RESEARCH ARTICLE OPEN ACCESS

A Comprehensive Review on Multi-Objective Optimization Using Genetic Algorithms

Amarbir Singh*
*(Department of Computer Science, Guru Nanak DevUniversity, Amritsar)

----------**********----------

Abstract:

In real world applications, most of the optimization problems involve more than one objective to be optimized. The objectives in most of engineering problems are often conflicting, i.e., maximize performance, minimize cost, maximize reliability, etc. In the case, one extreme solution would not satisfy both objective functions and the optimal solution of one objective will not necessary be the best solution for other objective(s). Therefore different solutions will produce trade-offs between different objectives and a set of solutions is required to represent the optimal solutions of all objectives. Multi-objective formulations are realistic models for many complex engineering optimization problems. Customized genetic algorithms have been demonstrated to be particularly effective to determine excellent solutions to these problems. A reasonable solution to a multi-objective problem is to investigate a set of solutions, each of which satisfies the objectives at an acceptable level without being dominated by any other solution. In this paper, an overview is presented describing various multi objective genetic algorithms developed to handle different problems with multiple objectives.

Keywords **—Genetic Algorithms, Elitist Algorithm, Multi-Objective Optimization, Mutation**

----------**********----------

I. INTRODUCTION

While most real world problems require the simultaneous optimization of multiple, often competing, criteria (or objectives), the solution to such problems is usually computed by combining them into a single criterion to be optimized, according to some utility function. In many cases, however, the utility function is not well known prior to the optimization process. The whole problem should then be treated as a multiobjective problem with non-commensurable objectives. In the recent past, multi-objective optimization techniques have been successfully utilized to solve the problems having multiple conflicting objective in spite of their computational expenses. The availability of fast machines and computational models has boosted the use of these techniques to solve many problems [1].

There are two general approaches to multiple-objective optimization. One is to combine the individual objective functions into a single composite function. Determination of a single objective is possible with methods such as utility theory, weighted sum method, etc., but the problem lies in the correct selection of the weights or utility functions to characterize the decision-makers preferences. In practice, it can be very difficult to precisely and accurately select these weights, even for someone very familiar with the problem domain. Unfortunately, small perturbations in the weights can lead to very different solutions. For this reason and others, decision-makers often prefer a set of promising solutions given the multiple objectives.

The second general approach is to determine an entire Pareto optimal solution set or a representative subset. A Pareto optimal set is a set of solutions that are non-dominated with respect to each other. While moving from one Pareto solution to another, there is always a certain amount of sacrifice in one objective to achieve a certain amount of gain in the other. Pareto optimal solution sets are often

preferred to single solutions because they can be practical when considering real-life problems, since the final solution of the decision maker is always a trade-off between crucial parameters. Pareto optimal sets can be of varied sizes, but the size of the Pareto set increases with the increase in the number of objectives.

II.MULTI-OBJECTIVE OPTIMIZATION FORMULATION

A multi-objective decision problem is defined as follows: Given an n-dimensional decision variable vector x={x1,…,xn} in the solution space X, find a vector x* that minimizes a given set of K objective functions z(x*)={z1(x*),…,zK(x*)}. The solution space X is generally restricted by a series of constraints, such as gj(x*)=bj for j = 1, …, m, and bounds on the decision variables. In many real-life problems, objectives under consideration conflict with each other. Hence, optimizing x with respect to a single objective often results in unacceptable results with respect to the other objectives. Therefore, a perfect multi-objective solution that simultaneously optimizes each objective function is almost impossible. A reasonable solution to a multi objective problem is to investigate a set of solutions, each of which satisfies the objectives at an acceptable level without being dominated by any other solution. If all objective functions are for minimization, a feasible solution x is said to dominate another feasible solution y (x y ;), if and only if, zi(x) ≤ zi(y) for i=1, …, K and zj(x) < zj(y) for least one objective function j. A solution is said to be Pareto optimal if it is not dominated by any other solution in the solution space. A Pareto optimal solution cannot be improved with respect to any objective without worsening at least one other objective. The set of all feasible 3 non-dominated solutions in X is referred to as the Pareto optimal set, and for a given Pareto optimal set, the corresponding objective function values in the objective space is called the Pareto front. For many problems, the number of Pareto optimal solutions is enormous (maybe infinite).

