2016 1st International Electrical Engineering Congress (IEEC 2016)
May. 13-14, 2016 in IEP Centre, Karachi, Pakistan

Performance Gaining for solving Many-objective Optimization Problems using Variation Operators

Syed Zaffar Qasim[1] and Muhammad Ali Ismail[2*]

[1,2] High Performance Computing Centre, Department of Computer & Information Systems Engineering,
NED University of Engineering and Technology,
Karachi, 75290, Pakistan
[1]zafarqas@neduet.edu.pk, [2]maismail@neduet.edu.pk

Abstract: The recent years have witnessed phenomenal growth of interest in solving many-objective optimization problems. Many-objective Optimization Problems (MaOPs) are the Multi-objective Optimization Problems (MOPs) with typically more than three objectives. For instance, the designing of an electric motor involves satisfying at least seven objectives simultaneously. These include the size or weight of the machine, material cost, maximum or average torque, torque ripple and efficiency or losses (core and copper). In general, some of these objectives are at odds with each other. Among the various reasons cited for this inefficiency is the non-suitability of conventional variation operators of intelligent meta-heuristics in generating solutions that attain the convergence goal satisfactorily. Many latest studies have asserted the need for developing new variation operators as well as using their systematic combination to augment their search performance for solving MaOPs. Accordingly the main objective of this review paper is to give short description of the various significant attempts in this particular problem solving approach along with future research directions.

Keywords: Many-objective optimization, Parameter control, Adaptive operator selection, Multi-armed Bandit.

I. INTRODUCTION

In many applications of business and engineering, multiple and often conflicting objectives need to be simultaneously optimized. For a computer engineer, the placing of more electronic components on a chip while minimizing that chip area and power loss are conflicting objectives [8]. The overall benefit usually gained with Multi-objective Optimization (MO) is the achieving of most valuable outcome for the organization as a whole along with due consideration of the requirements of all stakeholders.

MO Problems (MOPs) are in fact more challenging to solve, compared to Single-objective Optimization, because they have no unique solution; rather, there is a set of acceptable trade-off optimal solutions, called *Pareto front* (in the objective space). However the launch of intelligent meta-heuristic algorithms had made it possible to solve MOPs in a better way. Among the most significant meta-heuristic search techniques include population-based methods like Genetic Algorithms [9,10] (an Evolutionary Computation method) and Particle Swarm Optimization [10,11] (based on Swarm Intelligence). The two important goals of Multi-Objective Evolutionary Algorithms (MOEAs) are the *convergence* to the true Pareto-optimal (trade-off) solutions and a wide *diversity* among the solutions [8, 12, 13, 14].

However the performance of these algorithms is unsatisfactory in solving many-objective problems. Many-objective optimization problems (MaOPs) are the MOPs with more than three objectives. For instance, the design of an electric motor involves satisfying at least seven objectives simultaneously. These include the size or weight of the machine, material cost, maximum or average torque, torque ripple and efficiency or losses (core and copper). As another example, the Software Refactoring problem typically needs the optimization of around 15 distinct quality metrics [15]. Among the various reasons cited for this inefficiency is the non-suitability of conventional variation operators of intelligent meta-heuristics in generating solutions that achieve the convergence goal satisfactorily. The two conventional genetic variation operators on chromosomes (candidate solutions) are the *crossover* that facilitates the exchange of information between individual candidate solutions and *mutation* that introduces genetic diversity into the candidate solutions. Essentially the overall performance heavily depends on the suitability of the variation operator with respect to the solution representation, selection mechanism and the problem nature. Unquestionably, the choice of suitable variation operators and the tuning of their parameters, which is a formidable task, are extremely important [16].

Many recent studies have insinuated the need for new variation operators as well as using their systematic combination to enhance their search ability for solving MaOPs. Accordingly the main objective of this review paper is to give short description of the various attempts

made so far in this particular problem solving approach along with future research directions.

The rest of the paper is organized as follows. Section 2 elaborates on the issue of parameter control for adapting the behavior of variation operators to better suit the nature of the problem at hand. Section 3 describes the adaptive operator selection approach to dynamically select the operator showing promising performance at a particular time step. Section 4 covers some experimental results illustrating the impact of parameter variation on the performance of genetic algorithms. Conclusive remarks are finally covered in section 5.

II. ROLE OF PARAMETER CONTROL

Two major issues in using Evolutionary Algorithms (EAs), including MOEAs, are the setting of a number of control parameters and selecting genetic operators to which the algorithm performance is often very sensitive. The parameter setting is problem-specific task; people from other domains often require the consultation of EA experts for proper setting in their problems. For this reason, automatic parameter and operator configuration has been a very important and active research topic in the EA community [19].

