VILNIUS UNIVERSITY

FACULTY OF MATHEMATICS AND INFORMATICS

MODELLING AND DATA ANALYSIS MASTER'S STUDY PROGRAMME

MASTER'S THESIS

Parallelization of Multi-objective Optimization Methods

Daugiakriterio optimizavimo metodų lygiagretinimas

Mantas Nemura

Supervisor: Algirdas Lančinskas, PhD

Vilnius, 2021

Abstract

The purpose of this research is to create a novel parallel optimization algorithm while focusing mainly on efficient Pareto ranking strategy. A theoretical background of multi-objective optimization and parallel computing is provided at the first part of the report. Two different Pareto ranking approaches - dominance-rank and dominance-depth - are used for algorithm implementation and hence two separate instances are distinguished. Implementation details of proposed algorithm are comprehensively explained and the performance results of two parallel ranking variants are presented and discussed. The implemented algorithm is applied to solve a facility location problem where two existing companies compete for the market.

Keywords: multi-objective optimization, Pareto ranking, non-dominated sorting, genetic algorithm, parallel computing, parallel optimization

Santrauka

Pagrindinis šio baigiamojo darbo tikslas yra sukurti naują lygiagretųjį optimizavimo algoritmą pagrindinį dėmesį skiriant Pareto rangavimo proceso lygiagretinimui. Daugiakriterio optimizavimo ir lygiagrečiųjų skaičiavimų teorinis pagrindas yra pateiktas pirmojoje darbo dalyje. Du skirtingi Pareto rangavimo būdai yra naudojami kuriant algoritmą, todėl analizuojant rezultatus lyginamos dvi strategijos. Pasiūlyto algoritmo struktūra išsamiai aprašyta antrajame darbo skyriuje. Pateikiami sukurtų lygiagrečiųjų programų efektyvumo rezultatai ir atliekamas jų palyginimas. Galiausiai, sukurtas algoritmas pritaikomas objektų vietos parinkimo uždaviniui spręsti.

Raktiniai žodžiai: daugiakriteris optimizavimas, Pareto rangavimas, rikiavimas pagal nedominuojamumą, genetinis algoritmas, lygiagretusis skaičiavimas, lygiagretusis optimizavimas

Glossary

P	parent population
Q	offspring population
N	population size
k	number of objectives
n	number of optimization variables
p	number of processors
$\mathbb{N}$	set of all natural numbers
MOP	Multi-objective Optimization Problem
FLP	Facility Location Problem
NSGA-II	Non-dominated Sorting Genetic Algorithm
BOS	Best Order Sort
CPU	Central Processing Unit
GA	Genetic Algorithm
EA	Evolutionary Algorithm
MPI	Message Passing Interface

Contents

Introduction

The need of optimization is present in majority of various business, science and engineering fields. Constant improvement of different processes or artifacts requires more complex and more advanced solutions for emerging optimization problems. Many stochastic as well as deterministic optimization algorithms can be applied for solving various kinds of these problems. One of the most extensively used stochastic optimization algorithms is genetic algorithm, which can deal with both single-objective and multi-objective problems and is well known for its versatility and robustness.

However, genetic algorithm is computationally demanding and complex problems can often be solved in inordinate amount of time. This drawback can be mitigated by applying parallel computing techniques. In order to reach the desired benefit from employing additional computational resources, an efficient parallelization strategy is needed. Since genetic optimization algorithm, as well as all the other algorithms, are composed of several coherent procedures, the strategy of executing those procedures in parallel is usually nontrivial. The ability of optimization algorithms to be adapted for parallel computing systems reaching appropriate efficiency levels is still an unresolved matter.

Due to high relevance emphasized above, genetic optimization algorithm and its parallelization are the primary objectives of this research. The main aim of the research is to propose an effective parallel Pareto ranking strategy using the basis of genetic algorithm. Parallelization of objective values calculation is rather trivial and hence more focus was given to Pareto ranking process. In order to achieve the mentioned goal, the following tasks were performed:

- making an overview of current achievements in the field;
- designing a novel optimization algorithm while focusing on parallelization of Pareto ranking;
- implementing and evaluating the designed algorithm;
- performing the benchmarking of algorithm variants;
- applying the proposed algorithm to solve a real-world problem.

1 Literature review

Mathematical optimization is a branch of applied mathematics which is useful in such fields as manufacturing, economics, engineering, etc. Determining the best solution is a canonical interpretation of every optimization problem even though they can look very different. As such problems are very computational intensive, optimization becomes closely related to parallel computing. With the help of parallelization, computational time can be significantly reduced and because of that, additional relevant applications can be constructed. Main terms and concepts of optimization and parallel computing are discussed in further chapters.

1.1 Optimization

Dealing with optimization problems can be perceived in human behavior, nature shapes, laws of physics and many other areas. For instance, people tend to choose the shortest path to work or school, trees arrange their leaves in particular angles to maximize the amount of light captured. An infinite number of examples in which people or nature objects face and even unconsciously solve such problems can be thought of.

Pursuit of the best can be found as a common thing in the latter examples. However, generally it is not possible to fulfill all the needs in the finest way simultaneously. Getting better in one aspect usually causes contrary quantitative or qualitative changes in another. In addition to this, a variety of different constraints can be present. Therefore, finding the best option becomes a problem that needs to be solved. This is the point where mathematical optimization plays its role. Many optimization methods can be utilized to find the optimal solutions while taking different objectives and the constraints into consideration at the same time.

In order to solve the optimization problem in numerical way, optimality of a solution must be expressible by quantitative measures. Such measure, which enables the solutions to be compared among themselves, is called objective function (also referred as cost or fitness function). The value of objective function depends on optimization variables which correspond to characteristics of phenomenon investigated [1]. Several categories of optimization are distinguished in [5] and some of them are detailed in Table 1.1.

Multi-objective optimization methods are mainly investigated in this research. The details of this kind of optimization are provided further in the text.

1.1.1 Multi-objective optimization

Multi-objective optimization studies problems that involve more than one objective function. Such problems can be mathematically described as follows:

Table 1.1: Optimization categories.

Categorization criteria	Explanation
single / multiple variable	The optimization is one-dimensional if objective function depends on one variable and multi-dimensional if it depends on more than one variable.
static / dynamic	One of the arguments of an objective function is time in dynamic case while time does not play a role in static optimization.
constrained / unconstrained	Constrained optimization incorporates variable equalities and inequalities into the objective function. Unconstrained optimization allows the variables to get any value.
continuous / discrete	Set of possible values of optimization variables can be discrete and continuous.
single-objective / multi-objective	One or more objective functions can be sought to minimize using the same independent variables.

Definition 1. General Multi-Objective Optimization Problem (MOP): *In general, an MOP minimizes* $F(\vec{x}) = (f_1(\vec{x}), ..., f_k(\vec{x}))$ *subject to* $g_i(\vec{x}) \leq 0$, $i = 1, ..., m$, $\vec{x} \in \Omega$. *An MOP solution minimizes the components of a vector* $F(\vec{x})$ *where* $\vec{x}$ *is an n-dimensional decision variable vector* $(\vec{x} = x_1, ..., x_n)$ *from some universe* Ω [13].
There k is number of objectives, $g_i(\vec{x}) \leq 0$ is i-th out of m constraints and Ω is a set of all possible $\vec{x}$ that can be used for objective functions evaluation (further also referred as decision space). In majority of cases, it is not possible to find an optimal solution according to all objectives. Decreasing value of one objective usually causes opposite behavior of another. There are several fitness measurement methods used for dealing with such cases [6]:

- **Scalar** approach, where dimensionality of an MOP is reduced to 1D by applying aggregation of objectives.
- **Criterion-based** approach, where each objective function is treated separately. Solutions are compared according to a single objective function, independently from the others.
- **Dominance-based** approach, where solutions are classified based on dominance relation. The main techniques are:
  - *Dominance-rank*: counting number of solutions that dominate a solution which is being evaluated.
  - *Dominance-count*: counting number of solutions that are dominated by the one which is being evaluated.
  - *Dominance-depth*: classifying the solutions into different fronts (classes).

- **Indicator-based** approach, where characteristic of fitness is calculated by using predefined quality indicator.

The first two techniques are more often applied on simpler problems while dominance-based and dominance-depth approaches are suitable for more complex ones. Dominance-based comparison methods are used in this research, and therefore more detail on this is given in the following chapters.

1.1.2 Dominance relation

Dominance-based techniques rely on Pareto-dominance criterion which is a classic variant of dominance relation. It is considered that decision vector $u = (u_1, ..., u_n)$ dominates another vector $v = (v_1, ..., v_n)$ (denoted by $u \prec v$) if for all of objectives f_i, $i = 1, ..., k$ an inequality

$$f_i(v) \leq f_i(u) \tag{1.1}$$

holds true and exists at least one objective f_j, $j = 1, ..., k$ that

$$f_j(v) < f_j(u). \tag{1.2}$$

If $u \prec v$, then decision vector u is called a *dominator* of v and vector v is considered being *dominated* by u. Solution, which has no dominators, is called non-dominated or Pareto optimal. A set of such non-dominated solutions is called Pareto set and their corresponding objective vectors form a Pareto front.

1.1.3 Pareto ranking

Pareto ranking is a dominance-based technique for measuring solution fitness. Every decision vector gets assigned a dominance rank while executing this ranking process. There are variety of methods and definitions of Pareto ranking. Dominance-rank and dominance-depth approaches are used in this research.

