Volume 5, Issue 4, 2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

VEGA and MOGA an Approach to Multi-Objective Optimization

Indresh Kumar Gupta, Jeetendra Kumar
ABV-Indian Institute of Information Technology and Management,
Gwalior, Madhya Pradesh, India

Abstract— VEGA and MOGA are most common approaches to solve a multi-objective optimization problem. Multi-objective optimization problem arises when we have more than one conflicting objectives, for example: In two objectives the task is to maximize the first objective, while to minimize the other, then there exists no single solution while A set of solutions(trade-offs) is found in which we can never say any solution is better than others with additional information. . The fundamental principle of VEGA and MOGA are presented with manual calculations. We also present the advantage and disadvantage of each approach.

Keywords— VEGA (Vector evaluated genetic Algorithm), MOGA (multi- objective genetic algorithm), Fitness, Population, Dominate

I. INTRODUCTION

VEGA (vector evaluated genetic algorithm) and MOGA (multi-objective genetic algorithm) are two simple approaches to solve multi-objective optimization problem. Optimization refers to finding one or more feasible solutions which corresponds to extreme values of one or more objectives. The need for finding such optimal solutions in a problem comes mostly from the extreme purpose of either designing a solution for minimum cost of fabrication, or for maximum possible reliability, or others. when an optimization problem modelling a physical system involves only one objective function , the task of finding the optimal solution is called single objective optimization , similarly when has more than one objective functions, the task of finding one or more optimum solutions is known as multi-objective optimization[1][2][3][4]. The multi-objective optimization comes due to the conflicting objectives; there are no single optimum solutions. There exit a number of solutions which are optimal, and without any further information, no solution from the set of optimal solution can be said to be better than any other [5]. The multi-objective optimization problem (MOOP) is defined as,

$$\begin{aligned} &\text{Minimize/Maximize} \quad f_m(X), && m = 1,2,\ldots\ldots M;\\ &g_j(X) \geq 0, && j = 1,2,\ldots\ldots J;\\ &h_k(X) = 0, && k = 1,2\ldots\ldots K;\\ &x_i^l \leq x_i \leq x_i^u, && i = 1,2,\ldots\ldots n; \end{aligned}$$

Any solution is denoted as:

$$X = (x_1 x_2 \ldots\ldots x_n) \quad n-\text{dimensional problem}$$

In the context of optimization Deb [6] suggests that we can convert a maximization problem into minimization by multiplying the objective function by -1. For each solution in the decision space there exists a point in the objective space. The given below figure illustrate these two spaces and a mapping between them.

