Hierarchical Approach to Evolutionary Multi-Objective Optimization

Eryk Ciepiela[1], Joanna Kocot[1], Leszek Siwik[1], Rafał Dreżewski[1]

[1]Department of Computer Science, AGH University of Science and Technology, Kraków, Poland
{siwik,drezew}@agh.edu.pl

Abstract. In this paper a new "hierarchical" evolutionary approach to solving multi-objective optimization problems is introduced. The results of experiments with standard multi-objective test problems, which were aimed at comparing "hierarchical" and "classical" versions of multi-objective evolutionary algorithms, show that the proposed approach is a very promising technique.

1 Introduction

The most natural process of decision making for human being consists in analyzing many—often contradictory—factors and searching for peculiar compromise among them. Such decisive process is known as a *multi-criteria decision making (MCDM).* The most frequently, *MCDM* process is based on appropriately defined *multi-objective optimization problem (MOOP).* Following [2]—*multi-objective optimization problem* in its general form is defined as minimizing/maximizing the set of objectives $f_m(\bar{x}),\quad where \quad m = 1, 2 \ldots, M$, taking into account the set of constraints, which define all possible (feasible) decision alternatives ($\mathcal{D}$).

Because there are many criteria, to indicate which solution is better than the other, so called dominance relation is used [2]. A solution of the multi-objective optimization problem in the Pareto sense means determination of all non-dominated alternatives from the set $\mathcal{D}$.

During over 20 years of research on evolutionary multi-objective algorithms (EMOAs) quite many techniques have been proposed. Generally all of these techniques and algorithms can be classified as elitist (which give the best individuals the opportunity to be directly carried over to the next generation) or non-elitist ones [2].

The Hierarchical Genetic Strategy (HGS) was introduced by Kołodziej and Schaefer [4] as one of the multi-deme, parallel genetic algorithms models. The main idea of HGS is running a set of dependent evolutionary processes in parallel. Its dependency relation has a tree structure with fixed depth. The tree nodes which are closer to the root perform chaotic search with low accuracy—they detect promising regions of the optimization landscape—while more accurate searching is done in further successor nodes.

In the course of this paper the new “hierarchical” approach to multi-objective optimization based on HGS model is presented and compared experimentally with “classical” EMOAs. “Hierarchical” in this context means that presented algorithm is trying to identify more and more precisely (in a hierarchical way) the more and more accurate approximation of non-dominated points.

The paper is organized as follows. First the HGS model is described with more details. Next the “hierarchical” approach to multi-objective optimization (MOHGS) is presented. The preliminary experimental results comparing “hierarchical” and “classical” versions of multi-objective evolutionary algorithms conclude the paper.

2 Hierarchical Genetic Strategy

As it was said already, the main idea of HGS is running in parallel a set of dependent evolutionary processes organized as a tree with more and more accurate searching done in the nodes located far and far away from the root of the tree [4]. The HGS node’s individuals represent the solutions (phenotypes) with precision that increases with the node’s level.

After a *metaepoch* which lasts a predefined number of iterations of evolutionary algorithm, each HGS node chooses the best individuals. Each of them constitutes a new “child” population. This procedure is called *sprouting operation* and is performed conditionally, according to the outcome of the *branch comparison operation*. It is reasonable as long as such a comparison prevents the same or similar individual from sprouting identical or similar populations in the child HGS nodes.

Avoiding exploration of the same regions of the optimization landscape is also supported by further operation called *reduction*. However, unlike *branch comparison operation*, *reduction* is performed after branches have been sprouted. *Reductions* can be performed both within the scope of sibling HGS nodes—then it is said to be local, and globally when sibling scope is exceeded. Details on how the above mentioned operations are carried out depend strongly on the implementation of HGS.

The individual being sprouted, has to be prepared to find a new HGS node of increased precision, what means that the individual (phenotype) has to be specified more precisely. This is accomplished e.g. by appending randomly least significant bits to the floating point number’s binary representation.

