
Article:
Dubey, Rahul orcid.org/0000-0003-1524-7797, Hickinbotham, Simon John orcid.org/0000-0003-0880-4460, Price, Mark et al. (1 more author) (2023) Local Fitness Landscape Exploration Based Genetic Algorithms. IEEE Access. pp. 3324-3337. ISSN 2169-3536

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier xx.xxxx/ACCESS.xxxx.DOI

Local Fitness Landscape Exploration Based Genetic Algorithms

RAHUL DUBEY[1], SIMON HICKINBOTHAM[1] MARK PRICE[2], ANDY TYRRELL[1](Life Member, IEEE)

[1]Department of Electronics Engineering, University of York, UK (e-mail: rahul.dubey@york.ac.uk, simon.hickinbotham@york.ac.uk, andy.tyrrell@york.ac.uk)
[2]School of Mechanical and Aerospace Engineering, Queen's University Belfast, N. Ireland, UK (e-mail: M.Price@qub.ac.uk)

Corresponding author: Rahul Dubey (e-mail: rahul.dubey@york.ac.uk).

This work is supported by EPSRC Programme Grant (EP/V007335/1).

ABSTRACT Genetic algorithms (GAs) have been used to evolve optimal/sub-optimal solutions of many problems. When using GAs for evolving solutions, often fitness evaluation is the most computationally expensive, and this discourages researchers from applying GAs for computationally challenging problems. This paper presents an approach for generating offspring based on a local fitness landscape exploration to increase the speed of the search for optimal/sub-optimal solutions and to evolve better fitness solutions. The proposed algorithm, "Fitness Landscape Exploration based Genetic Algorithm" (FLEX-GA) can be applied to single and multi-objective optimization problems. Experiments were conducted on several single and multi-objective benchmark problems with and without constraints. The performance of the FLEX-based algorithm on single-objective problems is compared with a canonical GA and other algorithms. For multi-objective benchmark problems, the comparison is made with NSGA-II, and other multi-objective optimization algorithms. Lastly, Pareto solutions are evolved on eight real-world multi-objective optimization problems, and a comparative performance is presented with NSGA-II. Experimental results show that using FLEX on most of the single and multi-objective problems, the speed of the search improves up to 50% and the quality of solutions also improves. These results provide sufficient evidence of the applicability of fitness landscape approximation-based algorithms for solving real-world optimization problems.

INDEX TERMS genetic algorithms, fitness landscape approximation, multi-objective optimization, evolutionary search

I. INTRODUCTION

Genetic algorithms (GAs) are population-based optimization techniques that have been successfully used to tune parameters to maximize or minimize the fitness of non-linear problems [1]–[3]. Population-based algorithms typically begin with randomly generated candidate solutions and use selection and recombination operators to generate offspring for the next generation. These reproduction operators do not utilize the local fitness gradient to produce the offspring [4], and thus do not exploit any potential advantages from the fitness landscape (FL) that could be used to generate or select offspring with better fitness characteristics. The complete fitness landscape of a problem can be obtained by mapping all possible genotypes/solutions to their respective fitnesses. This approach is similar to "brute-force search" in that it gives a complete picture of the mapping but is computationally expensive. Often for real-world complex problems, the process of fitness evaluation of a genotype is computationally expensive, and this discourages researchers from applying GAs with fitness landscape analysis capabilities to computationally challenging problems. Each solution created by a genetic algorithm has information (a single position on the FL), and the availability of this information for a population of individuals could be used to model the local FL more efficiently. This contribution is motivated by the idea that the speed or quality of search for optimal/sub-optimal solutions could be improved by appropriate modelling of the local fitness landscape around each generation in an evolutionary run.

In this paper, a generic "Fitness Landscape Exploration" (FLEX) based genetic algorithm is presented that uses information from the local fitness landscape of a given problem. The aim is to generate some of the offspring of the next generation via FLEX alongside those generated via the con-

ventional genetic algorithm methods, but without increasing the overall time complexity of the GA. For local fitness landscape analysis, a **genome vector** is created and a **fitness vector** is computed corresponding to the genome vector, and these are used to approximate the local fitness landscape.

A genome vector is an n-dimensional vector, where n is the input dimension of the problem, obtained using two neighboring candidate solutions. A fitness vector is a m-dimensional vector computed by taking the difference of the fitnesses of the two candidate solutions, where m is the number of objectives. Since the genome and fitness vectors can be computed for any m-objective problem, this approach is applicable to single and multi-objective optimization problems. However, this paper only considers one, and two-objective problems. To approximate the local fitness landscape with sufficient accuracy, the distance between two solutions in the search/design space must be small [5]. Thus, solutions using FLEX should be encoded in real parameters (not binary) so that simulated binary crossover (SBX) can be used to exchange information between two selected candidate solutions. SBX allows the euclidean distance between parents and the newly generated offspring to be controlled using the spread factor as described in the NSGA-II algorithm [6].

Novel genetic algorithms using the FLEX concept can be derived for single and multi-objective problems. In order to verify the hypothesised performance improvements, the proposed FLEX-based genetic algorithms are compared with several other algorithms for both single and multi-objective problems. For single-objective problems, a simple GA is the base algorithm and is augmented using fitness landscape exploration to derive the proposed FLEX-GA. The performance of FLEX-GA is compared with a canonical GA, Differential Evolution(DE) [7], Particle Swarm Optimization (PSO) [8], and Evolutionary Strategies (ES) [9].

In the case of multi-objective problems, NSGA-II [6] is the base algorithm, and the version proposed in this paper is FLEX-NSGA-II. Additionally, the performance of the proposed FLEX-based approach with the baseline NSGA-II, multi-objective evolutionary algorithm based on decomposition (MOEA/D) [10], Strength Pareto Evolutionary Algorithm-II (SPEA-II) [11], and Adaptive Geometry Estimation based MOEA (AGE-MOEA) [12] are compared on benchmark and real-world problems. Experiments were conducted on 10 single-objective constrained optimization problems from the IEEE CEC-2006 competition [13], 5 two-objective test benchmark problems [6], and on eight real-world two-objective problems [14]. Results indicate that FLEX-GA generally takes fewer functional evaluations to find better solutions than other algorithms on more than half of the problems, and is inferior only in a few. These results demonstrate the effectiveness of augmenting the capabilities of existing GAs with fitness landscape exploration.

The three main contributions of this paper are as follows: 1) a GA that incorporates the linear approximation of local fitness landscape, called “FLEX-GA" is introduced, which enhances the speed of search for solutions and evolves better quality solutions compared to standard GAs, 2) it is shown how the proposed algorithm generalizes and is applicable to single and multi-objective optimization problems, and 3) The proposed FLEX approach does not increase the time complexity of the base algorithm.

The remainder of this paper is organized as follows. Section II describes prior work in fitness landscape analysis-based evolutionary algorithms. Section III describes the details of FLEX and local fitness landscape approximation. Section IV presents the details of FLEX-GA and also compares the results obtained using FLEX-GA, and other algorithms on single-objective benchmark problems. Similarly, section V presents FLEX-NSGA-II and compares the performance of different algorithms on two-objective benchmark problems. Finally, the last section VI provides conclusions and discusses future work.

II. RELATED WORK

The concept of the fitness landscape were studied by Wright [15] to emphasize the dynamics of evolutionary optimization. FL analysis can provide useful insights on a given problem [16] and is defined by a search space, objective function, and neighborhood operators [17]. Mathematically, an FL (S, f, d) of a problem is composed of a set of samples/points S, the fitness f assigned to each the samples, and a distance d between samples, which together define the spatial structure of the landscape [18]. In this paper, the values of S, f and d are used to estimate the local fitness landscape, but the samples S are taken from the current population. A number of research articles have been published that discuss how FL information can be utilized to improve the performance of algorithms, reviewed in [5].

The FL analysis presented by Ochoa [19] highlights the motivation for using FL analysis with case studies. An FL may contain valleys, peaks, ridges, plateaus, and landscapes that could be combinations of smooth and rugged regions. This information, if available, can be utilized to enhance the speed of the search. Earlier, in 1998, Ratle [20] studied how to accelerate the convergence of evolutionary algorithms by fitness landscape approximation. The author in that paper used a statistical model, Kriging interpolation, to approximate the fitness landscape using samples/ data points from the first generation and used this approximated FL model for evaluating individuals for the next few generations before updating the model again. The author created a surrogate model to approximate the fitness evaluator (which is computationally expensive) to compute the fitness of individuals. Note that it is difficult to approximate the entire FL using a small number of samples generated during evolution [5] and thus the approximated model does not best represent the actual fitness evaluator. FLEX-GA shares the same motivation of accelerating the convergence of evolutionary algorithms but without using surrogate models.

Recent trends [21] and a survey [5] show the effectiveness of different local/global fitness landscape approximation techniques and their usefulness in different domain-specific

problems. For evolutionary search enhancement, Yan [22] presented a method based on dimensionality reduction for fitness approximation and used a Fourier transform to obtain frequency information of the fitness landscape to drive the search acceleration. Huang [23] presented a self-feedback strategy for differential evolution using the FL to improve the performance of the algorithm. Takagi [24] proposed a method of estimating the convergence point using the solutions of different generations and used this convergence point to accelerate the search process. Yang [25] presented a genetic algorithm back-propagation neural network algorithm that uses the FL to improve solutions. They use the fitness landscape analysis to optimize the learning rate of the back-propagation algorithm, and the GA evolves a population of solutions.