A classification proposed in [17] classifies parameter setting into two types. One is for *parameter tuning* and the other is for *parameter control*. The former is a static approach that calculates the parameter values based on statistics gathered from several past runs that remain fixed while solving for a new instance. Parameter control methods dynamically adjust the parameter values at run-time for good performance.

These two problems are clearly related, but carry important differences. Theoretically speaking, the tuning problem is the stationary while the control problem is the non-stationary side of the same coin. From a practical perspective, tuning is indispensable for EA users, since no EA can be executed without giving some value to its parameters. There are very good parameter tuning methods developed and published in the last decade and the EC community is increasingly adopting them. For parameter control, the situation is quite different. The control problem is yet to be solved although the various advantages of control have been already identified in the past:

- It allows the EA to use appropriate parameter values in different stages of the search process. (e.g., search by big jumps in the beginning, fine tune the near-optimal solutions by small steps in the last stage of the search.)
- It allows the EA to adjust to the changing fitness landscapes when facing dynamic problems.
- It allows the EA to collect information about the fitness landscape during the search and use the accumulating information to improve performance in later stages.
- Using a parameter control mechanism also solves the tuning problem as it relieves the user from the task of choosing parameter values.

Parameter control methods can be further classified as deterministic control, adaptive control and self-adaptive control depending on the manner of adaptation.

Deterministic methods are uninformed; they follow a predetermined problem-specific schedule (or rules) for assigning new parameter values. Parameter values are predefined functions of time. *Adaptive methods* are informed as they receive feedback from the previous EA runs and assign values based upon that feedback. Parameter values are predefined functions of the whole history of the run. *Self-adaptive methods* encode parameter values in the genetic contents (in the chromosomes) along the solutions and allow them to co-evolve with the problem solutions. Note that this notion of self-adaptation is generic. It can concern any parameter and its use is not limited to denote the specific strategy used in ES to control the mutation step sizes.

While deterministic control suffers from the same problems as static control (finding the optimal pre-defined functions would require parameter tuning), self-adaptive and adaptive control have met some success over the years, but only in very specific domains, like in the framework of continuous parameter optimization.

One key objective of Parameter Control is thus to propose generic methods for the control of representation independent parameters: i) population size, ii) selection mode and parameters, iii) variation operator probabilities.

III. ADAPTIVE VARIATION OPERATOR

AOS is an increasingly popular approach to adaptively select operators for generating a new solution during the search. It calculates the application rates of

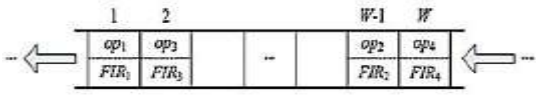

Fig. 1 Illustration of the FIFO sliding window structure

different operators in an online manner based on their recent performances within an optimization process. AOS involves two major tasks: the *credit assignment* and the *operator selection*. The former defines how to reward an operator based on its recent performance (usually based on fitness improvements) in the search process, while the latter uses these rewards to decide which operator should be applied next [18,19]. The combination of this operator selection and the credit assignment schemes constitutes the AOS method named fitness-rate-rank-based multi-armed bandit FRRMAB [19]. The incorporation of AOS in MOEAs has not been very successful so far; the main reason being the difficulty in quantitatively measuring the fitness improvement in most Pareto-dominance based MOEAs. Multiobjective evolutionary algorithm based on decomposition (MOEA/D) decomposes a MOP into a number of scalar optimization subproblems and optimizes them simultaneously. Thus, it is natural and feasible to use AOS in MOEA/D. We elaborate several important issues in using FRRMAB in MOEA/D.

A. Credit Assignment

The most commonly used metric in credit assignment for measuring the quality of an operator is based on the fitness improvement obtained by a new solution, compared with a given baseline solution. In [20], for instance, the best solution of the current population is considered baseline, while in [21] the offspring's parent is used for comparison. In credit assignment, one needs to address the following two issues:

a) how to measure the impact in the search process caused by the application of an operator;

b) how to assign an appropriate credit value to an operator based on this measured impact.