3 Multi-Objective Hierarchical Genetic Strategy

The first implementations of HGS ([6]) used simple genetic algorithm (SGA) as the main optimization algorithm, therefore being limited to optimizing single-objective problems. Our goal in this paper is to introduce multi-objective optimization algorithms to HGS—thus developing multi-objective HGS (MOHGS)—and provide it with new features to take advantage of these algorithms’ properties.

In our approach SGA algorithm was replaced by vector evaluated genetic algorithm (VEGA) [5], multiple objective genetic algorithm (MOGA) [3], strength pareto evolutionary algorithm (SPEA) [9] and non-dominated sorting genetic algorithm II (NSGA-II) [2]. However, any other multi-objective evolutionary algorithm could have been used as well.

3.1 Sprouting and Reduction Operators in MOHGS

Having more than one objectives to operate on, means that in the process of sprouting more than one criterion can be considered. Furthermore, these criteria do not have to be limited only to objectives, but other properties of the investigated population (generation) can be used—e.g. the domination level (see [2]). It can be also advisable to promote individuals which seem to differ from others in either the problem domain space or the objective space. It is also likely to take into consideration the criteria based on the ideas known from the multi-objective evolutionary algorithms, like niching or elitism (see [2]).

New HGS node sprouted with respect to one criterion creates a new population (generation), which neither can be compared with nor reduced to populations resulting from sprouting with regard to other criteria. In this way, HGS tree can be considered as a colored tree, where the node's color denotes the node sprouting criteria. Certainly, such an approach may cause existence of similar populations in HGS tree. However, it was found desirable as long as it ensures heavier computation of those populations that were promoted with respect to more than one criterion.

4 Selected Aspects of the System Realization

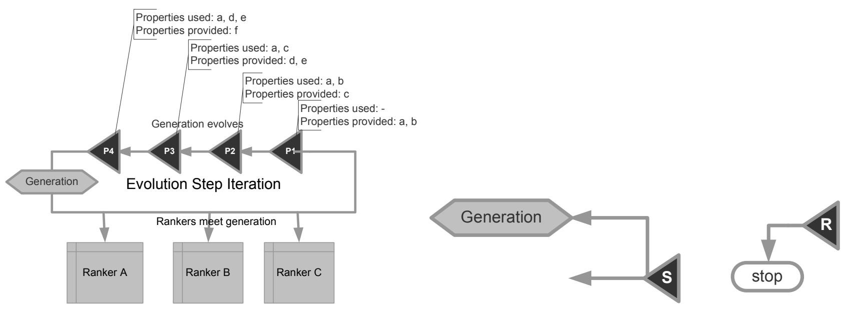
In the following section the key aspects of MOHGS implementation on the EA-AE platform ([1]) are described.

4.1 The EA-AE Platform

The EA-AE system is a runtime environment that provides the convenient approach to development, "composition", and running of different types of evolutionary algorithms. Its approach is based on the idea of multiple "processors" which, arranged in a sequence, create a single iteration of an algorithm. Each of the processors transforms the given generation of a population, in a way defined by the developer. A single processor can be for example responsible for performing mutation (or recombination) operation on the current generation of individuals.

Despite of being independent, the processors have to be able to exchange some information. As an example, the recombination based on the previously selected pool of individuals can be considered. If so, the information about pool has to be passed forward from pool selector processor. To enable such communication, the *individual properties* concept is introduced. Processors are able to annotate each individual of the generation they are transforming with "key-value" pairs. In this way, the information is exchanged in "per individual" context and is actually attached to the individual. These "key-value" pairs are called properties and are carried throughout the "evolution step" (see further in this section). The information the processor needs is expressed by property-existence requirements. Hence, the processors are characterized by properties they use and by those they provide.