In the case of dominance-rank approach (see left part of Figure 1.1), solution fitness is evaluated by simply calculating number of existing dominators. The less dominators a solution has, the more optimal it is. All non-dominated decision vectors (having no dominators) form a Pareto set. This ranking method is well suitable for optimization problems in which only the first Pareto front is relevant.

Dominance-depth technique (see right part of Figure 1.1) divides all population members into non-dominated fronts. Such ranking process is more computationally demanding compared to previous one as more comparisons of decision vectors have to be performed. First non-dominated front is determined using the same process as in dominance-rank case. Then all decision vectors that fall into the current front are temporarily removed from the population and the latter procedure is repeated. Such process is repeatedly executed until all solutions get classified. This method gives a better view of relations between popula-

tion items, however it is way more complex than the dominance-rank technique in terms of computational resources usage.

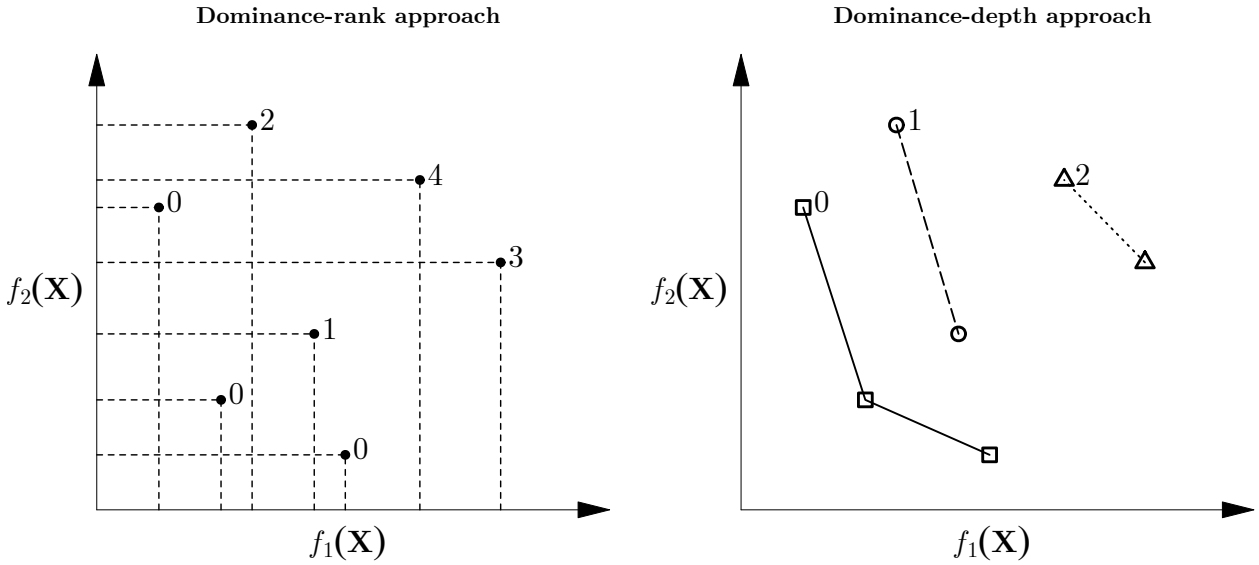

Figure 1.1: Dominance-based ranking techniques: dominance-rank (left) and dominance-depth (right) approaches.

Different Pareto fronts can be obtained when using different optimization methods or their execution parameters. Better or worse approximations of true Pareto front are made in terms of convergence and diversity metrics. Convergence indicator evaluates the closeness of Pareto front approximation to true Pareto front (the closer the better). Diversity indicator characterizes the scatteredness of points from a Pareto front approximation (the more evenly distributed they are, the better). To evaluate the goodness of Pareto front, several measures are used [7]. A few of them are described below:

- **Pareto size** defines, how many elements a Pareto front consists of.
- **Hypervolume**. This measure defines the hypervolume which is bounded by points from Pareto front and additional reference point. It reflects the spread of Pareto front approximation, distance to real Pareto front (convergence) and the diversity of points. The better the latter criteria are fulfilled, the grater value of hypervolume is. An illustration of hypervolume in two-objective case is given in Figure 1.2.
- **Coverage metric** defines, how many points from true Pareto front is within the evaluated front.

As hypervolume sheds a light on several quality indicators of the front, this metric is used further in this work for comparing quality of different fronts.

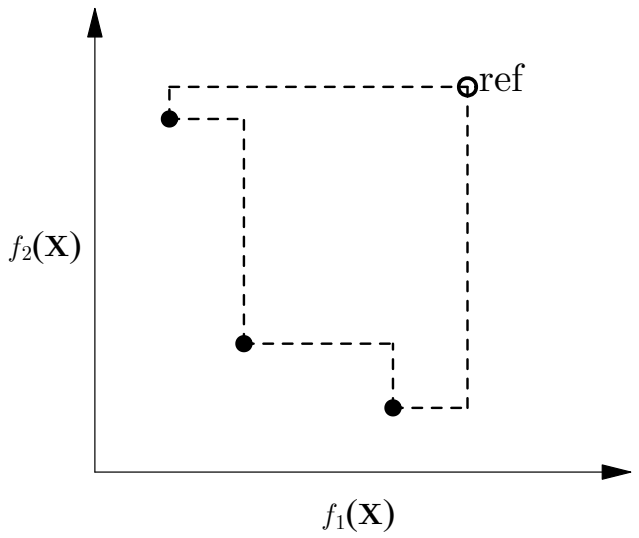

Figure 1.2: Hypervolume of Pareto front approximation.

1.1.4 Applications of multi-objective optimization

Multi-objective optimization is applied in many fields of science. This kind of optimization can be used to solve finance and risk-related problems in economics, to design optimal processes or infrastructures in various industries, etc. The proposed algorithm in this research is validated by solving facility location problem (FLP); therefore, FLP is described below as an illustrative application.

Facility location problem is a classical optimization problem that studies the optimal placement of facilities. Such problems are usually characterized by four basic components (without loss of generality, facility-customer case is described) [10]:

- Customers that are located at demand points or routes.
- Facilities that are being located.
- Geographic surface or space where demand points and facilities are located.
- Metric that indicates time, cost, distance or any other objective which quantifies the relation between facility and demand point.

A facility can represent any service establishment such as supermarket, business center, warehouse, transport hub or other similar object. The aim of FLP is to identify the best spacial location for facilities that various factors would be optimized.

A real-world example of FLP utilization could be the optimization of supply chain by choosing the right locations for new warehouses. In this problem, warehouses are considered as facilities and demand points are represented by stores of a company. Optimization can be invoked for minimizing rental and staffing costs, distances to suppliers and stores while at the same time maximizing, for instance, available area for construction of the warehouse.

Another example, which is worth to be discussed in more detail, is competitive facility location game. This problem is usually solved when an existing company seeks to establish

a new service center or a new company tries to enter the market (most often oligopoly). The purpose of careful investigation of the market before making the decision is to be able to maximize the market share which is going to be served by new facility. In case of existing company expansion, overlapping areas within its own controlled market share must be taken into account. New facility can take over a large share of the market, but it is completely possible that the majority of attracted customers had already been served by that company. In other words, company must maximize volume of new customers attracted and simultaneously minimize number of customers that are taken over from its own customer base when establishing a new facility.

Depending on the scale of search area, facility location problems can be discrete or continuous. If the location is searched on a worldwide level, the optimization problem could be considered being continuous because its feasible set is continuous space of geographical coordinates. When the problem is solved on, for instance, town level, it would rather be considered being discrete because it is impossible to place the facility wheresoever and a set of possible locations must be predefined.

1.2 Genetic algorithm

Genetic algorithm (GA) is an optimization technique based on biological evolution and natural selection principles. In this algorithm, population of solutions is iteratively improved with regard to the fitness. Biological terms are used to name different elements of mathematical genetic algorithm [2]:

- **Gene:** single element of decision vector; an optimization variable.
- **Chromosome:** a string of genes; a decision vector.
- **Individual:** chromosome together with an associated value of fitness measure.
- **Population:** a set of individuals.

A detailed schema of genetic algorithm is given in Figure 1.3. Each step of the algorithm is briefly explained further.

Initialization step covers definition of the problem, selection of genetic algorithm parameters and any other problem-specific operations that need to be performed beforehand. When the problem is clearly formulated, optimization variables and objective functions are defined. Optimization variables sometimes are expressed in non-scalar or other indirect forms, so various mathematical transformations or encoding operations can be applied. After that, different fitness functions for evaluation of decision vectors are formed. Initialization step is completed by specifying the GA execution parameters (number of iterations, population size, probabilities of crossover and mutation) and stopping criteria.

Initial population is usually generated by creating required amount of random decision vectors (chromosomes) from decision space. It is important to choose diverse values of optimization variables (genes) in order to avoid getting stuck in local minimum.

