^{LB} &= p_i^g - \delta_i, \\ \tilde{x}_i^{UB} &= p_i^g + \delta_i, \\ \delta_i &= c_r(x_i^{UB} - x_i^{LB}), \end{aligned} \tag{8}$$

where p_i^g – value of i-th variable of the best solution found so far, $c_r \in (0, 1)$ – coefficient of the reduction, $i = 1, 2, \ldots, d$. The smaller reduction coefficient leads to a smaller new search area.

Reduction of the search area is performed after a predefined number E_G of function evaluations. After the reduction, optimization process continues by generating population of candidate solutions, satisfying new search area $\tilde{D}$. The reduction procedure is illustrated in Figure 1.

PSO requires a lot of computational resources, especially if large population is used. On the other hand population-based algorithms are intrinsically parallel. The population of particles can be divided into subpopulations and distributed among the processors for evaluation of values of objective functions.

The main performance bottleneck in parallel computational environment is often communication latency between processors. Therefore, it is very important to design an efficient strategy for data exchange among processors in order to obtain efficient

parallel algorithm. The following strategies to exchange data between processors have been investigated in the dissertation.

Synchronous master-slave (sMS). One of processors (the master) controls data exchange. At the end of iteration the master processor gathers solutions from all processors, identifies the best one and distributes it among all processors. The master processor has to wait while all processors finish their iteration and then start data exchange procedure. The master processor has computational work too.

All-to-all (AtA). Each processor sends its best known solution to all other processors and receives information about the best solution from all others. Each processor continues to the next iteration with the best solution that is known after data exchange.

Hierarchic scatter (HS). Master processor gathers information from all processors in hierarchic fashion. Master processor identifies the best solution and sends it back to all processors following the hierarchic fashion.

Asynchronous master-slave (aMS). This strategy is similar to sMS strategy. The difference is that the master processor has no computational work. It just waits while others finish their iteration and request for data exchange. The master receives solutions from the slaves, determines the best solution and sends it back to the slave requesting data exchange. Algorithm does not stop to wait for all processors.

Multi-Objective Single Agent Stochastic Search

Single Agent Stochastic Search (SASS) has been successfully used as a local search strategy in evolutionary algorithms for single-criteria optimization problems. In order to apply it for multi-objective optimization a new version called Multi-Objective Single Agent Stochastic Search (MOSASS) has been developed.

MOSASS algorithm begins with an initial solution $\mathbf{x}$, and an empty archive A for storing non-dominated solutions. A new solution $\mathbf{x}'$ is generated in the same way as in SASS, objective vector $\mathbf{F}(\mathbf{x}')$ is evaluated, and the dominance relation between $\mathbf{x}'$ and $\mathbf{x}$ is evaluated. In the case of $\mathbf{x}' \succ \mathbf{x}$, the present solution $\mathbf{x}$ is changed to $\mathbf{x}'$ and the algorithm continues to the next iteration. Otherwise, if $\mathbf{x}'$ does not dominate $\mathbf{x}$ and is not dominated by any solution in $A \cup \{\mathbf{x}\}$ then the archive A is supplemented by $\mathbf{x}'$ and algorithm continues to the next iteration. If solution $\mathbf{x}'$ is dominated by any member of $A \cup \{\mathbf{x}\}$, then $\mathbf{x}'$ is rejected and the opposite solution $\mathbf{x}''$ is investigated in the same way as $\mathbf{x}'$. If archive size exceeds the size limit N_A then the half most crowded individuals are removed.

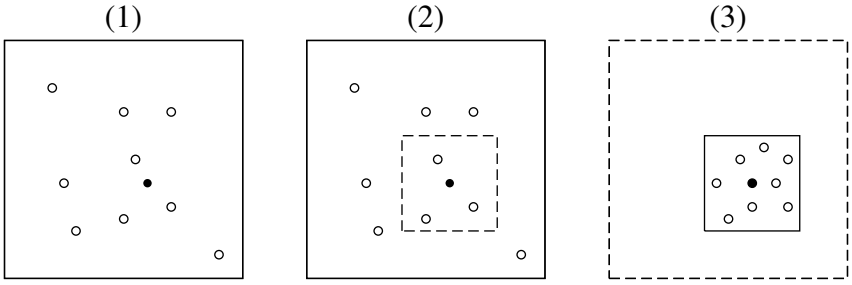

Figure 1: Reduction of the search area

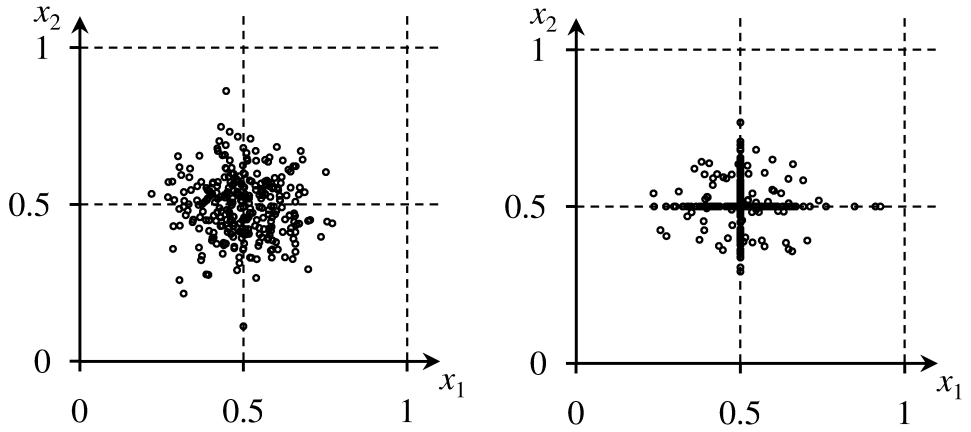

Figure 2: Generation of neighbor solution in MOSASS (left) and MOSASS/P (right) algorithms

If either a present solution $\mathbf{x}$ is changed or the archive is supplemented then the iteration is assumed to be successful, otherwise – failed. The mean and standard deviation parameters are dynamically adjusted as in SASS algorithm.

A strategy which is used in selecting neighbor solution can play an important role in random search techniques. In SASS as well as in MOSASS a neighbor solution is generated by changing values of all variables of $\mathbf{x}$ without any probabilistic choice. Therefore it is a large probability that obtained neighbor solution $\mathbf{x}'$ or $\mathbf{x}''$ differs from its precursor $\mathbf{x}$ by all parameters. However, sometimes it is necessary to make only a slight modification of the current solution – to alter only one variable in order to obtain a solution which would dominate its precursor or at least would be non-dominated by other solutions. This is especially important in a later stage of the algorithm when almost optimal solutions are used to produce new ones. In order to approve (or disprove) this modification, MOSASS has been modified by involving the probability in generating a neighbor solution. The modification has been done by generating a neighbor solution using

$$\xi_i = \begin{cases} N(b_i, \sigma_i), & \text{if} \quad r < \pi_{LS}, \\ 0, & \text{otherwise}, \end{cases} \tag{9}$$

where r is a random number uniformly generated in $[0, 1]$ and $\pi_{LS} \in (0, 1]$ is a predefined parameter. The larger p value is, the larger probability that the particular variable will be changed. The modified version of MOSASS algorithm has been called by MOSASS/P. The main difference in generation of neighbor decision vector in algorithms MOSASS and MOSASS/P is shown in Figure 2, where the most likely neighbor decision vectors $\mathbf{x}'$ to be generated by MOSASS (on the left) and MOSASS/P (on the right) are illustrated ($x_i = 0.5$, $\pi_{LS} = 0.5$, $\sigma = 0.1$ and $b_i = 0$, $i = 1, 2$).

There is also possible situation that all coordinates will remain unchanged. Then the generation of the new solution is repeated until at least one variable is changed.

In order to improve some performance metrics of original NSGA-II algorithm a hybrid algorithm, based on performing the local search towards a set of non-dominated decision vectors, has been developed. The hybrid algorithm begins with an initial

parent population P, consisting of N decision vectors randomly generated over the search space. Further, the algorithm continues with the following processes:

(1) A new child population, of the size N, is generated by applying genetic operations to the individuals of the parent population. The uniform crossover that combines pairs of parent population individuals, and mutation with mutation rate equal to $1/d$ are used.

(2) Parent and child populations are combined into one $2N$-size population, and each decision vector evaluated by counting the number of its dominators.

(3) The obtained population is reduced to the size of N by removing the required number of most dominated individuals.

(4) The counter of NSGA-II generations G is increased by one. If the value of G does not exceed the predefined number, then the algorithm returns to the first step. Otherwise the algorithm continues to the next step.