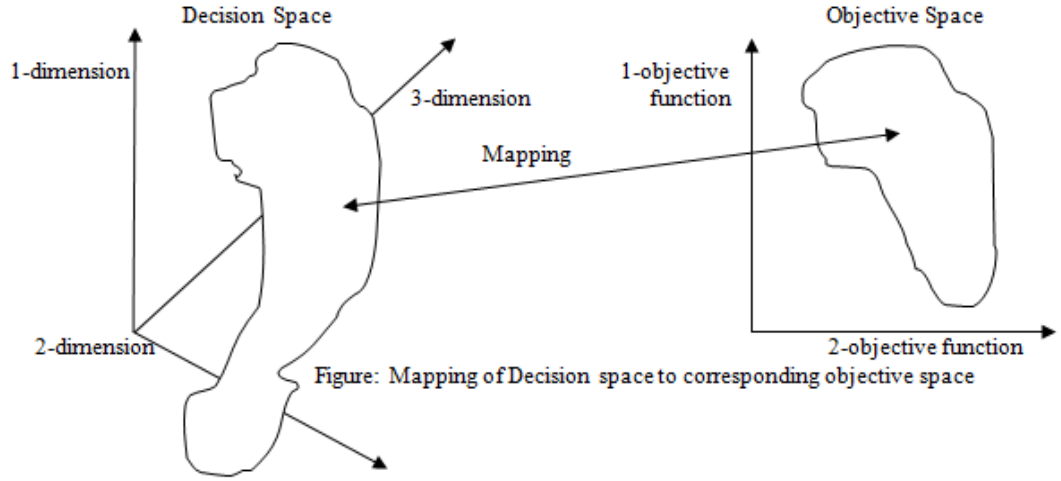

Figure: Mapping of Decision space to corresponding objective space

II. VEGA

VEGA (vector evaluated genetic algorithm) is the first multi-objective GA to find a set of non dominated solution given by Schaffer [7]. VEGA is simplest possible algorithm for multi-objective optimization using genetic algorithm and is a straightforward extension of single objective GA for multi-objective optimization. Since a number of objectives (say M) have to be handled, Schaffer thought of dividing the GA population at every generation into M equal subpopulation randomly. Each subpopulation is assigned a fitness based on a different objective function, after that proportionate selection operator is used to create the mating pool. In this way each of M objective functions is used to evaluate some members in the population.

"In proportionate Selection, if all population member fitness is F_{avg} , then i^{th}individual with fitness F_i gets an expected F_i/F_{avg} number of copies [8]."

VEGA (Vector Evaluated Genetic Algorithm) Procedure:

Step 1: Set an objective function counter $i = 1$ and define $q = N/M$.

Step 2: For all solution $j = 1 + (i - 1) * q$ to $j = i * q$, assign fitness as ,
$F(X^j) = f_i(X^j)$.

Step 3: Perform the proportionate selection on all q solutions to create a mating pool P_i.

Step 4: If $i = M$, go to Step 5. Otherwise increment i by one and go to step 2.

Step 5: Combine all mating pool together : $P = \cup_{i=1}^{M} P_i$. Perform crossover and mutation to create new a new population.

A little thought will reveal that the above algorithm emphasizes solutions which are good for individual objective functions. In order to find intermediate trade-off solutions, Schaffer allowed crossover between any two solutions in the entire populations. In this way Schaffer thought a crossover between too good solutions, each corresponding to a different objective, may find offspring which are good compromised solutions between the two objectives.

Manual Calculation: We consider the two objectives, two variable maximization optimization problem as shown below to better illustrate the working of the VEGA. Since a VEGA uses the proportionate selection operator, which in principle maximizes the fitness function, we have chosen the maximization problem. The randomly selected population is shown in figure 2. Now we show the step by step procedure of the VEGA.

Example:

$$\text{Maximize } f_1(X) = x_1,$$
$$\text{Maximize } f_2(X) = \frac{1 + x_2}{x_1},$$
$$\text{subject to } 0.1 \leq x_1 \leq 1,$$
$$0 \leq x_2 \leq 5,$$

Table 1: Six randomly chosen solution in the objective space for above maximization example.

Solution	x_1	x_2	f_1	f_2
1	0.31	0.89	0.79	53.90
2	0.43	1.92	0.67	53.21
3	0.22	0.56	0.88	52.91
4	0.59	3.63	0.51	52.15
5	0.66	1.41	0.44	56.35
6	0.83	2.51	0.27	55.77

Step 1: We Set the counter i=1 and parameter q=6/2=3. Since there are two objectives, we partition the population into two subpopulations. Let us say that partition 1 contains solution 1, 2 and 5, and partition 2 contains solutions 3, 4 and 6.

Step 2: Now, all solutions of first partition will be assigned a fitness based on the first objective function. Table 2 presents the assigned fitness to each of these three solutions of first population.

Step 3: It can be seen that solution 1 has the best fitness among all solutions in first partitions. Under a proportionate selection operator with scaling, solution 1 and solution 2 are expected to have two copies and one copy, respectively, in the first partition. It is likely that the worst solution (solution 5) will get eliminated after the selection operation.