Once the generation passes the chain of processors custom rankers rate (in the way defined by the developer) the generation's individuals. Each ranker keeps a ranking of the best individuals, to which the rated ones are compared. Depending on the comparison, they are inserted (or not) into the ranker's list.



(a) Evolution step

(b) Sprouting ("S") and reduction ("R") processors

Fig. 1. Principles of EA-AE platform functioning.

The sequence of processors along with rankers constitute the *evolution step* (see Fig. 1a). Evolution step is performed in the context of the problem's objective function and the search region constraints defining a phenotype space.

The phenotype space is responsible for bidirectional mapping between individual's genotype and phenotype. Since within EA-AE phenotypes and genotypes are represented in a generic, normalized, and uniform way, generation processors remain universal and independent of phenotype space. The generic binary genotype is provided as well as floating point phenotype feature (decision variable), etc. What is more, since the objective function is defined using generic phenotypes, it is also separated from the phenotype space. Therefore, the orthogonal concerns such as: search region constraints along with genetic representation, individuals rating, and applied evolutionary algorithm are independent.

The evolution step constitutes the iteration of the evolutionary algorithm. Since the evolution step can be composed of processors and rankers chosen by developer, ideas from different algorithms may be combined in order to find the best heuristic for solving various optimization problems.

4.2 Multi-Objective HGS Meets EA-AE

The EA-AE platform was used for the implementation of the MOHGS algorithms. It was possible because of the EA-AE platform mechanisms and features, which include: the generation processors ability to spawn computation (which enables the HGS branch sprouting), and computation controller, which maintains the structure of the spawned computations and therefore enables HGS tree structure and HGS branch reduction operation. The implementation of MOHGS was reduced to providing HGS-specific generation processors.

As it was previously said, every generation from the HGS node is rated with regard to several criteria. For each of those, a certain number of best individuals is marked and can be used for sprouting new HGS nodes—each initially containing a population of one individual, copied from the initial population. A HGS node containing the derived population evolves in the same way as its parent node does, still, representing the individuals with higher precision. Simultaneously the initial population's node continues the evolutionary algorithm's transformations.

The sprouting process is performed by generic sprouting processor (see Fig. 1b) which designates individuals annotated by property with certain name to be sprouted.

The reduction operator is used when populations of two different HGS nodes are operating on a very similar area of decision variables space. The similarity of the HGS nodes defined as the similarity of the areas of decision variables space they operate on, was estimated by the Equation (1).

$$sim(P_i, P_j) = \frac{1}{|P_i| \cdot |P_j|} \sum_{ind^i \in P_i} \sum_{ind^j \in P_j} dist(ind^i, ind^j) \qquad (1)$$

where: $sim(P_i, P_j)$ is the similarity measure of the two populations P_i and P_j (located in the i-th and j-th HGS nodes), $|P_i|$ is the number of individuals in the population P_i, ind^i is the individual from population P_i, $dist(ind^i, ind^j)$ is the distance between ind^i and ind^j individuals in the Euclidean metric ($ind \in \mathbb{R}^n$).

The nodes reduction operator is introduced as a custom generation processor (see Fig. 1b), which calculates the similarity at the same level of the HGS tree—not only locally between sibling nodes, but also globally. Since the measure (1) is symmetrical, in order to avoid computation redundancy as well as simultaneous reduction of two close populations, each node compares itself only to its ancestors. The node with a similarity measure value less than similarity threshold value does not continue computations. However, despite stopping the parent node, its children remains active.

To avoid the effectiveness losses, the reduction processor was placed before the processor responsible for sprouting in the HGS processor sequence. Otherwise, the amount of similar non-sibling child nodes would have been sprouted just before their parent nodes similarity evaluation.

Each of the HGS nodes is allowed to perform only a defined number of computing iterations. After that, like in the case of reduction, it is stopped. The whole strategy is completed when **all** the HGS nodes are stopped.