(5) An auxiliary set P_L of k decision vectors is created from the population P. Non-dominated decision vectors are chosen to be included into the set P_L. If $|P_L| > k$ then $|P_L| - k$ decision vectors are randomly removed from P_L. Otherwise, if $|P_L| < k$, then $k - |P_L|$ randomly chosen dominated decision vectors are added.

(6) All decision vectors in P_L are locally optimized by performing a predefined number of MOSASS or MOSASS/P iterations for each decision vector. Since MOSASS as well as MOSASS/P returns a set of non-dominated decision vectors and $|P_L|$ decision vectors are optimized, $|P_L|$ sets are resulted from the local optimization. All these sets are combined into one set together with the population P, which is reduced to the size of N. The same scheme as in third step is used to perform the reduction.

(7) The algorithm continues to the first step by reseting the generations counter G to 0, and using the obtained population as the parent population for performing NSGA-II generation.

Depending on the local search algorithm (MOSASS or MOSASS/P) used, the derived hybrid algorithm has been denoted by NSGA-II/LS and NSGA-II/LSP respectively. The number of NSGA-II iterations after which the local search is performed, the size of the auxiliary set P_L, and the number of local search iterations are given as input parameters.

Parallelization of NSGA-II

Most of the published parallel versions of NSGA-II are based on master-slave strategies and parallelization of function evaluations. On the other hand NSGA-II can be separated into 3 parts:

(1) evaluation of objective functions;

(2) Pareto ranking;

(3) other computations.

The first part – the evaluation of the objective functions – is frequently the largest part of the algorithm and intrinsically parallel. Therefore the easiest way to parallelize NSGA-II is to evaluate the objectives by distributing the population among all processors and leave the remaining parts sequential. We denote the parallel algorithm developed following this strategy by ParNSGA/PE (Parallel NSGA based on Parallelization of Evaluations).

This strategy is good if evaluation of objectives requires relatively large amount of time. For example if it requires 99% of execution time of sequential algorithm the maximum speed-up of the parallel algorithm will be 100. But if evaluations of functions require 95% or 90% of the execution time the maximum speed-up of the algorithm is respectively 20 or 10. These estimations are based on Amdahl's law.

Another part of the algorithm requiring relatively large computational time is part (2) – Pareto ranking. Therefore optimization and parallelization of this part can increase the maximum speed-up of the algorithm.

The population R consisting of 2 N-sized populations P and Q must be Pareto ranked in every iteration of NSGA-II. Lets denote by $r(P,Q)$ the vector, representing the number of dominators of each individual in P between individuals in Q:

$$r(P,Q) = (r_1, r_2, \ldots, r_{|P|}), \tag{10}$$

where

$$r_i = |\{y \in Q : y \succ x_i, x_i \in P\}|, \tag{11}$$

and $i = 1, 2, \ldots, |P|$.

Thus the ranks of all individuals of P can be described as

$$r(P,P) + r(P,Q), \tag{12}$$

and ranks of all individuals of Q can be described as

$$r(Q,Q) + r(Q,P). \tag{13}$$

Computation of $r(P,Q)$ requires N^2 Pareto comparison operations as each individual from P must be compared with each individual of Q. The same complexity has computation of $r(Q,P)$. Computations of $r(P,P)$ and $r(Q,Q)$ requires $N(N-1)$ Pareto comparisons per each, as it is not necessary to compare individual with itself. Therefore ranking of $P \bigcup Q$ requires

$$2N^2 + 2N(N-1) = 2N(2N-1) \tag{14}$$

Pareto comparison operations.

Proposition. If solution $\mathbf{x}_1$ dominates solution $\mathbf{x}_2$ then Pareto rank of solution $\mathbf{x}_1$ is strictly lower than Pareto rank of solution $\mathbf{x}_2$.

The proof, illustrated in Figure 3, follows from the transitivity property of the dominance relation.

Taking into account the proposition and the fact that new parent population P is created by removing the most dominated solutions, we can state that it is not necessary to calculate $r(P,P)$ in any generation except the first one. We just need to calculate $r(P,Q)$, $r(Q,Q)$ and $r(Q,P)$. Doing so the number of Pareto comparisons in any generation is reduced by $N(N-1)$ except the first generation which complexity does not changes.

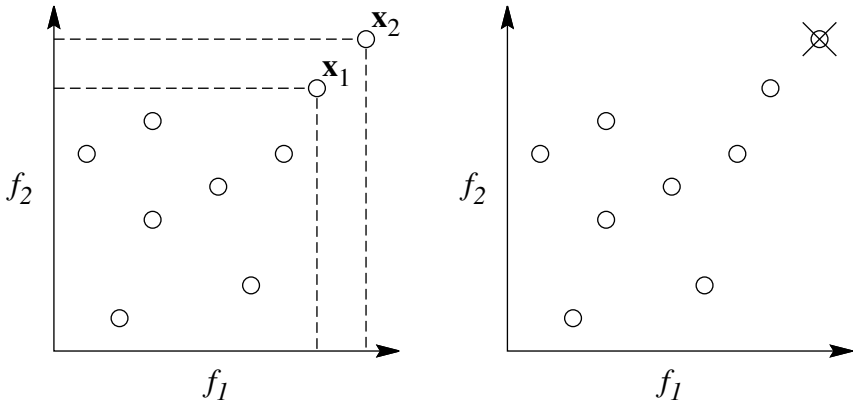

Figure 3: Illustration of proposition about Pareto ranks

Taking into account this statement, three strategies to parallelize NSGA-II algorithm have been proposed.

ParNSGA/HR-1. The population P is sent to all processors following the hierarchic scatter fashion. Each processor generates a part of offsprings' population Q_i of the size N/p (where p is the number of processors), evaluates objective values and sends information back to the master processor in hierarchic fashion. In each step of gathering data a partial Pareto ranking is performed as shown in Figure 4.

In this strategy all processors have to wait while 1st processor calculates $r(P, Q)$ at the final step of data transfer. Another strategy ParNSGA/HR-2 has been proposed in order to avoid such an idle time.

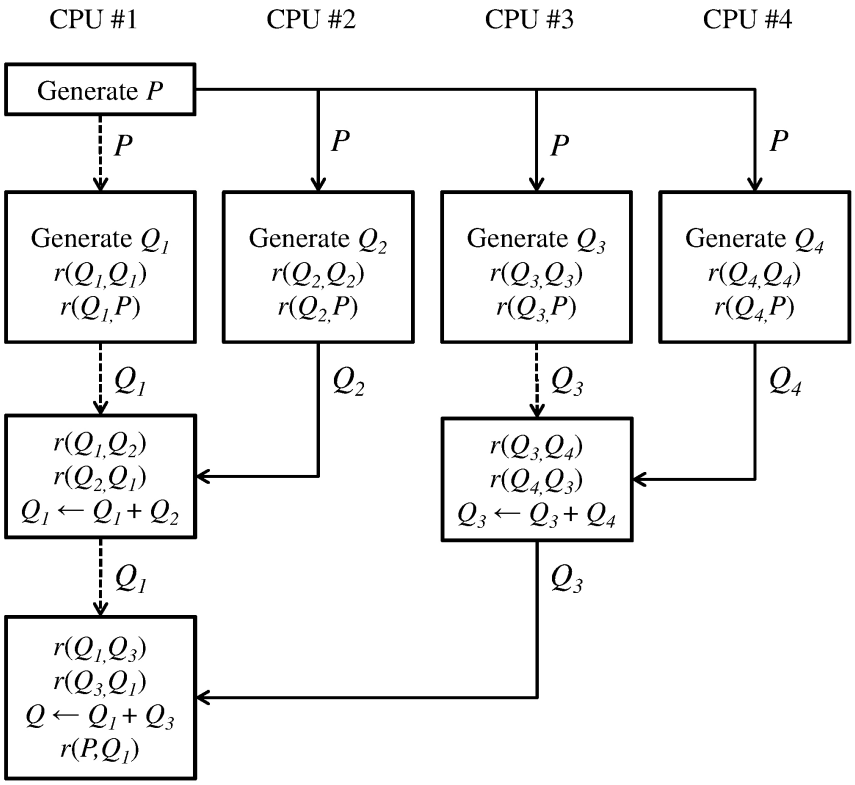

Figure 4: Scheme of ParNSGA/HR-1

ParNSGA/HR-2. In contrast with the ParNSGA/HR-1, processors additionally calculate $r(P, Q_i)$ in the first step of data exchange procedure (i – the number of processor). Although the processors have additional data (Pareto ranks of individuals of P) which must be transferred, it is not necessary to perform N^2 Pareto comparison operations at the end of data transfer procedure. The scheme of ParNSGA/HR-2 is given in Figure 5.

In both strategies described above the processors generate offsprings' subpopulations and send them whole. At the end of generation, a part of offsprings are rejected.