Step 4: Next we move to Step 2 for the second objective function (i=2).
Step 2: Solutions 3,4 and 6 belong to the second partition and are assigned fitness values based on the second objective function. These fitness values are presented in Table 2.
Step 3: The solution 6 has the best fitness among all solutions in the second partition. During the selection operation in the second partition, two copies of solution 6 and one copy of solution 3 are expected.
Step4: Now, we move to step 5, where crossover and mutation operation will be performed on the complete population obtained after both selection operations. This completes the one generation of the VEGA

Table 2: Fitness assignment procedure under a VEGA

Solution	x_1	x_2	f_1	f_2	Partition	Assigned fitness
1	0.31	0.89	0.79	53.90	1	0.79
2	0.43	1.92	0.67	53.21	1	0.67
3	0.22	0.56	0.88	52.91	2	52.91
4	0.59	3.63	0.51	52.15	2	52.15
5	0.66	1.41	0.44	56.35	1	0.44
6	0.83	2.51	0.27	55.77	2	55.77

Advantage: The main advantage of a VEGA is that it uses a simple idea and is easy to implement. Only minor changes are required to be made in a simple GA to convert it to a multi-objective GA and this does not incur any additional computational complexity. This is a definite advantage of the VEGA over all other multi-objective GAs.
Disadvantage: As each solution in a VEGA is evaluated only with one objective function. Thus every solution is not tested for other (M-1) objective functions, all of which are also important in the context of multi-objective optimization. During a simulation run of a VEGA, it is likely that solutions near the optimum of an individual objective function would be preferred by the selection operator in a subpopulation. Such preference s takes place in parallel with other objective functions in different subpopulations.

III. MOGA

Fonseca and Fleming [9]10][11] introduced the MOGA (multi-objective genetic algorithm), which uses the non-dominated classification of GA population. The MOGA uses different way for fitness assignment, for selection it uses the stochastic universal sampling, then crossover and mutation is applied.

"In Stochastic Universal Sampling [12] one random r is chosen for the whole selection process. Since μ different solutions have to be chosen, a set of N equi-spaced number is created:

$$R = \{r, r + 1/\mu, r + 2/\mu, \dots \dots r + (\mu - 1)/\mu\} \qquad \text{mod}1$$

Thereafter, a solution corresponding to each number of R is chosen from the cumulative probability values ."
First each solution is checked for its domination in the population. To a solution i a rank equal to one plus the number of solutions that dominate the solution i is assigned, In this way, non dominated solutions are assigned a rank equal to1, since no solution would dominate a non-dominated solution in a population. A little thought reveals that in any population, there must be at least one solution with rank equal to one and the maximum rank of any population member cannot be more than N (the population size). To maintain diversity among non dominated solution Fonseca and Fleming comes with rank niche count among solutions of each rank. Where niche count [13][14] is defined as ,

$$nc_i = \sum_{j=1}^{\mu(r_i)} Sh(d_{ij})$$

The Sharing function calculation [15] is only possible for individuals having same rank, and Sharing function between i and j- individual is calculated as:

$$Sh(d_{ij}) = \begin{cases} 1-\left(\frac{d_{ij}}{\sigma_{share}}\right)^{\alpha} & \text{if } d \le \sigma_{share} \\ 0 & \text{otherwise} \end{cases}$$

d_{ij} is the distance between i^{th} and j^{th} individuals, σ_{share} is the sharing parameter and normally taken $\alpha = 1$.

$$where\ d_{ij} = \sqrt{\sum_{l=1}^{M} \left(\frac{f_i^l - f_j^l}{f_{max}^l - f_{min}^l}\right)}$$

where f_{max}^l and f_{min}^l are the maximum and minimum objective funciton value of l^{th} objective.

MOGA Fitness Assignment Procedure:

Step 1: choose a σ_{share} (dynamically updated procedure for fixing σ_{share}). Initialize $\mu(j) = 0$ for all the possible ranks $j = 1, \ldots . N$. Set the solution counter $i = 1$.

