Multi-objective single agent stochastic search in non-dominated sorting genetic algorithm*

Algirdas Lančinskas[a], **Pilar Martinez Ortigosa**[b], **Julius Žilinskas**[a]

[a]Vilnius University Institute of Mathematics and Informatics
Akademijos str. 4, LT-08663 Vilnius, Lithuania
algirdas.lancinskas@mii.vu.lt; julius.zilinskas@mii.vu.lt

[b]Universidad de Almería, ceiA3
Ctra. Sacramento s/n, La Cañada de San Urbano 04120, Almería, Spain
ortigosa@ual.es

Received: 10 September 2012 / **Revised:** 13 May 2013 / **Published online:** 18 June 2013

Abstract. A hybrid multi-objective optimization algorithm based on genetic algorithm and stochastic local search is developed and evaluated. The single agent stochastic search local optimization algorithm has been modified in order to be suitable for multi-objective optimization where the local optimization is performed towards non-dominated points. The presented algorithm has been experimentally investigated by solving a set of well known test problems, and evaluated according to several metrics for measuring the performance of algorithms for multi-objective optimization. Results of the experimental investigation are presented and discussed.

Keywords: multi-objective optimization, Pareto set, non-dominated sorting genetic algorithm, single agent stochastic search.

1 Introduction

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, according to a single criterion. Without reducing the generality further 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

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

*This research was funded by a grant (No. MIP-063/2012) from the Research Council of Lithuania. This work has been funded by grants from the Spanish Ministry of Science and Innovation (TIN2008-01117 and TIN2011-23283), Junta de Andalucia (P10-TIC-6002), in part financed by the European Regional Development Fund (ERDF).

and a decision vector $\mathbf{x}^*$ such as

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

where $f(\mathbf{x})$ is an *objective function* which is subject to minimization while decision 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.

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 (MOP).

Mathematically, a MOP with d variables and m objectives in the objective vector

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

is to find an objective vector

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

Since there exist conflicts among objectives, it is natural that a single solution best according to all objectives does not exist. 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 decision vectors 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 [1].

Suppose two decision vectors $\mathbf{x}$ and $\mathbf{y}$ from a set D. It is said that decision vector $\mathbf{x}$ *dominates* decision vector $\mathbf{y}$ if

- decision vector $\mathbf{x}$ is not worse than $\mathbf{y}$ by all objectives and
- decision vector $\mathbf{x}$ is strictly better than $\mathbf{y}$ by at least one objective.

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

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

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

The set of all decision vectors that are non-dominated according to some set $D' \subset D$ is called *Pareto set*, and the set of decision vectors that are non-dominated according to the whole search space D is called *Pareto-Optimal Set*. The corresponding set of objective vectors are called *Pareto Front* and *Pareto-Optimal Front*, respectively.

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

2 Evolutionary multi-objective optimization

One well known class of optimization algorithms are Evolutionary Algorithms (EAs). They are well suited to multi-objective optimization problems as they are fundamentally based on biological processes which are inherently multi-objective. Multi-Objective EAs (MOEAs) can yield a whole set of potential solutions – which are all optimal in some sense – and give the option to assess the trade-offs between different solutions. Furthermore EAs require little knowledge about the problem being solved and they are robust, easy to implement and inherently parallel. The only requirement to solve a certain optimization problem is to be able to evaluate the objective functions for a given set of input parameters.

After the first works on multi-objective evolutionary optimization [2], several different algorithms have been proposed and successfully applied to various problems.

Although there are many techniques to design MOEAs, which have different characteristics, the advantages of a particular technique can be improved by hybridizing two or more different techniques, for example, hybridizing different search methods, search and updating methods or different search methods in different search phases. A special case of hybrid MOEA is memetic MOEA [3] – incorporation of local optimization techniques into an evolutionary algorithm.

In this paper we will focus on one of the most popular multi-objective evolutionary algorithm Non-dominated Sorting Genetic Algorithm (NSGA) [1], and local multi-objective optimization technique Multi-Objective Single Agent Stochastic Search (MOSASS) [4], derived from algorithm for single-objective optimization Single Agent Stochastic Search (SASS) [5].

The Non-dominated Sorting Genetic Algorithm (NSGA) was proposed in [1], and was one of the first MOEA. Since then NSGA was applied to various problems [6, 7]. The updated version of NSGA – NSGA-II has been proposed in [8].

The algorithm begins with an initial parent population, consisting of decision vectors randomly generated over the search space. Then the algorithm continues with the following generational process (see Fig. 1):

1. A new child population is created by applying genetic operators (selection, crossover and mutation) to the elements of the parent population. Usually a child population has the same size as the parent population.
2. Parent and child populations are combined into one population.
3. The population is sorted according to the number of dominators.
4. Obtained population is reduced to the size of parent population by removing the most dominated elements. If two or more decision vectors are equally dominated then crowding distances estimator [8] is used to choose the most promising decision vector.
5. The reduced population is used as a parent population in the next generation.

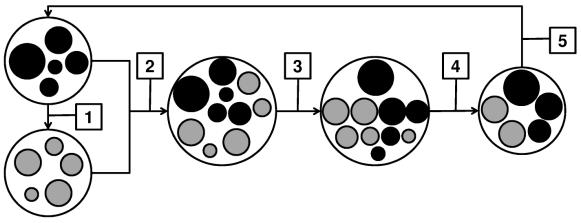

Fig. 1. Scheme of NSGA-II algorithm.

3 Hybrid multi-objective optimization

3.1 Modified single agent stochastic search

Single Agent Stochastic Search (SASS) is a random local optimization technique, based on a single agent stochastic search strategy [5]. SASS algorithm begins with an initial solution $\mathbf{x}$ which is assumed to be the best solution found so far, and a new candidate solution $\mathbf{x}'$ is generated by the expression

$$\mathbf{x}' = \mathbf{x} + \xi, \tag{6}$$

where $\xi = (\xi_1, \ldots, \xi_d)$ is a vector of random numbers

$$\xi_i = \mathcal{N}\big(b_i, \sigma(x_i^+ - x_i^-)\big), \tag{7}$$

generated following Gaussian distribution with the mean b_i, $i = 1, \ldots, d$, and the standard deviation σ. Here x_i^- and x_i^+ correspond to the lower and upper bounds of the value of variable x_i. In the case of

$$f(\mathbf{x}') < f(\mathbf{x}), \tag{8}$$

the solution $\mathbf{x}'$ is accepted as $\mathbf{x}$ – the best solution found so far – and the algorithm continues to the next iteration. Otherwise the opposite solution

$$\mathbf{x}'' = \mathbf{x} - \xi \tag{9}$$

is investigated and accepted as $\mathbf{x}$ if

$$f(\mathbf{x}'') < f(\mathbf{x}). \tag{10}$$

If either (8) or (10) is satisfied then the mean values b_i are recalculated by

$$b_i = \begin{cases} 0.2b_i + 0.4\xi_i & \text{if } f(\mathbf{x}') < f(\mathbf{x}), \\ b_i - 0.4\xi_i & \text{if } f(\mathbf{x}'') < f(\mathbf{x}), \end{cases} \quad i = 1, \ldots, d, \tag{11}$$

and the iteration is supposed to be successful. Otherwise, the iteration is supposed to be failed and the mean values are recalculated by

$$b_i = 0.5b_i, \quad i = 1, \ldots, d. \tag{12}$$

If the number of repetitive successful iterations ($scnt$) reaches the given value $Scnt$, then the standard deviation value is expanded by

$$\sigma = \sigma e, \tag{13}$$

Analogically, if the number of repetitive failed iterations ($fcnt$) reaches the given value $Fcnt$, then the standard deviation is contracted by

$$\sigma = \sigma c. \tag{14}$$

If standard deviation σ falls below the given minimum value $\sigma_{\min}$, it is increased to the given value $\sigma_{\max}$.