ParNSGA/HR-R. Performing the procedure $r(Q, P)$ in the first step of data sending procedure, solutions from Q_i, with Pareto ranks larger than maximum Pareto rank in P, are removed. Doing so the population Q_i is reduced thus reducing costs requiring for transferring data and performing Pareto ranking.

Although ParNSGA/HR-1, ParNSGA/HR-2 and ParNSGA/HR-R parallelize Pareto ranking without additional cost of communication, non-master processor may be idle for some time. On the one hand it could be insignificant performing computations on several or several tens of processors, on the other hand the idle time can be relative long when using several hundreds or thousands of processors. For comparison the strategy ParNSGA/DR (Distributed Ranking) which reduces the idle time but increases costs for communication has been proposed. The strategy is based on the following process:

(1) The master processor distributes the parent population among the slaves following the hierarchic fashion.

(2) Each processor generates an appropriate part Q_i of offsprings' population Q.

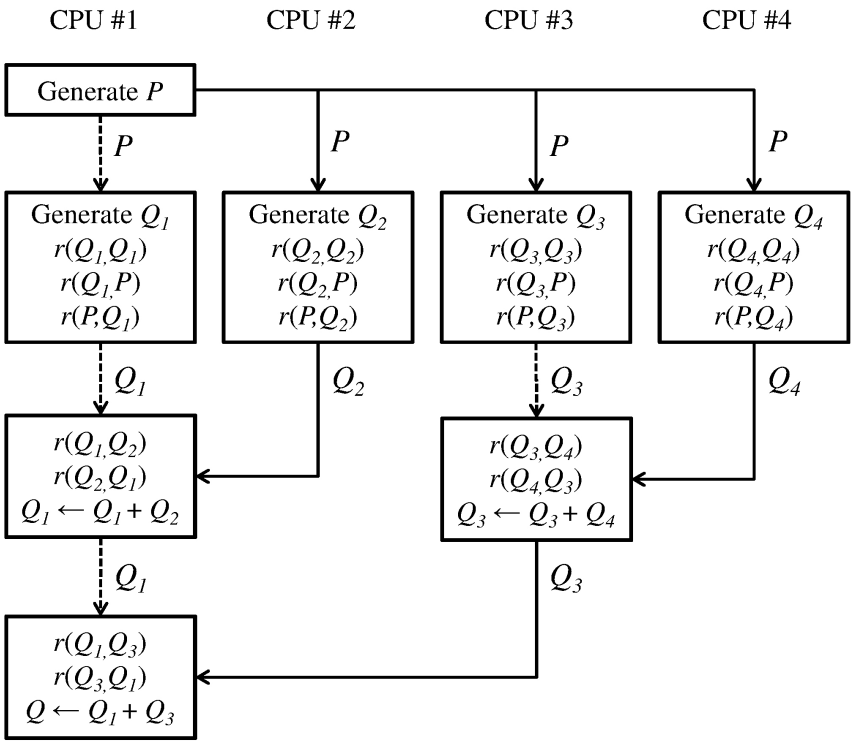

Figure 5: Scheme of ParNSGA/HR-2

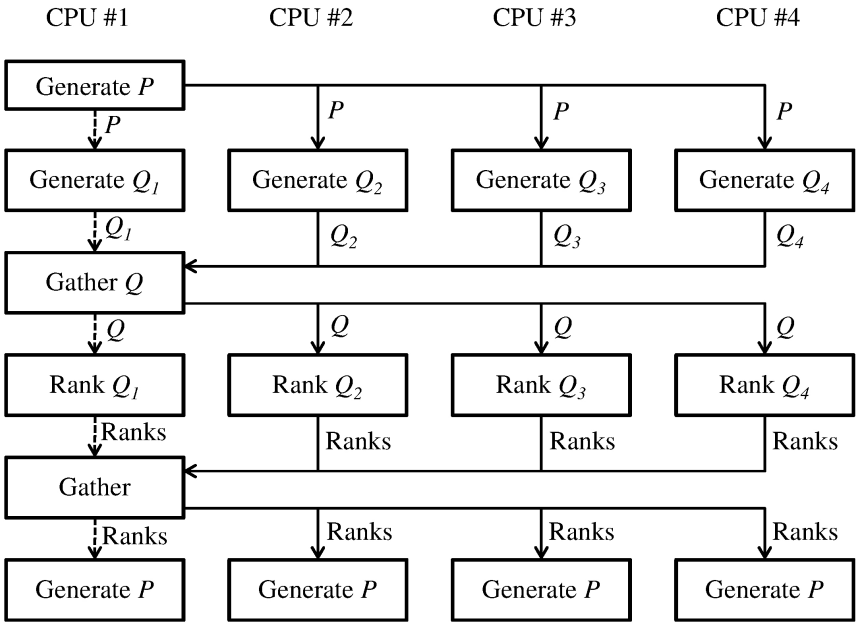

Figure 6: Scheme of ParNSGA/DR

(3) The master processor gathers subpopulations Q_i from the slaves, combines them into one population Q and distributes it among the slaves.

(4) Each processor evaluates the corresponding subpopulation $Q_i \in Q$ by performing $r(Q_i, Q)$, $r(Q_i, P)$ and $r(P, Q_i)$.

(5) The master processor gathers all the information about Pareto ranks, counts the total values of Pareto ranks of each individual in $P \cup Q$ and distributes them among slaves.

(6) Now all processors have the whole set $P \cup Q$ including the information about Pareto ranks, and can continue with rejecting most dominated solutions and proceeding to the next iteration (2nd step).

The scheme of ParNSGA/DR is given in Figure 6.

Application of NSGA-II to Solve Discrete CFL Problem

In discrete CFL problem locations for the new facilities must be selected from a set of candidate demand points $L = \{l_1, l_2, \ldots, l_r\}$. Thus the search space D can be described as the set of all subset of L such as $|X| = s$:

$$D = \{X \subset L : |X| = s\}, \tag{15}$$

where s – the number of facilities expected to be located (the number of variables).

Following the classical scheme of NSGA-II, the algorithm begins with generation of parent population $P^{(0)}$ consisting of N individuals – subsets of L satisfying equation (15).

Each individual $X^{(3)}$ of offsprings' population $Q^{(k)}$ is generated by applying crossover operations to the individuals $X^{(1)}$ and $X^{(2)}$ from $P^{(k)}$. The crossover is performed following the equation:

$$X^{(3)} \subset \left(X^{(1)} \cup X^{(2)} \right). \tag{16}$$

Mutation of i-th variable of offspring $X^{(3)}$ is performed by changing its value $x_i^{(3)} \in L$, to

$$l \in \left(L \setminus x_i^{(3)} \right), \tag{17}$$

On one hand the location l can be chosen at random from all possible candidate locations, however such a strategy leads to chaotic mutation. Therefore, we proposed to define a neighborhood of variable $x_i^{(3)}$ – a subset $L^{(h)}(x_i^{(3)}) \subset L$ of h locations which are nearest to the location $x_i^{(3)}$, and perform mutation within the neighborhood as follows:

$$x_i^{(3)} = l \in L^{(h)}(x_i^{(3)}) : l \neq x_j^{(3)}, j = 1, \ldots, r. \tag{18}$$

An important factor is the value of h which can vary from 1 to $|L|$. Usage of large values leads to chaotic mutation similar as using (17), while usage of lower values of h leads to inclusion of less scattered locations.

The local search algorithm based on selection of neighbor solution and aimed to improve Pareto front obtained by NSGA-II algorithm has been proposed. A single iteration of the algorithm consists of randomly choosing a single solution X from an approximation of Pareto set $\tilde{P}$ found so far, and generation of a neighbor solution X' using (18). After a neighbor solution X' is generated and the values of objectives are evaluated, the dominance relation against other members of $\tilde{P}$ is evaluated. If X' is not dominated by any other member of Pareto set, it is added to $\tilde{P}$ and all solutions which are dominated by X' are removed from $\tilde{P}$:

$$\tilde{P} \leftarrow \{X \in \tilde{P} : X' \not\succ X\} \cup \{X'\}. \tag{19}$$

The number of function evaluations we want to devote to local search must be defined in advance.

3 Experimental Investigation

Solution of MGA Problem Using PSO

PSO with reduction of the search area has been experimentally investigated and compared with original PSO by solving MGA problem. Different population sizes (20, 40, 60, 80, and 100 particles) and different reduction coefficients (0.15, 0.05, and 0.01) have been investigated. Due to the stochastic nature of the algorithm, each experiment has been performed 100 times using different initial populations, and average values have been evaluated. The quality of the algorithm has been evaluated by probability to achieve an acceptable solution – a solution which objective value is lower than 5.308.