Step 2: Calculate the number of solutions (n_i) that dominates solution i. Compute the rank of the i^{th} solution as $r_i = 1 + n_i$. Increment the count for the number of solutions in rank r_i by one , i. e. $\mu(r_i) = \mu(r_i) + 1$.

Step 3: If $i < N$, *increment i by one and go to Step*1. *Otherwise, go to Step* 4.

Step 4: Identify the maximum rank r^* by checking t he largest r_i which has $\mu(r_i) > 0$. The sorting rank according and fitness averaging yields the following assignment of the avrerage fitness to any solution i $= 1,2 \ldots . N$:

$$F_i = N - \sum_{k=1}^{r_i - 1} \mu(k) - 0.5(\mu(r_i) - 1)$$

To each solution i with rank $r_i = 1$, the above equation assign a fitness equal to $F_i = N - 0.5(\mu(1) - 1)$, which is the the average value of $\mu(1)$ consecutive integers from N to $N - \mu(1) + 1$. Set a rank counter $r = 1$.

Step 5: For each solution i in the rank r, calculate the niche count nc_i with other solutions $(\mu(r)$ of them) of the same rank by using equation $nc_i = \sum_{j=1}^{\mu(r_i)} Sh(d_{ij})$. Calculate the shared fitness using $F_j' = F_j / nc_j$. To preserve same average fitness, scale the shared fitness as follows:

$$F_j' = \frac{F_j \mu(r)}{\sum_{k=1}^{\mu(r)} F_k'} F_j'.$$

Step 6: If $r < r^*$, increment r by one and go to step 5. otherwise, the process is complete.

Manual Calculation: We consider the two objectives, two variable minimization optimization problem as shown below to better illustrate the working of the MOGA. The randomly selected population is shown in Table 2. Now we show the step by step procedure of the MOGA.

Example:

$$\text{Minimize } f_1(X) = x_1,$$
$$\text{Minimize } f_2(X) = \frac{1 + x_2}{x_1},$$
$$\text{subject to } 0.1 \leq x_1 \leq 1,$$
$$0 \leq x_2 \leq 5,$$

Table 3: Six randomly chosen solution in the objective space for above maximization example.

Solution	x_1	x_2	f_1	f_2
1	0.31	0.89	0.31	6.10
2	0.43	1.92	0.43	6.79
3	0.22	0.56	0.22	7.09
4	0.59	3.63	0.59	7.85
5	0.66	1.41	0.66	3.65
6	0.83	2.51	0.83	4.23

Step 1: We choose $\sigma_{share} = 0.5$ and $\mu(j) = 0$ for all $j = 1$ to 6 (since there are $N = 6$ population members).
Step 2 and 3 : For solution 1, we find that $n_1 = 0$ and thus, $r_1 = 1 + 0 = 1$. Similiary, We find r_i for all other solutions and list them in table 4. We observe that there are three solutions with rank1 $(\text{thus } \mu(1 = 3))$. Two with rank 2 $(\mu(2) = 2)$ and one solution with rank 4 $(\mu(4) = 1)$. Therefore , thereis no solution in rank 3,5 and 6, or $\mu(3) = \mu(4) = \mu(6) = 0$.

Step 4 : the maximum rank r^* assigned to any solution in the population is 4. Sorting of the population according to rank yields the following sequence (Table 4)

$$(1,3,5)(2,6)(.)(4)$$

Now the average fitness is calculated as given below,

$$F_1 = N - 0.5(\mu(1) - 1) = 6 - 0.5(3 - 1) = 5.0,$$
$$F_2 = N - \mu(1) - 0.5(\mu(2) - 1) = 6 - 3 - 0.5(2 - 1) = 2.5,$$
$$F_3 = N - 0.5(\mu(1) - 1) = 6 - 0.5(3 - 1) = 5.0,$$
$$F_4 = N - \mu(1) - \mu(2) - 0.5(\mu(4) - 1)$$
$$= 6 - 3 - 2 - 0 - 0.5(1 - 1) = 0,$$
$$F_5 = N - 0.5(\mu(1) - 1) = 6 - 0.5(3 - 1) = 5.0,$$
$$F_6 = N - \mu(1) - 0.5(\mu(2) - 1)$$
$$= 6 - 3 - 0.5(2 - 1) = 2.5,$$

It is seen that all three solutions of the first rank has the same average fitness of 5.0.