The values $e > 1$ and $c \in (0, 1)$ – expansion and contraction coefficients, respectively, are given as input parameters as well as the values $Scnt$ and $Fcnt$, the upper and lower ranges for standard deviation ($\sigma_{\min}$ and $\sigma_{\max}$), and the number of iterations to be performed (N_{iters}). Recommended values for the parameters are $e = 2$, $c = 0.5$, $Scnt = 5$, $Fcnt = 3$, $\sigma_{\min} = 10^{-5}$, $\sigma_{\max} = 1.0$ [5].

SASS algorithm has been successfully used in evolutionary algorithms for single-criterion optimization problems [9–11]. In order to apply it for multi-objective optimization problems a new version called Multi-Objective Single Agent Stochastic Search (MOSASS) has been developed. Primary version of MOSASS algorithm and its evaluation on several test functions has been presented in our previous work [4]. Further we will present a modified version of MOSASS algorithm where modifications are mainly based on involving probability in generation of neighbor solutions.

MOSASS algorithm begins with an initial solution $\mathbf{x}$, empty archive A and constant N_A, which defines how many solutions can be stored in the archive. Previously defined SASS parameters $Scnt$, $Fcnt$, e, c, $\sigma_{\min}$, $\sigma_{\max}$ and N_{iters} are given as initial parameters for MOSASS as well.

Since a new solution $\mathbf{x}'$ is generated, following equation (6), and objective vector $\mathbf{F}(\mathbf{x}')$ is calculated, 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 by $\mathbf{x}'$ 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 goes to the next iteration. If solution $\mathbf{x}'$ is dominated by any member of $A \cup \{\mathbf{x}\}$, then $\mathbf{x}'$ is rejected and opposite solution $\mathbf{x}''$ is investigated in the same way as $\mathbf{x}'$. If archive size exceeds the limit N_A then a half of elements are removed by utilizing crowding distance operator [8].

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 (6). Since values of all variables of $\mathbf{x}$ are changed without any probabilistic choice, it is a large probability that obtained neighbor solution $\mathbf{x}'$ or $\mathbf{x}''$ differs from its precursor $\mathbf{x}$ by all variables. However, sometimes it is necessary to make only a slight

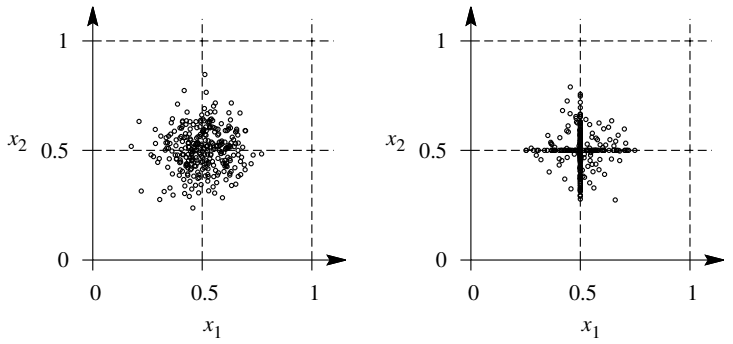

Fig. 2. Illustration of difference in generation of neighbor decision vector in MOSASS and MOSASS/P.

modification of the current solution – to alter only one variable in order to obtain a neighbor 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 slight modification, we previously modified MOSASS by involving the probability in generating a neighbor decision vector, and call the modified algorithm by MOSASS/P. The modification has been done by changing the expression (7) to the following one:

$$\xi_i = \begin{cases} N(b_i, \sigma_i) & \text{if } r \leqslant p, \\ 0 & \text{otherwise,} \end{cases} \tag{15}$$

where r is a random number uniformly generated in $[0, 1]$ and $p \in (0, 1]$ is a probability value, predefined in advance. The larger p value is, the larger probability that the particular coordinate will be changed. If p equals to 1, the algorithm behaves as MOSASS. The main difference in generation of neighbor decision vector in algorithms MOSASS and MOSASS/P is shown in Fig. 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 ($x_i = 0.5$, $p = 0.5$, $\sigma = 0.1$ and $b_i = 0$, $i = 1, 2$).

There is also possible situation that all coordinates will remain unchanged. Then the generation of the new decision vector is repeated until at least one coordinate will be changed.

3.2 Memetic algorithm

In order to improve some performance metrics of original NSGA-II algorithm a memetic algorithm, based on performing the local search towards a set of non-dominated decision vectors, has been developed. The memetic 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 elements of the parent population. The uniform crossover that combines pairs of parent population elements, 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 is 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 elements. In the case of removing two decision vectors that are equally dominated, the crowding distance operator [8] is used to choose the most promising one.

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, we denote the derived memetic algorithm by NSGA/LS and NSGA/LSP, respectively. The number of NSGA-II iterations after which local search is performed, the size of the auxiliary set P_L, and the number of local search iterations are given as input parameters by the user.

4 Experimental investigation

4.1 Description of experiments

The proposed memetic algorithms NSGA/LS and NSGA/LSP have been experimentally investigated by solving a set of multi-objective optimization problems which are listed in Table 1.

Table 1. List of test problems used in experimental investigation.

