Genetic Algorithms-Based Techniques for Solving Dynamic Optimization Problems with Unknown Active Variables and Boundaries

AbdelMonaem F.M. AbdAllah, Daryl L. Essam and Ruhul A. Sarker

Abstract In this paper, we consider a class of dynamic optimization problems in which the number of active variables and their boundaries vary as time passes (DOPUAVBs). We assume that such changes in different time periods are not known to decision makers due to certain internal and external factors. Here, we propose three variants of genetic algorithm to deal with a dynamic problem class. These proposed algorithms are compared with one another, as well as with a standard genetic algorithm based on the best of feasible generations and feasibility percentage. Experimental results and statistical tests clearly show the superiority of our proposed algorithms. Moreover, the proposed algorithm, which simultaneous addresses two sub-problems of such dynamic problems, shows superiority to other algorithms in most cases.

Keywords Active · Best of feasible generations
Dynamic optimization problems · Feasibility percentage
Genetic algorithms · Mask detection

1 Introduction

Optimization is one of the essential research fields that directly relates to everyday decision making problems, such as planning, transportation and logistics. There are different classes of optimization problems. Based on the variables that affect them, optimization problems can be categorized as discrete, continuous or mixed-variables,

A.F.M. AbdAllah (✉) · D.L. Essam · R.A. Sarker
School of Engineering and Information Technology, University of New South Wales Canberra (UNSW Canberra@ADFA), Canberra, ACT 2600, Australia
e-mail: a.abdallah@student.adfa.edu.au; abdo_system@yahoo.com

D.L. Essam
e-mail: d.essam@adfa.edu.au

R.A. Sarker
e-mail: r.sarker@adfa.edu.au

I. Zelinka et al. (eds.), *Innovative Computing, Optimization and Its Applications*, Studies in Computational Intelligence 741,
https://doi.org/10.1007/978-3-319-66984-7_9

as discussed in the following subsections. Also, optimization problems can be categorized based on existing constraints, with constrained problems generally considered to be more complicated than unconstrained ones. Furthermore, they can be categorized as static that do not change over time [1], or dynamic, where at least one part of a problem changes over time [2]. In many real-life situations, problems change as time passes, such as the demand and the capacity at different nodes and arcs in transportation systems. In Dynamic Optimization Problems (DOPs), at least one part of the problem, such as its objective function or constraints change over time. Therefore, for DOPs solving algorithms, it is important to not only locate optimal solutions, but to also track changes as time passes [3, 4]. As a result, DOP has become a challenging topic in computer science and operations research.

In the literature, most of the research carried out in DOPs deals with changes in the objective functions and/or constraints [3, 4]. However, the CEC2009 competition presented dynamic problems which are the only attempt that considers changes in problem dimensionality. In that competition, the number of variables is simply increased or decreased by adding or eliminating a variable from the end of the problem vector. So, to the best of our knowledge, there is not a detailed study taking into consideration changes in active variables and boundaries.

In an earlier work, we defined dynamic optimization problems with unknown active variables and also proposed a type of algorithm to solve such problems [5]. Furthermore, we conducted research on an initial version of dynamic optimization problems with known changeable boundaries [6].

Here, in this paper, we introduce a DOP with unknown active variables and boundaries (DOPUAVBs), in which both the active variables and their boundaries change as time passes. Therefore, a DOPUAVB consists of two sub-problems: DOPs with unknown active variables (DOPUAVs) and DOPs with unknown dynamic boundaries (DOPUBs). To solve such a dynamic problem, we develop three variants of genetic algorithms (GAs). The first algorithm considers the activeness of variables. The second considers the changeable boundaries of the variables, and the third simultaneously considers both sub-problems. The proposed algorithms were compared with one another, as well as with a simple GA (SGA), on the basis of the average of the feasible generations and percentage.

This paper is organized as follows. In Sect. 2, dynamic optimization problems with unknown active variables and boundaries are introduced and described, and a framework is provided for generating its test problems. Section 3 introduces three proposed GA-based techniques to solve such dynamic problems, along with SGA. Section 4 includes experimental results and comparisons among all GA-based techniques. Finally, conclusions and directions for future work are presented in Sect. 5.