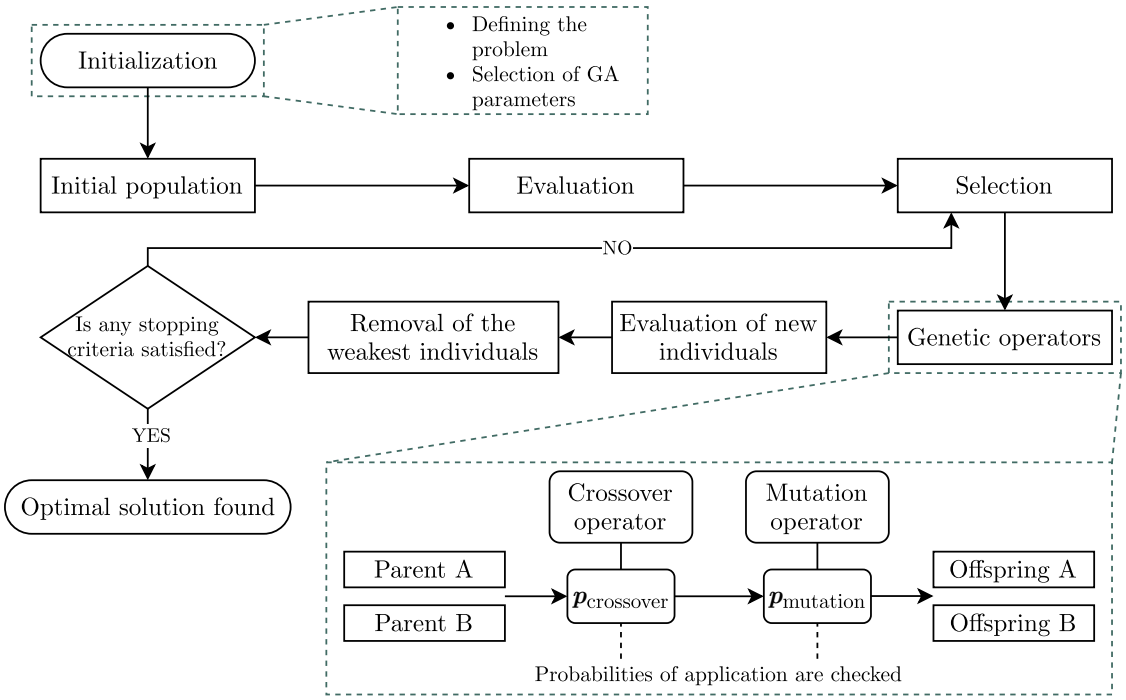

Figure 1.3: Flow chart of genetic algorithm.

Evaluation of initial population is performed by calculating values of objective functions for each of the individuals.

Selection. Parents are usually selected for mating based on their fitness. This is where natural selection principle comes into play as fitter solutions have a higher probability to be picked for mating. Such approach can be implemented by employing various methods, one of which is roulette-wheel selection (utilized for implementation of algorithms further in this research and explained in Appendix B).

Genetic operators (crossover and mutation) are applied in process of mating two individuals. These stochastic operators are used to derive a new offspring from two parent solutions. By using crossover operator it is achieved that an offspring has a set of genes inherited from both of its parent individuals. Many different approaches exist for crossing over but the most common form involves two parents that produce two offspring. Mutation operator alters some genes of the offspring with a certain probability. Increasing the mutation probability increases the search space of the algorithm. By applying these two operators on parent individuals, the offspring containing portions of parental genetic code (inheritance) with some genes adjustments (evolution) is created.

Evaluation of new individuals is carried out (values of objective functions are calculated) just after offspring population is generated.

Removal of the weakest individuals is performed on both the current and the offspring populations in some arbitrary way. Elitist strategy is usually used here, when half of the best solutions are selected from joint parent-offspring population.

1.3 Algorithms

Two multi-objective optimization-related algorithms, NSGA-II and BOS, are outlined in this chapter. These algorithms are a foundation for a method proposed in this research.

1.3.1 NSGA-II

NSGA-II (Non-dominated Sorting Genetic Algorithm-II) is a multi-objective evolutionary algorithm (EA) prominent for fast non-dominated sorting procedure. It is a successor of primary NSGA algorithm which was criticized for high computational complexity, non-elitism approach and the need for specifying sharing parameter. NSGA-II alleviated these shortcomings by reducing the complexity, incorporating elitism and eliminating sharing parameter. It is claimed that this algorithm outperforms many other multi-objective EAs in terms of various aspects [4]. Pseudo-code of NSGA-II is given in Algorithm 1 and the most significant steps of it are briefly detailed below.

1. Population initialization. Parent population is created by generating a set of random solutions within the decision space. Any prior information about optimal solutions can be used to create the initial population what could help to improve the convergence speed.

2. Offspring population generation step is covered in chapter 1.2.

3. Non-dominated sorting. Dominance-depth fitness assignment approach is utilized in NSGA-II. A general dominance-depth sorting technique which requires $O(mN^3)$ (there m - number of objectives, N - population size) computational time is described in chapter 1.1.3. In order to reduce the computational complexity to $O(mN^2)$, a new procedure was designed. For each solution i two data entities are calculated: a number (n_i) of dominators and a set (S_i) of solutions dominated by i. Here the repetitious process starts and all solutions with $n_i = 0$ are put into current front $\mathcal{F}_j$. Then sets S_i play a part by pointing to the counters n_i that need to be decremented. The process is continued until all solutions become classified. A more detailed explanation can be found in [4].

4. Crowding distance. Comparison of individuals is based on their rank (front index). When comparing solutions from the same front, the crowding distance is calculated. This measure helps to obtain better spread of solutions. The idea of crowding distance calculation is represented in Figure 1.4. Cumulative distance between two points on either side of a particular solution along all the objectives is calculated. This distance is an estimate of cuboid (see figure) enclosing that solution without including any other point from the population.

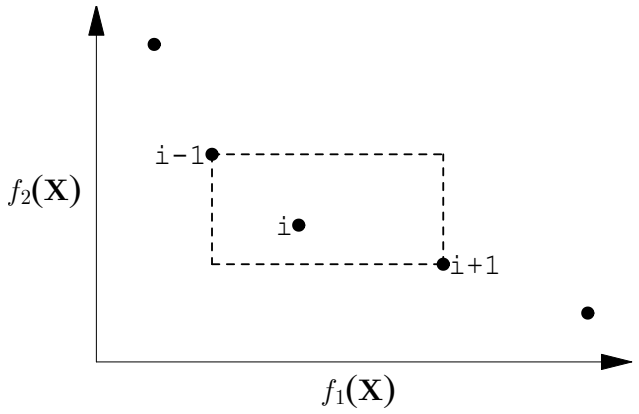

Figure 1.4: Crowding distance illustration.

5. Selection. All solutions from the lowest fronts are added to new parent population as long as population size is not exceeded. When the size of subsequent front is higher than a number of individuals that are needed to fill up the parent population, solutions with the largest crowding distance are selected.

Algorithm 1: NSGA-II

generate random parent population P_0
while *not done* **do**
  generate offspring population Q_t from P_t
  $R_t = P_t \cup Q_t$
  $\mathcal{F} = (\mathcal{F}_1, \mathcal{F}_2, ...) =$ `non-dominated-sort`$(R_t, n_{[\![1,N]\!]}, S_{[\![1,N]\!]})$
  $P_{t+1} = \emptyset$
  $i = 1$
  `/* vertical bars denote cardinality of the sets */`
  while $|P_{t+1}| + |\mathcal{F}_i| < N$ **do**
    $P_{t+1} = P_{t+1} \cup \mathcal{F}_i$
    $i = i + 1$
  end
  `crowding-distance-assignment`$(\mathcal{F}_i)$
  sort $\mathcal{F}_i$ by crowding distance in descending order
  $P_{t+1} = P_{t+1} \cup \mathcal{F}_i[1 : (N - |P_{t+1}|)]$
  $t = t + 1$
end
return $\mathcal{F}$

1.3.2 Best Order Sort (BOS)

Best Order Sort (BOS) is an algorithm for non-dominated sorting. The main advantage of it is reduced number of comparisons in non-dominated sorting procedure [11]. This algorithm is based on solutions sorting according to each objective, meaning that for each of the

objectives a sorted list is created. This helps to pre-determine candidate dominators and hence to skip unpromising comparisons.

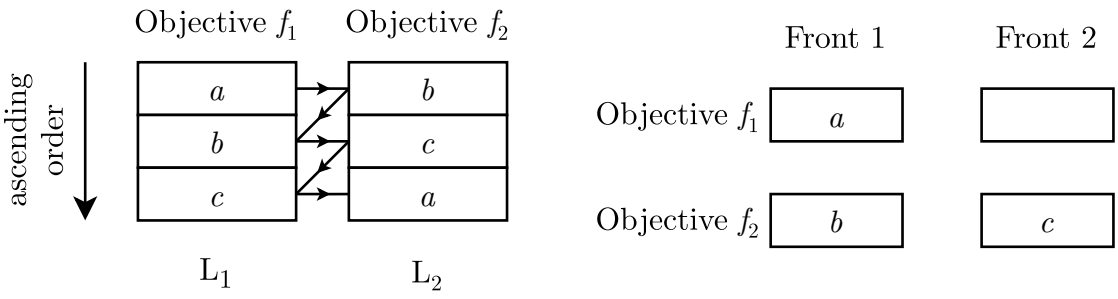

Figure 1.5: BOS example.

The algorithm will be explained through an example - two-objective case is represented in Figure 1.5. At the beginning, first element of list L_1 is classified. Since it has no candidate dominators (individual a is the best according to first objective), it goes straight to corresponding box of Front 1. Then, first element of L_2 (b) is processed in the same way (the lists are iterated over as shown in the figure). Process returns to L_1 and encounters b which is already ranked; therefore, this individual is omitted and c from L_2 is processed further. c is compared with solutions from all existing fronts for corresponding objective until a front without dominators is found. As c is dominated by b (b is above c in both lists), it cannot be placed in Front 1 and hence the second one is created. Once all solutions become ranked, algorithm execution is halted. At the very end of the process, the final fronts are formed by merging corresponding boxes.