Cheng [26] used FL approximation and local search methods for finding solutions to multi-objective multi-modal problems. For multi-modal problems, *niching* approaches have been used to preserve diverse solutions. Authors [26] show how a multi-objective multi-modal problem can be recast as a uni-model multi-objective problem, and using FL approximation, all optimal solutions/peaks can be obtained in a single run. Tan [27], [28] presented an adaptive mutation strategy based on FL analysis for differential evolution. These approaches gather information about the FL during the evolutionary search and use them in subsequent generations. When solving an optimization problem, FL analysis can be used to take advantage of the underlying structure of a given problem where the structure is defined by the search space, objective function, and neighborhood operator. Traore [29] presented the fitness landscape footprint, a framework (tool) to compare search problems. Using the tool, the degree of difficulty of a network architecture based search of the fitness landscape is evaluated, with the aim of finding a better architecture in fewer training cycles.

Techniques based on fitness landscape analysis have been used not only for parameter tuning but also for traveling salesman problems (TSP) and to understand different machine learning problems [30]. Yafrani [31] presented FL analysis of TSP using Local Optima Networks (LONs) (originally presented in [32]). Matheus [33] analyzed the fitness landscape to characterize the search space explored by neural architecture search methods for graph neural networks. In recent years, evolutionary algorithms have been used as a tool for Neural Architecture Search (NAS) by encoding convolutional/recurrent neural network (CNN/RNN) architectures in genomes [34], [35], in evolving strategies to win intense war simulation games [36], physical simulation [37], manufacturing process [38], and other expensive-to-evaluate functions [39]. Since all of these problems are computationally expensive, they require a fast search algorithm that can find optimal or near-optimal solutions through fewer functional evaluations. Both FLEX-GA and FLEX-NSGA-II have potential applications in these domain-specific problems.

In this paper, canonical genetic algorithms are modified without increasing the time complexity of the algorithms. FL information is leveraged to generate new offspring to increase the speed of the search and/or to find optimal/near-optimal solutions to encourage researchers to apply GAs to more computationally expensive problems.

Algorithm 1: FLEX based Genetic Algorithms

Input : $Pop, Gen, P_x, P_m, \lambda$
$P_0 \leftarrow Initialize(Pop)$
$Evaluate(P_0)$
for *t in Gen* **do**
  P_c = []
  for *i in* λ **do**
    $p_1, p_2, c_1, c_2 = Reproduction(P_t)$
    $Evaluate(c_1, c_2)$
    $P_c.add(c_1, c_2)$
    $i = i + 2$
    $c_n = LFLA([p_1, p_2], [c_1, c_2])$
    for *j in* $len(c_n)$ **do**
      if $i < \lambda$ **then**
        $Evaluate(c_{nj})$
        $P_c.add(c_{nj})$
        $i = i + 1$
      end
    end
  end
  $P_{t+1} \leftarrow NextGenIndividuals(P_t, P_c)$
end

III. METHODOLOGY

Genetic algorithms are popular stochastic, gradient-free evolutionary algorithms, that use selection, crossover, and mutation operators to randomly generate new offspring. GAs have been used extensively to evolve solutions for poorly understood non-linear problems [1]. In this paper, a $\mu + \lambda$ elitist GA is used to preserve good solutions found during evolution. Readers are advised to use [40] and [6] as reference algorithms for single and multi-objective optimization respectively.

A. FLEX BASED GENETIC ALGORITHMS

Algorithm 1 shows the general form of fitness landscape examination based genetic algorithms where Pop is the population size, Gen is the maximum number of iterations for evolution, P_x, P_m are probabilities of crossover and mutation. The algorithm generates new offspring using tournament selection, SBX crossover, polynomial mutation, and the proposed local fitness landscape approximation (LFLA) approach until λ offspring are generated in a generation. Here, λ is the population size.

An FL may contain plateaus, peaks, valleys, and provides information about both local and global optima. When the fitness of neighbouring genomes are the same, they create a plateau region, and this does not help the GA's crossover and mutation operators to evolve fitter solutions. However, for a minimization problem, peaks (low fitness solutions)

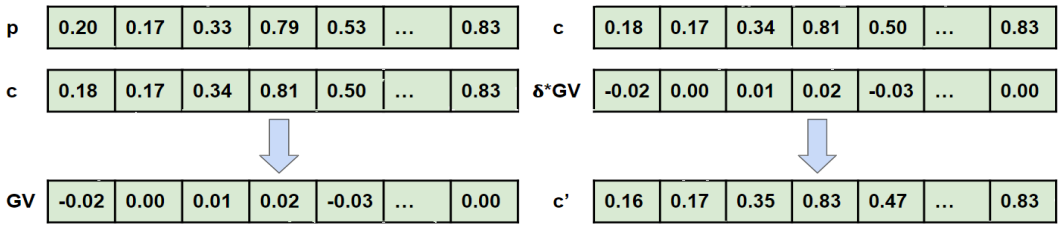


FIGURE 1: An example of genome vector (GV) generated using a parent and a child genome. Here δ is one.

and valleys (high fitness solutions) can provide gradient information that can be used to determine better fitness solutions in the search space [19]. These peaks and valleys generally lie on non-linear surfaces, and due to these non-linearities, it is difficult to predict the fitness of neighbouring solutions/genomes reliably.

The FLEX based genetic algorithms presented herein use a local fitness landscape approximation (LFLA) as shown in Algorithm 2. In LFLA, a pair of parent (p) and a pair of child (c) solutions are used to compute a genome vector locally, a fitness vector/gradient, and LFLA uses these vectors to generate new offspring. However, before using local fitness landscape approximations a few questions must be kept in mind, 1) since the fitness landscape is often highly non-linear and deceptive, following a local vector and fitness vector/gradient may lead the search to a non-desirable region of the search space, 2) following a local vector and fitness gradient, solutions may get stuck into local minima, and it would be difficult to recover or come out of sub-optimal valley/peak, and 3) the methodology must consider how local FL approximation can be used to generate offspring for multi-objective problems where for a genome vector and multiple fitness gradients exist. This paper attempts to answer these questions.

B. LOCAL FITNESS LANDSCAPE APPROXIMATION

Algorithm 2 presents the proposed local fitness landscape approximation based technique for generating offspring. The algorithm takes a pair of parent (p_1, p_2) and a pair of child (c_1, c_2) genomes and generates new offspring (C_n) if the conditions of the local fitness landscape approximation satisfy.

$$c_1, c_2 = \frac{p_1 + p_2}{2} \pm \frac{1}{2}\beta(|p_2 - p_1|) \tag{1}$$

Equation 1 shows the relationship between p_1, p_2 and c_1, c_2, where p_1, p_2 are the two parents selected from the population of genomes/individuals using a binary tournament selection, and c_1, c_2 are the two children generated using SBX crossover. In this equation, β is the spread factor which is the ratio of the spread of child points to that of the parent points.

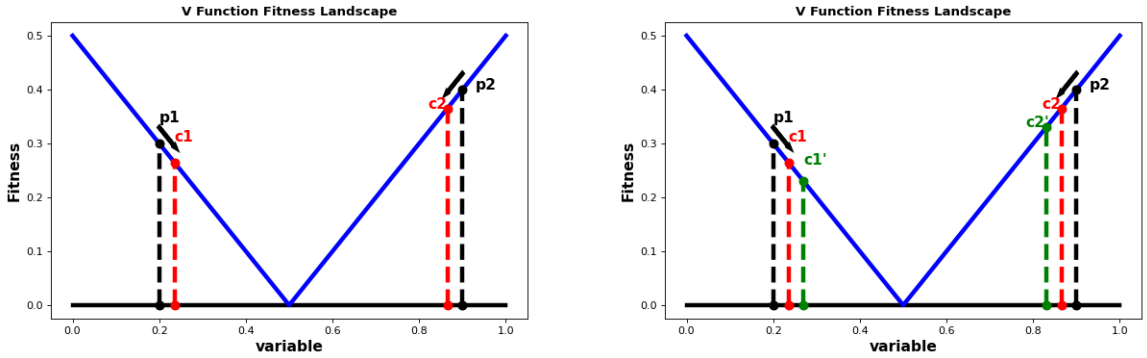
In this context, the spread of points refers to the euclidean distance between two points. When $\beta = 1$, the spread of p_1, p_2 and c_1, c_2 are the same whereas for $\beta > 1$ the distance between the two child genomes is greater compared to the two parent genomes and vice-versa for $\beta < 1$. Thus, the spread factor can be varied to generate two children that are close to the selected parents in the search/design space, allowing the distance between parents and children to be controlled. Such solutions are desirable as it allows a linear relationship between search/design and fitness landscape (linear mapping). This type of linear mapping has been studied for local decision-making in machine learning algorithms [41] [42]. To find a local fitness landscape approximation, two terms, namely genome vector, and fitness vector are defined. In the next two subsections, these two terms are explained.

Algorithm 2: Local Fitness Landscape Approximation (LFLA)

Input : p, c
$C_n = empty\ list$
for *i in len(p)* **do**
  $idx = Closest\ Individual(c)$
  $d = getDist(p_i, c_{idx})$
  $GV = (c_{idx} - p_i)$
  $FV = c_{idx}.fit - p_i.fit$
  $Gradients = \Delta(FV)$
  if $d > d_{th}$ *and* $Gradients < 0$ **then**
    $c'_i = c_{idx} + \delta GV$
    $C_n.append(c'_i)$
  end
end
return C_n

1) Genome Vector

A genome vector (GV) is created by subtracting each gene of two genomes as shown by Figure 1(a) where a $GV(c-p)$ is created using two n dimensional genomes p and c. The genome vector is then used to create a new offspring (c') according to equation 2 and shown by Figure 1(b). Here, $\delta = 1$ guarantees that the distance $d(p, c)$ is the same as distance $d(c, c')$. Remember that the distance between p, c is controlled by the spread factor β, and in order to make linear approximation (locally) $d(p, c)$ must be small. Thus, $\beta \approx 1$



(a) Genome Vectors (black arrows)

(b) New Child (green points)

FIGURE 2: Fitness landscape of V-function. Here, black points show the location of the two parents, red points represent the location of the two children generated using crossover and mutation operator, and green points represent the two children generated using genome vector and fitness gradient. The back arrows show the genome vectors.

is recommended.

$$c' = c + \delta \vec{GV} \tag{2}$$

In the equation 2, GV is a vector, δ is a scalar, and thus the distance between each gene of p, c and c, c' is the same. However, varying the δ for each gene may guide the evolutionary search to more promising regions in the search space. A study of the effect of varying δ for each gene is for future work. The next subsection explains the fitness vector, gradient, and how the fitness gradient decides whether to generate a new offspring using a genome vector.