Problem	d	m	Ref.	Problem	d	m	Ref.
ZDT1	30	2	[12]	UP4	30	2	[14]
ZDT2	30	2	[12]	UP7	30	2	[14]
ZDT3	30	2	[12]	UP8	30	3	[14]
ZDT4	10	2	[12]	UP10	30	3	[14]
ZDT6	30	2	[12]	KNO1	2	2	[15]
DTLZ1	6	2	[13]	OKA1	2	2	[15]
DTLZ2	12	2	[13]	OKA2	3	2	[15]
DTLZ3	6	2	[13]	VLMOP2	2	2	[15]
DTLZ4	12	2	[13]	Kursawe	3	2	[16]
DTLZ7	12	2	[13]	DMM	2	2	[17]
UP1	30	2	[14]	VLMOP3	2	3	[15]
UP2	30	2	[14]	Comet	3	3	[13]
UP3	30	2	[14]	SPH	10	3	[18]

4.2 Performance metrics

The following performance metrics were used in experimental investigation to evaluate the performance of the algorithms:

- *Pareto Front Size* (PS) – number of non-dominated objective vectors in the obtained Pareto front.
- *Coverage of Pareto Optimal Front* (C) – number of decision vectors which are non-dominated regarding to members of Pareto-optimal front. This metric is based on Coverage metric proposed in [12]:

$$C\big(\tilde{P}, P^*\big) = \big|\{\mathbf{x} \in \tilde{P}\colon\ \mathbf{x} \sim \mathbf{x}'\ \forall \mathbf{x}' \in P^*\}\big|, \tag{16}$$

  where P^* denotes the Pareto-optimal front, and $\tilde{P}$ – the obtained approximation. Naturally $C \leqslant |\tilde{P}|$ and larger C value implies better quality of $\tilde{P}$. The value $C(\tilde{P}, P^*) = 0$ means that all decision vectors in $\tilde{P}$ are dominated by at least one decision vector in P^*. The opposite, $C(\tilde{P}, P^*) = |\tilde{P}|$ represents the situation when all decision vectors in $\tilde{P}$ are indifferent to decision vectors in P^*.
- *Hyper-Volume* (HV) – the volume of a region made by the members of obtained Pareto front and the given reference point (*Ref*) [19]. The larger HV value means the better quality of Pareto Front. Two examples of HV are presented in Fig. 3, where Pareto front on the left has more points and better (more similar to uniform) distribution than Pareto front on the right. Therefore the dominated space of the Pareto front on the left is noticeably larger.
- *Inverted Generational Distance* (IGD) – the average distance from the element of the Pareto-optimal front P^* to the nearest element of the approximated Pareto front $\tilde{P}$:

$$IGD\big(\tilde{P}, P^*\big) = \frac{1}{|P^*|} \sum_{\mathbf{x}' \in P^*} \min_{\mathbf{x} \in \tilde{P}} \big\|\mathbf{F}(\mathbf{x}') - \mathbf{F}(\mathbf{x})\big\|. \tag{17}$$

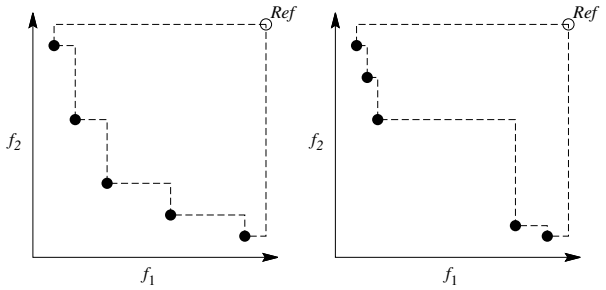

Fig. 3. Illustration of hyper-volume.

Since it is expected to find decision vectors which would be as close as possible to the members of the Pareto-optimal front, lower IGD value implies better quality of the approximation.

- *Pareto Spread* (Δ) shows how well members of Pareto front are spread. The original metric is based on calculating the distance between two consecutive solutions and is dedicated to bi-objective problems. We use an extended version of spread metric which is based on calculating the distance from a point to its nearest neighbor [19]:

$$\Delta = \frac{\sum_{i=1}^{m} d(\mathbf{e}_i, \tilde{P}) + \sum_{\mathbf{x}\in\tilde{P}} |d(\mathbf{x}, \tilde{P}) - \bar{d}(P^*, \tilde{P})|}{\sum_{i=1}^{m} d(\mathbf{e}_i, \tilde{P}) + |P^*|\bar{d}(P^*, \tilde{P})}, \tag{18}$$

where $\tilde{P}$ is an obtained approximation of Pareto front, P^* – Pareto-optimal front, $\mathbf{e}_1, \ldots, \mathbf{e}_m$ are m decision vectors, corresponding to extreme objective vectors in P^* and

$$d(\mathbf{x}, \tilde{P}) = \min_{\mathbf{y}\in\tilde{P}, \mathbf{y}\neq\mathbf{x}} \left\|\mathbf{F}(\mathbf{x}) - \mathbf{F}(\mathbf{y})\right\|^2, \tag{19}$$

$$\bar{d}(P^*, \tilde{P}) = \frac{1}{|P^*|} \sum_{\mathbf{x}\in P^*} d(\mathbf{x}, \tilde{P}). \tag{20}$$

If the members of approximated Pareto front are well distributed and include extreme solutions, then $\Delta = 0$.

4.3 Results

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 2, 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.

Sensitivity of the algorithm to the values of parameters (E_G, E_L) depends on test function being solved. For 5 test functions the loss of HV when choosing the worst combination of parameters was less than 5%, while for 15 test functions – less than 10%, and for 20 test functions – less than 20%. The algorithm was most sensitive to the values of the parameters when solving test functions ZDT4, UP8, OKA2, ZDT6 and UP10 – loss of HV was 28%, 35%, 37%, 56%, 91%, respectively.

The second experiment was dedicated to investigate performance of proposed algorithms NSGA/LS and NSGA/LSP. The performance of the algorithms has been compared with each other and with classical NSGA-II, implemented as given in literature [8]. 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 (see Section 3.2) for local optimization. The probability in generating new decision vector in NSGA/LSP was chosen to be equal to $1/d$ – the same as mutation rate in NSGA-II.

Results of the experimental investigation are presented in Tables 3–7, where values of PS, C, HV, IGD and Δ metrics are presented respectively. 100 independent runs were performed for each test function. Average values and standard deviations are presented in the tables. The best values of performance metric are marked in bold.

Table 2. Numbers of test functions for which particular values (E_G, E_L) was 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

Table 3. Pareto size.

Problem	NSGA-II		NSGA/LS		NSGA/LSP	
	Mean	StDev	Mean	StDev	Mean	StDev
ZDT1	**96.70**	15.30	94.40	2.30	96.10	2.30
ZDT2	91.50	20.60	77.70	17.20	**95.60**	2.44
ZDT3	**94.90**	14.10	86.00	4.60	92.20	6.43
ZDT4	**72.40**	15.60	67.00	8.10	59.60	12.80
ZDT6	16.10	3.49	14.10	2.30	**77.20**	7.93
DTLZ1	62.10	31.10	**87.70**	3.70	84.00	18.90
DTLZ2	92.90	3.90	**93.40**	2.00	92.50	2.34

Table 3. (*Continued.*)