1.4 Parallel computing

Parallelism is one of the most actively developed fields of computer science. Parallel computing is a type of computation where calculations are carried out utilizing more than one processing unit processing in parallel. Parallel computing is based on a principle that computational problems often can be divided into separate independent smaller tasks that can be executed simultaneously. This way, amount of time required for solving a particular problem can be significantly reduced. The specifics of the problem determines to which extent it can be parallelized - some problems can be easily divided into independent parts while other can be purely sequential and not parallelizable at all.

The principal measure of parallelization efficiency is the speedup S_p, defined to be the ratio of the time to execute the computational workload on a single processor T_1 to the time on p processors T_p. Speedup can be expressed by the following formula:

$$S_p = \frac{T_1}{T_p}. \tag{1.3}$$

Usually $1 \leq S_p \leq p$, but in case of inappropriate implementation T_p can be even larger than T_1 and hence $S_p < 1$. On the other hand, due to computer architecture-related reasons (e.g.

whole working set fits into higher level of CPU cache), the speedup can become super-linear, meaning that $S_p > p$.

A problem can usually be divided into parallel and sequential parts. The relative size of sequential part determines the maximum speedup which cannot be exceeded with any number of processors. This statement is defined by Amdahl's law [3]:

$$S_{max} = \frac{1}{\alpha}, \tag{1.4}$$

there $\alpha \in [0, 1]$ - proportion of sequential part.

Parallelization efficiency sometimes is also measured by efficiency coefficient e_p, which is defined as a ratio of speedup and number of processors:

$$e_p = \frac{S_p}{p}. \tag{1.5}$$

The higher the e_p value, the more efficiently the processors are utilized.

Parallel computing systems can roughly be divided into two groups: shared and distributed memory systems. Shared memory systems have a joint memory pool which can be accessed by all processors. A shared memory system is relatively simple since all processors share a single view of data and the communication between processors does not need to be separately ensured. In the case of distributed memory system, each processor has its own memory unit and communication between them must be organized. Such communication is carried out by passing messages via some communication channels. In parallelization context, a message is a block of data that is sent from one processing unit to another.

MPI (Message Passing Interface) standard was utilized for the algorithm implementation in this research. The standard defines the syntax and semantics of communication routines used for creating portable message-passing programs in C, C++, and Fortran.

2 Proposed algorithm

Various features of NSGA-II and BOS algorithms were employed for creating a new parallel optimization strategy. NSGA-II structure and whole framework is better suitable for parallelization while BOS helps to reduce number of dominance checks. Two different Pareto-ranking strategies were selected to be used as a part of the algorithm: dominance-rank and dominance-depth approaches. As a basis for parallelization, distributed memory parallel programming model was utilized. The proposed strategy is comprehensively described in further chapters.

2.1 Data structures

To begin with, data structures that were used for implementation need to be explained. Custom data entities are introduced to store solutions, their objective values and views of sorted lists. Using one-dimensional arrays (vectors) for data storing is a sound option while using MPI parallelization standard because of its message passing specifics described in chapter 1.4. As optimization problems as well as decision vectors are multi-dimensional, arbitrary rules of vectors interpretation are established. Structures of the arrays are represented in Figure 2.1 where N denotes population size, n - number of optimization variables and k - number of objectives.

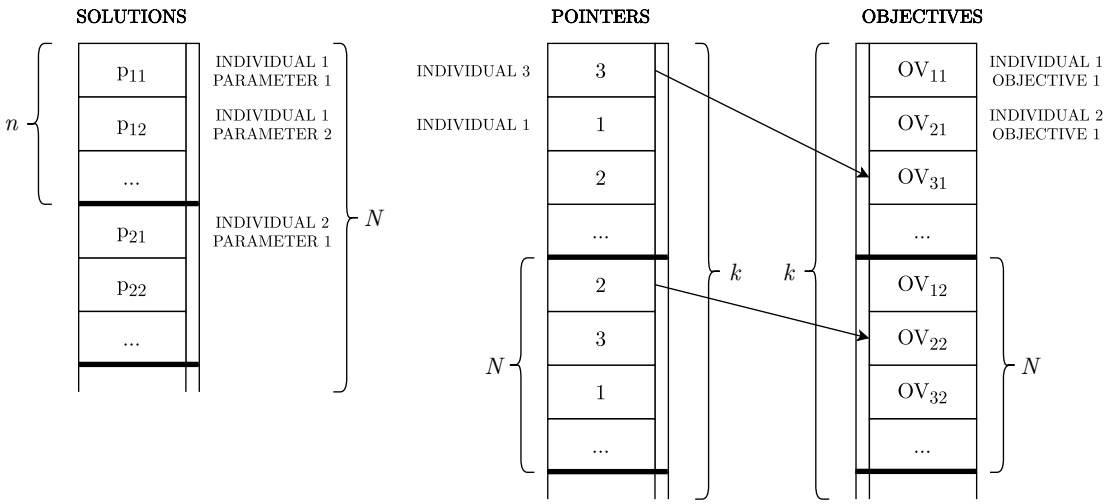

Figure 2.1: Main data structures used in the algorithm.

Solutions array stores values of optimization variables (function parameters). Once a new solution is generated, the vector is simply appended with all n values. When the j-th ($j \in \mathbb{N}^{+}_{\leq n}$) parameter of i-th solution needs to be retrieved, corresponding index is equal to $n \cdot (i-1) + j$.

Objectives array contains values of objective functions. Vector is divided into k sequential logical blocks of size N, each corresponding to one of the objectives. The values in these blocks are stored without any particular order, meaning that once the solution is created and evaluated, the objective value is put just below the last non-empty element in the

corresponding block. For example, if the i-th decision vector is evaluated according to j-th ($j \in \mathbb{N}^{+}_{\leq k}$) objective, the value is inserted at $(N \cdot (j-1) + i)$-th position.

Pointers array represents the order of solutions sorted according to each of the objectives. This vector is also constructed as a set of sequential blocks of size N. Every block contains indices of all solutions arranged in ascending order of respective objective value. This means that, for example, the first element of j-th block represents an index of a solution which has the lowest value of j-th fitness function. When a new solution is generated and its fitness is assessed, an index is inserted into the position identified by utilizing binary search algorithm. The classical binary search is adjusted in such a way that it could operate on pointers array along with the objective values. Arrows in Figure 2.1 illustrate connections between elements of vectors what is the basis of that adjustment.

Moreover, an array storing numbers of dominators is operated. It is an ordinary numeric vector in which each element corresponds to one solution and defines how many dominators that solution has. In the case of dominance-depth ranking, additional data entity for index sets of dominated solutions (denoted by S in chapter 1.3.1) is needed. As size of these sets varies, it is problematic to express them by one-dimensional vector while keeping its length at minimum level. An arbitrary negative integer (-1) was introduced as a separator of the sets (see Figure 2.2).

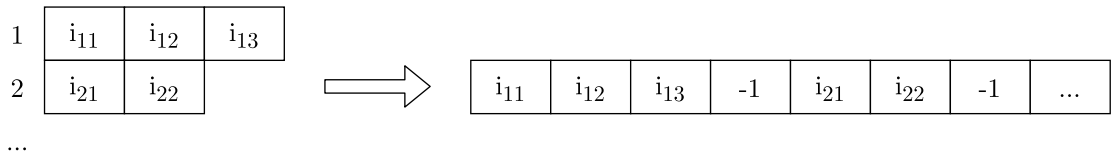

Figure 2.2: Index sets conversion to one-dimensional array.

2.2 Algorithm in detail

The suggested algorithm is designed for distributed memory parallel computing system and can be executed using $p = 2^m, m \in \mathbb{N}^0$ processing units. The latter restriction arises from particular data scattering and gathering approach that is utilized. As stated before, the strategy is mainly based on NSGA-II and BOS algorithms - iterative optimization procedure is constructed using NSGA-II framework while some features of BOS are incorporated as well. Data migration and Pareto ranking techniques ($p = 4$) are shown in Figure 2.3 and described below.

1. At the very first iteration, master processing unit generates and evaluates initial parent population. Three data arrays described before (see chapter 2.1) are created.

2. Dominance relations among decision vectors of parent population are identified. Also, arrays of solutions, objective values and pointers are distributed over all CPUs. Data distribution is carried out using binomial tree scatter approach (hierarchical scattering) [12] which is shown in Appendix A.

3. All processors generate and evaluate a portion of offspring population. The size of this portion depends on total number of processors used (p) and is equal to N/p. Then partial Pareto ranking using custom BOS procedure is performed for particular combinations of available populations (more detail about partial ranking is provided further).