III. GENETIC ALGORITHMS

The concept of genetic algorithms (GA) was developed by Holland and his colleagues in the 1960s and 1970s [2]. GA is inspired by the evolutionist theory explaining the origin of species. In nature, weak and unfit species within their environment are faced with extinction by natural selection. The strong ones have greater opportunity to pass their genes to future generations via reproduction. In the long run, species carrying the correct combination in their genes become dominant in their population. Sometimes, during the slow process of evolution, random changes may occur in genes. If these changes provide additional advantages in the challenge for survival, new species evolve from the old ones. Unsuccessful changes are eliminated by natural selection.

In GA terminology, a solution vector **x∈X** is called an individual or a *chromosome*.Chromosomes are made of discrete units called *genes*. Each gene controls one or more features of the chromosome. In the original implementation of GA by Holland, genes are assumed to be binary numbers. In later implementations, more varied gene types have been introduced. Normally, a chromosome corresponds to a unique solution **x** in the solution space. This requires a mapping mechanism between the solution space and the chromosomes. This mapping is called an encoding. In fact, GA works on the *encoding* of a problem, not on the problem itself. GA operates with a collection of chromosomes, called a *population*. The population is normally randomly initialized. As the search evolves, the population includes fitter and fitter solutions, and eventually it converges, meaning that it is dominated by a single solution. Holland also presented a proof of convergence (the schema theorem) to the global optimum where chromosomes are binary vectors.

GA uses two operators to generate new solutions from existing ones: *crossover* and*mutation*. The crossover operator is the most important operator of GA. In crossover, generally two chromosomes, called *parents*, are combined together to form new chromosomes, called *offspring*. The parents are selected among existing chromosomes in the population with preference towards fitness so that offspring is expected to inherit good genes which make the parents fitter. By iteratively applying the crossover operator, genes of good chromosomes are expected to appear more frequently in the

population, eventually leading to convergence to an overall good solution. The mutation operator introduces random changes into characteristics of chromosomes. Mutation is generally applied at the gene level. In typical GA implementations, the mutation rate (probability of changing the properties of a gene) is very small, typically less than 1%. Therefore, the new chromosome produced by mutation will not be very different from the original one. Mutation plays a critical role in GA. As discussed earlier, crossover leads the population to converge by making the chromosomes in the population alike. Mutation reintroduces genetic diversity back into the population and assists the search escape from local optima.

Reproduction involves selection of chromosomes for the next generation. In the most general case, the fitness of an individual determines the probability of its survival for the next generation. There are different selection procedures in GA depending on how the fitness values are used. Proportional selection, ranking, and tournament selection are the most popular selection procedures. The procedure of a generic GA is given as follows:

Step 1.

Set t =1. Randomly generate N solutions to form the first population, $P1$. Evaluate
the fitness of solutions in $P1$.

Step 2.

Crossover: Generate an offspring population Qt as follows.

2.1. Choose two solutions **x** and **y** from Pt based on the fitness values.

2.2. Using a crossover operator, generate offspring and add them to Qt.

Step 3.

Mutation: Mutate each solution $\mathbf{x} \in Qt$ with a predefined mutation rate.

Step 4.

Fitness Assignment: Evaluate and assign a fitness value to each solution $\mathbf{x} \in Qt$ based its objective function value and infeasibility.

Step 5.

Selection: Select N solutions from Qt based on their fitness and assigned them
$Pt+1$.

Step 6.

If the stopping criterion is satisfied, terminate the search and return the current
population, else, set $t=t+1$ go to Step 2.

IV. MULTI-OBJECTIVE GENETIC ALGORITHMS

Being a population based approach, GA are well suited to solve multi-objective optimization problems. A generic single-objective GA can be easily modified to find a set of multiple non-dominated solutions in a single run. The ability of GA to simultaneously search different regions of a solution space makes it possible to find a diverse set of solutions for difficult problems with non-convex, discontinuous, and multi-modal solutions spaces. The crossover operator of GA may exploit structures of good solutions with respect to different objectives to create new non-dominated solutions in unexplored parts of the Pareto front. In addition, most multi-objective GA do not require the user to prioritize, scale, or weigh objectives. Therefore, GA has been the most popular heuristic approach to multi-objective design and optimization problems. Jones et al. [3] reported that 90% of the approaches to multiobjective optimization aimed to approximate the true Pareto front for the underlying problem. A majority of these used a meta-heuristic technique, and 70% of all meta-heuristics approaches were based on evolutionary approaches. The first multi-objective GA, called Vector Evaluated Genetic Algorithms (or VEGA), was proposed by Schaffer [4]. Afterward, several major multi-objective evolutionary algorithms were developed such as Multi-objective Genetic Algorithm (MOGA), Niched Pareto 6 Genetic Algorithm, Random Weighted Genetic Algorithm (RWGA), Nondominated Sorting Genetic Algorithm (NSGA), Strength Pareto Evolutionary Algorithm (SPEA), Pareto-Archived Evolution Strategy (PAES), Fast Non-dominated Sorting Genetic Algorithm (NSGA-II), Multi-objective Evolutionary Algorithm (MEA), Rank-Density Based Genetic Algorithm (RDGA). It is to be noted that although there are many variations of multiobjective GA in the literature, these cited GA are well-known and credible algorithms that have been used in many applications and their performances were tested in several comparative