2) Fitness Vector and Fitness Gradient

Similar to the GV, a fitness vector (FV) is computed by subtracting two neighbouring genomes' fitness ($f_c - f_p$). Equation 3 provides a fitness vector computed using $p's$ and $c's$ 2-dimensional fitness, and Equation 4 shows the gradient of fitness with respect to each objective.

$$\begin{aligned} f_p &= \langle f_{p1}, f_{p2} \rangle \\ f_c &= \langle f_{c1}, f_{c2} \rangle \\ FV &= \langle f_1, f_2 \rangle = \langle f_{c1} - f_{p1}, f_{c2} - f_{p2} \rangle \end{aligned} \tag{3}$$

$$\begin{aligned} \frac{\partial}{\partial f_1} FV &= \langle f_{c1} - f_{p1} \rangle \\ \frac{\partial}{\partial f_2} FV &= \langle f_{c2} - f_{p2} \rangle \end{aligned} \tag{4}$$

In the equation 3, f_p and f_c are two-dimensional objectives, and an FV computed using f_p and f_c provides a direction of evolutionary search movement in the fitness space. For minimization problems, FV with a negative fitness gradient is of importance, and vice-versa for maximization problems. A negative fitness gradient of FV with respect to an objective indicates that the quality of solution improved for that objective as shown by equation 4.

3) Generating New Offspring

Using the GV, FV, and gradient Algorithm 2 generates new offspring. To avoid the generation of an offspring c' too close to a child c, the magnitude of GV must be larger than a threshold distance (d_{th}) of 10^{-10} [6]. For each parent, Algorithm 2 finds the closest child (c_{idx}) and then compares their fitness. If the fitness of the child is smaller than the parent, then this provides an indication that moving along a vector (GV) in the design space (search space) from parent to child results in the reduction of the fitness (thus has negative fitness gradient). By assuming a local linear mapping between the fitness of neighboring genomes, the algorithm generates a new child ($c_i^{'}$) from the location of c_{idx} in the direction of the genome vector using equation 2. To compare FLEX based genetic algorithms with existing GAs, the number of children generated in each generation is kept fixed to λ.

IV. SINGLE-OBJECTIVE FLEX

In this section, the FLEX based approach for single-objective problems is illustrated. Since the mapping from genotype to fitness is unknown, a genome vector that leads to a negative fitness gradient may be difficult to identify. Thus the genome vector must be carefully chosen. For single-objective problems, a simple GA is the base algorithm, and the proposed algorithm is FLEX-GA.

A. FLEX-GA

In this section, a simple one-dimensional minimization problem is used to illustrate how local fitness landscape approximation works within FLEX-GA. This takes the form of a V-function $f(x) = |x - 0.5|$, as shown in Figure 2, where variable $x \in (0, 1)$ with the global minimum of $f(x) = 0$ at $x = 0.5$. Through tournament selection, two parents p_1, p_2 are selected for crossover with $\beta = 0.90$. Assuming that $p_1 = 0.2$, $p_2 = 0.9$, the two children will be generated at $c_1 = 0.235$ and $c_2 = 0.865$ using equation 1. For simplicity, the mutation is not considered here. The fitness of $f(p_1, p_2, c_1, c_2)$ are $(0.3, 0.4, 0.265, 0.365)$ as shown in Figure 2(a) where two black dots show the locations of the two

TABLE 1: Genome vector and fitness gradient computed using two parents and two children

	Genome Vector $(c-p)$		*Fitness Gradient*	
	c_1	c_2	c_1	c_2
p_1	$+0.035$	$+0.665$	-0.035	$+0.065$
p_2	-0.665	-0.035	-0.135	-0.035

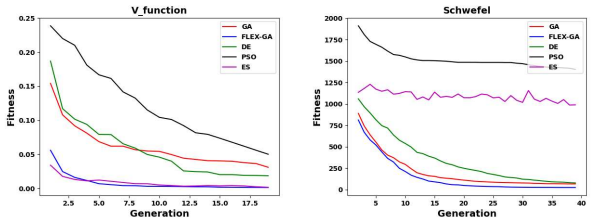
parents, two red dots show the locations of the two children, and two black arrows represent genome vectors. Equation 2 computes GVs, equation 4 computes fitness gradients, and Table 1 shows these values obtained using p_1, p_2, c_1, and c_2. For this example, a spread factor ($\beta \approx 1$) $\in \{0, 1\}$ is selected as the global optimal solution lies in the middle of the search space. For other problems, it's recommended to use $\beta = 1 \pm \gamma$, where γ is a small value.

As shown in Table 1, negative fitness gradients are obtained while moving along vectors (p_1, c_1), (p_2, c_2), and (p_2, c_1). Since the mapping between genotype and fitness is generally not known *a priori*, finding a vector from two points in the design space that are close to each other provides an indication about fitness mapping locally. For example, a GV $(c_1 - p_1) = 0.0\vec{3}5x$ reduces the fitness by 0.035, thus assuming a linear genotype-fitness mapping, a new child can be computed using equation 2. A new child c_1' generated from c_1 in the direction of GV is at 0.30 where $\delta = 1$ (moving one vector length). These small incremental steps ensure that a search does not lead, abruptly, to non-desirable regions of the search space. Using the same concept, a child c_2' can be generated from c_2 at 0.83 following a vector $GV = (c_2 - p_2)$. However, following a vector that reduces fitness might be deceptive and can generate a bad solution. For instance, a vector from p_2 to c_1 indicates that there will be a reduction in fitness, but a child generated using a $GV(c_1 - p_2)$ will be at point -0.43 which lies outside the search space. Thus, Algorithm 2 generates a child using a vector whose magnitude is smaller than the other vectors. However, this does not guarantee that the search will not diverge, rather the divergence will be slow and bad solutions will be eliminated using elitist selection.

As mentioned earlier, the genome vector sometimes can be misleading and a search could get stuck in a local minimum if it was used instead of conventional reproduction operators. To minimize the effect of excessive gradient-based search, as a maximum, only half of the offspring are generated using local genome vectors and fitness gradients, the remaining are generated using the three genetic operators. In future work, limiting the number of offspring generated using FLEX to less than 50% will be studied. In the next subsection, results on several single-objective benchmark problems are presented and a comparison is made with other algorithms.

B. SINGLE-OBJECTIVE RESULTS AND DISCUSSIONS

In this section, the results obtained from using FLEX-GA on several single-objective benchmark problems with and without constraints are presented. Note that since the proposed advantage of FLEX is to increase the speed of search for optimal/near-optimal solutions, the different algorithms are compared in terms of 1) the number of evaluations used for searching solutions (convergence), and 2) the quality of the evolved solution (fitness).



(a) V-function (b) Schwefel

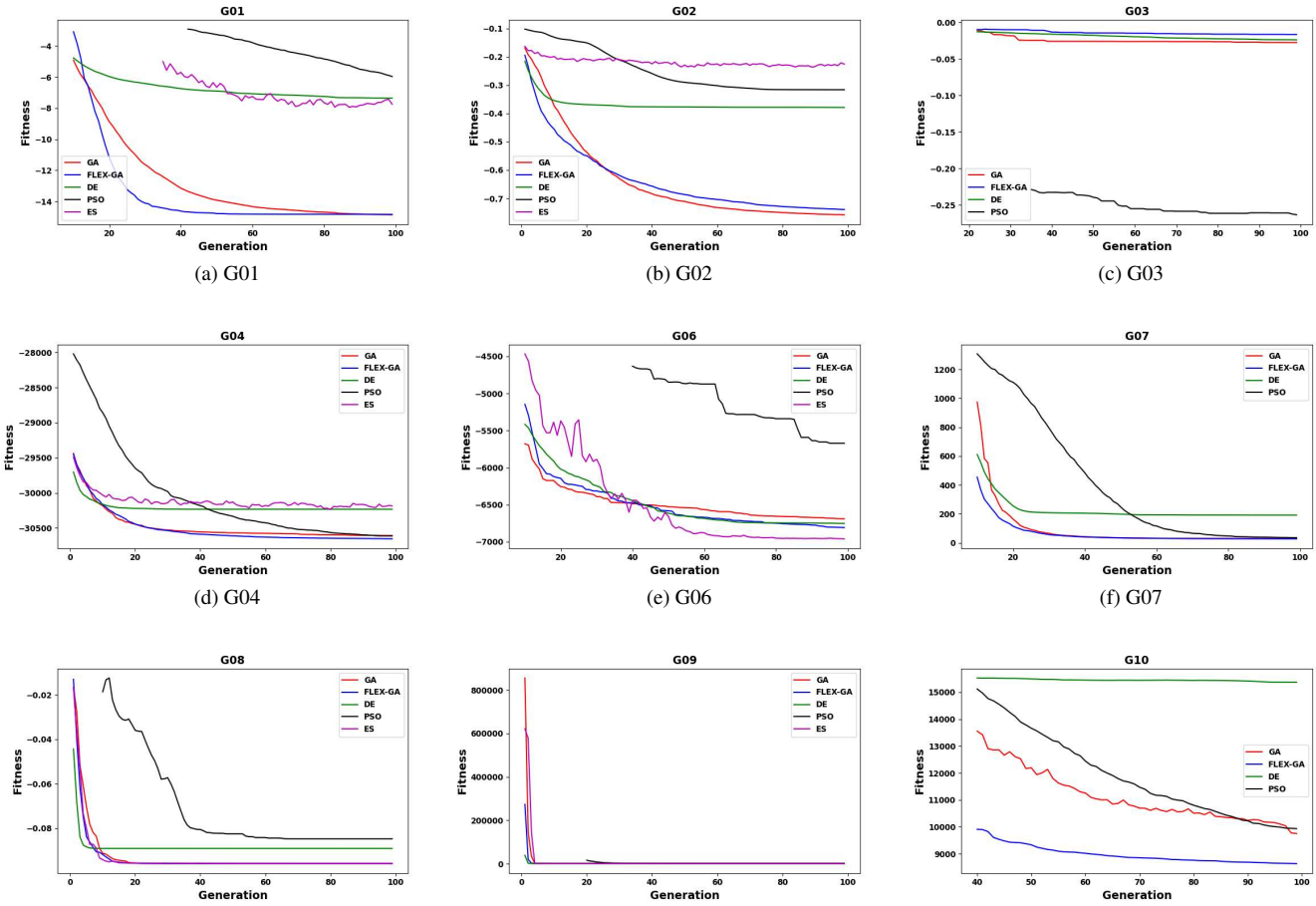
FIGURE 3: Comparing the average of the best fitness over 30 runs of each generation on two single-objective test problems obtained using different algorithms.