Probabilities to achieve acceptable solution using original PSO and PSO with different reduction coefficients (1.5, 0.5, and 0.01) are given in Figure 7. Quality of

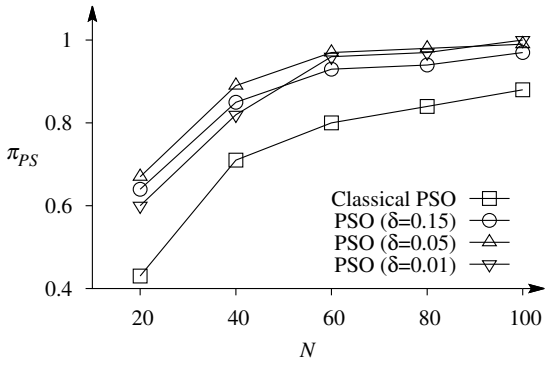

Figure 7: Probability to achieve acceptable solution using different population size and reduction coefficient

solution hardly depends on population size – more particles in population leads to larger probability to achieve acceptable solution. Reduction of the search area gives significant advantage. Best results have been achieved using reduction coefficient 0.01 and 100 particles in population – acceptable solution has been achieved in all 100 independent runs. However using smaller population it is better to use reduction coefficient 0.05 due to error of solution that is achieved after 10000 iterations using small population.

Once the best sequential strategy and the best population size have been selected, the sequential algorithm has been compared to the four parallel strategies. The average sequential execution time using 100 particles in population has been 141.08 seconds. Using parallel computing methods it was reduced to 18.02 seconds using HS strategy and 13.6 seconds using aMS strategy on 16 processors. Comparison of speed-up an efficiency of the algorithm using synchronous data exchange strategies and different number of processors are given in Figure 8. The best speed-up and efficiency of synchronous strategies have been achieved using HS strategy.

Speed-up and efficiency achieved with aMS strategy have been compared with HS strategy which seems to be the best of all synchronous strategies. Results, given in Figure 9, showed that performing computations on a small number of processors (2, 4 or 6) synchronous strategy produces better speed-up and efficiency. Therefore,

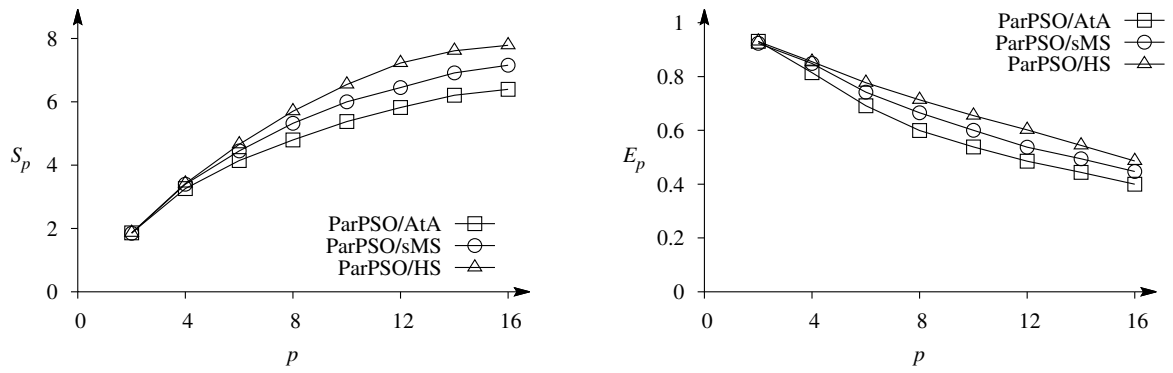

Figure 8: Speed-up (left) efficiency (right) of ParPSO/AtA, ParPSO/sMS, and ParPSO/HS

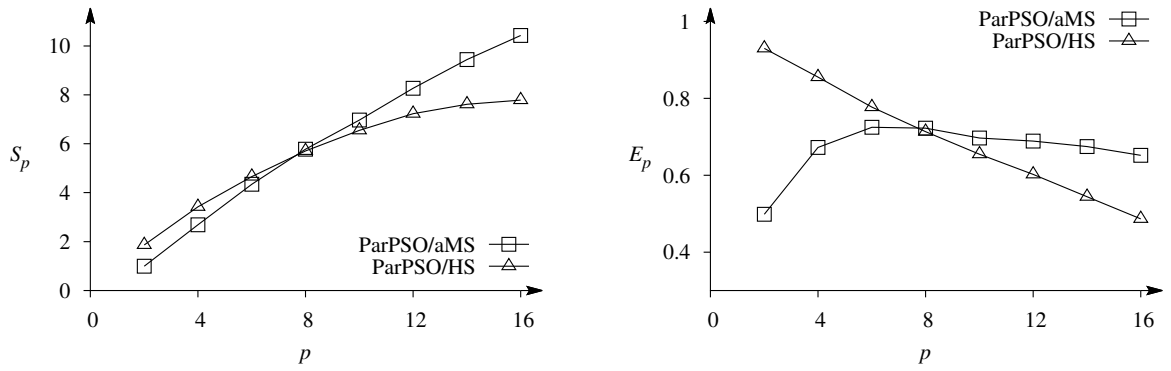

Figure 9: Speed-up (left) efficiency (right) of ParPSO/aMS and ParPSO/HS

it is useful to get computational work for all processors. On the other hand if the number of processors is larger (8 or more) it is useful to leave one processor without computational work, but only for management of communication (aMS strategy).

Investigation of NSGA-II/LS and NSGA-II/LSP

The proposed memetic algorithms NSGA-II/LS and NSGA-II/LSP have been experimentally investigated by solving a set of 26 multi-objective optimization problems. The quality of the algorithms has been measured by 5 metrics: Pareto Size (PS); Hypervolume (HV); Coverage of Pareto Fronts (C); Inverted Generational Distance (IGD); Pareto Spread (Δ).

The first set of experiments was aimed at choosing the appropriate number (E_G) of function evaluations after which local search is performed, and the number (N_L) of function evaluations to be performed during each local search. A set of 36 different combinations of parameters (E_G, E_L) has been used. The results of the investigation are presented in Table 1, where the numbers of test functions for which particular set of parameters (E_G, E_L) was the best by hyper-volume metric are given. From the table we can see that it is worth to perform local search every 500–2000 function evaluations. The number of function evaluations, performed in each local search, should be 300–500. We have chosen a median point – (1000,400) as an appropriate parameter set (E_G, E_L) for further experiments.

The second experiment was dedicated to investigate performance of proposed algorithms NSGA-II/LS and NSGA-II/LSP. The performance of the algorithms has been compared with each other and with classical NSGA-II, implemented as given in literature. All three algorithms have been performed for 100 independent runs on all 26 test functions. The crossover probability equal to 0.8, mutation rate equal to $1/d$ and population size equal to 100 have been used for NSGA-II. Local optimization has been performed after every 10 NSGA generations (1000 function evaluations) for 400 function evaluations, as this combination of parameters gives the best statistical results in previous experiment. A set of 10 decision vectors have been selected for local optimization. The probability in generating new decision vector in NSGA-II/LSP was chosen to be equal to $1/d$ – the same as mutation rate in NSGA-II.

Each algorithm was evaluated by counting the number of test problems for which the evaluated algorithm gives the best result according to a particular metric of

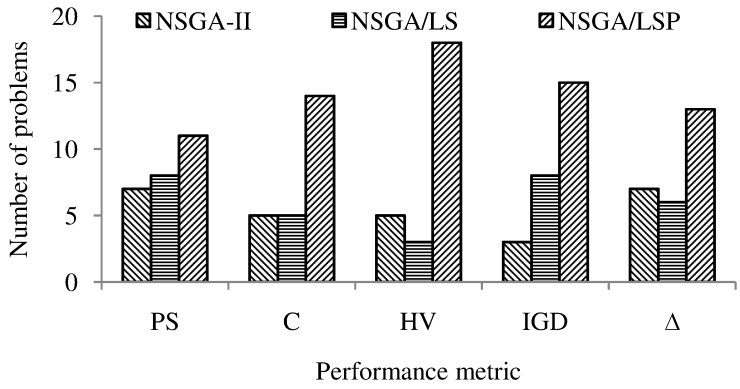

Figure 10: Comparison of the algorithms by number of problems solved best.

performance. All three algorithms were evaluated according to all five performance metrics used in the experimental investigation. The results are illustrated in Figure 10, where the vertical axis represents the number of problems which were solved best, and horizontal axis – performance metric. Different columns in a group represent different algorithms: NSGA-II, NSGA-II/LS and NSGA-II/LSP respectively. As illustrated in the figure, NSGA-II gives best Pareto size values for 7 test problems, while NSGA-II/LS was the best for 8, and NSGA-II/LSP – for 11. According to coverage metric, NSGA-II was the best for 5, NSGA-II/LS – for 5, NSG-II/LSP – for 14 test problems, and for 1 test problem all three algorithms gives zero values of coverage metric. According to HV and δ, the best algorithm was NSGA-II/LSP, and the worst – NSGA-II/LS. According to IGD, the worst algorithm NSGA-II. Algorithm NSGA-II/LSP solves best the largest number of test problems according to all performance metrics.