studies. In the next section details of important multi-objective genetic algorithms are discussed.

V. REVIEW OF ALGORITHMS PROPOSED IN LITERATURE

Various MOGAs proposed in the literature are described in subsequent subsections.

A. VIGA

An important implementation of MOGA is called Vector Evaluated Genetic Algorithm (VEGA), incorporates a modified selection process to cope with multiple evaluation criteria [5]. In this paper a modified selection procedure to handle multiple objectives. He suggested dividing the whole population into groups equal to a number of objectives. Selection procedure in each group is based on a single objective. Mating limits help limited combinations of individual solutions in the same group. Pairwise comparison helps to recognize the dominated solutions. After a few generations, a set of non-dominated solutions are recognized to represent the Pareto front [6]. But, the major difficulty with this algorithm is that it prevents to determine the location of the Pareto front. Another problem of the selection procedure, the individual solutions that are better in one objective are given preference over the other individual solutions. This leads the algorithm to converge to individually best solutions only.

B. MOGA

In 1993, another implementation of MOGA was proposed which employed the concept of niching and dominance along with the rank based fitness assignment. The non-dominated solutions are categorized into groups. The individual solutions are assigned same ranks in each group. The other groups of the solutions which are dominated by the current group are assigned next ranks. To maintain the diversity among the groups, the author proposed to use dynamically updated sharing. But, the major problem of this method is slow convergence that prevent from finding the optimum Pareto front [6].

C. *NSGA and Its Variants*

Non dominated Sorting Genetic Algorithm (NSGA) is based on the concept of dominance and sharing and it is quite similar to multi-objective genetic algorithms [7]. The NSGA is based on several layers of classification of individuals as suggested by [8]. Before selection is performed, the population is ranked on the basis of non-domination: all non-dominated individuals are classified into one category (with a dummy fitness value, which is proportional to the population size, to provide an equal reproductive potential for these individuals). To maintain the diversity of the population, these classified individuals are shared with their dummy fitness values. Then this group of classified individuals is ignored, and another layer of non-dominated individuals is considered. The process continues until all individuals in the population are classified. Since individuals in the first front have the maximum fitness value, they always get more copies than the rest of the population. The diversity among the individual solutions is maintained by using the sharing concept. How- ever, NSGA does not involve dynamic updating of any niche that makes it faster than MOGA. The algorithm of the NSGA is not very efficient, because Pareto ranking has to be repeated again. Evidently, it is possible to achieve the same goal in a more efficient way.

Niched Pareto Genetic Algorithm (NPGA) is based on the concept of dominance and sharing. NPGA differs from earlier approaches in the selection of individual solutions [9]. Here, the selection is based on a modified tournament selection than the proportional selection (as in NSGA). The basic idea of the algorithm is quite clever: two individuals are randomly chosen and compared against a subset of the entire population (typically, around 10% of the population). If one of them is dominated (by the individuals randomly chosen from the population) and the other is not, the the non- dominated individual wins. All the other situations are considered as a tie (i.e., both competitors are either dominated or non-dominated). When there is a tie, the result of the tournament is decided through fitness sharing.