1) Experiment Parameters and Simple Test Problems

In the previous section, a simple single-objective V-function was used to explain local fitness landscape examination/approximation to generate new offspring. On this simple problem, solutions are evolved with a small population size of 20 for 20 iterations/generations. Figure 3(a) shows the average of minimum fitness, over 30 runs, obtained in each generation using FLEX-GA, GA, DE, PSO, and ES. The figure shows that FLEX-GA quickly converges on the optimal solution compared to GA, DE, PSO, and comparable to ES. The fitness landscape of this type of problem is ideal for FLEX where it can leverage local fitness landscape information to produce better-quality offspring. To test the performance of the proposed algorithm on problems with more complex fitness landscape, another test problem was chosen, Schwefel [43]. The Schwefel function has many local minima and thus is a good test problem to see how FLEX based approach performs. Figure 3(b) shows the average of the best fitness of each generation over 30 runs on Schwefel, and it again clearly shows that FLEX-GA convergence is faster. FLEX-GA found better quality solutions in fewer evaluations compared to the other algorithms. These two simple test experiments provide evidence that when leveraging fitness landscape information, the speed of the search improves (against the base GA), and thus the computational cost/budget reduces.

2) CEC Benchmark Problems

Following the positive results on the proof-of-concept fitness landscapes, the FLEX-GA algorithm's performance was next tested on 10 IEEE CEC-2006 single-objective benchmark problems (G01-G10) [13]. The CEC-06 competition provides highly constrained single-objective optimization benchmark functions and has been used extensively [13]. Experiments were conducted 30 times with different random initialization



(a) G01 (b) G02 (c) G03

(d) G04 (e) G06 (f) G07

(g) G08 (h) G09 (i) G10

FIGURE 4: Comparing the average of the best fitness obtained in each generation over 30 runs on G01-G10 (except G05) obtained using different algorithms. (On problem G05 feasible solutions are not found using any algorithm).

on each problem with a population size of 100 for 100 generations. (The population size and number of iterations were varied, and a similar performance pattern has been observed.) To compare the performance in terms of the number of evaluations required to find solutions, the minimum fitness of each generation plotted, averaged over 30 runs, in Figure 4.

Figure 4 shows the convergence of average fitness on all problems (except G05). Since all 10 problems are constrained optimization problems, the figure only shows the average of the best feasible solution of each generation. In the early stage of these runs, evolved solutions are usually infeasible, so for some problems the fitness of initial generations are not presented. In all figures, the blue line represents the fitness curve of FLEX-GA, the red line for GA, green for DE, black for PSO, and magenta for ES. On G01, Figure 4(a) shows that the fitness obtained in the 50^{th} generation with FLEX-GA is equal to the fitness obtained on and after the 80^{th} generation using the GA, which leads to a saving of more than 30% of computational cost/budget. Similarly, on G04, the performance of FLEX-GA and GA are comparable until 30^{th} generation, and thereafter FLEX-GA's performance is better than GA. Figure 4(d) shows that the average GA's fitness of the 100^{th} generation is similar to the average FLEX-GA's fitness of the 50^{th} generation. On this problem, FLEX-GA's search for optimal/near-optimal solution is twice as fast compared to than canonical GA's. The figure also shows that the convergence of FLEX-GA is much faster than PSO whereas the quality of solution is better compared to DE and ES. A similar performance pattern is observed on G07 and G10 where FLEX-GA's convergence is better (faster) than others leading to saving of computational budget.

On G08, FLEX-GA's, GA's, and ES's convergence are comparable but better than DE and PSO whereas, on G09, each algorithm converges quickly. On G02, GA's performance is better than others and on G06, ES performed the best. On G03, PSO's performance is slightly better than others, however far from optimal. G03 is a polynomial equation with an equality constraint and none of the algorithms achieve near-optimal solution. In depth analysis suggests that on G03, genome vectors and fitness gradients lead the search towards infeasible regions of the search space and thus performance is worse than the base GA. Similarly, G05

TABLE 2: Comparing the best, worst, and median fitness obtained using different algorithm over 30 runs

Problems		FLEX-GA	GA	DE	PSO	ES
G01 (-15)	Best	**-14.99**	-14.96	-11.25	-9.799	-9.990
	Worst	-12.44	**-14.56**	-4.482	-2.868	-5.694
	Median	**-14.99**	-14.80	-7.159	-6.113	-7.748
G02 (-0.8036)	Best	-0.786	**-0.791**	-0.519	-0.384	-0.275
	Worst	-0.629	**-0.706**	-0.269	-0.132	-0.195
	Median	-0.740	**-0.761**	-0.345	-0.325	-0.216
G03 (-1.0)	Best	-0.085	-0.184	-0.195	**-0.640**	NA
	Worst	0.0	0.0	0.0	0.0	NA
	Median	-0.007	-0.014	-0.009	**-0.251**	NA
G04 (-30665)	Best	**-30665**	-30663	-30633	**-30665**	-30378
	Worst	**-30614**	-30475	-29599	-30242	-29902
	Median	**-30660**	-30609	-30276	-30639	-30194
G06 (-6961)	Best	-6955	-6935	**-6961**	-6932	**-6961**
	Worst	-6584	-6391	-703.6	-978.2	**-6951**
	Median	-6823	-6720	**-6961**	-5985	6959
G07 (24.30)	Best	**24.30**	25.37	32.03	27.35	NA
	Worst	**34.61**	39.02	2151	57.37	NA
	Median	**27.18**	27.60	66.17	32.90	NA
G08 (-0.0958)	Best	**-0.0958**	**-0.0958**	**-0.0958**	**-0.0958**	**-0.0958**
	Worst	**-0.0958**	**-0.0958**	-0.0258	-0.0291	**-0.0958**
	Median	**-0.0958**	**-0.0958**	**-0.0958**	**-0.0958**	**-0.0958**
G09 (680)	Best	**684.01**	684.35	684.63	684.51	703.10
	Worst	**688.73**	694.74	1010.2	700.49	902.65
	Median	**684.61**	685.58	705.93	685.43	757.80
G10 (7049)	Best	**7436.3**	7694.2	7634.1	7533.7	NA
	Worst	**11438**	18207	28610	14544	NA
	Median	**8408.9**	8484.9	14302	19321	NA

TABLE 3: Ranking different algorithms in terms of the speed of the search and quality of the evolved solutions

Algorithms	Problems								
	G1	G2	G3	G4	G6	G7	G8	G9	G10
FLEX-GA	I	II	IV	I	III	I	I	I	I
GA	II	I	II	II	IV	II	I	II	II
DE	IV	III	III	IV	II	IV	II	IV	IV
PSO	V	IV	I	III	V	III	III	III	III
ES	III	V	NA	V	I	NA	I	V	NA

has three equality constraints, and no feasible solutions were found using any of the five algorithms and thus not shown in Figure 4. Additionally, it has been found that FLEX-GA is inferior to its base algorithm on problems with equality constraints. These figures show that on most problems either FLEX-GA found better fitness solutions compared to others, or if the quality of solutions are the same, then FLEX-GA took less functional evaluations (computational cost) for searching those solutions. This shows that the FLEX approach can yield better convergence in shorter time on many benchmark problems.

While Figure 4 shows the average of the best fitness of each generation to see how these algorithms are converging over generations, to comment on the best-evolved solution Table 2 shows the best, worst, and median fitness obtained using the five algorithms over 30 runs. Table 3 ranks these algorithms using the information from Figure 4 and Table 2. Table 2 shows that on G01, G04, G07, G09, and G10 the best, worst, and median performance of FLEX-GA is better than others, whereas comparable to others on G08. On G02, the FLEX-GA's performance is the second best in terms of the convergence (from Figure 4(b)) and evolved solution (from Table 2), after GA. Similarly, on G06 FLEX-GA is the third best. On problems G03, G07, and G10, ES did not find

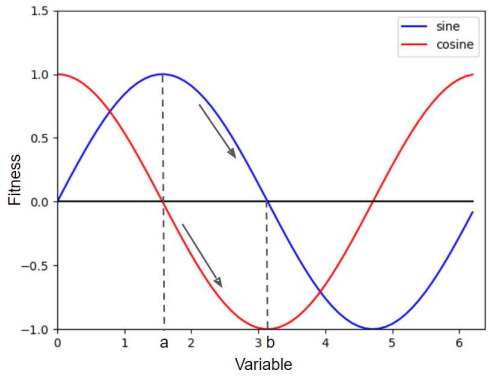


FIGURE 5: Fitness landscape of a simple two objective problem. The two arrows show that when $x \leq \pi$ and $x \geq \pi/2$, a genome vector can generate better solutions.

feasible solutions and thus not shown figures and Table2.