As for the first issue, the most commonly used approach is to directly use the raw values of the fitness improvements caused by the recent uses of the operator under assessment. However, the range of raw fitness improvements varies from problem to problem and even at the different stages of an optimization process. It is common that the raw fitness improvement value is much larger at early stages than at later ones. Therefore the direct use of raw fitness improvement could deteriorate the algorithm's robustness. To alleviate this problem, method proposed in [19] uses the fitness improvement rates (FIR). More specifically, the FIR achieved by an operator i at time point t is defined as

$$FIR_{i,t} = \frac{pf_{i,t} - cf_{i,t}}{pf_{i,t}}$$

Where $pf_{i,t}$ is the fitness value of the parent, and $cf_{i,t}$ is the fitness value of the offspring.

A sliding window with fixed size W is used to store the FIR values of the recently used operators. It is organized as a first-in, first-out (FIFO) queue, i.e., the FIR value of the most recently used operator is added at the tail of the sliding window, while the oldest record (the item at the head of the queue) is removed to keep the window size constant. Fig. 1 [19] illustrates the structure of a sliding window. Each slot in the sliding window stores two components:

1) the index of the operator *op* used;
2) its FIR value.

The major reason for using the sliding window is that, in dynamic AOS environments, the performance of an operator in a very early stage may be irrelevant to its current performance. The sliding window ensures that the stored FIR information is for the current situation of the search.

To address the second issue set at the outset of this subsection, we first compute $Reward_i$, the sum of all FIR values for each operator i in the current sliding window. Then, we rank all these $Reward_i$ values in a descending order. Let $Rank_i$ be the rank value of operator i, inspired by other recently proposed rank-based credit assignment schemes [22, 23], and to give more chances to the best operators, we introduce a decaying factor $D \in [0, 1]$ to transform $Reward_i$ to

$$Decay_i = D^{Ranki} \times Reward_i \quad (1)$$

Then, we assign the following credit value to operator i:

$$FRR_{i,t} = \frac{Decay_i}{\sum_{j=1}^{K} Decay_j}$$

Clearly, the smaller the value of D, the larger the influence for the best operator. Fig. 2 [19] illustrates *FRR* versus *Rank* with three different values of D in a case of 15 distinct rank values.

B. Operator Selection

Guided by the reward values, operator selection methods select operators for generating new solutions. The operator selection problem can be regarded as an instance of the Exploration vs Exploitation dilemma: one should exploit the operators-set by selecting good operators as often as possible, while also doing some exploration to give chances to poor operators since they may become better in the future search.