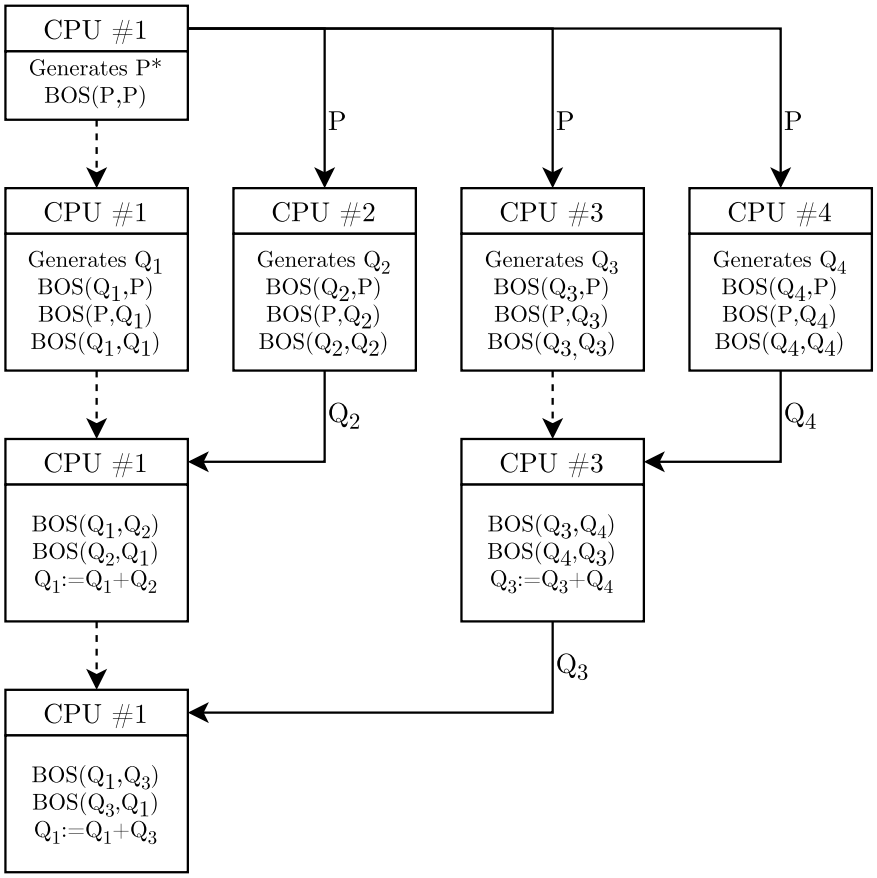

Figure 2.3: Algorithm parallelization approach.

4. Generated and calculated data is collected from the processors using hierarchical data gathering approach. It requires to perform $\log_2 p$ steps in order to gather all available data into the master processor. At every intermediate data gathering step, partial ranking of offspring population chunks is performed. Additionally, those chunks are merged into one set - receiver's offspring population is appended with solutions received from the sender. At the end of the step, respective data arrays are updated with new information while adhering to all the previously described rules.

5. When the gathering is finished and the master process holds whole offspring population together with its ranking data, final procedures of the iteration are carried out. Based on the nature of these procedures two variants of the strategy are distinguished. The following denotation (a and b) is also used when referring to them further in the text.

*Parent population is generated only at the first iteration of the algorithm.

(a) In the case of measuring the fitness by number of existing dominators (dominance-rank approach), N solutions with the least number of dominators are selected from both parent and offspring populations as parent individuals for the next iteration (as the best individuals in case of the last iteration). When there are more individuals having the same fitness indicator than the number of vacancies left in the prospective parent population, solutions are chosen based on the crowding distance measure.

(b) When ranking is carried out using dominance-depth approach, non-dominated sorting procedure on a combined parent-offspring population needs to be performed additionally (for more detail, see 1.3.1). Then, N solutions from the lowest non-dominated fronts are selected. Same as in variant (a), ties are solved by crowding distance comparisons.

6. At the end of iteration, the main data arrays are prepared - selected decision vectors together with their objective values are picked from old arrays and put into new ones. At the same time, pointers vector is formed. Then process returns to the second step unless a pre-defined number of iterations to be run has already been reached.

As it is shown in Figure 2.3, ranking is performed for particular combinations of available populations. Populations are not merged beforehand to avoid performing the same checks. For example, $Q_1 := Q_1 + Q_2$ merging operation is done after ranking these pieces separately because otherwise Q_1 and Q_2 would be compared with themselves what is already done in a previous step. Notation BOS(A, B) denotes a procedure during which each individual from population A gets assigned a number of dominators existing in population B. In the case of strategy (b), a set of dominated individuals needs to be determined as well. When it is identified that, e.g., a solution i from population A is dominated by j from B, a counter n_i is incremented and a set S_j is appended with index i simultaneously. Therefore, in order to determine the set of dominated solutions S for solution i, BOS(B, A) must be executed as well. An example of the latter case is given in Figure 2.4, where a single solution from A is assessed against all solutions from B. It is dominated by one and dominates two solutions from B and the data entities depicted in the graph are preserved and used in the final procedures.

Partial Pareto ranking procedure is a key part of the algorithm that is executed in intermediate data gathering steps. This approach, which is introduced in [8], helps to utilize available processing units in an efficient way. As stated previously, each processor generates its own part of offspring population and it does not have any information about solutions generated or objective values calculated in other CPUs. Seeking to distribute the computational work across the processors as evenly as possible, ranking of incomplete populations must be carried out. This requires an adjustment to usual BOS ranking process where a whole population is ranked at once. At first, BOS algorithm was customized by eliminating non-dominated sorting part. As suggested strategies either does not include non-dominated sorting at all or executes it at the end of iteration, this operation is unnecessary.

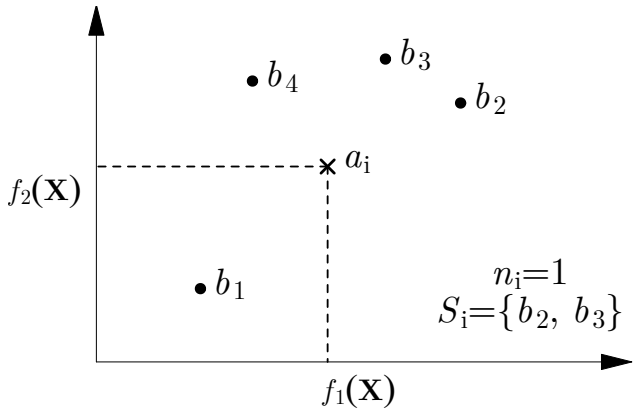

Figure 2.4: Partial ranking example of strategy (b).

In the first step of every iteration, dominance relations within parent population are calculated. Similar calculations are also executed when a generated portion of the offspring population is assessed against itself. Such ranking is performed similarly as described in chapter 1.3.2, using custom BOS described above. Another adjustment is needed when two different populations are evaluated. As they cannot be merged due to the previously explained reason, a plain BOS ranking technique becomes insufficient. For that reason, each individual from population A (referring to the example in Figure 2.4) is processed as follows:

1. Logical positions for solution a_i are found in the sorted lists using binary search method.
2. An objective with the least number of candidate dominators is identified (i.e, selection of the objective is determined by the uppermost logical position found).
3. Solution is checked with the candidate dominators from the corresponding list of selected objective.

This procedure is illustrated in Figure 2.5. It is seen that a_i would be placed at third and second positions of the respective lists. Therefore, f_2 should be chosen where only one dominance check is needed.

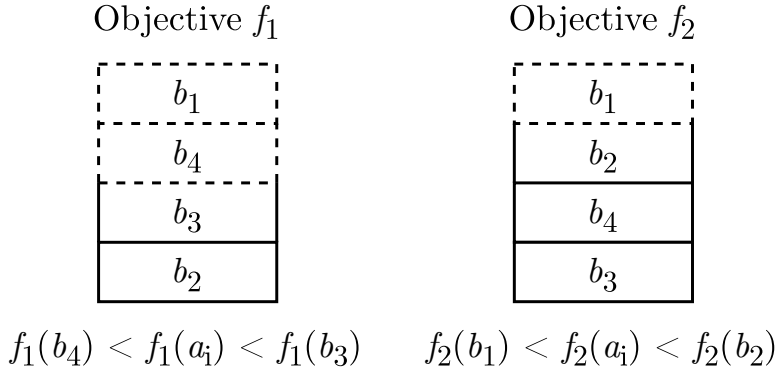

Figure 2.5: Procedure of determining an objective that requires the least number of domination checks.

3 Results

Proposed optimization strategies were implemented in C++ programming language using distributed memory parallel computing model (MPI standard). Separate programs for each strategy were developed and then tested utilizing VU MIF high-performance computing resources. Performance of both strategies was evaluated from parallelization efficiency and general quality perspectives.

3.1 Key findings

The strategies were evaluated by solving a simulated two-dimensional optimization problem on 1, 2, 4, 8 and 16 processors. As the principle measure of parallelization performance is the speedup, program execution time was mainly investigated. In order to correctly assess the efficiency of Pareto ranking parallelization, the execution of each algorithm was divided into two parts: (1) objective values calculation and (2) the rest of procedures. The whole second part is considered Pareto ranking as the final output of the programs is a set of Pareto optimal solutions. Assessing total program execution time is not very meaningful because time which is needed for objective values calculation is problem-dependent and can distort the results. Therefore, the speedups of both parts were measured separately.

First of all, relation between the execution time and number of processors was measured for objective values calculation part. A population of 1600 individuals was used and 500 iterations were performed (the same runs were investigated for ranking part analysis as well). The same results were obtained using both strategies an they are depicted in Figure 3.1. It is observed that objectives calculation part is well parallelized and linear speedup is achieved. This outcome confirmed an initial conjecture for such speedup which was implied by the fact that equal portions of offspring population are evaluated by each of the processors.