Problem	NSGA-II		NSGA/LS		NSGA/LSP	
	Mean	StDev	Mean	StDev	Mean	StDev
DTLZ3	50.70	23.90	82.70	6.20	**86.50**	14.50
DTLZ4	62.60	30.10	63.40	35.30	**92.50**	3.80
DTLZ7	86.60	5.23	**93.50**	1.90	86.10	4.45
UP1	39.70	6.55	64.20	14.60	**78.60**	6.91
UP2	**92.40**	6.96	92.00	3.70	90.60	3.46
UP3	16.90	5.73	28.50	6.80	**63.60**	13.50
UP4	**77.60**	3.39	73.60	4.20	51.00	6.75
UP7	46.80	4.15	70.80	19.10	**82.30**	7.78
UP8	83.80	7.69	77.40	11.80	**86.20**	5.74
UP10	23.10	21.70	23.10	15.00	**26.40**	21.00
KNO1	89.30	5.68	**95.60**	1.90	81.50	6.67
OKA1	28.20	14.70	48.00	6.90	**63.20**	9.93
OKA2	5.90	8.00	13.20	4.80	**24.00**	6.48
VLMOP2	**86.30**	16.70	85.70	5.30	79.90	3.92
Kursawe	64.30	8.30	**66.10**	3.90	60.80	4.32
DMM	66.60	11.60	**80.50**	7.60	76.00	7.59
Comet	70.10	10.10	**73.10**	10.90	71.90	8.22
VLMOP3	85.30	5.33	**85.90**	6.20	82.70	5.71
SPH	**86.80**	3.55	86.50	4.90	74.80	6.47

Table 4. Coverage of Pareto front.

Problem	NSGA-II		NSGA/LS		NSGA/LSP	
	Mean	StDev	Mean	StDev	Mean	StDev
ZDT1	0.00	0.00	1.80	1.50	**88.10**	3.56
ZDT2	0.00	0.00	0.10	0.40	**88.50**	3.75
ZDT3	0.90	1.10	12.40	4.40	**85.40**	6.85
ZDT4	2.80	2.00	**15.00**	4.10	8.50	12.40
ZDT6	0.00	0.00	0.00	0.00	**21.40**	7.87
DTLZ1	0.00	0.00	38.90	15.80	**64.00**	19.30
DTLZ2	51.00	12.00	25.60	9.50	**70.30**	5.98
DTLZ3	0.00	0.00	34.20	15.10	**61.10**	15.60
DTLZ4	0.10	1.30	32.80	28.50	**84.10**	5.18
DTLZ7	6.60	4.20	67.20	10.10	**78.40**	5.24
UP1	4.00	5.60	0.40	1.60	**40.70**	10.70
UP2	0.00	0.20	0.00	0.10	**14.40**	4.50
UP3	0.00	0.00	0.00	0.00	**1.85**	7.37
UP4	0.00	0.00	**0.80**	0.00	0.00	0.00
UP7	3.90	5.70	2.90	8.10	**37.00**	18.60
UP8	**58.30**	6.20	0.00	0.00	12.50	7.01
UP10	**15.40**	22.00	0.00	0.00	0.90	5.09
KNO1	11.40	10.00	**39.20**	4.60	25.60	9.13
OKA1	0.00	0.10	**0.03**	0.20	0.00	0.00
OKA2	0.00	0.00	0.00	0.00	0.00	0.00

(*continued on next page*)

Table 4. (*Continued.*)

Problem	NSGA-II		NSGA/LS		NSGA/LSP	
	Mean	StDev	Mean	StDev	Mean	StDev
VLMOP2	63.20	6.10	**63.70**	5.90	58.80	5.35
Kursawe	1.00	0.60	1.40	0.60	**1.64**	0.75
DMM	51.50	32.00	**77.20**	8.20	70.80	7.70
Comet	**65.20**	9.40	28.50	7.00	32.30	8.12
VLMOP3	**79.10**	6.40	52.40	4.90	48.20	5.21
SPH	**10.50**	4.90	0.37	0.60	0.00	0.00

Table 5. Hyper-volume.

Problem	NSGA-II		NSGA/LS		NSGA/LSP	
	Mean	StDev	Mean	StDev	Mean	StDev
ZDT1	0.6466	0.0020	0.6558	0.0011	**0.6603**	0.0005
ZDT2	0.3046	0.0120	0.2452	0.0828	**0.3272**	0.0004
ZDT3	0.5061	0.0071	0.5109	0.0014	**0.5137**	0.0020
ZDT4	**0.6103**	0.0220	0.5245	0.0446	0.5372	0.0519
ZDT6	0.0098	0.0023	0.0251	0.0065	**0.3892**	0.0035
DTLZ1	0.4601	0.0180	0.4315	0.0468	**0.4759**	0.0510
DTLZ2	**0.2097**	0.0002	0.2086	0.0004	0.2093	0.0004
DTLZ3	0.1769	0.0170	0.1621	0.0265	**0.2008**	0.0226
DTLZ4	0.1595	0.0770	0.1379	0.0877	**0.2091**	0.0007
DTLZ7	0.4135	0.0130	0.4176	0.0019	**0.4199**	0.0007
UP1	0.5053	0.0400	0.5130	0.0476	**0.5505**	0.0266
UP2	0.5915	0.0070	0.6024	0.0089	**0.6217**	0.0106
UP3	0.3738	0.0260	0.3742	0.0306	**0.4253**	0.0384
UP4	0.2395	0.0035	0.2363	0.0043	**0.2570**	0.0044
UP7	0.3026	0.0700	0.3183	0.0763	**0.3554**	0.0795
UP8	0.5794	0.0140	0.5700	0.0182	**0.6175**	0.0175
UP10	0.0447	0.0420	**0.0465**	0.0308	0.0434	0.0400
KNO1	0.6641	0.0120	**0.6864**	0.0070	0.6716	0.0132
OKA1	0.5320	0.0140	0.5749	0.0115	**0.5766**	0.0117
OKA2	0.0928	0.0180	0.1259	0.0234	**0.1681**	0.0397
VLMOP2	**0.3086**	0.0005	0.3084	0.0006	0.3068	0.0012
Kursawe	0.2908	0.0011	0.2914	0.0013	**0.2918**	0.0014
DMM	0.7708	0.0800	**0.8199**	0.0006	0.8191	0.0011
Comet	0.5116	0.0034	0.5136	0.0033	**0.5164**	0.0034
VLMOP3	**0.8272**	0.0025	0.8253	0.0028	0.8258	0.0024
SPH	**0.5539**	0.0073	0.5462	0.0053	0.5502	0.0061

Table 6. Inverted generational distance.