The performance of NSGA/LSP has been compared with performance of Multi-objective Optimization Evolutionary Algorithm based on Decomposition (MOEA/D). MOEA/D has been recognized as the best algorithm for solution of test problems UP1–UP10, In CEC 2009 Special Session and Competition. Both algorithms NSGA/LSP and MOEA/D have been ran for a certain time period – 20 seconds. Each experiment has been repeated 100 times on test problems UP1, UP2, UP3, UP4, UP7, UP8 and UP10 and average results have been evaluated. Performance of the algorithms has been evaluated by Hypervolume metric, which has been

Table 1: Numbers of test problems for which particular set of parameters (E_G, E_L) were the best by HV value

E_L	E_G					
	500	1000	2000	3000	4000	5000
50	1	1	0	0	0	0
100	0	0	0	0	0	0
200	1	1	0	2	1	3
300	2	1	1	0	1	0
400	1	3	0	0	0	0
500	0	3	1	0	0	2

measured at discrete time moments – 3-th, 5-th, 10-th, 15-th and 20-th second of the experiment. Results of the experiments are presented in Figure 11, where the horizontal axis in each graph represents time moments, and the vertical axis – value of Hyper-Volume. Different graphs represent different test problem and different curves – different algorithm. The results show that NSGA/LSP produces better results for almost all test functions within fixed time period. MOEA/D was significant better for one test problem – UP3, however MOEA/D was unable to find any evaluable approximation of Pareto front of test problem UP10 within 20 seconds.

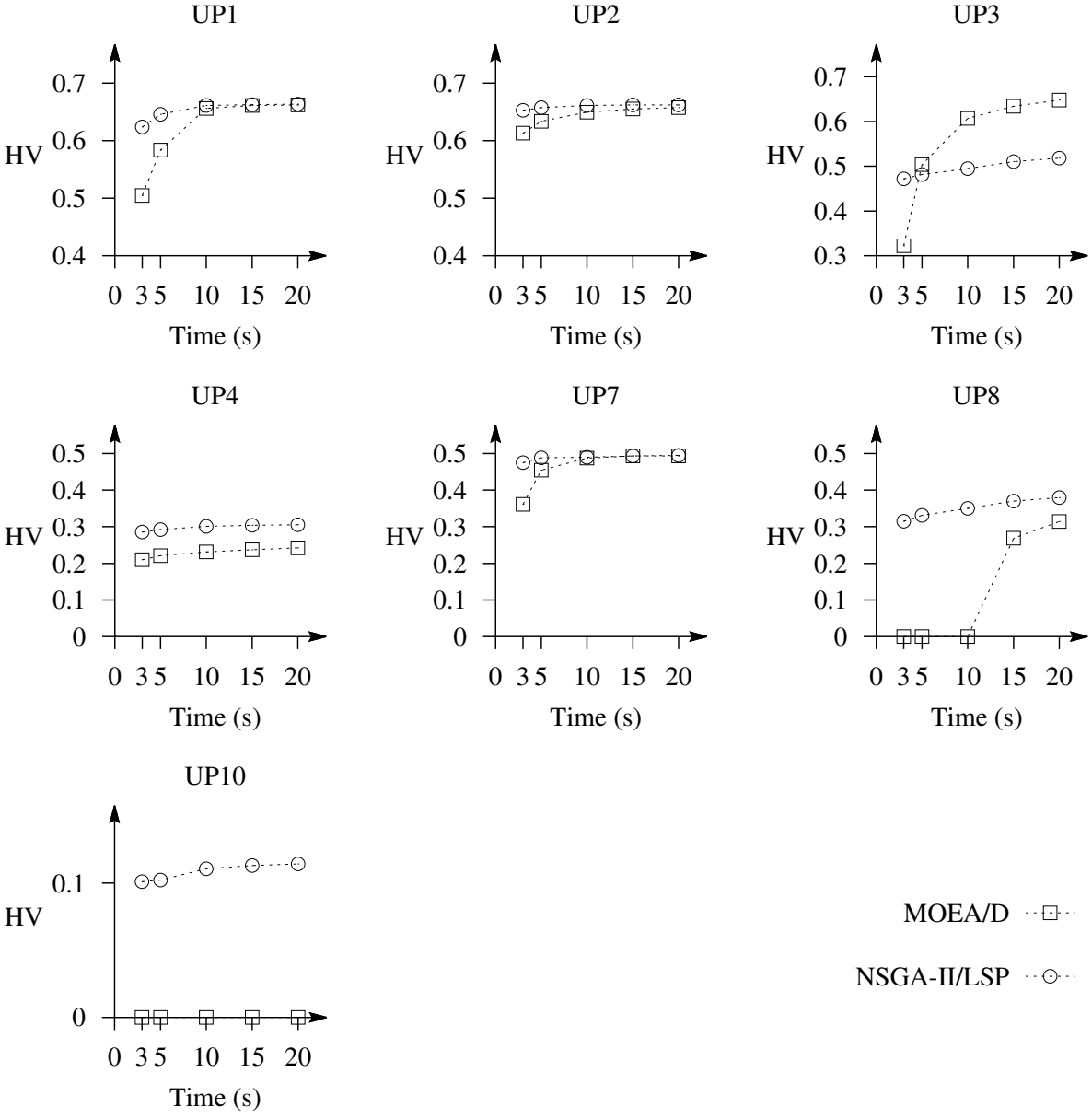

Figure 11: Values of HV, obtained by MOEA/D and NSGA-II/LSP within fixed time period

Investigation of Strategies to Parallelize NSGA-II

Parallel algorithms ParNSGA/PE, ParNSGA/HR-1, ParNSGA/HR-2 and ParNSGA/HR-R have been experimentally investigated by solving multi-objective optimization problem ZDT1 with 2 objectives and 10 variables. The size of population was 512 and the number of generations performed – 250. Ten independent runs have been made to estimate duration of the execution. The evaluations of functions last about 90% of the sequential algorithm duration. Computations have been performed on 2, 4, 8, 16, 32 and 64 processors and the speed-up of the algorithm has been measured. Hardware, containing 16 Intel(R) Core(TM) i5 CPU 760 @ 2.80GHz nodes with 4 cores on each and 4GB of RAM has been used during experimental investigation. The results of the experiments (Figure 12) show that the proposed strategies give significant impact to the speed-up of the algorithm. Using ParNSGA/PE the speed-up on 64 processors was 8.86 while speed-up of ParNSGA/HR-1, ParNSGA/HR-2 and ParNSGA/HR-R was 12.41, 16.9 and 23.25 respectively. Reduction of offsprings' population in ParNSGA/HR-R gives a significant advantage – the speed-up increases by 37.2% comparing with the the ParNSGA/HR-2.

Solution of CFL Problem Using Parallel NSGA-II

Performance of parallel NSGA-II has been also investigated by solving Competitive Facility Location (CFL) problem.

In the first instance the performance of two parallel algorithms ParNSGA/DR and ParNSGA/HR which utilize distributed memory parallel programming model has been investigated. The multi-objective competitive facility location problem has been solved using two different sizes of the population: 256 and 512 individuals. It was expected to locate 5 new facilities, therefore the number of variables of the problem was 10 as each facility has two coordinates. Later the number of facilities expected to locate has been increased to 25 thus increasing to 50 the number of problem variables. The algorithm has been performed for 250 generations in each experiment. Depending on population size and the number of problem variables, duration of sequential algorithm varies from around 18 minutes (locating 5 facilities using population of 256 decision vectors) to more than 12 hours (locating 25 facilities using population of 2048 decision vectors). The number of processors varies up to