These results show that FLEX-GA has the potential to leverage local fitness landscape information to increase the speed of search significantly (up to 50% faster) and to find better fit solutions. In the next section, experimental results on multi-objective problems are presented.

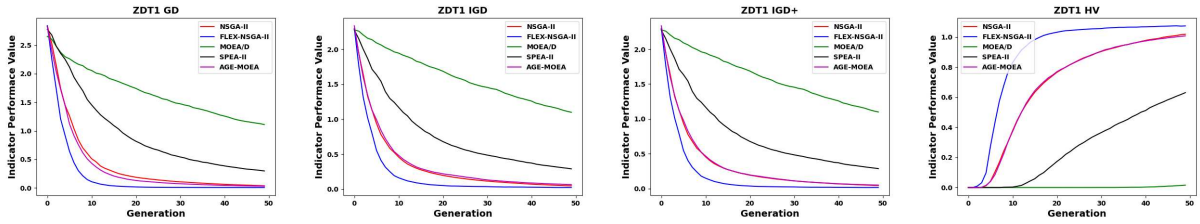
V. MULTI-OBJECTIVE FLEX

The procedure of local FL approximation of multi-objective problems is similar to that for single-objective problems with only one difference, that is instead of having one fitness gradient, multiple fitness gradients exist. Here, NSGA-II is the base algorithm for multi-objective optimization, and the proposed algorithm is referred to as FLEX-NSGA-II.

A. FLEX-NSGA-II

Just as with single-objective problems, in the case of a multi-objective optimization problem, Algorithm 2 generates a GV using a parent (p_i) and a child (c_i), and multiple fitness gradients that are associated with the GV. Algorithm 2 generates a new child solution (c_i') only when the fitness gradient with respect to each objective is negative, that is $\langle \frac{\partial}{\partial f_1} FV, .., \frac{\partial}{\partial f_m} FV \rangle < 0$ where m is the number of objectives. For two-objective problems, if $\langle \frac{\partial}{\partial f_1} FV, \frac{\partial}{\partial f_2} FV \rangle < 0$, then c_i dominates p_i and a new child is generated. To explain FLEX-NSGA-II domination, assume a simple two-objective problem where $f_1(x) = sin(x)$, $f_2(x) = cos(x)$, and $x \in \{0, 2\pi\}$. The fitness landscapes for both objectives are shown in Figure 5 where the x-axis represents the input variable (x) and the y-axis shows the fitness.

Consider this as a one variable problem to explain the concept here, but the concepts apply to problems represented by an l-dimensional chromosome as well. In Figure 5, two arrows represent genome vectors associated with negative fitness gradients i.e. following these GVs, both objectives will reduce when $x \geq a(\pi/2)$ *and* $x \leq b(\pi)$. Thus Algorithm 2 can only generate new child using equation 2 in this region only. The three questions raised in previous subsection are applicable here as well. Remembering the issue that



(a) GD (b) IGD (c) IGD+ (d) HV

FIGURE 6: Comparing the average GD, IGD, IGD+, and HV of the best Pareto front of each generation obtained over 30 runs using different algorithms on ZDT1 problem. Figures show that FLEX-NSGA-II outperformed others on all indicator measures.

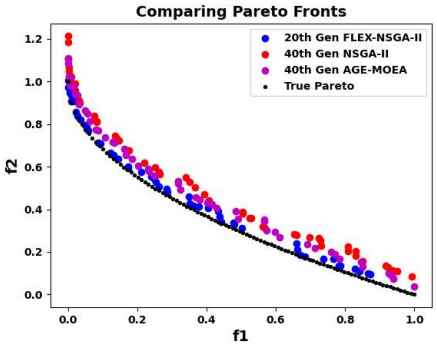


FIGURE 7: Figure shows that the best Pareto front obtained in the 20^{th} generation using FLEX-NSGA-II is closer to the true Pareto front compared to fronts obtained in the 40^{th} generation using NSGA-II and AGE-MOEA.

following a vector may lead the search to undesirable parts of the search space. This issue is less a cause of worry for multi-objective optimization problems compared to single-objective problems. Since, Algorithm 2 generates a new solution only when c_i dominates p_i, thus until a vector is found that improves both objective values, a new child will not be created. The multi-objective nature of a given problem has inherent characteristics that discourage the divergence of solutions generated using local fitness landscape approximation given by Algorithm 2. Hence it is safe to expect that Algorithm 2 will generate more child solutions for single-objective problems compared to multi-objective problems.

B. MULTI-OBJECTIVE RESULTS AND DISCUSSIONS

In this paper, only two-objective problems are considered. For these two-objective problems FLEX-NSGA-II is compared with the original NSGA-II, MOEA/D, SPEA-II, and AGE-MOEA. For all experiments, chromosomes are real-value encoded, simulated binary crossover is used to create offspring with the probability of 0.90, the polynomial mutation mutates an offspring with the probability of 0.05. The population size and number of generations were varied for different problems between $40 - 100$. Finally, to preserve good solutions found during evolution, $\mu + \lambda$ elitism is used. Studies on selection schemes [44] have shown that quality of solutions improves if convergence is slowed and greater diversity is allowed in the populations, thus binary tournament selection is used to select two parent solutions to produce two offspring.

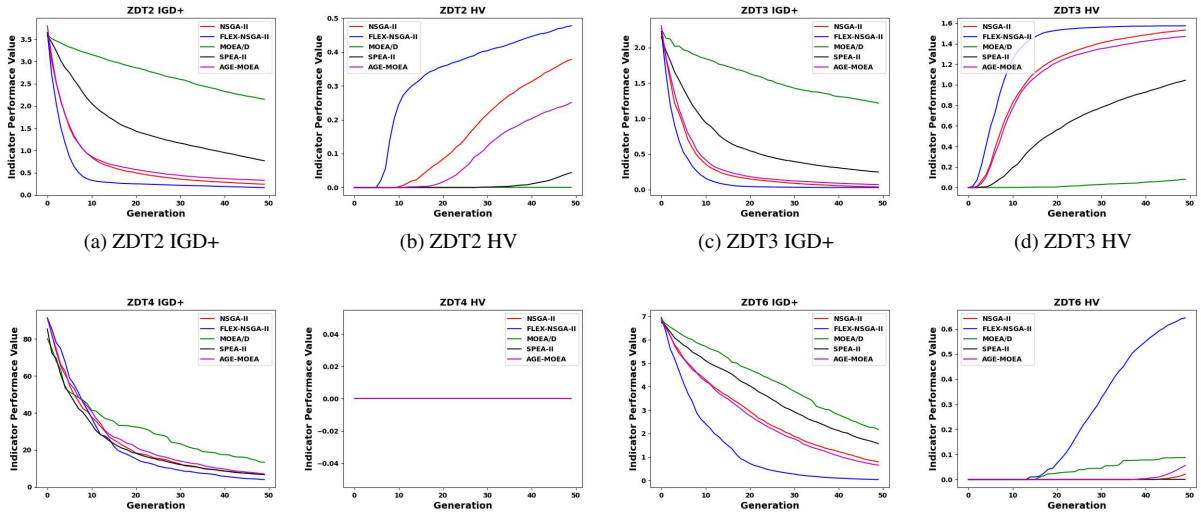
1) Performance Indicators

Comparing two Pareto fronts is not straightforward as in the case of single-objective problems. In the literature, generational distance (GD), inverted generational distance (IGD), inverted generational distance plus (IGD+), and hypervolume (HV) [45], [46] have been used to compare Pareto solutions. From these, only IGD+ and HV are Pareto-compliant indicators, and thus used for a detailed comparison of FLEX-NSGA-II's performance with other algorithms. GD, IGD, and IGD+ compute the distance from the true Pareto front where a lower distance means that the evolved Pareto front is closer to the optimal. However when using the above three indicators, a true/target Pareto front is required, and thus these are not applicable to problems with unknown true Pareto fronts. The HV indicator does not require the true front [47], and uses a reference point to compute the area enclose by points in the best Pareto front.

2) Unconstrained multi-objective Benchmark Problems

First, five unconstrained ZDT benchmark problems are considered from [6], and results obtained for these problems (ZDT1, ZDT2, ZDT3, ZDT4, ZDT6) are presented here. For each of these problems, the true Pareto front is known [48]. On each problem, solutions are evolved, 30 times with different initialization, with the population size of 50 for 50 generations and computed GD, IGD, IGD+, and HV of the best Pareto front (rank zero) from each generation. Figure 6 shows the comparison of average (a) GD, (b) IGD, (c) IGD+, and (d) HV per generation obtained over 30 runs using different algorithms on ZDT1. Figure 6 (a), (b), and (c) show that the average GD, IGD, IGD+ of FLEX-NSGA-II converges much faster than other algorithms. For comparison, FLEX-NSGA-II converges around the 20^{th} generation whereas NSGA-II, and AGE-MOEA converge around the 40^{th} generation. MOEA/D's convergence is always slower than others.

Note that, the lower the values of GD, IGD, and IGD+ the



(a) ZDT2 IGD+ (b) ZDT2 HV (c) ZDT3 IGD+ (d) ZDT3 HV

(e) ZDT4 IGD+ (f) ZDT4 HV (g) ZDT6 IGD+ (h) ZDT6 HV

FIGURE 8: Comparing the average IGD+ and HV per generation over 30 runs obtained using the five algorithms on ZDT2, ZDT3, ZDT4, and ZDT6. Figures show that FLEX-NSGA-II's performance is better than the others.

TABLE 4: Comparing the five algorithms using different performance indicators on ZDT1, ZDT2, and ZDT3 over 30 runs.