Step 5: For three solutions of the first rank, we will now calculate the niche count in the objective function space. Assuming, $f_1^{min} = 0.1, f_1^{max} = 1, f_2^{min} = 1$ and $f_2^{max} = 10$, we have the following normalized distances:

$$d_{13} = 0.149, d_{15} = 0.475, d_{35} = 0.621$$

Using there values and $\sigma_{share} = 0.5$, we calculate sharing function values (with $\alpha = 1$) :

$$Sh(d_{13}) = 0.702, \quad Sh(d_{15}) = 0.050 \quad Sh(d_{35}) = 0.000.$$

Of course, $Sh(d_{11}) = Sh(d_{33}) = Sh(d_{55}) = 1$. Thus the niche counts for solutions 1,3, and 5 are as follows:

$$nc_1 = 1 + 0.702 + 0.050 = 1.752,$$
$$nc_2 = 1 + 0.702 + 0 = 1.702,$$
$$nc_3 = 1 + 0.050 + 0 = 1.050,$$

Now, dividing the average fitness values by the corresponding niche count , we calculate the shared fitness F_j'(column 9 in table 4).

Now, we scale these fitness values so that their average is same as original fitness value. The scaling factor is $F_1\mu(1)/(F_1' + F_3' + F_5')$ or $5 * 3/(2.854 + 2.938 + 4.762)$ or 1.421. we multiply column 9 in Table 4 by 1.421 to calculate the scaled fitness values of solutions 1,3, and 5.

Step 6: We now move to step 5 with solutions of rank 2.

Step 5: we compute $d_{26} = 0.528$. Thus, $Sh(d_{26}) = 0$ (soluion 2 and 6 have no effect on each other) and niche count $nc_1 = nc_6 = 1$. Thus, the shared fitness values are the same as the average fitness values and scaling factor is one. Table 4 shows the corresponding scaled fitness values.

Step 7: Now, we move to Step 5 again for rank 3 solutions. Since there are none ,we move to Step 5 for rank 4 solutions. Since there is only one solution in rank 4,its scaled fitness value is same as its average fitness value. Since the maximum rank is 4 in the same as its average fitness value. Since the maximum rank is 4 in the population, we stop here. This completes the fitness assignment procedure of the MOGA.

Table 4: Fitness assignment procedure of the MOGA

Solution	x_1	x_2	f_1	f_2	Rank	Average fitness	Niche count	Shared fitness	Scaled fitness
1	0.31	0.89	0.31	6.10	1	5.0	1.752	2.854	4.056
2	0.43	1.92	0.43	6.79	2	2.5	1.000	2.500	2.500
3	0.22	0.56	0.22	7.09	1	5.0	1.702	2.938	4.176
4	0.59	3.63	0.59	7.85	4	1.0	1.000	1.000	1.000
5	0.66	1.41	0.66	3.65	1	5.0	1.050	4.762	6.768
6	0.83	2.51	0.83	4.23	2	2.5	1.000	2.500	2.500

Advantage: The fitness assignment Scheme is simple in MOGA. Since niching is performed in objective space, the MOGA can be easily applied to other optimization problems, such as combinatorial problems. If a Spread of Pareto-optimal solution is required on the objective space, the MOGA is suitable choice.