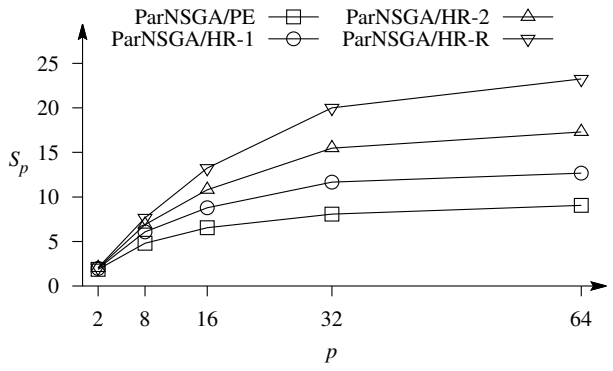

Figure 12: The speed-up of parallel NSGA-II algorithm using different strategies

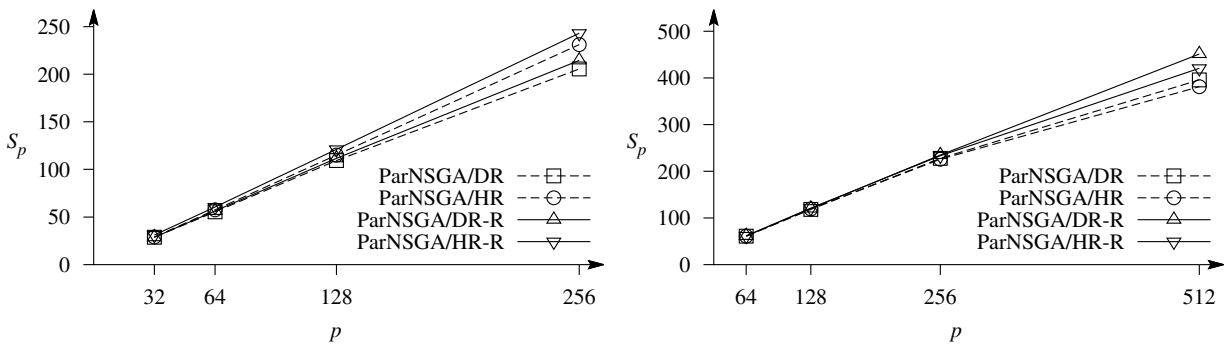

Figure 13: The speed-up of the algorithms when solving CFL problem with 10 variables using different population sizes: 256 (left) and 512 (right)

the maximum so that the workload (generation of child population and evaluation of objective functions) could be equally divided among all processors – up to 256 processors for population of 256 individuals, and up to 512 processors for population of 512 individuals. The results of the investigation are presented in Figure 13. Results show that reduction of offsprings' population gives significant advantage in both HR and DR strategies. Using 256 individuals population, algorithm ParNSGA/HR-R gives better performance utilizing parallelization of Pareto ranking without additional cost for communication, but forces processors to be idle. When the size of the population has been increased to 512 individuals, the performance of both algorithms was similar when 256 processors have been used, however ParNSGA/DR-R gives advantage against ParNSGA/HR-R when 512 processors have been used. It could be explained as overmuch large idle time of processors using large population in ParNSGA/HR-R.

Another experimental investigation has been performed to investigate the influence of utilization of shared memory parallel programing model to the performance of the algorithm. The same optimization problem has been solved using ParNSGA/DR-R. Two different populations consisting of 1024 and 2048 individuals have been investigated by performing computations on different number of processors which varies up to 1024 and 2048 depending on the size of the population. Different number of threads per shared memory processing unit (4 and 16 threads) have been also investigated. Results of the investigation are presented in Figure 14. Results show that utilization of shared memory parallel programing model gives significant advantages to the performance of the algorithm when using up to 512 processors for population of 1024 individuals and up to 1024 processors for population of 2048 individuals. Increasing the number of processors to the maximum, the performance falls down using either 4 or 16 threads per processor. Note that using the maximum number of processing units the workload per processor is very small comparing with initialization an synchronization costs using shared memory parallel programming subroutines.

Consequently the number of problem variables has been increased from 10 to 50 thus increasing five times the computational costs per processor. The size of the population has been chosen to be 1024 individuals and the number of processors varies up to 1024 as before while investigating the performance of the algorithms with the population of the same size. Results of the investigation are illustrated in Figure 15. Results show that hybrid distributed-shared memory algorithm gives

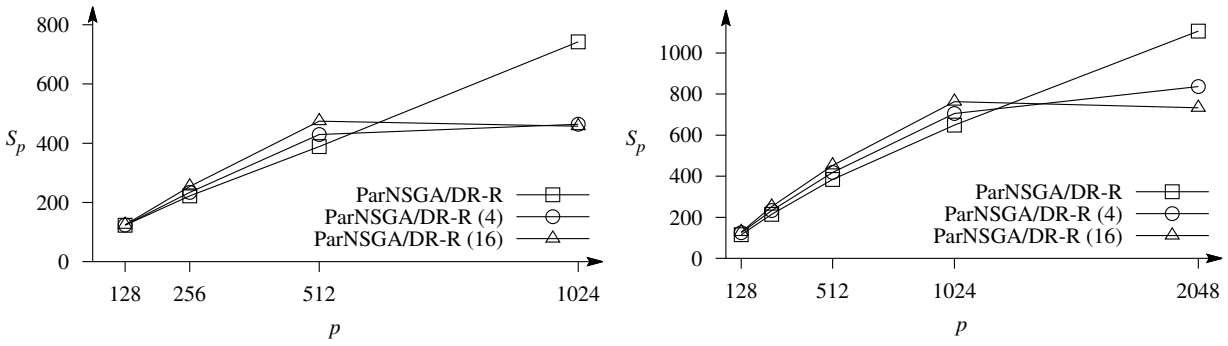

Figure 14: The speed-up of the algorithms when solving CFL problem with 10 variables using different population sizes: 1024 (left) and 2048 (right)

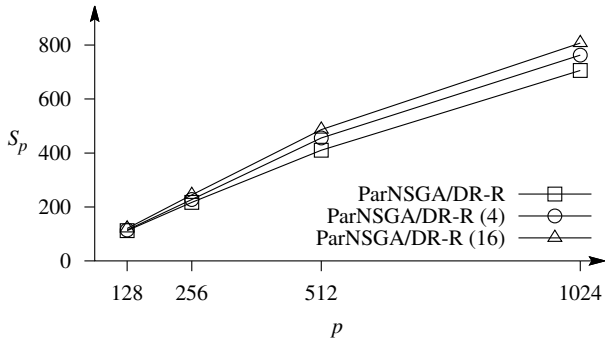

Figure 15: The speed-up of the algorithms when solving CFL problem with 50 variables using different population of 1024 individuals

significant advantage, moreover using more threads per shared memory processing unit, the hybrid algorithm was more superior comparing with the algorithm which utilizes distributed memory parallel programing model only.

Solution of Discrete CFL Problem Using NSGA-II

The performance of NSGA-II when solving discrete CFL problem has been experimentally investigated using real geographical and population data. It was supposed that each of firms has 3 preexisting facilities located in the most populated demand points. A set of candidate demand points L consists of required number of the remaining largest demand points including those in which facility of firm B is already located.

Four different configurations of parameters of the problem have been investigated (see Table 2) which vary on the number of facilities expected to locate (number of variables, r) and the number of candidate locations (size of the search space, k).

The value of attractiveness that customers from demand point i feels to the facility j has been calculated by

$$a_{ij} = \frac{q_j}{1 + d_{ij}}, \tag{20}$$

Table 2: Four configuration of parameters of the discrete CFL problem.

	I	II	III	IV
r	3	3	5	10
k	100	500	100	500

where d_{ij} is a distance between i-th demand point and j-th facility, and q_j is a predefined value of quality of j-th facility. Quality values q_i, $i = 1, 2, \ldots, 6$ of preexisting facilities have been chosen to be $(57, 60, 59, 45, 56, 36)$. All new facilities have been assumed to be of the same quality $q_n = 35$. In particular case, if quality values of all facilities (new and pre-existing) are equal, the patronizing behavior become the distance-based, what means that all customers from a demand point are served by the closest facility. This variant of the model has been also investigated.

A complete enumeration algorithm has been used to determine true Pareto fronts of CFL problem with first three configurations of parameters. NSGA-II with population size of 250 individuals and pure random mutation has been run for 250 generations to investigate the opportunities to approximate Pareto fronts. One hundred independent runs have been performed to obtain average values of performance metrics.

The optimization problem seems to be quite simple as it requires C^3_{100} function evaluations to perform the complete enumeration, however it requires around 15 minutes (using Intel(R) Core(TM) i5 CPU 760 @ 2.80GHz hardware). Increasing value of k to 500 increases the number of of function evaluations to C^5_{500} and the complete enumeration algorithm requires around 18 hours. Increasing the number of facilities expected to locate from 3 to 5 (third configuration of parameters), duration of computations increases to around 78 hours.

Although, depending on initial parameters, time required to perform complete enumeration increases up to 78 hours in any of these three cases Pareto front can be quite precisely approximated by classical version NSGA-II algorithm within 25000 function evaluations – the maximum value of Inverted Generational Distance (IGD) metric was less than 0.004.

More intractable was the problem with the last configuration of parameters – expecting to choose location for 10 facilities from the set of 500 candidates. Since it was almost impossible to perform the complete enumeration in reasonable time, an approximation of true Pareto front has been constructed from all approximations (around 1500) obtained by all experiments made during the investigation.