NSGA- II is fast, elitist algorithm proposed by [10] and it is a generational algorithm that works upon the concept upon dominance. Instead of sharing, NSGAII uses the crowding distance to maintain the diversity among the individual solutions. Here, the author proposed to use

tournament selection strategy for selection of individual solutions. In this algorithm, to sort a population of as- signed size according to the level of nondomination, each solution must be compared with every other solution in the population to find if it is dominated. Solutions of the first non-dominated front are stored in the first Pareto front, solutions of the second front on the second Pareto front and so on. The new population is constituted by solutions on the first Pareto front, if they are less than the initial population size: solutions from the next front are taken according to their ranks. In the NSGA-II, for each solution one has to determine how many solutions dominate it and the set of solutions to which it dominates. The NSGA-II estimates the density of solutions surrounding a particular solution in the population by computing the average distance of two points on either side of this point along each of the objectives of the problem. This value is the so-called crowding distance. During selection, the NSGA-II uses a crowded-comparison operator which takes into consideration both the non-domination rank of an individual in the population and its crowding distance (i.e., non-dominated solutions are preferred over dominated solutions, but between two solutions with the same non-domination rank, the one that resides in the less crowded region is preferred). The NSGA-II does not use an external memory as the other MOEAs previously discussed. Instead, the elitist mechanism of the NSGA-II consists of combining the best parents with the best offspring obtained (i.e. non-dominated solutions are preferred over dominated solutions, but between two solutions with the same non-domination rank, the one that resides in the less crowded region is preferred). The NSGA-II does not use an external memory as the other MOEAs previously discussed. Instead, the elitist mechanism of the NSGA-II consists of combining the best parents with the best offspring obtained (i.e. a $(\mu + \lambda)$ selection). Due to its clever mechanisms, the NSGA-II is much more efficient (computationally speaking) than its predecessor, and its performance is so good, that it has become very popular in the last few years, becoming a landmark against which other multi-objective evolutionary algorithms have to be compared.

D. SPEA and Its Variants

Several researches were carried to improve the performance of VEGA, NPGA and NSGA. One such algorithm was proposed by [11] called the Strength Pareto Evolutionary Algorithm (SPEA). SPEA is an elitist MOGA where elitism helps to improve its convergence properties. Here, they proposed to maintain an archive of non-dominated solutions from the beginning of the algorithm. The archive is external to the main population, and it takes part in fitness computation. With the use of the archive, its size may increase very large so its pruning may be done to keep it in limits. Limited archive size helps in the selection of individual solutions. SPEA uses an archive containing non-dominated solutions previously found (the so-called external non-dominated set). At each generation, non- dominated individuals are copied to the external non-dominated set. For each individual in this external set, a strength value is computed. This strength is similar to the ranking value of MOGA [12], since it is proportional to the number of solutions to which a certain individual dominates. In SPEA, the fitness of each member of the current population is computed according to the strengths of all external non- dominated solutions that dominate it. The fitness assignment process of SPEA considers both close-ness to the true Pareto front and even distribution of solutions at the same time. Thus, instead of using Niches based on distance, Pareto dominance is used to ensure that the solutions are properly distributed along the Pareto front. Although this approach does not require a niche radius, its effectiveness relies on the size of the external non-dominated set. Since, the external non-dominated set participates in the selection process of SPEA, if its size grows too large, it might reduce the selection pressure, thus slowing down the search. Because of this, the authors decided to adopt a technique that prunes the contents of the external nondominated set so that its size remains below a certain threshold.

SPEA2 is an improved method for pruning the size of the archive that retain the boundary solutions in the archive. SPEA2 involves a fine grained fitness function based number of individual solutions that dominate a current solution and how many it

dominates. Fitness function also includes density information based on a k-NN algorithm. SPEA2 is found to be superior in performance than NSGA-II, especially in high dimensional search spaces [13]. SPEA2 has three main differences with respect to its predecessor.

1) It incorporates a fine-grained fitness assignment strategy which takes into account for each individual the number of individuals that dominate it and the number of individuals by which it is dominated

2) It uses a nearest neighbour density estimation technique which guides the search more efficiently.

3) It has an enhanced archive truncation method that guarantees the preservation of boundary solutions.

E. PAES

Pareto Archived Evolutionary Strategy (PAES) algorithm was proposed by [14]. It is a simple multi-objective evolutionary algorithm using a single parent- single child strategy. In this strategy, binary strings and bitwise mutation are used to create children in replacement of real parameters. PAES algorithm consists of a (1 + 1) Evolution strategy (i.e., a single parent that generates a single offspring) in combination with a historical archive that records the non-dominated solutions previously found [15]. This archive is used as a reference set against which each mutated individual is being compared. Such a historical archive is the elitist mechanism adopted in PAES. However, an interesting aspect of this algorithm is the procedure used to maintain diversity which consists of a crowding procedure that divides the objective space in a recursive manner. Each solution is placed in a certain grid location based on the values of its objectives (which are used as its coordinates or geographical location). A map of such a grid is maintained, indicating the number of solutions that reside in each grid location. Since the procedure is adaptive, no extra parameters are required (except for the number of divisions of the objective space).