2 Dynamic Optimization Problems with Unknown Active Variables and Boundaries (DOPUAVBs)

In this section, we propose a new type of dynamic problem, called dynamic optimization problem with unknown active variables and boundaries (DOPUAVB). In such dynamic problems, the activeness of variables and their boundaries change as

time passes. Therefore, a DOPUAVB consists of two sub-problems: a DOP with unknown active variables (DOPUAV) and a DOP with unknown dynamic boundaries (DOPUB). In a DOPUAV, active variables change, while in a DOPUB, the boundaries of variables change as time passes. Without loss of generality, this paper considers minimization problems.

To generate an instance for the DOPUAVB, its two sub-problems are considered. First, for a DOPUAV, active variables affect a decision during the time slot, while inactive variables do not. To simulate such dynamic problems, a mask with variable coefficients of 0s and 1s is randomly generated to determine inactive and active variables respectively. Let us consider a simple example of an absolute function with 5 variables: $\text{abs}(x_1 + x_2 + x_3 + x_4 + x_5)$; the minimal value for this function is when x_1: x_5 equal 0. Suppose that two of these variables are inactive; let x_2 and x_5 be chosen to be inactive (its mask value is set equal to 0) while the others are active (1). In such a case, the optimal occurs when x_1, x_3 and x_4 converge to 0s, while x_2 and x_5 have any other values. This is because the values of the inactive variables, x_2 and x_5, are ignored; to do this they are always evaluated as 0. Moreover, due to mutation, crossover and lack of selection pressure processes, x_2 and x_5 tend to diverge to different random values. Hence, the efficiency of an algorithm for solving DOPUAVs depends on how active and inactive variables are handled as time passes. In [1], to solve DOPUAVs, it is suggested that an algorithm needs to periodically detect the mask of the problem every a specified number of generations. Here in this paper, to save fitness evaluations, the solving algorithm tries to detect whether or not the problem has changed before detecting the problem mask.

Second, in DOPUB, the original/default boundaries of the variables are [−5, +5] with initial width equal to 10, and are shifted randomly inside these boundaries by a range of [−3, +3]; where "−3" and "+3" shifts the dynamic/feasible/changeable/shifted boundaries to the left and right by 3 steps respectively, while maintaining a minimum width of these dynamic/feasible boundaries being 2. Then, if any of the variable values of a solution are within current feasible boundaries, the fitness function will be assigned its actual fitness value; otherwise, a maximum value (DBL_MAX) will be assigned. This is because for such an infeasible solution, the objective function does not have any function value or information about infeasible areas. In this problem, in contrast to constrained optimization problems, in DOPUB the objective function cannot assign any constraint violation value to the infeasible solution(s). Moreover, the objective function cannot tell which variable is outside of its feasible boundaries. Note that for evaluating either feasible or infeasible solution, the number of conducted fitness evaluations is increased by 1.

3 Genetic-Based Algorithms for Solving DOPUAVBs

In this paper, four genetic algorithms (GAs) are used to solve dynamic optimization problems with unknown active variables and boundaries (DOPUAVBs). These GAs proposed to illustrate how they deal with the proposed problem under different considerations. These GA-based techniques are presented as follows:

3.1 SGA

The first GA is a simple GA (SGA), in which its operators work normally without any modifications. In other words, processes of selection, crossover and mutation deal with the original boundaries for all variables without any consideration of variables activeness and/or their current dynamic boundaries. Figure 1 shows the basic structure of SGA.