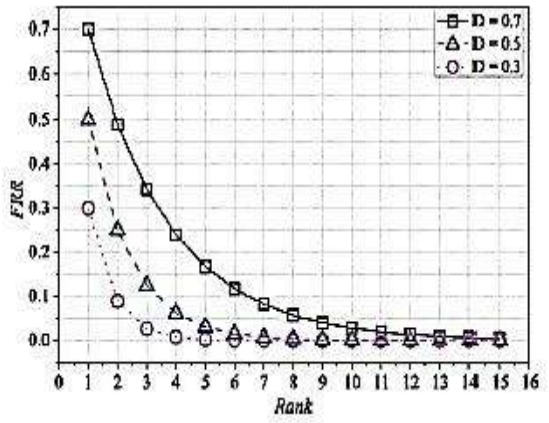

Fig. 2 Comparison between different decaying mechanisms

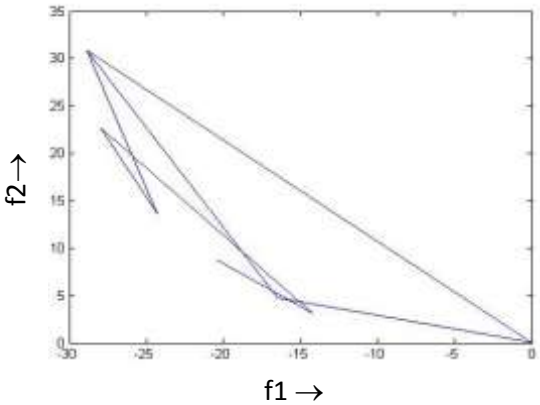

Fig. 3 Pareto-front for 2-objective problem with crossover probability pc=0.85

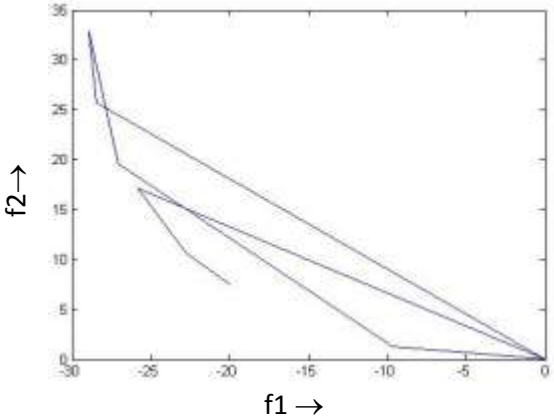

Fig. 4 Pareto-front for 2-objective problem with crossover probability pc=0.90

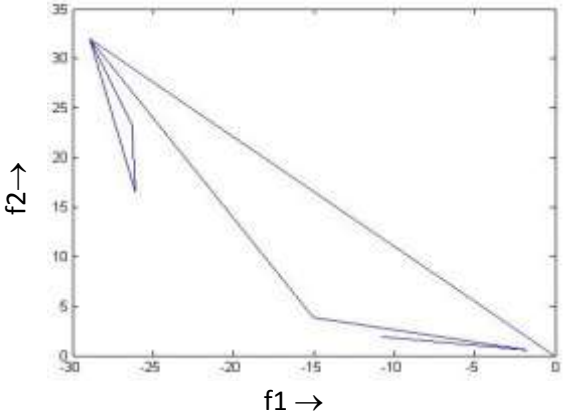

Fig. 5 Pareto-front for 2-objective problem with crossover probability pc=0.95

Bandit-based operator selection has received much attention in recent studies. A multi-armed bandit involves K independent arms (equivalent to variation operators or strategies in the AOS literature). The i-th arm is characterized by its (fixed, unknown) reward probability p_i ($p_i \in [0, 1]$). An optimal arm selection strategy is the one that maximizes the cumulative reward along time. Many MAB algorithms have been proposed to tackle this problem. Most of the recent ones are based on the UCB algorithm [24], which provides asymptotic optimality guarantees. In an UCB-based MAB algorithm, the *i*th arm has an empirical quality estimate $\hat{q}_i$ and a confidence interval that depends on the number of times n_i it has been tried before. At each time point *t*, the arm maximizing the following function is selected:

$$\hat{q}_{i,t} + C \times \sqrt{\frac{2 \times ln \sum_{j=1}^{K} n_{j,t}}{n_{i,t}}}$$

The scheme in [19] is similar to the original UCB algorithm [24]. The major difference is that we use *FRR* values as the quality index instead of the average of all the rewards received so far for an operator. In addition, n_i indicates the number of times operator *i* has been selected in the recent *W* applications. It is worth noting that no operator has yet been applied at the beginning of the search; thus, we give each operator an equal chance to be selected in this case. FRRMAB is not employed until each operator has been applied at least once.

Recent empirical studies [19] on five-objective instances of test problems (WFG1, WFG8 and WFG9) have shown that FRRMAB has improved the abilities of MOEA/D for dealing with MaOPs.

IV. EXPERIMENTAL STUDIES

The impact of parameter variation was investigated in MATLAB on a two-objective minimization problem. The two objectives on two decision variables x_1 and x_2 are defined as follows:-

$f_1(x) = x_1^4 - 10x_1^2 + x_1x_2 + x_2^4 - x_1^2x_2^2$

$f_2(x) = x_2^4 - x_1^2x_2^2 + x_1^4 + x_1x_2$

The parameter selected for variation was *crossover probability*, p_c. The pareto-fronts obtained for different values of p_c are shown below in figures 3, 4 and 5. The graphs indicate that pareto-front with p_c=0.90 is more regular with less fluctuations than the front at p_c=0.85 and p_c=0.90. However the appropriate value of p_c depends mainly on the nature of the problem.

V. CONCLUSION

This paper has emphasized the importance of parameter control and adaptive selection of variation operators for efficiently solving many-objective optimization problems. The self-adaptive and adaptive parameter controls have shown successes in recent past. Adaptive operator selection on the other hand involves credit assignment and operator selection. Numerous techniques have been proposed for both these processes in AOS. This paper studied the issues in incorporating FRRMAB in MOEA/D. In summary, the AOS approach needs further enhancement and sophistication. Furthermore it still needs to be applied to the state-of-the-art MaOPs including NSGA-III [6] (and its variants) and indicator-based MOEAs.

REFERENCES

[1] Ishibuchi, Hisao, Noritaka Tsukamoto, and Yusuke Nojima. "Evolutionary many-objective optimization: A short review." *IEEE congress on evolutionary computation*. 2008.

[2] Kukkonen, Saku, and Jouni Lampinen. "Ranking-dominance and many-objective optimization." *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*. IEEE, 2007.

[3] Deb, Kalyanmoy. "Multi-objective optimisation using evolutionary algorithms: an introduction." *Multi-objective evolutionary optimisation for product design and manufacturing*. Springer London, 2011. 3-34.