Problem	NSGA-II		NSGA/LS		NSGA/LSP	
	Mean	StDev	Mean	StDev	Mean	StDev
ZDT1	0.0121	0.0016	0.0074	0.0017	**0.0051**	0.0005
ZDT2	0.0290	0.0150	0.1094	0.1084	**0.0054**	0.0006

Table 6. (*Continued.*)

Problem	NSGA-II		NSGA/LS		NSGA/LSP	
	Mean	StDev	Mean	StDev	Mean	StDev
ZDT3	0.0081	0.0062	**0.0051**	0.0008	0.0053	0.0018
ZDT4	**0.0581**	0.0340	0.1599	0.0623	0.1068	0.0427
ZDT6	0.5541	0.0190	0.4707	0.0265	**0.0119**	0.0026
DTLZ1	0.0243	0.0130	0.0480	0.0331	**0.0190**	0.0518
DTLZ2	**0.0045**	0.0002	0.0050	0.0005	0.0049	0.0005
DTLZ3	0.0501	0.0270	0.0483	0.0275	**0.0145**	0.0296
DTLZ4	0.1123	0.1700	0.1613	0.1956	**0.0059**	0.0011
DTLZ7	0.0166	0.0170	0.0090	0.0095	**0.0047**	0.0007
UP1	0.1280	0.0380	0.1123	0.0336	**0.0904**	0.0229
UP2	0.0659	0.0150	0.0551	0.0175	**0.0469**	0.0166
UP3	0.2180	0.0320	0.2112	0.0302	**0.1877**	0.0391
UP4	0.0913	0.0077	0.0921	0.0074	**0.0654**	0.0089
UP7	0.2326	0.1200	0.2019	0.1280	**0.1708**	0.1228
UP8	0.1500	0.0300	0.1339	0.0267	**0.1140**	0.0351
UP10	0.4928	0.1400	**0.4654**	0.1200	0.4691	0.1184
KNO1	0.0846	0.0140	**0.0624**	0.0064	0.0751	0.0123
OKA1	0.0853	0.0150	**0.0603**	0.0193	0.0604	0.0198
OKA2	0.3668	0.0510	0.3301	0.0536	**0.2377**	0.0932
VLMOP2	**0.0059**	0.0005	0.0066	0.0012	0.0079	0.0021
Kursawe	0.1329	0.0011	**0.1326**	0.0010	0.1332	0.0015
DMM	0.0409	0.0520	**0.0089**	0.0014	0.0100	0.0020
Comet	0.0315	0.0034	0.0295	0.0025	**0.0280**	0.0022
VLMOP3	0.0209	0.0380	**0.0118**	0.0016	0.0172	0.0071
SPH	0.0717	0.0044	**0.0709**	0.0038	0.0712	0.0041

Table 7. Spread.

Problem	NSGA-II		NSGA/LS		NSGA/LSP	
	Mean	StDev	Mean	StDev	Mean	StDev
ZDT1	**0.00009**	0.00004	0.00015	0.00005	0.00011	0.00004
ZDT2	0.00063	0.00081	0.00673	0.00773	**0.00011**	0.00004
ZDT3	0.00019	0.00026	**0.00011**	0.00015	0.00013	0.00011
ZDT4	**0.00146**	0.00120	0.00477	0.00241	0.00249	0.00144
ZDT6	0.07628	0.01500	0.06772	0.01330	**0.00032**	0.00013
DTLZ1	0.00031	0.00013	**0.00018**	0.00017	0.00557	0.03590
DTLZ2	0.00017	0.00003	0.00019	0.00005	**0.00016**	0.00005
DTLZ3	0.00270	0.00190	**0.00025**	0.00016	0.00337	0.02242
DTLZ4	0.01586	0.02700	0.02268	0.03742	**0.00014**	0.00003
DTLZ7	0.00069	0.00079	0.00034	0.00054	**0.00009**	0.00002
UP1	0.00508	0.00630	0.00219	0.00180	**0.00146**	0.00101
UP2	0.00101	0.00083	0.00064	0.00062	**0.00056**	0.00062
UP3	0.03557	0.02300	0.01605	0.00864	**0.00923**	0.00395
UP4	**0.00146**	0.00032	0.00151	0.00030	0.00176	0.00060
UP7	0.01846	0.01900	0.00858	0.00832	**0.00512**	0.00464

(*continued on next page*)

Table 7. (*Continued.*)

Problem	NSGA-II		NSGA/LS		NSGA/LSP	
	Mean	StDev	Mean	StDev	Mean	StDev
UP8	**0.00162**	0.00150	0.00303	0.00188	0.00163	0.00155
UP10	0.17940	0.18000	0.12659	0.15271	**0.12065**	0.10789
KNO1	0.00239	0.00045	**0.00163**	0.00015	0.00221	0.00041
OKA1	0.00341	0.00220	0.00139	0.00129	**0.00110**	0.00100
OKA2	0.11960	0.05100	0.05850	0.03328	**0.02227**	0.01488
VLMOP2	**0.00011**	0.00002	0.00012	0.00003	0.00015	0.00007
Kursawe	**0.00216**	0.00019	0.00227	0.00035	0.00254	0.00032
DMM	0.00057	0.00026	**0.00040**	0.00010	0.00046	0.00018
Comet	**0.00136**	0.00035	0.00152	0.00040	0.00146	0.00035
VLMOP3	0.00094	0.00027	0.00091	0.00022	**0.00086**	0.00024
SPH	0.00596	0.00098	**0.00562**	0.00086	0.00572	0.00111

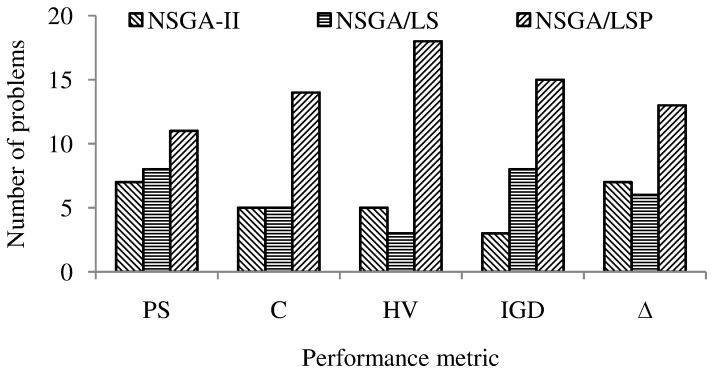

Fig. 4. Comparison of the algorithms by number of problems solved best.

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 performance. All three algorithms were evaluated according to all five performance metrics used in the experimental investigation. The results are illustrated in Fig. 4, 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/LS and NSGA/LSP, respectively. As illustrated in the figure, NSGA-II gives the best Pareto size values for 7 test problems, while NSGA/LS was the best for 8, and NSGA/LSP – for 11. According to coverage metric, NSGA-II was the best for 5, NSGA/LS – for 5, NSG-II/LSP – for 14 test problems, and for 1 test problem all three algorithms give zero values of coverage metric. According to the hyper-volume metric, NSGA-II solves best 5 test problems, NSGA/LS – 3 and NSGA/LSP – 18. Three test problems were solved best by NSGA-II according to IGD metric, while NSGA/LS and NSGA/LSP solve best 8 and 15 test problems, respectively. According to spread metric, NSGA-II solves best 7, NSGA/LS – 6, and NSGA/LSP – 13 test problems. Thus we can conclude that algorithm NSGA/LSP gives the best result of the evaluation according to any metric of performance. One test problem (VLMOP2) was

solved best by NSGA-II, however advantages were insignificant, and 9 test problems were solved best by NSGA/LSP according to all metrics.