The average value of IGD was 0.0144. In order to improve the performance of the algorithm the local mutation strategy (see eq. (18)) with different values of parameter h (3, 5, 10, 20 and 30) has been utilized. The results of the investigation are presented in Figure 16 (left), where different columns represent different models of behavior of customers: distance-based and attractive-based. The results show that change of the value has significant impact on the value of IGD, and it is worth to use $h = 10$.

Further an impact of incorporation of local search algorithm into NSGA-II has been investigated. The same CFL problem has been solved by performing NSGA-II with local mutation strategy and $h = 10$ for 24000, 20000, 15000 and 10000 function evaluations (240, 200, 150, 100 generations respectively). The remaining evaluations have been used for local search. Results of the investigation presented in Figure 16 (right) show that devotion of function evaluations to local search can significantly

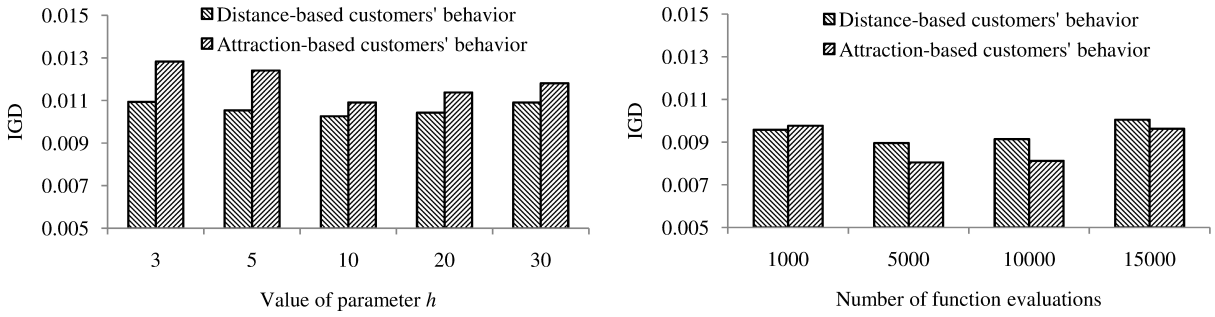

Figure 16: Values of IGD, obtained using different values of parameter h (left) and devoting different number of function evaluations for the local search (right)

improve quality of final Pareto front approximation. From the results we can see that it is worth to devote 5000–10000 function evaluations to local search.

General Conclusions

1. The modification of Particle Swarm Optimization algorithm, based on reduction of the search area has been proposed. Results of the experimental investigation showed that appropriate reduction of the search area can increase performance of the algorithm by 25% when solving Multiple Gravity Assist problem.

2. Experimental investigation of parallel Particle Swarm Optimization algorithm showed that the best performance is achieved using asynchronous strategy, based on devotion of one processors for management of communication.

3. The multi-objective local search algorithm has been developed by modifying single-objective local search strategy for multi-objective optimization. The developed algorithm has been incorporated into Non-dominated Sorting Genetic Algorithm thus developing a hybrid algorithm for global multi-objective optimization. Results of the experimental investigation showed that the proposed algorithm solves better than classical genetic algorithm from 53% to 77% test problems, depending on performance metric.

4. Several strategies to parallelize Pareto ranking in multi-objective optimization algorithms have been proposed and experimentally investigated. Results of the investigation showed that usage of the proposed strategies can increase speed-up of the parallel algorithm 1.6 times.

5. The strategy for mutation in genetic algorithm when solving Competitive Facility Location problem has been proposed and investigated. The local search based on the proposed strategy has been developed and investigated. Results of the experimental investigation showed that appropriate usage of the proposed strategy for mutation and the local search algorithm can increase performance of genetic algorithm by $\sim$43% when solving competitive facility location problem.

List of Publications on the Topic of Dissertation

[A1] **A. Lančinskas**, J. Žilinskas. Methods for generation of random numbers in parallel stochastic algorithms for global optimization. *Journal of Young Scientists*, 2(27):118–122, 2010. ISSN 1648-8776.

[A2] **A. Lančinskas**, J. Žilinskas, P.M. Ortigosa. Investigation of parallel particle swarm optimization algorithm with reduction of the search area. In *2010 IEEE International Conference on Cluster Computing (Cluster Workshops)*, p. 1–5, Sept. 2010. ISBN 978-1-4244-8395-2.

[A3] **A. Lančinskas**, J. Žilinskas, P.M. Ortigosa. Local optimization in global multi-objective optimization algorithms. In *2011 Third World Congress on Nature and Biologically Inspired Computing (NaBIC)*, p. 323–328, 2011. ISBN 978-1-4577-1122-0.

[A4] **A. Lančinskas**, J. Žilinskas. Approaches to parallelize Pareto ranking in NSGA-II algorithm. In *Parallel Processing and Applied Mathematics, Vol. 7204 of Lecture Notes in Computer Science*, p. 371–380, 2012. ISSN 0302-9743.

[A5] **A. Lančinskas**, J. Žilinskas. Solution of multi-objective competitive facility location problems using parallel NSGA-II on large scale computing systems. In *Applied Parallel and Scientific Computing, Vol. 7782 of Lecture Notes in Computer Science*, p. 422–433, 2013. ISSN 0302-9743.

[A6] **A. Lančinskas**, P.M. Ortigosa, J. Žilinskas. Multi-objective single agent stochastic search in Non-dominated Sorting Genetic Algorithm. *Nonlinear Analysis: Modelling and Control*. ISSN 1392-5113. [Accepted]

About the Author

Algirdas Lančinskas was born on the 29th of January, 1985 in Kaišiadorys district, Žiežmariai. He received a Bachelor's degree in Mathematics from Vilnius Pedagogical University in 2007, and Master's degree in Informatics from Vilnius Pedagogical University in 2009. From 2009 till 2013 he was a PhD student of Vilnius University, Institute of Mathematics and Informatics. From 2006 till 2008 he was working as laboratory assistant, and from 2008–2010 – as a head of laboratory at Vilnius Pedagogical University, Faculty of Mathematics and Informatics. From 2010 he became an assistant at Vilnius Pedagogical University, Faculty of Mathematics and Informatics. At present he is lecturer at Lithuanian University of Educational Sciences and junior researcher at Vilnius University. Algirdas Lančinskas studied for 3 months as ERASMUS student at University of Almeria (Spain) in 2010, 3 months worked on research visit at EPCC (the super computing center at University of Edinburgh), United Kingdom in 2011.

ATSITIKTINĖS PAIEŠKOS GLOBALIOJO OPTIMIZAVIMO ALGORITMŲ LYGIAGRETINIMAS

Tyrimų sritis ir darbo aktualumas

Optimizavimo uždaviniai moksle ir praktikoje kyla dar nuo antikinių laikų. Yra pasiūlyta daug įvairių optimizavimo metodų, skirtų spręsti optimizavimo uždaviniams, kylantiems įvairiose srityse, tokiose kaip chemija, biologija, biomedicina, operacijų tyrimas ir pan.

Paprastai efektyviausiai sprendžiami uždaviniai, turintys tam tikras savybes, tokias kaip tikslo funkcijų tiesiškumas, iškilumas, diferencijuojamumas ir pan. Tačiau ne visi praktikoje kylantys optimizavimo uždaviniai tenkina šias savybes, o kartais iš vis negali būti išreiškiami adekvačia matematine išraiška. Be to, optimizavimo uždaviniai, kurių tikslo funkcijos netenkina iškilumo savybės, gali turėti daug lokalių sprendinių, kurių geriausiojo radimas yra sudėtingas uždavinys. Dėl šių priežasčių yra populiarūs atsitiktinės paieškos optimizavimo metodai, kurių įvairovė apima tiek optimizavimą pagal vieną kriterijų, tiek daugiakriterį optimizavimą.

Šiuolaikinės technologijos lygiagrečiųjų skaičiavimų srityje leidžia išspręsti daug skaičiavimo resursų reikalaujančius optimizavimo uždavinius. Aktuali problema yra optimalus uždavinio suskaidymas į viena nuo kitos nepriklausomas užduotis, siekiant optimaliai paskirstyti darbą procesoriams ir minimizuoti duomenų perdavimo kaštus. Sprendžiant optimizavimo uždavinius atsitiktinės paieškos algoritmais yra aktualu užtikrinti, kad skirtingi procesoriai generuotų skirtingas atsitiktinių skaičių sekas. Ši problema ypač aktuali naudojant bendrosios atminties lygiagrečiųjų skaičiavimų sistemas.

Todėl šio darbo tyrimų sritis yra atsitiktinės paieškos globaliojo optimizavimo algoritmai, jų lygiagretinimas ir taikymas praktikoje pasitaikantiems uždaviniams spręsti.