[4] Adra, Salem F., and Peter J. Fleming. "Diversity management in evolutionary many-objective optimization." *Evolutionary Computation, IEEE Transactions on* 15.2 (2011): 183-195.

[5] Rodrigo Silva et al. Visualization and Analysis of Trade-offs in Many-Objective Optimization: A Case Study on the Interior Permanent Magnet Motor Design. In: Compumag 2015, 2015, Montreal. Proceedings of the Compumag. 2015.

[6] Deb, Kaushik, and Himanshu Jain. "An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: Solving problems with box constraints." *Evolutionary Computation, IEEE Transactions on* 18.4 (2014): 577-601.

[7] Chand, Shelvin, and Markus Wagner. "Evolutionary many-objective optimization: A quick-start guide." *Surveys in Operations Research and Management Science* 20.2 (2015): 35-42.

[8] Ngatchou, Patrick, Anahita Zarei, and M. A. El-Sharkawi. "Pareto multi objective optimization." *Intelligent Systems Application to Power Systems, 2005. Proceedings of the 13th International Conference on*. IEEE, 2005.

[9] Mitchell, Melanie. *An introduction to genetic algorithms*. MIT press, 1998.

[10] Luke, Sean. "Essentials of metaheuristics. Lulu, 2009." *Available for free at http://cs. gmu. edu/sean/book/metaheuristics/. There is no corresponding record for this reference* (2011).

[11] Yang, Xin-She. *Engineering optimization: an introduction with metaheuristic applications*. John Wiley & Sons, 2010.

[12] C. A. Coello Coello, D. A. Van Veldhuizen, and G. B. Lamont, *Evolutionary Algorithms for Solving Multi-Objective Problems*. Norwell, MA: Kluwer, 2002, ISBN 0-3064-6762-3.

[13] Deb, K., 2011. Multi-Objective Optimization Using Evolutionary Algorithms: an Introduction, Department of Mechanical Engineering, KanGAL Report Number 2011003, Indian Institute of Technology Kanpur.

[14] K. Deb, Multiobjective Optimization Using Evolutionary Algorithms. Chichester, U.K.: Wiley, 2001.

[15] Mkaouer, Mohamed Wiem, et al. "High dimensional search-based software engineering: finding tradeoffs among 15 objectives for automating software refactoring using NSGA-III." *Proceedings of the 2014 conference on Genetic and evolutionary computation*. ACM, 2014.

[16] Tan, Kay Chen, et al. "Balancing exploration and exploitation with adaptive variation for evolutionary multi-objective optimization." *European Journal of Operational Research* 197.2 (2009): 701-713.

[17] Eiben, Agoston Endre, Robert Hinterding, and Zbigniew Michalewicz. "Parameter control in evolutionary algorithms." *Evolutionary Computation, IEEE Transactions on* 3.2 (1999): 124-141.

[18] DaCosta, Luis, et al. "Adaptive operator selection with dynamic multi-armed bandits." *Proceedings of the 10th annual conference on Genetic and evolutionary computation*. ACM, 2008.

[19] Li, Ke, et al. "Adaptive operator selection with bandits for a multiobjective evolutionary algorithm based on decomposition." *Evolutionary Computation, IEEE Transactions on* 18.1 (2014): 114-130.

[20] Davis, Lawrence. "Adapting operator probabilities in genetic algorithms."*proc. 3rd International conference on genetic algorithms*. 1989.

[21] Gong, Wenyin, Álvaro Fialho, and Zhihua Cai. "Adaptive strategy selection in differential evolution." *Proceedings of the 12th annual conference on Genetic and evolutionary computation*. ACM, 2010.

[22] Fialho, Álvaro, Marc Schoenauer, and Michèle Sebag. "Toward comparison-based adaptive operator selection." *Proceedings of the 12th annual conference on Genetic and evolutionary computation*. ACM, 2010.

[23] Fialho, Alvaro, et al. "Comparison-based adaptive strategy selection with bandits in differential evolution." *Parallel Problem Solving from Nature, PPSN XI*. Springer Berlin Heidelberg, 2010. 194-203.

[24] Auer, Peter, Nicolo Cesa-Bianchi, and Paul Fischer. "Finite-time analysis of the multiarmed bandit problem." *Machine learning* 47.2-3 (2002): 235-256.

[25] Zitzler, Eckart, and Simon Künzli. "Indicator-based selection in multiobjective search." *Parallel Problem Solving from Nature-PPSN VIII*. Springer Berlin Heidelberg, 2004.