Disadvantage: Although the concept of domination is used to assign fitness, all solutions in a particular non-dominated front need not have same assignment fitness. This may introduce an unwanted bias towards some solutions in the search region. In particular, this algorithm may be sensitive to the shape of the Pareto optimal front and to the density of solutions in the search space

IV. CONCLUSION

VEGA and MOGA are widely used approaches to solve multi-objective optimization problem (MOOP). MOOP have conflicting objectives , In this paper VEGA and MOGA are presented to find the solutions for MOOP , VEGA is a straightforward extension of single objective GA for multi-objective optimization, In which GA population at every generation is divided into M (say M objective to be handled) equal subpopulation randomly. Then each subpopulation is assigned a fitness based on a different objective functions, Apply the others operator of genetic algorithm. In MOGA each solution is checked for its domination, and then a rank is assigned to each solution, Solutions having same rank made the groups. Then shared fitness is calculated for every solutions group wise, followed by to calculate the Scaled fitness , Apply the others genetic algorithm operator in this way one generation of MOGA completes . We also presented the manual calculation, advantage and disadvantage of each approach.

REFERENCES

[1] Coello, C. A. C. (1999). A comprehensive survey of evolutionary-based multiobjective optimization techniques. *Knowledge and Information systems*, *1*(3), 269-308.

[2] Zitzler, E., Laumanns, M., & Bleuler, S. (2004). A tutorial on evolutionary multiobjective optimization. In *Metaheuristics for multiobjective optimisation* (pp. 3-37). Springer Berlin Heidelberg.

[3] Suman, B., & Kumar, P. (2006). A survey of simulated annealing as a tool for single and multiobjective optimization. *Journal of the operational research society*, *57*(10), 1143-1160.

[4] Corne, D. W., Knowles, J. D., & Oates, M. J. (2000, January). The Pareto envelope-based selection algorithm for multiobjective optimization. In *Parallel Problem Solving from Nature PPSN VI* (pp. 839-848). Springer Berlin Heidelberg.

[5] Fonseca, C. M., & Fleming, P. J. (1993, June). Genetic Algorithms for Multiobjective Optimization: FormulationDiscussion and Generalization. In *ICGA* (Vol. 93, pp. 416-423).

[6] Deb, K. (2012). Optimization for engineering design: Algorithms and examples. PHI Learning Pvt. Ltd..

[7] Schaffer, J. D. (1985, January). Multiple objective optimization with vector evaluated genetic algorithms. In *Proceedings of the 1st International Conference on Genetic Algorithms, Pittsburgh, PA, USA, July 1985* (pp. 93-100).

[8] Goldberg, D. E., & Deb, K. (1991). A comparative analysis of selection schemes used in genetic algorithms. *Foundations of genetic algorithms*, *1*, 69-93.

[9] Fonseca, C. M., & Fleming, P. J. (1993, June). Genetic Algorithms for Multiobjective Optimization: FormulationDiscussion and Generalization. In *ICGA* (Vol. 93, pp. 416-423).

[10] Fonseca, C. M., & Fleming, P. J. (1995). An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary computation*, *3*(1), 1-16.

[11] Fonseca, C. M., & Fleming, P. J. (1998). Multiobjective optimization and multiple constraint handling with evolutionary algorithms. I. A unified formulation. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, *28*(1), 26-37.

[12] Baker, J. E. (1987, July). Reducing bias and inefficiency in the selection algorithm. In *Proceedings of the second international conference on genetic algorithms* (pp. 14-21).

[13] Horn, J., Nafpliotis, N., & Goldberg, D. E. (1994, June). A niched Pareto genetic algorithm for multiobjective optimization. In *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on* (pp. 82-87). Ieee.

[14] Horn, J., Nafpliotis, N., & Goldberg, D. E. (1994, June). A niched Pareto genetic algorithm for multiobjective optimization. In *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on* (pp. 82-87). Ieee.

[15] Zitzler, E., & Thiele, L. (1999). Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *evolutionary computation, IEEE transactions on*, *3*(4), 257-271.