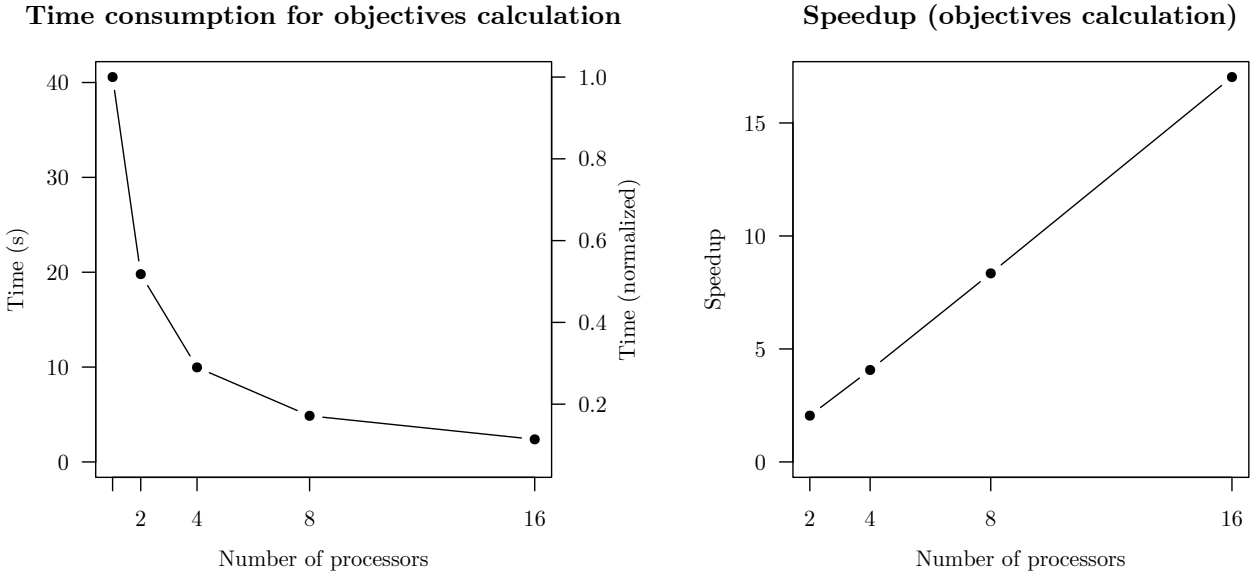

Figure 3.1: Time consumption of objectives calculation part (left) and speedup (right) against number of processors.

Furthermore, parallelization of Pareto ranking part was assessed. Figure 3.2 shows that different ranking strategies perform completely differently in terms of the time needed for ranking. It is seen that total ranking duration in the case of strategy (b) is several times longer than in the case of (a) regardless of the number of processors. When (a) is run on 8 processors, it is capable of reducing the required amount of time by more than half. The another strategy performs worse and under the same circumstances can shorten the time by less than 20%. Such poor performance of strategy (b) was investigated further and it was found out that non-dominated sorting procedure which is performed only by master processor takes almost half of total execution time. In order to achieve better results, parallelization of this particular part should be considered as this is the main drawback of (b). It is also notable that doubling number of processors from 8 to 16 does not result in a considerable improvement in either case.

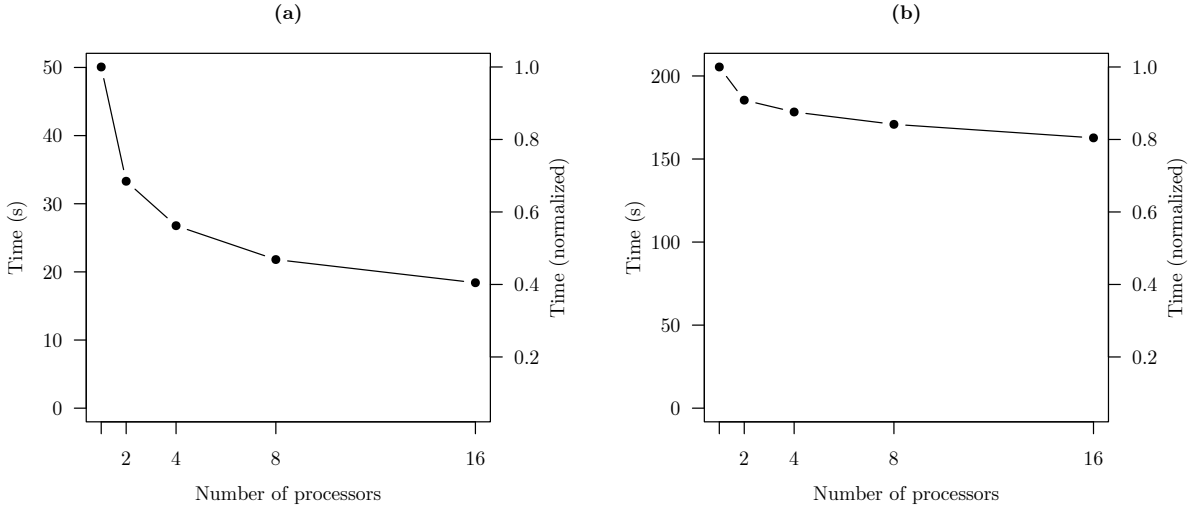

Figure 3.2: Time consumption of Pareto ranking part against number of processors. Dominance-rank (left) and dominance-depth (right) approaches.

The above insights are also illustrated in Figure 3.3, where speedups of both strategies are plotted together. The speedup of strategy (a) is much higher than of (b) with any number of processors.

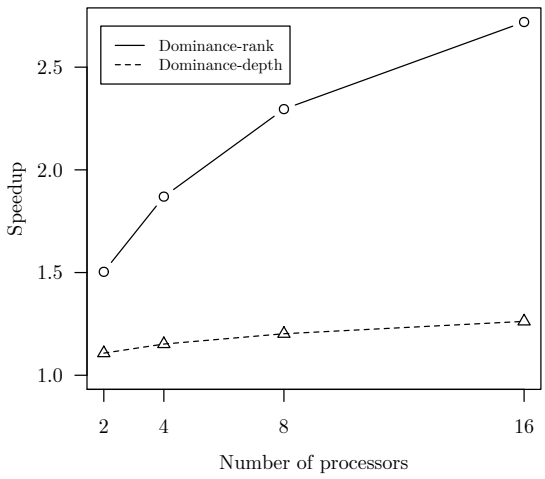

Figure 3.3: Comparison of strategies by speedup.

Additionally, master processor workload dependence on total number of utilized processors was evaluated. The workload was measured by the number of objective values comparisons (also defined as elementary operations) performed during program execution. As both strategies are equivalent with respect to quantity of such comparisons, the results of only strategy (b) are provided in Figure 3.4. It is observed that the master processor can get rid of up to half of the elementary operations when utilizing 16 processors. Curve quickly becomes flat and the difference between numbers when using 8 and 16 processors is not very significant. This can be explained by reference to the algorithm's design - master processor carries out either full or partial ranking of every individual existing in the offspring population while only parts of this population get through all other processors. Larger number of processors causes higher relative difference between the quantities of individuals processed by master and the majority of other processing units.

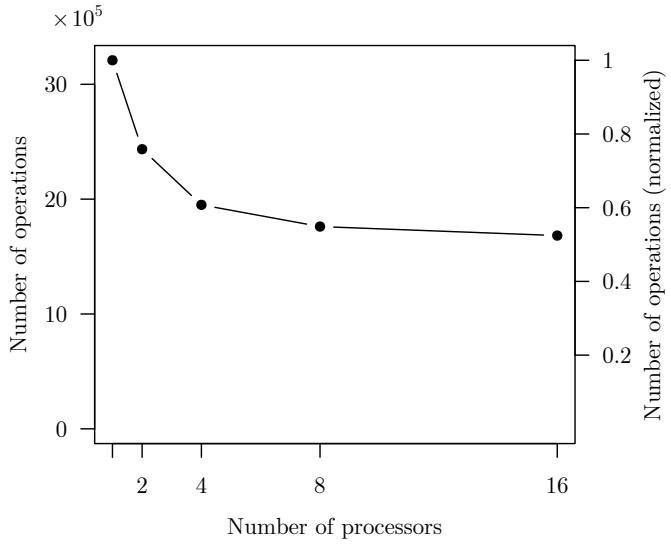

Figure 3.4: Master processor workload dependence on number of processors.

As proposed algorithm employ solutions sorting with the purpose of reducing the quantity of dominance checks, and hence of elementary operations, it was compared with original NSGA-II (which was implemented as described in [4]) with respect to these quantities. Due to the previously explained reason, only the outcome of strategy (b) is compared with that of original NSGA-II. The latter algorithm is sequential, so several different runs of both programs were run using one processor, altering only the population size and number of iterations. The purpose of this check is to assess the efficiency of solutions sorting, so the number of processing units is irrelevant here. Results provided in Table 3.1 indicated that regardless of the altered execution parameters, the proposed method performs roughly 8.5% less elementary operations.

Table 3.1: Number of elementary operations.

N	Iterations	Number of operations		Difference
		NSGA-II	Proposed method	
64	100	2477492	2258584	8.84%
64	200	4989299	4550870	8.79%
128	100	9896971	9073726	8.32%
128	200	19924004	18324889	8.03%

3.2 Hypervolume comparison

The above results showed that proposed strategies are very different in terms of the speedup. Therefore, it is crucial to find out if they also perform differently with regard to quality of the solutions. Hypervolume measure was used as an indicator of the quality. Having 64 individuals in the initial population, one hundred independent test runs of 100 iterations were completed for each strategy. The hypervolume value of first Pareto front was calculated for each run and mean hypervolume values of each strategy were compared using Student's t-test for two independent samples.