5 Experimental Results

The experiments were intended to compare "hierarchical" realizations of various multi-objective evolutionary algorithms with their "classical" versions. To achieve this the Zitzler-Deb-Thiele ZDT1, ZDT3, and ZDT6 problems ([8]) were used.

5.1 The Methodology of the Experiments

In order to ensure comparable results, both classical and HGS-based algorithms were allowed to evolve for the same overall number of iterations. The results' quality was compared using various metrics. The measured time of execution also gave an outlook on the overhead introduced by the HGS strategy.

For the experiments the following multi-objective evolutionary algorithms were chosen: vector evaluated genetic algorithm (VEGA), multiple objective genetic algorithm (MOGA), strength pareto evolutionary algorithm (SPEA) and non-dominated sorting genetic algorithm II (NSGA-II) [2].

The metrics used for measuring the quality of the obtained results are described in [7]. The first trivial metric that was taken into consideration was the *number of non-dominated individuals (solutions)*, which form the approximation of the Pareto frontier. Generally, the greater the value of this metric is, the better is the quality of the found solution.

The *Inferior Region (IR)* metric calculates the size of the area dominated by obtained non-dominated individuals. The greater is the Inferior Region Metric' value, the better is the approximation of the ideal Pareto frontier.

The *Dominant Region Metric (DR)* measures the size of the area that dominates the obtained non-dominated individuals. Generally the smaller is the Dominant Region Metric' value, the better is the approximation of the ideal Pareto frontier. However, its value decreases also in the case of increasing concentration of the Pareto frontier without moving towards the "good point", thus local variations of the metric' value are possible.

The *Accuracy Frontier Metric (AF)* calculates the size of the area that neither dominates the obtained individuals, nor is dominated by them. It can be observed that the value of this metric can be expressed by the formula $1 - DR - IR$, where DR and IR are values of the DR and the IR metrics, respectively. The smaller is the AF metric value, the more complete and concentrated the Pareto frontier approximation is. However, the Accuracy Frontier Metric actually does not measure the quality of the ideal Pareto frontier approximation.

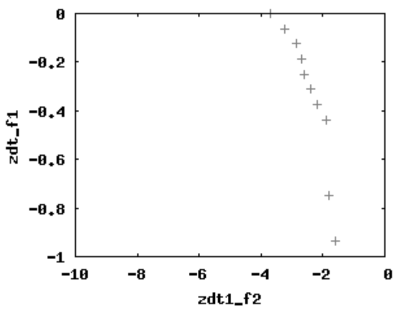
The *Pareto Spread Metric* compares the range of the obtained Pareto frontier and the ideal Pareto frontier (approximated by "good point" and "bad point").

The tests for all implemented evolutionary algorithms were carried out on a "flat" HGS structure (without sprouting and reduction—see Section 3.1) as well as on a regular HGS tree. This way the equal conditions were assured for both "classical" evolutionary algorithm (the "flat" structure) and HGS using this algorithm.

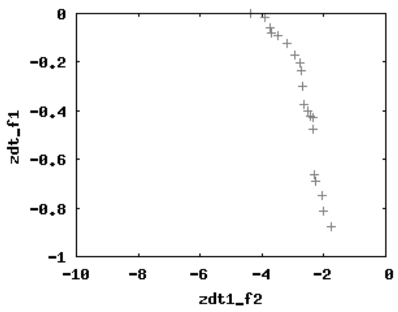
To ensure that the number of iterations will be the same in both cases, after HGS execution, iterations on all its nodes were counted and the compared algorithm was iterated exactly the same number of times. The number of individuals per population was also exactly the same.

5.2 Discussion of the Results

The Figures 2, 3, and 4 present plots of the obtained solutions for ZDT-1 with VEGA, ZDT-3 with MOGA, and ZDT-6 with SPEA, respectively. These three were chosen from 20 experiment runs, each per every combination of the problem and the algorithm. Each of the presented experiments completed in 450 iterations.