Although NSGA/LSP increases the performance metrics for most of test problems which were investigated, for some test problems some performance metrics were reduced. Also there is no information about how much a performance metric was increased or reduced. In order to elicit such an information, algorithms NSGA/LS and NSGA/LSP were evaluated by the percentage improvement of values of performance metrics obtained by the classical algorithm NSGA-II. The results of the evaluations are illustrated in Figs. 5–8. In order to avoid the uncertainty which would arise zero values of C metric values produced by NSGA-II for most test problems, this metric was not included in the evaluation. Instead, the results of evaluation by improvement in amount of decision vectors is presented in Fig. 9. Results presented in the figures show that performance metrics have been more or less improved for most of test problems investigated.

Several illustrations of approximations of Pareto fronts obtained by NSGA-II and NSGA/LSP are illustrated in Fig. 10.

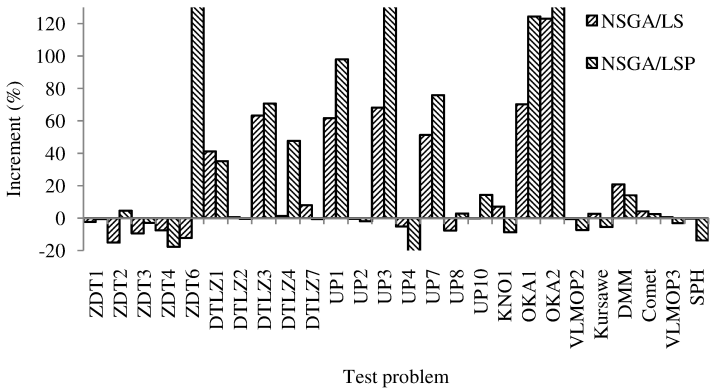

Fig. 5. Percentage improvement of performance metric PS.

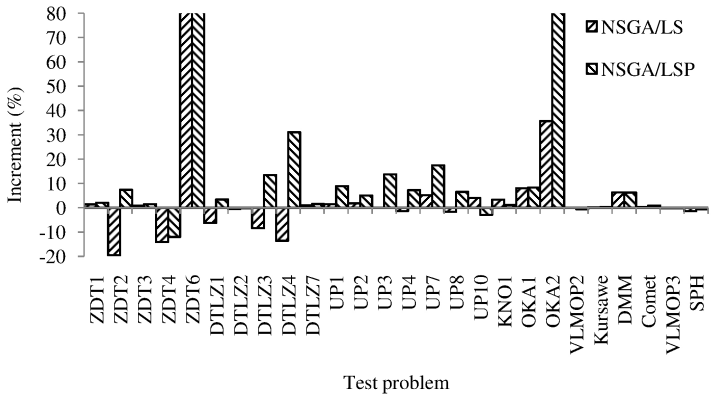

Fig. 6. Percentage improvement of performance metric HV.

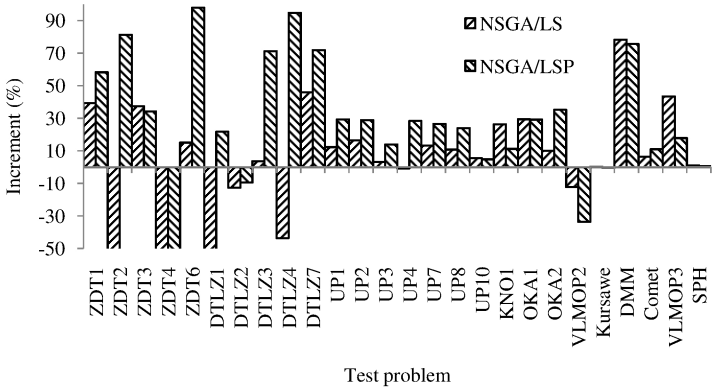

Fig. 7. Percentage improvement of performance metric IGD.

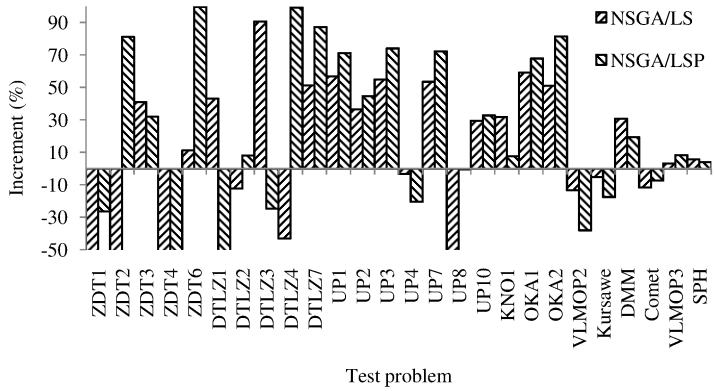

Fig. 8. Percentage improvement of performance metric Δ.

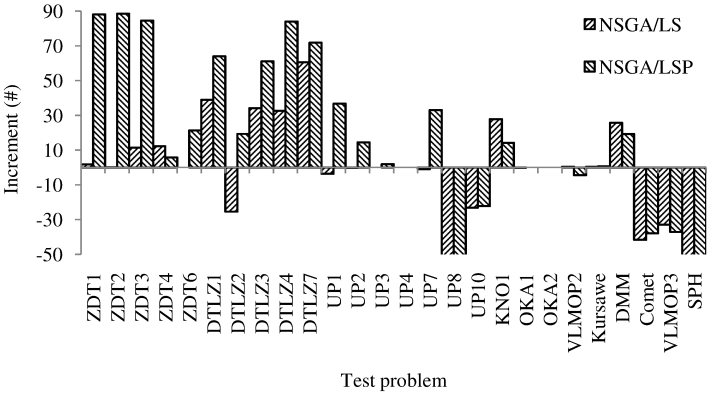

Fig. 9. Improvement in units of performance metric C.