Indicators		ZDT1					ZDT2					ZDT3				
		FLEX-NSGA-II	NSGA-II	MOEA/D	SPEA-II	AGE MOEA	FLEX-NSGA-II	NSGA-II	MOEA/D	SPEA-II	AGE MOEA	FLEX-NSGA-II	NSGA-II	MOEA/D	SPEA-II	AGE MOEA
IGD+	Best	**0.0117**	0.0223	0.7391	0.1453	0.0278	**0.0134**	0.0404	1.5923	0.3603	0.0668	**0.0067**	0.0151	0.8049	0.1617	0.0163
	Worst	**0.0252**	0.0922	1.5917	0.4033	0.1149	**0.3108**	0.3776	2.7932	1.3882	0.4282	0.1542	**0.09**	1.7319	0.3722	0.1956
	Median	**0.0145**	0.0425	1.0774	0.2954	0.0407	**0.1958**	0.3208	2.1543	0.7274	0.3417	**0.0083**	0.0279	1.2341	0.2494	0.0474
HV	Best	**1.0829**	1.0616	0.1405	0.8424	1.048	**0.745**	0.6843	0.0	0.2361	0.6310	**1.6246**	1.5929	0.3509	1.2266	1.5883
	Worst	**1.0467**	0.9385	0.0	0.477	0.9068	**0.2698**	0.1889	0.0	0.0	0.1282	1.325	**1.3987**	0.0	0.8709	1.2441
	Median	**1.0767**	1.0186	0.0	0.6148	1.0231	**0.4191**	0.2578	0.0	0.0	0.2320	**1.6187**	1.5485	0.0352	1.0149	1.5171

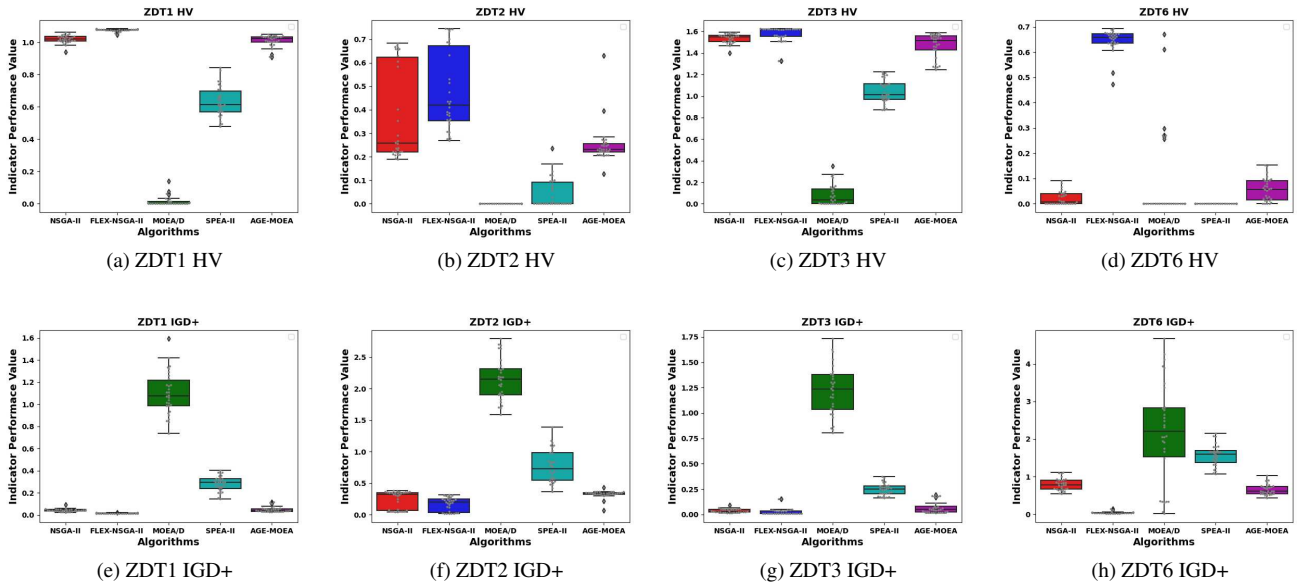
TABLE 5: Comparing the five algorithms using different performance indicators on ZDT4 and ZDT6 over 30 runs.

Indicators		ZDT4					ZDT6				
		FLEX-NSGA-II	NSGA-II	MOEA/D	SPEA-II	AGE MOEA	FLEX-NSGA-II	NSGA-II	MOEA/D	SPEA-II	AGE MOEA
IGD+	Best	**1.7815**	2.3398	5.585	3.7854	3.3586	**0.0105**	0.5419	0.0211	1.0718	0.4358
	Worst	**6.3919**	10.4432	22.4721	11.0587	14.0639	**0.1392**	1.1069	4.6694	2.1465	1.0343
	Median	**3.4395**	7.1427	13.3954	6.1816	6.8213	**0.031**	0.773	2.2037	1.6014	0.6096
HV	Best	0.0	0.0	0.0	0.0	0.0	**0.6924**	0.0922	0.6707	0.0	0.1524
	Worst	0.0	0.0	0.0	0.0	0.0	**0.472**	0.0	0.0	0.0	0.0
	Median	0.0	0.0	0.0	0.0	0.0	**0.6572**	0.0076	0.0	0.0	0.0573

closer they are (evolved Pareto front) to the actual/true Pareto front. Since figure 6 shows that FLEX-NSGA-II converges much faster than others, the evolved Pareto front from the 20^{th} generation of FLEX-NSGA-II and from the 40^{th} generation of NSGA-II, and AGE-MOEA are plotted in Figure 7. The other two algorithms performed worst compared to these three, and thus their Pareto fronts are not compared. The figure shows that the blue Pareto front (FLEX-NSGA-II) is closer to the true Pareto front (black front) compared to the red front (NSGA-II's), and magenta front (AGE-MOEA's) while taking 50% fewer function evaluations.

Unlike the distance-based indicators, a larger value of HV refers to better Pareto solution [47]. To compute the HV for each of the ZDT problems, a reference point at (1.2, 1.2) is chosen because the optimal solution's fitness lies between 0 to 1 for both objectives. Figure 6(d) shows that the HV of FLEX-NSGA-II Pareto front solutions is significantly better (larger) than the other four algorithms, again indicating that FLEX-NSGA-II evolved Pareto front solutions faster, and contains better solutions. These results provide evidence that by using fitness landscape information locally, the speed of search for optimal/near-optimal solution increases significantly. The figure clearly shows that FLEX-NSGA-II takes at least 50% less computational cost to find solutions.

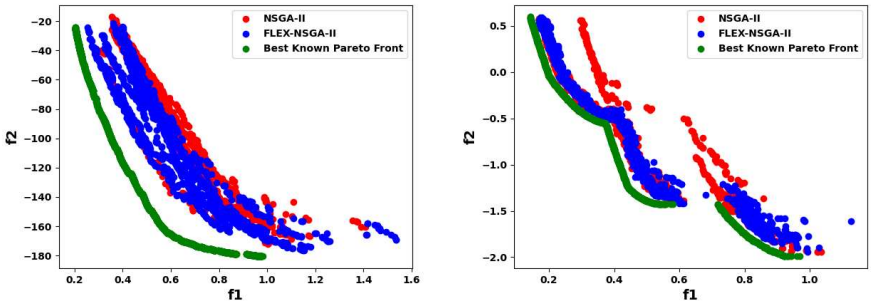
Similar experiments were conducted on the remaining ZDT problems and for comparison, only IGD+ and HV performance indicators are shown as only these two are Pareto-compliant indicators. Similar to ZDT1, Figure 8 shows that except on ZDT4, FLEX-NSGA-II takes at least 50% less computational evaluations to find solutions (in terms of IGD+ and HV) that are better (or comparable) than the other four algorithms. With a population size of 50, and running for 50 generations, the fitnesses of evolved solutions on ZDT4 are larger than the reference point of (1.2, 1.2), so



(a) ZDT1 HV (b) ZDT2 HV (c) ZDT3 HV (d) ZDT6 HV

(e) ZDT1 IGD+ (f) ZDT2 IGD+ (g) ZDT3 IGD+ (h) ZDT6 IGD+

FIGURE 9: Comparing the distribution of HV and IGD+ of the last generation Pareto fronts obtained over 30 runs. Note that higher HV and lower IGD+ refer to better quality Pareto solutions.

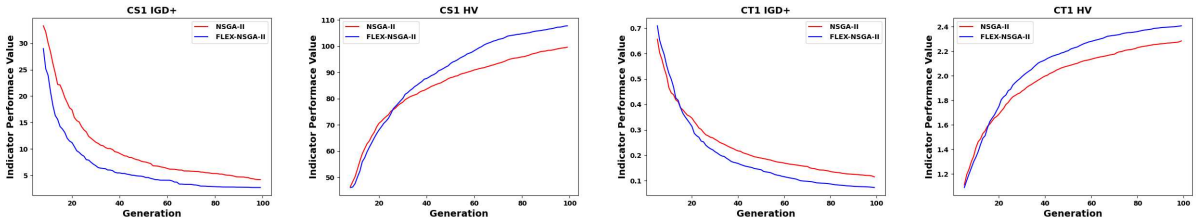


(a) CS1 (b) CT1

FIGURE 10: Rank zero Pareto fronts from the last generation of each run on CS1 and CT1. Green, blue, and red points represent the best know Pareto front so far [14], solutions evolved using FLEX-NSGA-II, and using NSGA-II in 10 runs respectively.

the hypervolume is zero. For ZDT4 more evaluations are required to compute better solutions. Tables 4 and 5 compare the best, worst, and median values of the IGD+ and HV of the best Pareto front from the last generation obtained using the five algorithms. On ZDT1, ZDT2, and ZDT6 in all combinations of indicators, FLEX-NSGA-II performed better than others. The hypervolume of the last generation Pareto front obtained using MOEA/D is zero because the fitness values of Pareto solutions are larger than the reference point. When comparing the best and median of IGD+ and HV over 30 runs on ZDT3, FLEX-NSGA-II is better and NSGA-II performance is better on the remaining performance comparison matrix. Results on ZDT problems show that using the FLEX approach, the speed of the search and quality of solutions improve significantly compared to NSGA-II, MOEA/D, SPEA-II, and AGE-MOEA.