First of all, test assumptions (data normality and variance homogeneity) were checked. p-values obtained using Shapiro–Wilk test are higher than the standard significance level 0.05 for both samples (see Appendix C). This result implies that the null hypothesis (that the data follows normal distribution) cannot be rejected. Taking into account data distribution and quantile-quantile plots, data from both samples are considered normally distributed. Homogeneity of variances was tested by Levene's test - $p = 0.359$ also means that null hypothesis of equal variances cannot be rejected.

After ensuring that t-test assumptions are satisfied, the test was performed. Null hypothesis that mean values of two samples are equal was not rejected because $p = 0.878$ is higher than significance level. Therefore, a conclusion was reached that means of the samples are not significantly different. It indicates that the quality of Pareto fronts which are found using both methods does not significantly differ and a faster strategy (a) can be used anytime when non-dominated sorting is not particularly required.

3.3 Real-world application

The proposed algorithm was applied to solve a two-objective facility location problem. A case of oligopolistic market with only two competing companies is investigated. It is assumed that one of the existing companies plans to establish a few new service centers and tries to determine the best locations for these prospective facilities. The fitness of candidate location is measured by the quantities of customers attracted away from competitor (first objective, has to be maximized) and attracted away from itself (second objective, has to be minimized). The particular problem which was solved is summarized by the following bullet list.

- Customer's choice of service facility is based only on haversine* distance measured between one's residence place and the facility (the closest facility is chosen).
- New facilities can be placed in top 50 largest settlements (population-wise) and only one service center can be located in a single city.
- Initially, company has 2 service centers while its competitor has 3. The company establishes 3 new facilities.

Data of 10000 settlements in Lithuania was used for simulating this problem. Dataset contains geographic cordinates of the settlements and number of inhabitants living in each of them. A few exemplary rows from the dataset are given in Table 3.2.

Table 3.2: Fragment of the dataset.

No.	Latitude	Longitude	Population
1.	54.683	25.317	543071
2.	54.900	23.900	336817
3.	55.717	21.117	177823

Initial market situation is depicted in Figure 3.5. This problem is solved from the perspective of the company which facilities are denoted by black dots and its controlled market share is colored blue. The rest of the settlements are controlled by its competitor.

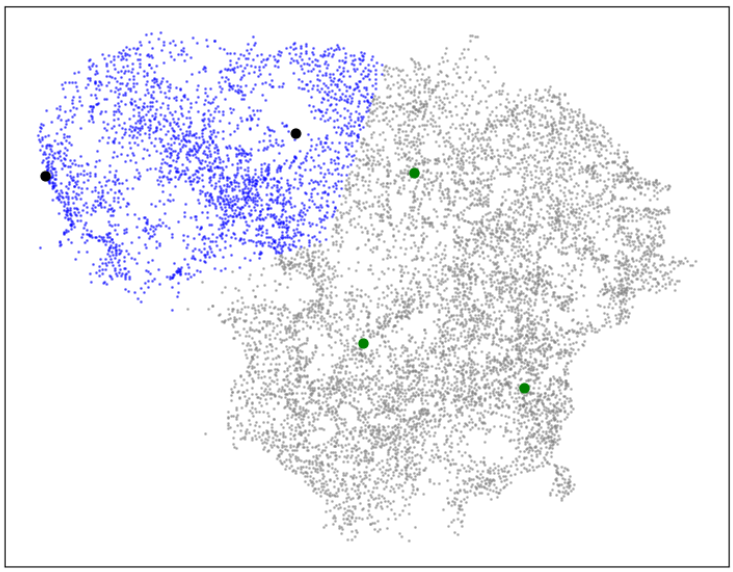

Figure 3.5: Market situation without new facilities.

There were 7 unique Pareto optimal solutions found and one of them is represented in Figure 3.6. Eventual market share taken over by new facilities is colored yellow and the red dots designate locations suggested by the algorithm. It is observed that new facilities

*Haversine distance defines the great-circle distance between two points on a sphere given their longitudes and latitudes.

have no impact on current market share which is covered by the company what means that second objective was minimized. New facilities kind of isolate competitor's service units and take over a large share of its controlled market (first objective was maximized).

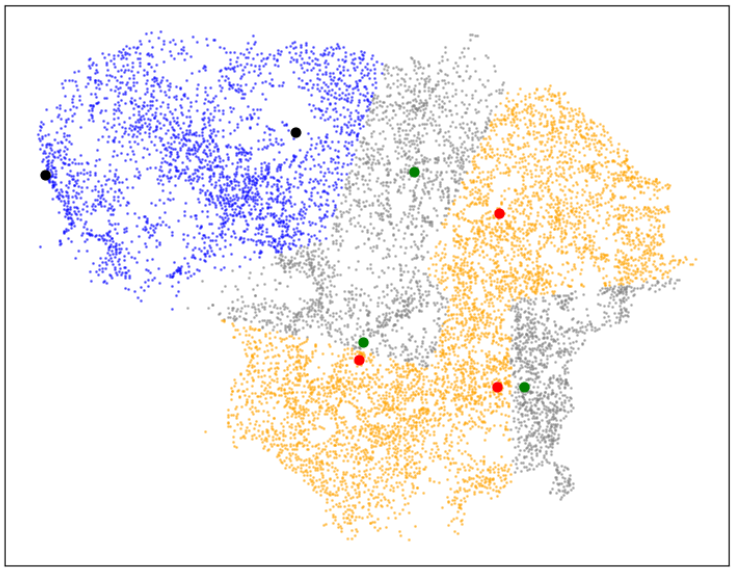

Figure 3.6: Prospective market situation.

Conclusions

After performing the planned tasks, the following conclusions were obtained:

1. A novel parallel optimization algorithm based on the features of NSGA-II and BOS algorithms was designed and implemented. Two Pareto ranking approaches - dominance-rank and dominance-depth - were incorporated and hence separate instances of the algorithm were distinguished.

2. Objective values calculation part was evaluated separately from the rest of the procedures. Analysis showed that a linear speedup of objective values calculation is achieved. This implies that the suggested algorithm can be efficiently utilized when such calculations takes a major part of algorithm execution time.

3. The proposed Pareto ranking strategies were compared to each other. A strategy that is based on dominance-rank approach is adequately parallelizable and a speedup of 2.7 on 16 processing units was achieved. Considerably different results were observed while testing another ranking strategy - ranking time using the same 16 processing units was shortened by only 20% comparing to a sequential run. Such poor speedup is caused by sequential non-dominated sorting procedure which takes a significant part of the ranking time.

4. Hypervolume metric was used to compare the quality of non-dominated fronts obtained using each strategy. Student's t-test showed that mean hypervolume values does not significantly differ across strategies. It implies that a quicker strategy can be used whenever the non-dominated sorting is not required.

5. The benefit of solutions sorting was evaluated by comparing the proposed algorithm to an original NSGA-II algorithm. Results indicated that regardless of the population size or number of iterations, proposed method carries out roughly 8.5% less elementary operations.

6. Implemented algorithms were validated by solving simulated facility location problem. Results clearly indicate that algorithms were correctly implemented.

References

[1] I. Aizpurua-Udabe. "Parallel optimization algorithms for High Performance Computing. Application to thermal systems". PhD thesis. Universitat Politècnica de Catalunya, 2017.

[2] E. Alba and J. Troya. "A Survey of Parallel Distributed Genetic Algorithms". In: *Complex.* 4.4 (1999-03), pp. 31–52. ISSN: 1076-2787.

[3] G. M. Amdahl. "Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities, Reprinted from the AFIPS Conference Proceedings, Vol. 30 (Atlantic City, N.J., Apr. 18–20), AFIPS Press, Reston, Va., 1967, pp. 483–485, when Dr. Amdahl was at International Business Machines Corporation, Sunnyvale, California". In: *IEEE Solid-State Circuits Society Newsletter* 12.3 (2007), pp. 19–20. DOI: 10.1109/N-SSC.2007.4785615.

[4] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. "A Fast Elitist Non-dominated Sorting Genetic Algorithm for Multi-objective Optimization: NSGA-II". In: *Parallel Problem Solving from Nature PPSN VI*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 849–858. ISBN: 978-3-540-45356-7. DOI: 10.1007/3-540-45356-3_83.

[5] R. Haupt and S. Haupt. *Practical Genetic Algorithms*. 3-5 p. Hoboken, New Jersey, USA: John Wiley & Sons, Inc., 2004.

[6] L. Jourdan, A. Liefooghe, and E. Talbi. "A unified model for evolutionary multi-objective optimization and its implementation in a general purpose software framework". In: *2009 IEEE Symposium on Computational Intelligence in Multi-Criteria Decision-Making(MCDM)*. 2009, pp. 88–95. DOI: 10.1109/MCDM.2009.4938833.

[7] A. Lančinskas. "Atsitiktinės paieškos globaliojo optimizavimo algoritmų lygiagretinimas". Disert. Vilnius university, 2013.

[8] A. Lančinskas and J. Žilinskas. "Solution of Multi-Objective Competitive Facility Location Problems Using Parallel NSGA-II on Large Scale Computing Systems". In: *Proceedings of the 11th International Conference on Applied Parallel and Scientific Computing*. PARA'12. Helsinki, Finland: Springer-Verlag, 2012, pp. 422–433. ISBN: 9783642368028. DOI: 10.1007/978-3-642-36803-5_31.

[9] A. Lipowski and D. Lipowska. "Roulette-wheel selection via stochastic acceptance". In: *Physica A: Statistical Mechanics and its Applications* 391.6 (2012), pp. 2193–2196. ISSN: 0378-4371. DOI: https://doi.org/10.1016/j.physa.2011.12.004.