F. PESA and Its Variants

Pareto Envelope based Selection Algorithm (PESA) was proposed by [16] which uses a small internal population and a large external population. The diversity is maintained by borrowing the concept of the hyper grid division of the phenotype space from PAES. But, the selection process is performed by the crowding distance method. In PESA, the external population plays an important role as it is considered to determine selection as well as to maintain diversity among the solutions.

The PESA is further revised to a new version called PESA-II [17]. The PESA-II uses region based selection. In the region based selection, the unit of selection is hyper box rather than an individual. Further, an individual is selected randomly from the hyperbox. The main objective of a PESA set of algorithms is to reduce the computational overhead associated with other representative methods.

G. AMGA and Its Variants

In Micro Genetic Algorithm (MGA) was originally proposed by [18]. It is a GA with a small population and reinitialization process. The working of the MGA involves the generation of random population, which gets loaded into memory in two different portions named replaceable and non-replaceable portion. The contents of replaceable portion get changed after each cycle of MGA whereas the contents of non-replaceable portion never changes during execution of the algorithm. The population of MGA is randomly taken as a mixture of individuals from both the portions. During each iteration, the algorithm experiences genetic operators. At the end of each iteration, two non-dominated individuals from final population are selected to compare with contents of the external memory. In this way, all dominated solutions from the external memory are removed. The MGA uses three forms of elitism:

1) It retains non-dominated solutions found within the internal iteration of the algorithm.

2) It uses a replaceable portion of the memory whose contents are partly restored at certain intervals.

3) It exchanges the population of the algorithm by nominal solutions created

The Archive based Micro Genetic Algorithm (AMGA) is a constrained multi-objective evolutionary optimization algorithm [19]. It is a generational genetic algorithm since during a particular iteration (generation), only solutions

created before that iteration takes part in the selection process. AMGA uses genetic variation operators such as crossover and mutation to create new solutions. For the purpose of selection, AMGA uses a two tier fitness assignment mechanism; the primary fitness is the rank which is based on the domination level and the secondary fitness is based on the diversity of the solutions in the entire population. This is in contrast to NSGA-II, where diversity is computed only among the solutions belonging to the same rank. The AMGA generates a very small number of new solutions at each iteration, and can be classified as a micro- GA. Generation of a very small number of solutions at every iteration helps in reducing the number of function evaluations by minimizing exploration of less promising search regions and directions. The AMGA maintains an external archive of good solutions obtained. Use of the external archive helps AMGA in reporting a large number of non-dominated solutions at the end of the simulation. It also provides information about its search history which is exploited by the algorithm during the selection operation. In each iteration, the parent population is created from the archive and the binary tournament selection is performed on the parent population to create the mating population. The off- spring population is created from the mating pool, and is used to update the archive. The size of the archive determines the computational complexity of the AMGA. The design of the algorithm is independent of the encoding of the variables and thus the proposed algorithm can work with almost any kind of encoding (so long as suitable genetic variation operators are provided to the algorithm). The algorithm uses the concept of Pareto ranking borrowed from NSGA-II and includes improved diversity computation and preservation techniques. The diversity measure is based on efficient nearest neighbor search and modified crowding distance formulation.

An improved Archive-based Micro Genetic Algorithm (referred to as AMGA2) for constrained multiobjective optimization is proposed in [20]. AMGA2 is designed to obtain fast and reliable convergence on a wide variety of optimization problems. AMGA2 benefits from the existing literature in that it borrows and improves upon several concepts from the existing multi-objective optimization algorithms. Improvements and modifications to the existing diversity assessment techniques and genetic variation operators are also proposed. AMGA2 employs a new kind of selection strategy that attempts to reduce the probability of exploring undesirable search regions. The proposed AMGA2 is a steady-state genetic algorithm that maintains an external archive of the best and diverse solutions and a very small working population. AMGA2 has been designed to facilitate the decoupling of the working population, the external archive, and the number of solutions desired as the outcome of the optimization process. Comprehensive benchmarks and comparison of AMGA2 with the current state-of-the-art multi-objective optimization algorithms demonstrate its improved search capability

VI. CONCLUSIONS

The MOGA has been successfully employed to solve the various problems of many domains having multiple conflictingobjectives. This paperpresented the background details of multi-objective genetic algorithms and highlighted the importance of NSGA and other algorithms proposed in the literature.