Figure 9 shows the distribution of HV (top) and IGD+ (bottom) values of the best Pareto front from the last generation obtained using the five algorithms over 30 runs. Remember that the higher the values of HV, the better the quality of Pareto solutions. HV distribution shows that the median performance of FLEX-NSGA-II is better than the others. The p-values of HV for all ZDT problems, except ZDT4, are less than 0.05 indicating strong statistical significance of results. In contrast to the HV measure, smaller IGD+ values suggest that the evolved Pareto fronts are close to the true Pareto front solutions. The distribution in Figure 9(bottom row) shows that FLEX-NSGA-II's IGD+ values are smaller than the rest of the four algorithms and p-values are smaller than 0.05. The effect sizes (A) for HV and IGD+ for all ZDT problems



(a) CS1 IGD+ (b) CS1 HV (c) CT1 IGD+ (d) CT1 HV

FIGURE 11: Comparing the average IGD+ and HV values of the best Pareto front from the last generation on CS1 and CT1 obtained using FLEX-NSGA-II and NSGA-II over 10 runs.

TABLE 6: Comparing the best, worst, and median values of IGD+ and HV on CS problems over 10 runs

Indicators		FLEX-NSGA-II	NSGA-II	FLEX-NSGA-II	NSGA-II	FLEX-NSGA-II	NSGA-II	FLEX-NSGA-II	NSGA-II
		CS1		CS2		CS3		CS4	
IGD+	Best	**0.4914**	1.03	**1.4981**	1.5257	30.4956	**24.4225**	**29.5131**	32.6533
	Worst	**7.1144**	7.6886	**4.4153**	5.5952	48.4414	**47.3848**	101.227	**98.4323**
	Median	**2.3922**	4.6027	**2.5046**	2.9762	**37.6297**	39.7891	64.2075	**63.714**
HV	Best	122.5058	**123.6587**	**119.3586**	116.4083	28.3348	**35.9575**	37.3196	**45.1831**
	Worst	**94.2387**	90.9137	58.4571	**60.4826**	**10.22**	9.6398	6.67	**8.2226**
	Median	**105.655**	98.4762	**96.7949**	96.502	18.206	**19.6642**	**24.8328**	23.6266

TABLE 7: Comparing the best, worst, and median values of IGD+ and HV on CT problems over 10 runs

Indicators		FLEX-NSGA-II	NSGA-II	FLEX-NSGA-II	NSGA-II	FLEX-NSGA-II	NSGA-II	FLEX-NSGA-II	NSGA-II
		CT1		CT2		CT3		CT4	
IGD+	Best	**0.0556**	0.0578	**0.0673**	0.0674	**0.0814**	0.1297	**0.3163**	0.3666
	Worst	**0.0939**	0.2507	0.1558	**0.1434**	0.712	**0.5942**	1.0178	**0.8757**
	Median	**0.0735**	0.0868	**0.0803**	0.0988	0.2347	**0.2211**	**0.483**	0.5193
HV	Best	**2.4778**	2.4567	2.3984	**2.4301**	**1.6696**	1.3969	**1.4362**	1.3372
	Worst	**2.3439**	1.8509	2.1512	**2.1952**	0.4157	**0.586**	0.2894	**0.4429**
	Median	**2.4162**	2.3681	**2.364**	2.3128	1.1241	**1.1621**	**1.0834**	0.9794

are greater than 0.5 again indicates that FLEX-NSGA-II performance is significantly better than the others. These results indicate that FLEX-NSGA-II ranks top compared to other four algorithms. In the next subsection, real-world multi-objective optimization problems are considered.

3) Constrained multi-objective Real World Problems

Picard [14] in 2021 presented 20 real-world industrial constrained multi/many-objective benchmark problems to design electro-mechanical actuators. Eight of those 20 are two-objective problems that aim to minimize cost (C), maximize the minimum torque excess (T), and maximize the safety factor (S) of actuators. These eight problems are combinations of CS (CS1, CS2, CS3, CS4) and CT (CT1, CT2, CT3, CT4) with different constraints. Each problem is of 20 dimensions and has between 6 and 11 constraints. For more details regarding the problem statement and constraints see Picard [14]. Note that since the local fitness landscape information is only used to create new offspring without looking at the constraints, constraint handling in FLEX-NSGA-II is identical to NSGA-II.

These real-world problems are computationally expensive to evaluate, and thus Pareto optimal solutions are evolved 10 times using FLEX-NSGA-II, NSGA-II on each problem with the population size of 100 and for 100 generations. Figure 10 shows the best evolved Pareto front of each run using FLEX-NSGA-II and NSGA-II on two problems (CS1, CT1) where blue points represent Pareto solution of FLEX-NSGA-II, red points refer to Pareto solutions of NSGA-II, and green points show the actual/true Pareto solutions. Both Figures 10(a) and (b) show that FLEX-NSGA-II evolved Pareto solutions are closer to the true Pareto compared to NSGA-II evolved Pareto solutions.

Figure 11 shows the comparison of average IGD+ and HV of the best front from the last generation on CS1 and CT1 over 10 runs. Recall that lower IGD+ and higher HV refer to better solution/front and the figure shows that FLEX-NSGA-II performance is better than NSGA-II on these two problems. Both CS1 and CT1 are constrained two-objective problems and results show that in the initial few generations no feasible solutions are found, thus the average IGD+ and HV values are plotted only when feasible solutions are found. Figure 11(a) and (b) show the IGD+ and HV values from the 8^{th} generation for CS1. Similarly Figure 11 (c), and (d) show the IGD+ and HV from the 5^{th} generation for CT1. These figures show that FLEX-NSGA-II's convergence (IGD+ and HV values over generations) is faster than NSGA-II, and thus takes fewer evaluations to evolve similar quality solutions. Table 6 and 7 list the best, worst, and median IGD+ and HV values obtained from the best Pareto front of the last generation evolved using FLEX-NSGA-II and NSGA-II on the eight problems. Table 6 shows that the

performance of FLEX-NSGA-II on CS1, and CS2 is better than NSGA-II, comparable and/or inferior on CS3 and CS4 to NSGA-II. Similar observations can be drawn from Table 7 where FLEX-NSGA-II is performing better on CT1 and CT4, comparable to NSGA-II on CT2, but inferior to NSGA-II on CT3.

Results on these eight real-world multi-objective problems show that FLEX-NSGA-II improved the speed of the search and quality of evolved solutions on four of the eight problems and inferior only in two against NSGA-II. This provides further evidence that using the local FLEX approach, the speed of the search and/or quality of solutions improves.

VI. CONCLUSION

This paper presents fitness landscape exploration-based genetic algorithms. The main aim of FLEX-GA is to leverage the FL information to increase the speed of the search and to find better quality solutions. Assuming a local linear relationship between neighbouring/local genomes' fitness, a genome vector is created and the fitness vector/gradient corresponding to the GV is computed. The GV and fitness gradient are used to generate a new child in the search/design space. Experiments were conducted on several single and two-objective optimization test benchmark problems, and eight real-world two-objective problems. Results show that on single-objective problems, FLEX-GA evolved better or comparable solutions while only requiring as much as 50% fewer evaluations compared to existing simple GAs and other algorithms. On most of the problems, the best solutions of the last generation were also better than the other four algorithms. In the case of two-objective problems, again FLEX-NSGA-II took as much as 50% fewer functional evaluations to generate better or comparable Pareto solutions/fronts than standard NSGA-II and other multi-objective optimization algorithms. Lastly, solutions evolved on eight real-world multi-objective problems show that FLEX-NSGA-II evolved better Pareto solutions than NSGA-II.

These results provide evidence that FL information can be utilized effectively to enhance the search capacity of genetic algorithms and thus encourages researchers to use GAs for computationally expensive problems. Results also show that on a few problems FLEX approach did not work well as compared to GA. These can happen because the fitness landscape might be too rugged and local fitness approximation is not able to produce better results. To deal with such problems global FL approximation might be well suited. Future work will extend this work in the following ways: 1) what will be the impact of generating a genome vector using the entire population information, not just with the two selected parents, 2) how to vary delta for each gene of a genome vector, and 3) how to scale the FLEX approach for three or more objective problems.

REFERENCES

[1] Sourabh Katoch, Sumit Singh Chauhan, and Vijay Kumar. A review on genetic algorithm: past, present, and future. Multimedia Tools and Applications, 80(5):8091–8126, 2021.

[2] Carmen Kar Hang Lee. A review of applications of genetic algorithms in operations management. Engineering Applications of Artificial Intelligence, 76:1–12, 2018.

[3] Darrell Whitley and Andrew M Sutton. Genetic algorithms-a survey of models and methods. In Handbook of natural computing, pages 637–671. Springer Berlin Heidelberg, 2012.

[4] Moustafa Alzantot, Yash Sharma, Supriyo Chakraborty, Huan Zhang, Cho-Jui Hsieh, and Mani B Srivastava. Genattack: Practical black-box attacks with gradient-free optimization. In Proceedings of the Genetic and Evolutionary Computation Conference, pages 1111–1119, 2019.

[5] Katherine Mary Malan. A survey of advances in landscape analysis for optimisation. Algorithms, 14(2):40, 2021.

[6] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. IEEE Transactions on Evolutionary Computation, 6(2):182–197, 2002.

[7] Kenneth V Price. Differential evolution. In Handbook of optimization, pages 187–214. Springer, 2013.

[8] Riccardo Poli, James Kennedy, and Tim Blackwell. Particle swarm optimization. Swarm intelligence, 1(1):33–57, 2007.

[9] Torsten Asselmeyer, Werner Ebeling, and Helge Rosé. Evolutionary strategies of optimization. Physical Review E, 56(1):1171, 1997.