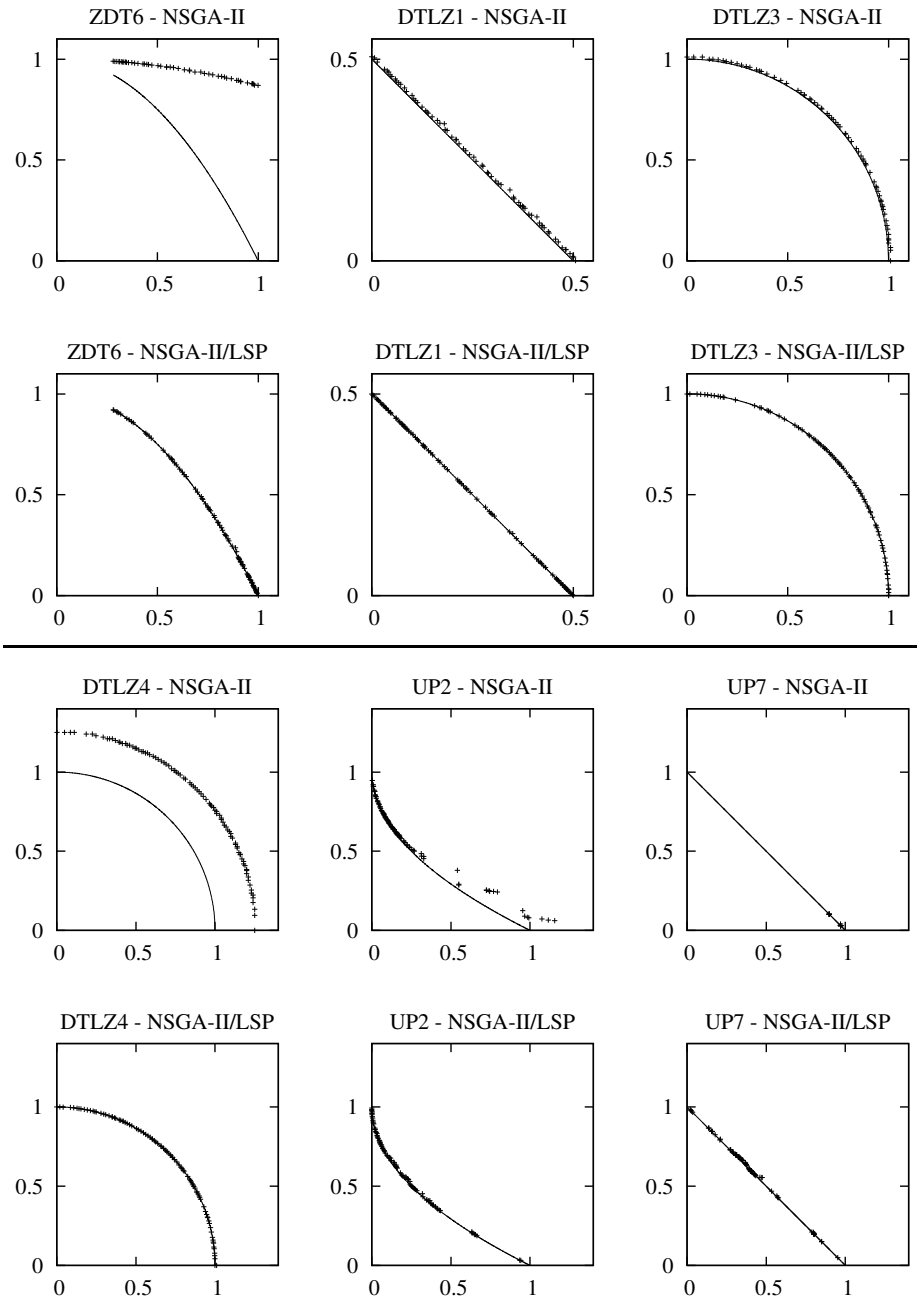

Fig. 10. Illustration of approximations of Pareto fronts of several test functions, obtained by NSGA-II and NSGA/LSP.

Third experiment was dedicated to compare the performance of the algorithm NSGA/LSP with performance of the Multi-objective Optimization Evolutionary Algorithm based on Decomposition (MOEA/D), which has been presented by Zhang and Li in [20]. MOEA/D decomposes a multiobjective optimization problem into a number of scalar optimization subproblems and optimizes them simultaneously. MOEA/D has the best performance on test functions UP1–UP10 in the CEC 2009 Special Session and Competition [14]. Implementation of the algorithm has been downloaded from the MOEA/D Homepage. Both algorithms NSGA/LSP and MOEA/D have been ran for a certain time period – 20 seconds, within which at least one of the algorithms provide valuable solution. The size of the population in NSGA/LSP has been chosen to be 200.

Local search has been performed for 800 function evaluations after every 2000 function evaluations performed by global search. The parameters of MOEA/D have been chosen to be the same as in CEC 2009 Special Session and Competition. Both algorithms have been ran for 100 independent runs on test functions UP1, UP2, UP3, UP4, UP7, UP8 and UP10. Performance of the algorithms has been evaluated by hyper-volume metric, which was measured at discrete time moments – 3rd, 5th, 10th, 15th and 20th second of experiment. Results of the experiment are presented in Fig. 11,

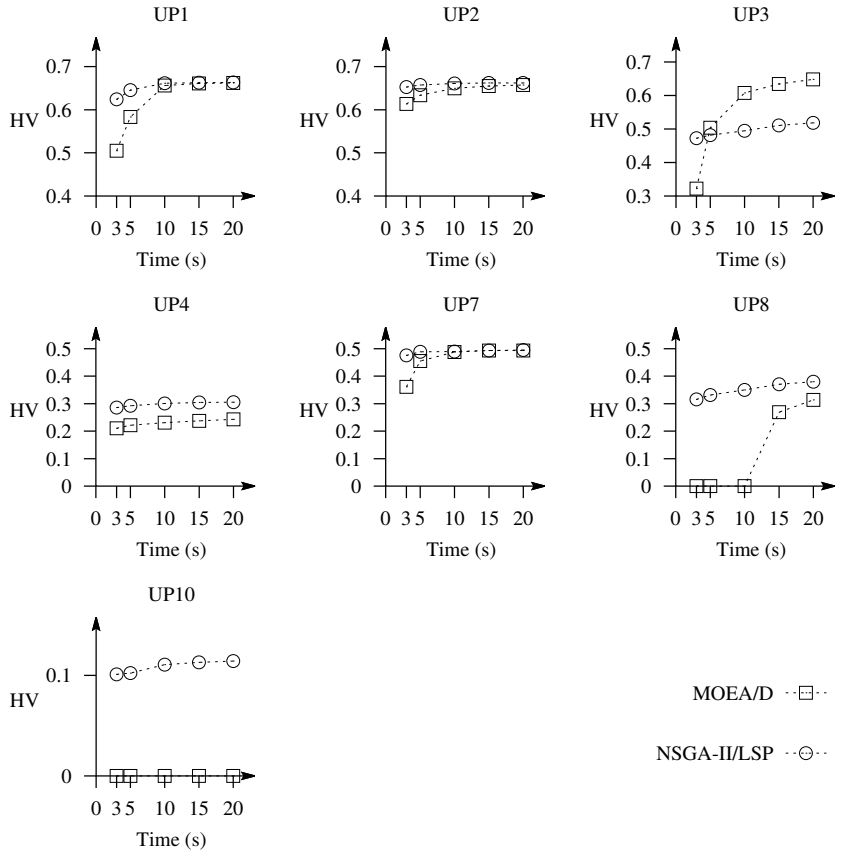

Fig. 11. HV values of different test functions, obtained by algorithms MOEAD/D and NSGA/LSP at discrete time moments.