ACKNOWLEDGMENT

The Author is thankful to Punjab Technical University, Jalandhar for the support and motivation for research.

REFERENCES

[1] Tiwari, S.: Development and integration of geometric and optimization algorithms for packing and layout design. Ph.D. thesis, Clemson University 2009.

[2] Holland, J.H., Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, 1975.

[3] Jones, D.F., Mirrazavi, S.K., and Tamiz, M., Multiobjective meta-heuristics: an overview of the current state-of-the-art, *European Journal of Operational Research* 137(1), pp. 1-9, 2009.

[4] Schaffer, J.D. Multiple Objective optimization with vector evaluated genetic

algorithms.International Conference on Genetic Algorithm and their applications, 1985.
[5] Ahmadian, K., Golestani, A., Mozayani, N., Kabiri, P., Anew multi-objective evolutionary approach for creatingensemble of classifiers. In: Proc. of IEEE InternationalConference on Systems, Man and Cybernetics (ISIC), pp.1031–1036. IEEE, 2007.
[6] Engen, V.: Machine learning for network based intrusiondetection: an investigation into discrepancies in findingswith the kdd cup'99 data set and multi-objective evolutionof neural network classifier ensembles from imbalanceddata. Ph.D. thesis, Bournemouth University 2010.
[7] Srinivas, N., Deb, K.: Muiltiobjective optimization usingnondominated sorting in genetic algorithms. Evolution- ary computation 2(3), pp. 221–248 ,1994.
[8] Goldberg, D., Holland, J.: Genetic algorithms and machinelearning. Machine Learning 3(2), pp. 95–99, 1988.
[9] Horn, J., Nafpliotis, N., Goldberg, D.: A niched paretogenetic algorithm for multiobjective optimization. In: Proc.of the First IEEE Conference on Evolutionary Computation, pp. 82–87. IEEE 1994.
[10] Deb, K., Agrawal, S., Pratap, A., Meyarivan, T.: A fastelitist non-dominated sorting genetic algorithm for multiobjectiveoptimization: Nsga-ii. Lecture notes in computer science 1917, pp. 849–858 2000.
[11] Zitzler, E., Deb, K., Thiele, L.: Comparison of multiobjectiveevolutionary algorithms: Empirical results. Evolutionary computation 8(2), pp. 173–195 2000.
[12] Fonseca, C., Fleming, P., et al.: Genetic algorithms formultiobjective optimization: Formulation, discussion andgeneralization. In: Proc. of the fifth international conferenceon genetic algorithms, vol. 1, p. 416. San Mateo, California 1993.
[13] Ziztler, E., Laumanns, M., Thiele, L.: Spea2: Improvingthe strength pareto evolutionary algorithm for multiobjectiveoptimization. Evolutionary Methods for Design,Optimization, and Control pp. 95–100 (2002)
[14] Knowles, J., Corne, D.: The pareto archived evolutionstrategy: A new baseline algorithm for pareto multiobjectiveoptimisation. In: Proc. of the 1999 Congress onEvolutionary Computation (CEC), vol. 1. IEEE , 1999.
[15] Coello Coello, C.: Evolutionary multi-objective optimization:a historical view of the field. Computational IntelligenceMagazine, IEEE 1(1), pp. 28–36, 2006.
[16] Corne, D., Knowles, J., Oates, M.: The pareto envelopebasedselection algorithm for multiobjective optimization.In: Proc. of Parallel Problem Solving from NaturePPSN VI, pp. 839–848. Springer 2000.
[17] Corne, D., Jerram, N., Knowles, J., Oates, M., et al.:Pesa-ii: Region-based selection in evolutionary multiobjectiveoptimization. In: Proc. of the Genetic and EvolutionaryComputation Conference (GECCO2001). Cite- seer, 2001.
[18] Coello, C.C., Toscano, P.G.: A micro-genetic algorithm formultiobjective optimization. In: Proc. of Evolution- aryMulti-Criterion Optimization, pp. 126–140. Springer, 2001.
[19] Tiwari, S., Koch, P., Fadel, G., Deb, K.: Amga: anarchive-based micro genetic algorithm for multi-objectiveoptimization. In: Proc. of Genetic and EvolutionaryComputation conference (GECCO-2008), Atlanta, USA, pp.729–736, 2008.
[20] Tiwari, S., Fadel, G., Deb, K.: Amga2: improving the performanceof the archive-based micro-genetic algorithm formulti-objective optimization. Engineering Optimization43(4), pp. 377–401, 2011.