Tyrimų objektas

Disertacijos tyrimų objektas yra:

- atsitiktinės paieškos globaliojo optimizavimo algoritmai;
- lygiagrečiųjų skaičiavimų sistemos;
- atsitiktinių skaičių sekų generavimo metodai.

Darbo tikslas ir uždaviniai

Darbo tikslas – modifikuoti esamus ir pasiūlyti naujus atsitiktinės paieškos globaliojo optimizavimo lygiagrečiuosius algoritmus, siekiant efektyvesnio optimizavimo uždavinių sprendimo. Siekiant šio tikslo buvo sprendžiami tokie uždaviniai:

- apžvelgti esamus atsitiktinės paieškos globaliojo optimizavimo metodus ir algoritmus bei jų lygiagretinimo būdus, išskirti tiriamų algoritmų grupę;

- ištirti atsitiktinių skaičių generavimo lygiagrečiųjų skaičiavimų sistemose galimybes ir metodus;
- siekiant efektyvesnio optimizavimo uždavinių sprendimo, modifikuoti ir išlygiagretinti tiriamus algoritmus, atsižvelgiant į sprendžiamą uždavinį, techninę ir programinę įrangą;
- eksperimentiniu būdu ištirti pasiūlytų algoritmų efektyvumą;
- gautus rezultatus palyginti su kitais rezultatais, gautais gerai žinomais atsitiktinės paieškos algoritmais;
- ištirti pasiūlytų algoritmų taikymo galimybes sprendžiant įvairių tipų uždavinius.

Mokslinis darbo naujumas

Pasiūlyta dalelių spiečiaus optimizavimo algoritmo modifikacija erdvėlaivių skrydžių trajektorijų optimizavimui, grįsta leistinosios paieškos srities mažinimu. Ištirtos kelios informacijos apsikeitimo tarp kompiuterių dalelių spiečiaus optimizavimo lygiagrečiajame algoritme strategijos.

Pasiūlytas lokaliojo daugiakriterio optimizavimo algoritmas, grįstas vieno agento stochastinės paieškos strategija. Pasiūlytas lokaliojo optimizavimo algoritmas įkomponuotas į gerai žinomą globaliojo daugiakriterio optimizavimo genetinį algoritmą, taip gaunant hibridinį globaliojo daugiakriterio optimizavimo algoritmą.

Pasiūlyta sprendinių vertinimo (rangavimo) daugiakriterio optimizavimo algoritmuose lygiagretinimo strategija. Pasiūlytos strategijos pagrindu išlygiagretintas daugiakriterio optimizavimo genetinis algoritmas, skirtas tiek paskirstytosios, tiek bendrosios atminties lygiagrečiųjų skaičiavimų sistemoms. Algoritmo efektyvumas eksperimentiniu būdu ištirtas sprendžiant daugiakriterio optimizavimo uždavinius, skaičiavimams naudojant iki 2048 procesorių.

Ginamieji disertacijos teiginiai

1. Tinkamas leistinosios srities siaurinimas gali ženkliai padidinti dalelių spiečiaus optimizavimo algoritmo efektyvumą sprendžiant erdvėlaivių trajektorijų optimizavimo uždavinį.
2. Pasiūlytas hibridinis daugiakriterio globaliojo optimizavimo algoritmas efektyviai sprendžia eksperimentiniam tyrimui naudotus daugiakriterio optimizavimo testo uždavinius.
3. Pasiūlytos daugiakriterio genetinio algoritmo modifikacijos ir lygiagretinimo strategijos leidžia efektyviai spręsti optimizavimo uždavinius didelio našumo lygiagrečiųjų skaičiavimų sistemose.
4. Pasiūlyti algoritmai efektyviai sprendžia erdvėlaivių skrydžių trajektorijų optimizavimo ir konkuruojančių objektų vietos parinkimo uždavinius.

Darbo rezultatų aprobavimas

Pagrindiniai disertacijos rezultatai paskelbti 6 mokslinėse publikacijose: 4 periodiniuose recenzuojamuose mokslo žurnaluose; 2 konferencijų pranešimų medžiagoje.

Disertacijos rezultatai pristatyti 4 nacionalinėse ir 6 tarptautinėse konferencijose ir seminaruose.

Disertacijos struktūra

Disertaciją sudaro įvadas, 3 skyriai ir bendrosios išvados. Papildomai disertacijoje pateikta: naudotų žymėjimų ir santrumpų sąrašas, iliustracijų sąrašas ir literatūros sąrašas. Bendra disertacijos apimtis yra 106 puslapiai, kuriuose pateikta 37 paveikslai, 10 lentelių ir 4 algoritmai. Disertacijoje remtasi 77 literatūros šaltiniais.

Bendrosios išvados

Išsprendus darbe suformuluotus uždavinius gautos tokios išvados:

1. Įgyvendinta dalelių spiečiaus optimizavimo algoritmo modifikacija, grįsta leistinosios srities mažinimu. Eksperimentiniu tyrimu nustatyta, kad tinkamas leistinosios srities mažinimas gali padidinti algoritmo efektyvumą iki 25 % sprendžiant erdvėlaivių skrydžių trajektorijų optimizavimo uždavinį.

2. Lygiagrečiojo dalelių spiečiaus optimizavimo algoritmo eksperimentinis tyrimas parodė, kad geriausias algoritmo spartinimo koeficientas pasiekiamas lygiagretinant asinchroninę dalelių spiečiaus optimizavimo algoritmo versiją, vieną iš procesorių išskiriant informacijos mainų užtikrinimui.

3. Pasiūlytas lokaliojo daugiakriterio optimizavimo algoritmas, modifkuojant vieno agento stochastine paiešką grįstą algoritmą. Siūlomas algoritmas įkomponuotas į daugiakriterį genetinį algoritmą, taip sudarant hibridinį globaliojo daugiakriterio optimizavimo algoritmą. Eksperimentiniu tyrimu nustatyta, kad pasiūlytas algoritmas sprendžia geriau už klasikinį daugiakriterio optimizavimo genetinį algoritmą nuo 53 % iki 77 % testo uždavinių, priklausomai nuo vertinimo kriterijaus.

4. Pasiūlytos sprendinių vertinimo daugiakriterio optimizavimo algoritmuose lygiagretinimo strategijos. Eksperimentinio tyrimo rezultatai parodė, kad siūlomos algoritmo lygiagretinimo strategijos gali padidinti lygiagrečiojo algoritmo spartinimo koeficientą 1,6 karto.

5. Pasiūlyta mutavimo genetiniame algoritme strategija, taikytina sprendžiant konkuruojančių objektų vietos parinkimo uždavinį. Pasiūlytos strategijos pagrindu įgyvendintas lokaliojo optimizavimo algoritmas. Eksperimentinis tyrimas parodė, kad tinkamas siūlomos mutacijos strategijos ir lokaliosios paieškos naudojimas gali padidinti genetinio algoritmo efektyvumą ∼43 %, sprendžiant konkuruojančių objektų vietos parinkimo uždavinį.

Trumpai apie autorių

Algirdas Lančinskas gimė 1985 m. sausio 29 d. Žiežmariuose, Kaišiadorių rajone. 2007 metais įgijo matematikos bakalauro laipsnį Vilniaus pedagoginiame universitete. 2009 metais Vilniaus pedagoginiame universitete įgijo informatikos magistro laipsnį. Nuo 2009 iki 2013 buvo Vilniaus universiteto Matematikos ir informatikos instituto doktorantas.

Nuo 2006 iki 2008 metų dirbo laborantu, o nuo 2008 iki 2010 – laboratorijos vedėju Vilniaus pedagoginio universiteto Matematikos ir informatikos fakultete. Nuo 2010 metų pradėjo dirbti asistentu Vilniaus pedagoginio universiteto Informatikos katedroje. Šiuo metu yra Lietuvos edukologijos universiteto Informatikos katedros lektorius ir Vilniaus universiteto jaunesnysis mokslo darbuotojas.

2010 metais 3 mėnesius studijavo Almerijos universitete (Ispanijoje) pagal studentų mainų programą ERASMUS. 2011 metais 3 mėnesius stažavosi Edinburgo superskaičiavimų centre EPCC (Jungtinėje Karalystėje).

Algirdas Lančinskas

PARALLELIZATION OF RANDOM SEARCH
GLOBAL OPTIMIZATION ALGORITHMS

Doctoral dissertation

Physical Sciences (P 000)
Informatics (09 P)
Informatics, Systems Theory (P 175)

Algirdas Lančinskas

ATSITIKTINĖS PAIEŠKOS GLOBALIOJO OPTIMIZAVIMO
ALGORITMŲ LYGIAGRETINIMAS

Daktaro disertacija

Fiziniai mokslai (P 000)
Informatika (09 P)
Informatika, sistemų teorija (P 175)