[10] Qingfu Zhang and Hui Li. Moea/d: A multiobjective evolutionary algorithm based on decomposition. IEEE Transactions on evolutionary computation, 11(6):712–731, 2007.

[11] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. Spea2: Improving the strength pareto evolutionary algorithm. TIK-report, 103, 2001.

[12] Annibale Panichella. An adaptive evolutionary algorithm based on non-euclidean geometry for many-objective optimization. In Proceedings of the Genetic and Evolutionary Computation Conference, pages 595–603, 2019.

[13] Jing J Liang, Thomas Philip Runarsson, Efren Mezura-Montes, Maurice Clerc, Ponnuthurai Nagaratnam Suganthan, CA Coello Coello, and Kalyanmoy Deb. Problem definitions and evaluation criteria for the cec 2006 special session on constrained real-parameter optimization. Journal of Applied Mechanics, 41(8):8–31, 2006.

[14] Cyril Picard and Jürg Schiffmann. Realistic constrained multiobjective optimization benchmark problems from design. IEEE Transactions on Evolutionary Computation, 25(2):234–246, 2020.

[15] Sewall Wright et al. The roles of mutation, inbreeding, crossbreeding, and selection in evolution. 1932.

[16] Erik Pitzer and Michael Affenzeller. A comprehensive survey on fitness landscape analysis. Recent advances in intelligent engineering systems, pages 161–191, 2012.

[17] Jean-Paul Watson. An introduction to fitness landscape analysis and cost models for local search. In Handbook of metaheuristics, pages 599–623. Springer, 2010.

[18] Peter Merz and Bernd Freisleben. Fitness landscape analysis and memetic algorithms for the quadratic assignment problem. IEEE transactions on evolutionary computation, 4(4):337–352, 2000.

[19] Gabriela Ochoa and Katherine Malan. Recent advances in fitness landscape analysis. In Proceedings of the Genetic and Evolutionary Computation Conference Companion, pages 1077–1094, 2019.

[20] Alain Ratle. Accelerating the convergence of evolutionary algorithms by fitness landscape approximation. In International Conference on Parallel Problem Solving from Nature, pages 87–96. Springer, 1998.

[21] Yan Pei. Trends on fitness landscape analysis in evolutionary computation and meta-heuristics. In Frontier Applications of Nature Inspired Computation, pages 78–99. Springer, 2020.

[22] Yan Pei and Hideyuki Takagi. Fourier analysis of the fitness landscape for evolutionary search acceleration. In 2012 IEEE Congress on Evolutionary Computation, pages 1–7. IEEE, 2012.

[23] Ying Huang, Wei Li, Chengtian Ouyang, and Yan Chen. A self-feedback strategy differential evolution with fitness landscape analysis. Soft Computing, 22(23):7773–7785, 2018.

[24] Jun Yu, Yuhao Li, Yan Pei, and Hideyuki Takagi. Accelerating evolutionary computation using a convergence point estimated by weighted moving vectors. Complex & Intelligent Systems, 6(1):55–65, 2020.

[25] Jing Yang, Yingpeng Hu, Kaixi Zhang, and Yanghui Wu. An improved evolution algorithm using population competition genetic algorithm and self-correction bp neural network based on fitness landscape. Soft Computing, 25(3):1751–1776, 2021.

[26] Ran Cheng, Miqing Li, Ke Li, and Xin Yao. Evolutionary multiobjective optimization-based multimodal optimization: Fitness landscape

approximation and peak detection. IEEE Transactions on Evolutionary Computation, 22(5):692–706, 2017.
[27] Zhiping Tan, Kangshun Li, and Yi Wang. Differential evolution with adaptive mutation strategy based on fitness landscape analysis. Information Sciences, 549:142–163, 2021.
[28] Zhiping Tan and Kangshun Li. Differential evolution with mixed mutation strategy based on deep reinforcement learning. Applied Soft Computing, 111:107678, 2021.
[29] Kalifou René Traoré, Andrés Camero, and Xiao Xiang Zhu. Fitness landscape footprint: A framework to compare neural architecture search problems. arXiv preprint arXiv:2111.01584, 2021.
[30] Cristiano G Pimenta, Alex GC de Sá, Gabriela Ochoa, and Gisele L Pappa. Fitness landscape analysis of automated machine learning search spaces. In European Conference on Evolutionary Computation in Combinatorial Optimization (Part of EvoStar), pages 114–130. Springer, 2020.
[31] Mohamed El Yafrani, Marcella SR Martins, Mehdi El Krari, Markus Wagner, Myriam RBS Delgado, Belaïd Ahiod, and Ricardo Lüders. A fitness landscape analysis of the travelling thief problem. In Proceedings of the Genetic and Evolutionary Computation Conference, pages 277–284, 2018.
[32] Gabriela Ochoa, Marco Tomassini, Sébastien Vérel, and Christian Darabos. A study of nk landscapes' basins and local optima networks. In Proceedings of the 10th annual conference on Genetic and evolutionary computation, pages 555–562, 2008.
[33] Matheus Nunes, Paulo M Fraga, and Gisele L Pappa. Fitness landscape analysis of graph neural network architecture search spaces. In Proceedings of the Genetic and Evolutionary Computation Conference, pages 876–884, 2021.
[34] Lingxi Xie and Alan Yuille. Genetic cnn. In Proceedings of the IEEE international conference on computer vision, pages 1379–1388, 2017.
[35] Zhichao Lu, Ian Whalen, Vishnu Boddeti, Yashesh Dhebar, Kalyanmoy Deb, Erik Goodman, and Wolfgang Banzhaf. Nsga-net: neural architecture search using multi-objective genetic algorithm. In Proceedings of the genetic and evolutionary computation conference, pages 419–427, 2019.
[36] Rahul Dubey, Joseph Ghantous, Sushil Louis, and Siming Liu. Evolutionary multi-objective optimization of real-time strategy micro. In 2018 IEEE Conference on Computational Intelligence and Games (CIG), pages 1–8, 2018.
[37] Yujiang Xiang, Jasbir S Arora, and Karim Abdel-Malek. Physics-based modeling and simulation of human walking: a review of optimization-based and other approaches. Structural and Multidisciplinary Optimization, 42(1):1–23, 2010.
[38] Muhammad Imran, Changwook Kang, Young Hae Lee, Mirza Jahanzaib, and Haris Aziz. Cell formation in a cellular manufacturing system using simulation integrated hybrid genetic algorithm. Computers & Industrial Engineering, 105:123–135, 2017.
[39] Nicolai Peremezhney, E Hines, A Lapkin, and C Connaughton. Combining gaussian processes, mutual information and a genetic algorithm for multi-target optimization of expensive-to-evaluate functions. Engineering Optimization, 46(11):1593–1607, 2014.
[40] John H Holland. Genetic algorithms. Scientific american, 267(1):66–73, 1992.
[41] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, pages 1135–1144, 2016.
[42] Matthew D Zeiler. Adadelta: an adaptive learning rate method. arXiv preprint arXiv:1212.5701, 2012.
[43] Momin Jamil and Xin-She Yang. A literature survey of benchmark functions for global optimisation problems. International Journal of Mathematical Modelling and Numerical Optimisation, 4(2):150–194, 2013.
[44] David E Goldberg and Kalyanmoy Deb. A comparative analysis of selection schemes used in genetic algorithms. In Foundations of genetic algorithms, volume 1, pages 69–93. Elsevier, 1991.
[45] Jesús Guillermo Falcón-Cardona and Carlos A Coello Coello. Indicator-based multi-objective evolutionary algorithms: A comprehensive survey. ACM Computing Surveys (CSUR), 53(2):1–35, 2020.
[46] Miqing Li, Shengxiang Yang, and Xiaohui Liu. A performance comparison indicator for pareto front approximations in many-objective optimization. In Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation, pages 703–710, 2015.
[47] Carlos M Fonseca, Luís Paquete, and Manuel López-Ibánez. An improved dimension-sweep algorithm for the hypervolume indicator. In 2006 IEEE international conference on evolutionary computation, pages 1157–1163. IEEE, 2006.
[48] J. Blank and K. Deb. pymoo: Multi-objective optimization in python. IEEE Access, 8:89497–89509, 2020.

RAHUL DUBEY is a postdoctoral research associate in the department of Electronics Engineering at the University of York UK. He received a PhD in computer science and engineering from the University of Nevada Reno USA in 2021. His research interests are in the areas of evolutionary computing, machine learning, explainable AI, and multi agent systems.

SIMON HICKINBOTHAM is a research fellow in the department of Electronics Engineering at the University of York, UK. He received a DPhil in 2000 (University of York) in Computer Vision. He joined the Electronic Engineering Department at the University of York in 2019. His main research intersts are in self-organisation of evolutionary systems, artificial life, and pattern recognition in big data. He has published over 50 papers in these areas.

MARK PRICE is a professor in the department of Mechanical Engineering at the Queen's University Belfast. He received a 1^{st} class honours in Aeronautical Engineering from Queen's University in 1987 and a PhD in Mechanical Engineering also from Queen's in 1993. After a period in industry, he returned to academia focusing his research on integrating engineering methods with manufacturing processes and systems. He is developing novel bio-inspired engineering design methods for multi-disciplinary problems collaborating closely with academic and industrial partners. He is a Fellow of the IMechE and the RAeS and a Member of the British Computer Society.

ANDY TYRRELL received a 1st class honours degree in 1982 and a PhD in 1985 (Aston University), both in Electrical and Electronic Engineering. He joined the Electronic Engineering Department at the University of York in April 1990, he was promoted to the Chair of Digital Electronics in 1998. His main research interests are in the design of biologically-inspired architectures, evolutionary robotics, evolvable hardware and novel engineering design methods. This work has included the creation of embryonic processing array, intrinsic evolvable hardware systems and the autonomous robot evolutionary system. He is Head of the Department of Electronic Engineering at York. He has published over 350 papers in these areas. He is a Senior member of the IEEE and a Fellow of the IET.