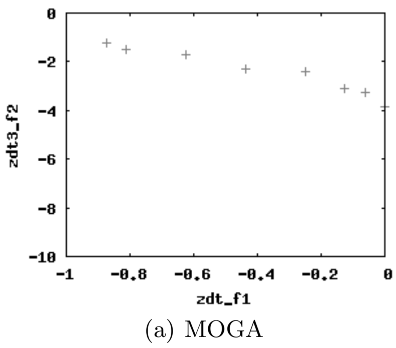


(a) VEGA

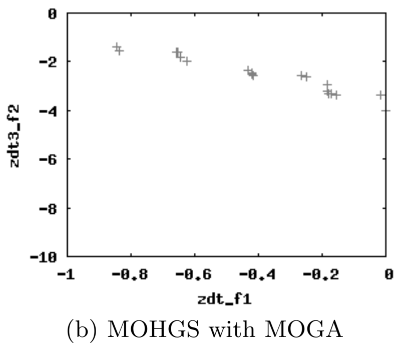


(b) MOHGS with VEGA

Fig. 2. The plot of obtained Pareto frontier for the ZDT-1 test problem.

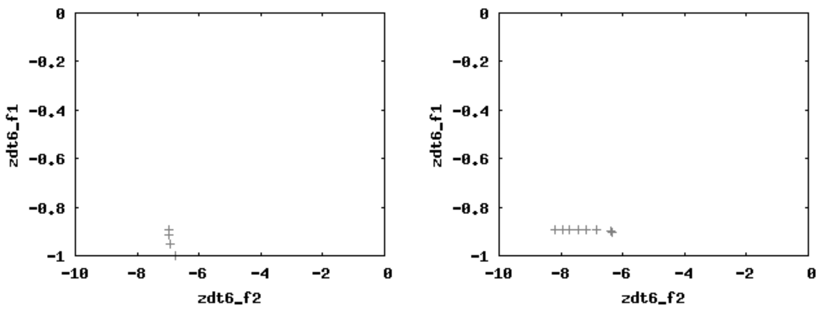


(a) MOGA



(b) MOHGS with MOGA

Fig. 3. The plot of obtained Pareto frontier for the ZDT-3 test problem.



(a) SPEA (b) MOHGS with SPEA

Fig. 4. The plot of obtained Pareto frontier for the ZDT-6 test problem.

Table 1. Number of non-dominated individuals metric values comparison.

test problem / algorithm	classic strategy	HGS
ZDT-1 / VEGA	10	21
ZDT-3 / MOGA	8	21
ZDT-6 / SPEA	7	8

Detailed comparison of the quality metrics results is shown in the Tables 1–5. The time of computation in each mentioned case was gathered in the Table 6.

On the basis of the experiment's results the HGS-based algorithms can be considered regarding to the following criteria:

1. Number of non-dominated individuals. In the case of HGS the number of non-dominated individuals is usually significantly greater than in the case of "classic" algorithm (see Table 1). It means that HGS offers a wider and more varied range of acceptable solutions.
2. Other Pareto frontier's quality metrics values. The "classic" strategies' Inferior Region metric values (Table 2) are in the most cases slightly better (the values are 1-5% higher). Similarly, the Dominant Region (Table 3) remains a little better (below 10%) in the case of "classic" algorithm. On the other hand, HGS obtains better Accuracy Frontier metric values (Table 4), which is implied by densely filled Pareto frontier. Moreover, the Overall Pareto

Table 2. Inferior Region metric values comparison.

test problem / algorithm	classic strategy	HGS
ZDT-1 / VEGA	0.7741	0.7284
ZDT-3 / MOGA	0.7715	0.7673
ZDT-6 / SPEA	0.0323	0.0393

Table 3. Dominant Region metric values comparison.

test problem / algorithm	classic strategy	HGS
ZDT-1 / VEGA	0.1981	0.2290
ZDT-3 / MOGA	0.1813	0.1981
ZDT-6 / SPEA	0.6958	0.7382

Table 4. Accuracy Frontier metric values comparison.

test problem / algorithm	classic strategy	HGS
ZDT-1 / VEGA	0.0279	0.0426
ZDT-3 / MOGA	0.0472	0.0345
ZDT-6 / SPEA	0.2719	0.2225

Spread is more satisfactory in HGS (see Table 5). The differences in Accuracy Frontier and Overall Pareto Spread metrics decrease with the test problem complexity.