[10] C. ReVelle and H. Eiselt. "Location analysis: A synthesis and survey". In: *European Journal of Operational Research* 165.1 (2005), pp. 1–19. ISSN: 0377-2217. DOI: 10.1016/j.ejor.2003.11.032.

[11] P. Roy, M. Islam, and K. Deb. “Best Order Sort: A New Algorithm to Non-Dominated Sorting for Evolutionary Multi-Objective Optimization”. In: GECCO ’16 Companion. Denver, Colorado, USA: Association for Computing Machinery, 2016, pp. 1113–1120. ISBN: 9781450343237. DOI: 10.1145/2908961.2931684.

[12] J. Träff. “Hierarchical gather/scatter algorithms with graceful degradation”. In: *18th International Parallel and Distributed Processing Symposium, 2004. Proceedings.* 2004, pp. 80–. DOI: 10.1109/IPDPS.2004.1303019.

[13] D. Van Veldhuizen. “Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations”. PhD thesis. Air University (USAF), 1999.

A Data broadcasting

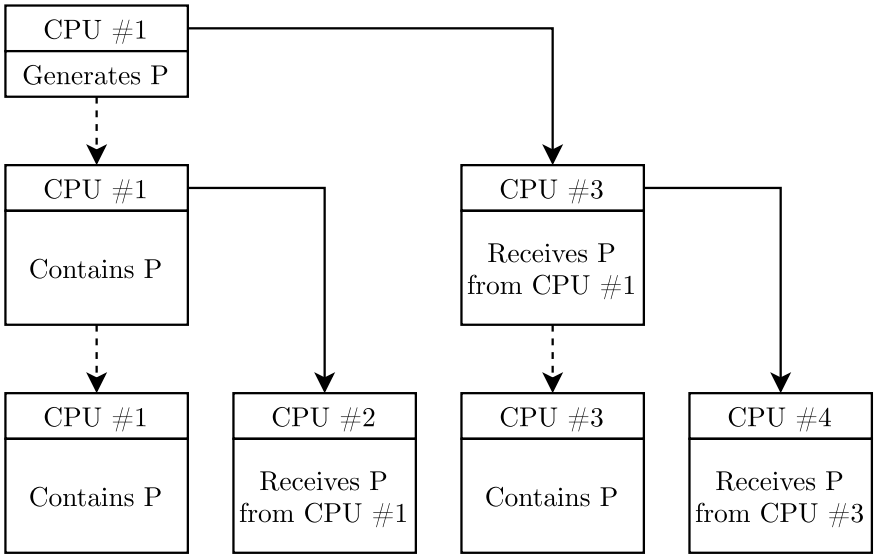

Figure A.1: Communication scheme of a binomial tree scatter with 4 processors.

B Roulette wheel selection

Roulette wheel selection, also known as fitness proportionate selection, is an operator used in genetic algorithms for selecting potentially optimal solutions for mating. This operator can be applied whenever fitness of solutions is expressed by quantitative measure. This method assumes that the probability of selection is proportional to the fitness of an individual. Considering N individuals, each characterized by its fitness $w_i > 0$ $(i = 1, 2, ..., N)$, the selection probability of i-th individual is expressed as [9]

$$p_i = \frac{w_i}{\sum_{i=1}^{N} w_i}. \tag{B.1}$$

Figure B.1 depicts a simple example of roulette wheel selection. Fitness values follow the relation $w_1 < w_2 < w_3$ and it is proportionally reflected on the roulette wheel. The weakest individual has the smallest share of the wheel and hence the lowest chance to become an outcome of roulette spinning.

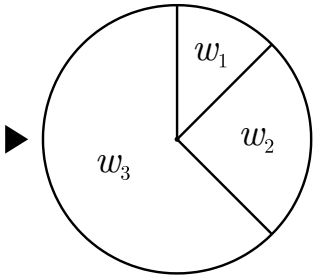

Figure B.1: An illustration of roulette wheel.

In the case of evaluating the solution fitness by quantity of dominators or by rank of non-dominated front, simple inverse values of these indicators can be used (appropriate way of dealing with 0 values needs to be introduced). Such conversion is necessary because lower values of these measures indicate better fitness.

C Comparison of strategies using t-test

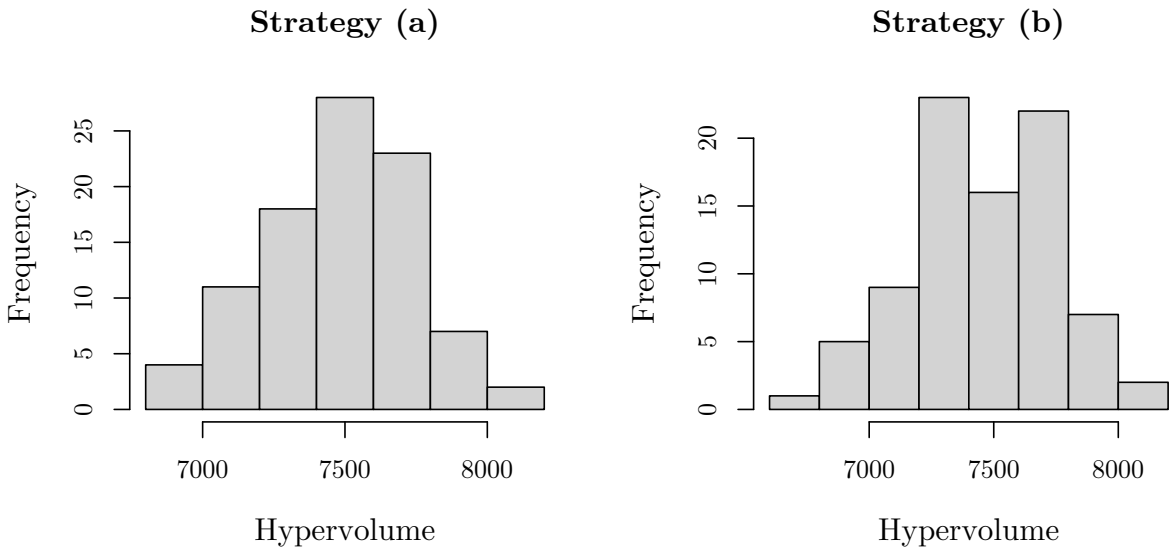

Figure C.1: Distribution of hypervolume values. Strategy (a) (left) and Strategy (b) (right).

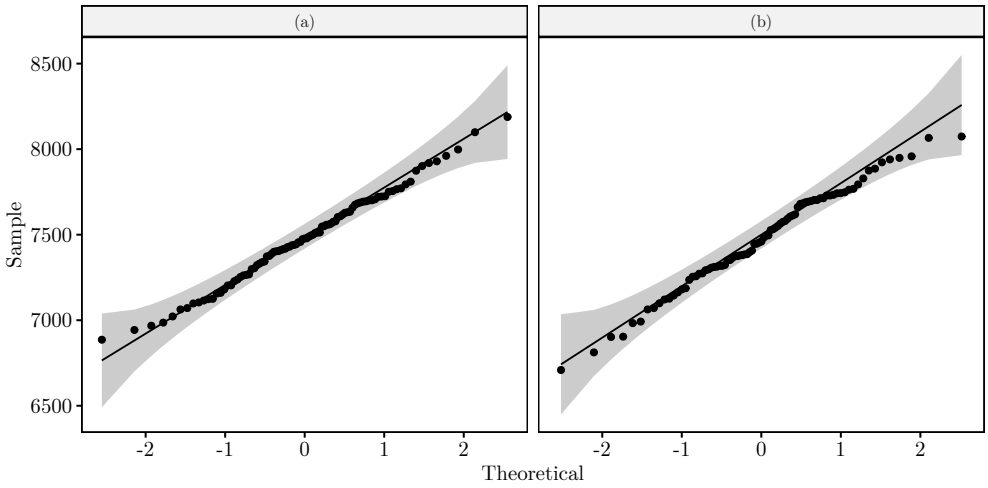

Figure C.2: Q-Q plot for each sample.

Table C.1: Data normality check using Shapiro-Wilk test.

Strategy	t-value	p
(a)	0.993	0.934
(b)	0.988	0.616

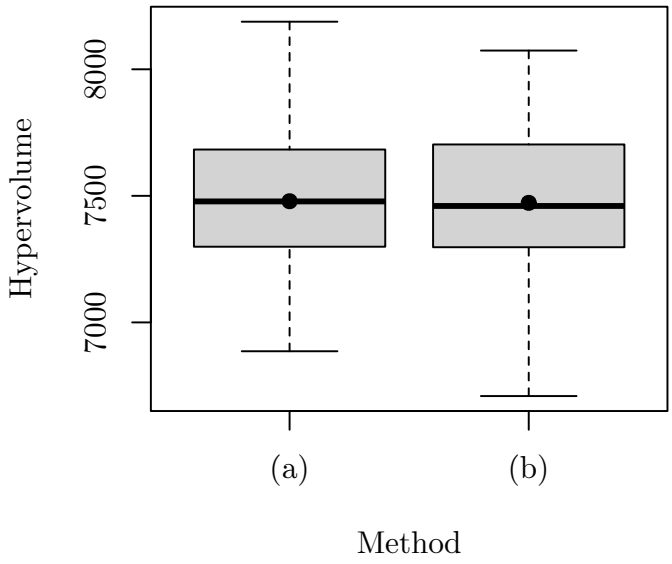

Figure C.3: Box plot for each sample.