Table 8. IGD values obtained solving test problems with 10 objectives.

Problem	NSGA-II		NSGA/LSP	
	Mean	StDev	Mean	StDev
DTLZ1	20.350	14.530	0.523	0.237
DTLZ2	1.641	0.238	0.677	0.104
DTLZ3	25.020	15.480	0.912	0.258
DTLZ4	2.126	0.201	0.781	0.139

where the horizontal axis in each graph represents time moments, and the vertical axis – value of hyper-volume. Different graphs represent different test function and different curves – different algorithm. The results show that NSGA/LSP produces significantly better results for almost all test functions within fixed time period. MOEA/D was significantly better for one test function – UP3, however MOEA/D was unable to find any evaluable approximation of Pareto front of test function UP10 within 20 seconds.

The last experiment was dedicated to investigate the ability of NSGA/LSP to solve hard test problems – DTLZ1 to DTLZ4 with 10 objectives. Number of variables have been chosen to be 14 for test problems DTLZ1 and DTLZ3, and 20 – for DTLZ2 and DTLZ4, as it was suggested in [13]. The algorithm has been ran for 1×10^6 function evaluations with population size of 1000. Local search has been performed for 5000 function evaluation after every 10000 function evaluations of global search. Every test function has been optimized by 100 independent runs. Performance has been measured and compared with NSGA-II by IGD metric. Results, presented in Table 8, showed that NSGA/LSP gives significantly better IGD value for all test functions investigated.

5 Conclusions

The paper proposes a hybrid multi-objective optimization algorithm based on Non-dominated Sorting Genetic Algorithm (NSGA-II) and improved Multi-Objective Single Agent Stochastic Search algorithm. The developed algorithm NSGA/LSP has been experimentally investigated and evaluated by solving a set of 26 test problems and measuring five different metrics of performance. Given results show that involving the presented local search method into the evolutionary algorithm NSGA-II gives significant advantage for most of test functions according to different performance metrics. Performance metrics are significantly improved for most of test problems by involving improvement in Multi-objective Single Agent Stochastic Search based on changing randomly only part of coordinate values.

References

1. N. Srinivas, K. Deb, Multiobjective optimization using nondominated sorting in genetic algorithms, *Evolutionary Computation*, **2**(3):221–248, 1994.

2. J.D. Schaffner, Multiple objective optimization with vector evaluated genetic algorithms, in: J.J. Grefenstette (Ed.), *Proceedings of the First International Conference on Genetic Algorithms and Their Applications, July 24–26, 1985 at the Carnegie-Mellon University, Pittsburg, PA*,1985, pp. 93–100.

3. A. Zhoua, B.Y. Qu, H. Li, S.Z. Zhao, P.N. Suganthan, Q. Zhangd, Multiobjective evolutionary algorithms: A survey of the state of the art, *Swarm and Evolutionary Computation*, **1**(1):32–49, 2011.

4. A. Lančinskas, J. Žilinskas, P.M. Ortigosa, Local optimization in global multi-objective optimization algorithms, in: *Third World Congress on Nature and Biologically Inspired Computing (NaBIC 2011), Salamanca, Spain, October 19–21*, 2011, pp. 323–328.

5. F.J. Solis, R.J. Wets, Minimization by random search techniques, *Math. Oper. Res.*, **6**(1):19–30, 1981.

6. K. Mitra, K. Deb, S.K. Gupta, Multiobjective dynamic optimization of an industrial Nylon 6 semibatch reactor using genetic algorithms, *J. Appl. Polym. Sci.*, **69**(1):69–87, 1998.

7. D.S. Weile, E. Michielssen, D.E. Goldberg, Genetic algorithm design of Pareto-optimal broad band microwave absorbers, *IEEE Trans. Electromagn. Compat.*, **38**(3):518–525, 1996.

8. K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, *IEEE Trans. Evol. Comput.*, **6**(2):182–197, 2002.

9. P.M. Ortigosa, I. García, M. Jelasity, Reliability and performance of UEGO, a clustering-based global optimizer, *J. Glob. Optim.*, **19**(3):265–289, 2001.

10. P.M. Ortigosa, J.L. Redondo, I. García, J.J. Fernández, A population global optimization algorithm to solve the image alignment problem in electron crystallography, *J. Glob. Optim.*, **37**(4):527–539, 2007.

11. J.L. Redondo, J. Fernández, I. García, P.M. Ortigosa, Heuristics for the facility location and design $(1|1)$-centroid problem on the plane, *Comput. Optim. Appl.*, **45**(1):111–141, 2010.

12. E. Zitzler, K. Deb, L. Thiele, Comparison of multiobjective evolutionary algorithms: Empirical results, *Evolutionary Computation*, **8**(2):173–195, 2000.

13. K. Deb, L. Thiele, M. Laumanns, E. Zitzler, Scalable test problems for evolutionary multi-objective optimization, in: A. Abraham, L. Jain, R. Goldberg (Eds.), *Evolutionary Multi-objective Optimization: Theoretical Advances and Applications*, Advanced Information and Knowledge Processing, Springer, 2005, pp. 105–145.

14. Q. Zhang, A. Zhou, S. Zhao, P.N. Suganthan, W. Liu, S. Tiwari, *Multiobjective Optimization Test Instances for the CEC 2009 Special Session and Competition*, Working Report, School of Computer Science and Electrical Engineering, University of Essex, 2008.

15. C. Grosan, A. Abraham, Approximating Pareto frontier using a hybrid line search approach, *Inf. Sci.*, **180**(14):2674–2695, 2010.

16. F. Kursawe, A variant of evolution strategies for vector optimization, in: H.P. Schwefel, R. Männer (Eds.), *Parallel Problem Solving for Nature*, Lect. Notes Comput. Sci., Vol. 496, Springer-Verlag, Berlin, 1990, pp. 193–197.

17. K. Deb, Multi-objective genetic algorithms: Problem difficulties and construction of test problems, *Evolutionary Computation*, **7**(3):205–230, 1999.

18. E. Zitzler, M. Laumanns, L. Thiele, SPEA2: Improving the strength Pareto evolutionary algorithms for multiobjective optimization, in: *Evolutionary Methods for Design, Optimization and Control with Application to Industrial Problems: Proceedings of the EUROGEN 2001 Conference, Athens, Greece, 19–21 September 2001*, 2002, pp. 95–100.

19. A. Zhou, Y. Jin, Q. Zhang, B. Sendhoff, E. Tsang, Combining model-based and genetics-based offspring generation for multi-objective optimization using a convergence criterion, in: *Proceedings of the IEEE Congress on Evolutionary Computation (CEC), Vancouver, Canada, 16–21 July 2006*, IEEE Press, 2006, pp. 3234–3241.

20. Q. Zhang, H. Li, MOEA/D: A multiobjective evolutionary algorithm based on decomposition, *IEEE Trans. Evol. Comput.*, **11**(6):712–731, 2007.