3. Efficiency. The overhead introduced by the MOHGS usually varies between 10-30% (see Table 6).
4. Even Pareto Spread. However not measured by any metric formula, worth taking into consideration is the observation that HGS much more evenly spreads the Pareto frontiers' individuals (compare Figures 2, 3, and 4).

While performing the experiments a strong and non-linear impact of HGS parameters such as node similarity threshold value, population size or metaepoch length on the strategy course was observed.

It is also worth mentioning that HGS enables distribution and parallelization of computations and does not cause high communication overhead, which involves only sending sprouted individual. However, the reduction operation in distributed environment remains a non-trivial and extremely important issue.

6 Concluding Remarks

In this paper the "hierarchical" approach to evolutionary multi-objective optimization was presented. The results of preliminary experiments show that still more research is needed on the proposed technique—however it seems to be a

Table 5. Overall Pareto Spread metric values comparison.

test problem / algorithm	classic strategy	HGS
ZDT-1 / VEGA	0.1993	0.2283
ZDT-3 / MOGA	0.2276	0.2231
ZDT-6 / SPEA	0.0020	0.0019

Table 6. Time elapsed for computing [ms].

test problem / algorithm	iterations	classic strategy	HGS
ZDT-1 / VEGA	450	17906	23110
ZDT-3 / MOGA	450	27875	34188
ZDT-6 / SPEA	450	21766	19766

very promising approach especially in the case of difficult multi-objective problems, and the goal of this paper was to present the idea of MOHGS approach from the general point of view.

The future research could concentrate on additional verification of the proposed approach especially with the use of hard multi-objective problems and on the introduction of additional mechanisms (like niching and elitism) improving the obtained results. The agent-based realization of MOHGS seems to be the specially interesting direction of research. Such system will allow for introducing additional mechanisms—for example mechanisms of maintaining population diversity based on co-evolutionary interactions between evolving agents.

References

1. E. Ciepiela, J. Kocot, and L. Siwik. Composable runtime environment for building evolutionary algorithms. Technical report, Department of Computer Science, AGH University of Science and Technology, 2006.
2. K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms.* John Wiley & Sons, 2001.
3. C. Fonseca and P. Fleming. Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In *Genetic Algorithms: Proceedings of the Fifth International Conference*, pages 416–423. Morgan Kaufmann, 1993.
4. R. Schaefer and J. Kołodziej. Genetic search reinforced by the population hierarchy. In *Foundations of Genetic Algorithms 7*, pages 383–399. Morgan Kaufman Publisher, 2003.
5. J. D. Schaffer. *Some experiments in machine learning using vector evaluated genetic algorithms.* PhD thesis, Vanderbilt University, 1984.
6. B. Wierzba, A. Semczuk, J. Kołodziej, and R. Schaefer. Hierarchical genetic strategy with real number encoding. Technical report, Institute of Computer Science, Jagiellonian University, 2003.
7. J. Wu and S. Azarm. Metrics for quality assessment of a multiobjective design optimization solution set. *Transactions of the ASME, Journal of Mechanical Design*, 123:18–25, 2001.
8. E. Zitzler. *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications.* PhD thesis, Swiss Federal Institute of Technology, 2001.
9. E. Zitzler and L. Thiele. An evolutionary algorithm for multiobjective optimization: The strength pareto approach. Technical Report 43, Swiss Federal Institute of Technology, Zurich, Gloriastrasse 35, CH-8092 Zurich, Switzerland, 1998.