3.2 GAUAV

The second GA deals with unknown active variables (GAUAV). This GA considers the first sub-problem (DOPUAV). The GAUAV consists of two processes as follows:

Problem change detection. This process is used to detect whether or not a problem has changed. For problem change detection, some experiments were conducted. From those experiments, to reduce the probability of false problem change detection, a *nonZeroNotEqualAbsChromosome* is used. A *nonZeroNotEqualAbsChromosome* is a

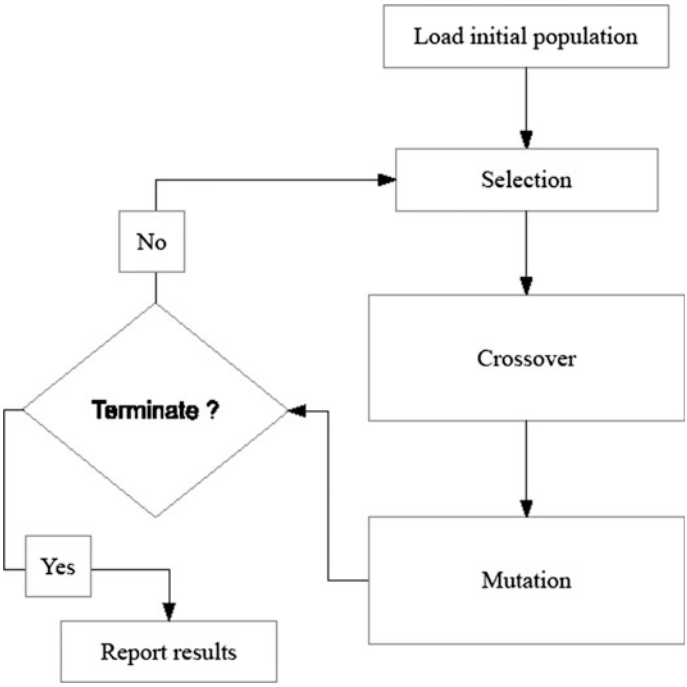

Fig. 1 The basic structure for SGA

chromosome that has non-zero, not-equal and unique absolute values, for example, if there is a chromosome with five genes, it might be (1, 2, 5, 4, 3). This chromosome is re-evaluated every generation, if its fitness is changed, then a change is detected. Once a change is detected, the GAUAV tries to detect the current problem mask using the mask detection process.

Mask detection process. This process is used to detect the mask of the inactive and active variables. Here, mask detection is done by using the single-sample mask detection procedure as follows:

- Choose a random chromosome.
- Calculate its actual fitness, let it be *F1*.
- Then for each variable, its value is replaced by a new random value within its boundary which is generated, where the absolute value of this new value is not equal to its old value:

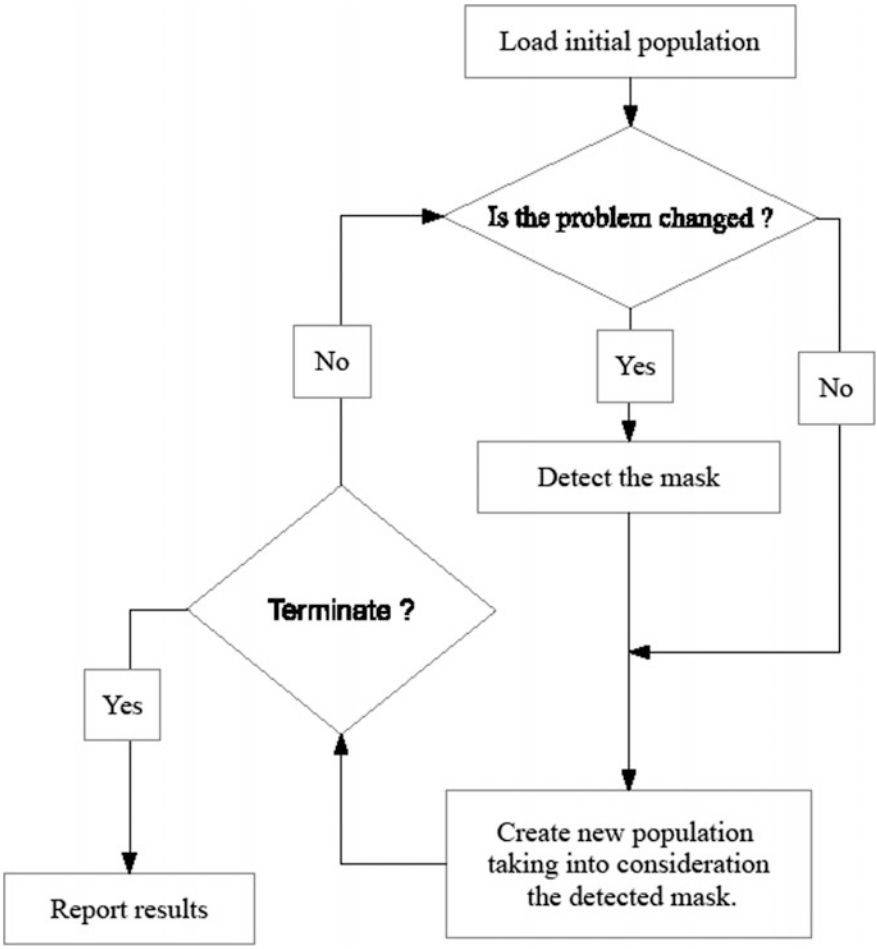

Fig. 2 The basic structure for GAUAV

- The fitness is re-calculated, let it be *F2*.
- If *F1* equals *F2*, then this variable is assumed to be inactive (its detected mask value is equal to 0); otherwise, it is assumed to be active (its detected mask value is equal to 1).

Figure 2 shows the basic structure of GAUAV.

3.3 GAUB

The third GA deals with unknown dynamic boundaries (GAUB). This GA tries to detect and use feasibility during the course of a search process. To do this, the GAUB keeps track of the feasible boundaries for feasible chromosomes, where the current lower and upper boundaries of the feasible area is the minimum and maximum variable values of feasible chromosomes respectively. Then, GAUB uses the detected feasible boundaries to evaluate infeasible chromosomes, by considering the distance between them and the centroid of the detected feasible boundaries as a degree of constraint violation. This is to guide GAUB during its selection process; it is calculated as follows:

$$X_{i(constraint\ violation)} = \sum_{j=1}^{N} d\left(X_{ij}, feasible\ centroid_j\right), \tag{1}$$

where X_i is an infeasible chromosome, N is the number of variables, and d is the distance metric. Figure 3 shows the basic structure of GAUB.

Here an illustrative example is used to show how GAUB computes the constraint violation value of an infeasible solution. Suppose we used the Manhattan distance as the distance metric and there is a problem consists of 2 variables that have boundaries [−5, 5]. An infeasible solution (−2, 2) is exist, and the currently detected feasible boundaries are [−1, −3] and [2, 0], so the feasible centroid is (1, −1). So, using Eq. 1, the constraint violation of this infeasible solution equals (abs (−2 − (1)) + abs (2 − (−1))) = (3 + 3) = 6, where abs is the absolute function.

3.4 GAUAVB

The fourth GA is a hybrid GA of the second and the third GAs (GAUAVB). This GA tries to solve the complex DOPUAVB by simultaneously considering its sub-problems. It is shown in Fig. 4. Furthermore, it considers only active variables when calculating the constraint violation value as follows:

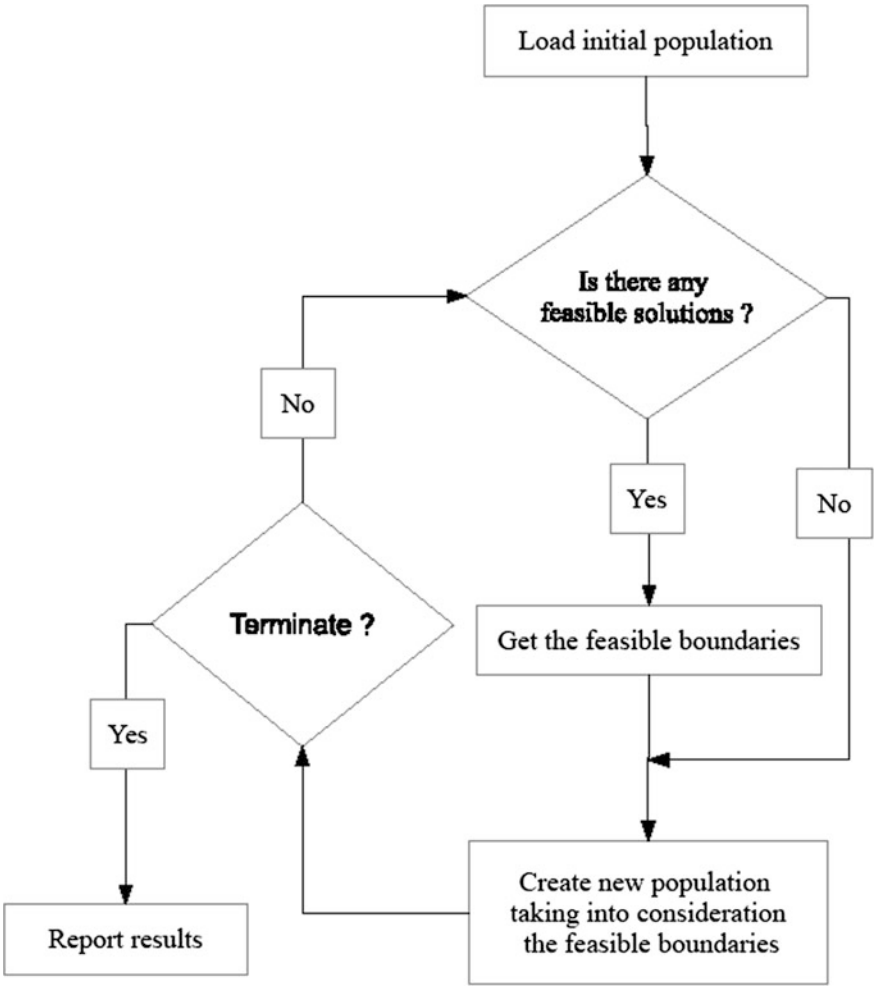

Fig. 3 The basic structure for GAUB

$$X_{i(constraint\ violation)} = \sum_{j=1}^{N} d\left(X_{ij}, feasible\ centroid_j\right),\ if\ variable\ j\ is\ active, \quad (2)$$

Using the previously used example in Sect. 3.3, suppose that the first variable is detected as an inactive variable. In this case, the distance violation of the first variable is excluded from the constraint violation calculations, So, using Eq. 2, the constraint violation of this infeasible solution equals (abs $(2 - (-1))) = 3$.

Note that for GAUAV and GAUAVB, when variables are detected as inactive, they are prevented from being mutated. Furthermore, the tournament selection in both GAUB and GAUAVB is adapted by using feasibility rules [2]. It works as follows:

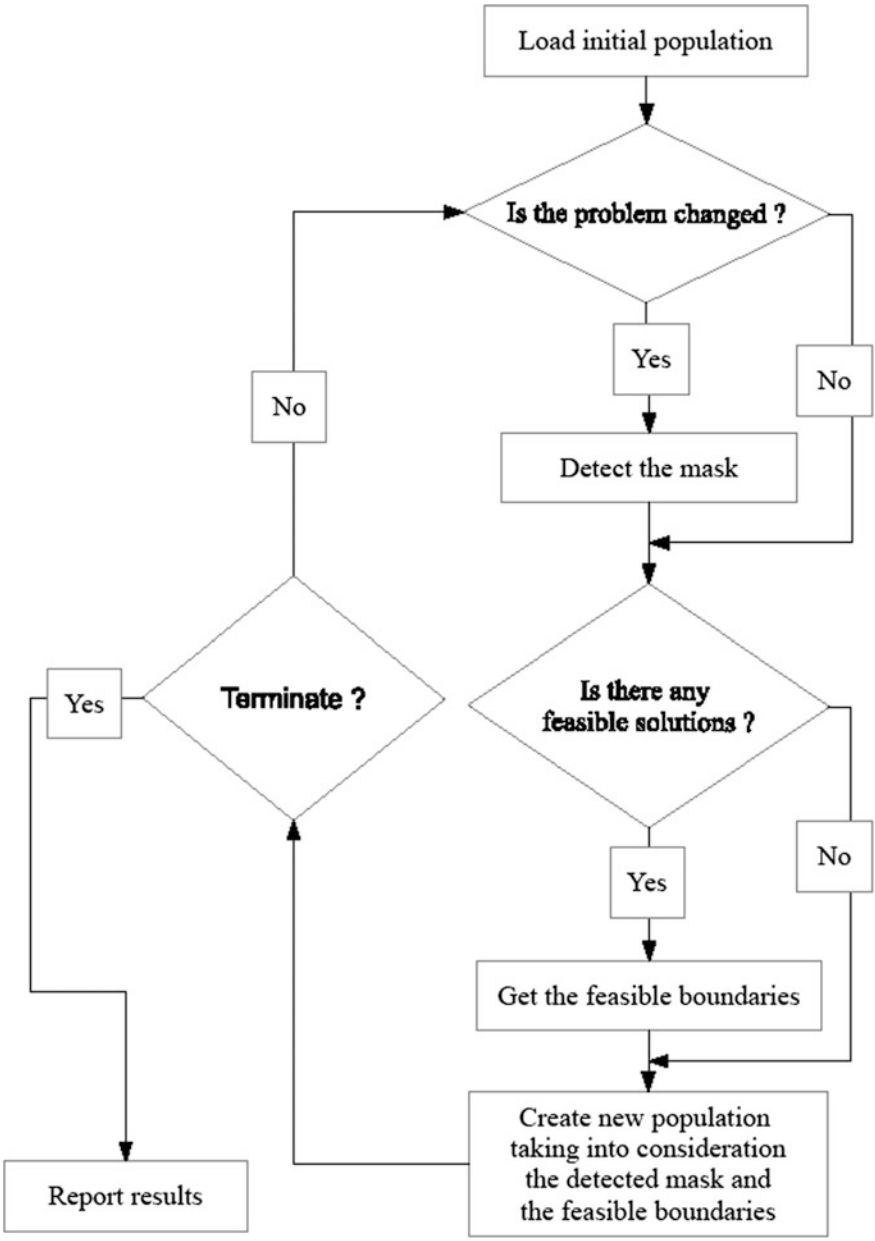

Fig. 4 The basic structure for GAUAVB

(1) If two compared solutions are both feasible, select the solution based on the minimum value of the fitness function.
(2) If one of the two compared solutions is a feasible and the other is infeasible, the feasible solution is selected.
(3) If both solutions are infeasible, select solutions based on the constraint violation (the distance between the solution and the centroid of the current feasible boundaries). The solution with the lowest violation wins.

4 Experimental Setup, Analysis and Discussion

To test the performance of the previously presented genetic algorithms (GAs)-based techniques for solving DOPUAVBs, real-coded GA-based algorithms with the same processes were implemented. The crossover is one-point, mutation is uniform and the selection is tournament. In this paper, a set of unconstrained benchmark functions, namely Sphere, Rastrigin, Weierstrass, Griewank, Ackley, Rosenbrock, Levy and NeumaierTrid are used to form these functions. Of these functions, five are completely separable problems, and three functions are non-separable. The five separable problems, namely, Sphere, Ackley, Griewank, Restrigin and Weierstrass, were used in previous test suites of dynamic problems [3], while the other three non-separable functions, namely, Levy, Rosenbrock and Trid, are taken from Surjanovic and Bingham (2015) [4, 5]. The global optimum for all these functions is $F(x^*) = 0$ with all solution variables $[x]^* = 0$, except Rosenbrock's where $x^* = 1$, and Trid where the global depends on the number of dimensions. From these five separable functions, Sphere is unimodal, while Griewank's, Rastrigin's, Ackley's and Weierstrass are multimodal. Unimodal modal functions contain only one optimum and hence are easier to optimise. On the other hand, multimodal functions are complex and contain many local optima and one global optimum. Of the three non-separable functions, Trid and Rosenbrock are unimodal, and Levy is a multimodal function. Optimising non-separable functions is generally harder than optimising separable functions. Therefore, a multimodal non-separable function is more challenging than optimising a unimodal separable function. Table 1 summarises some characteristics of the used functions followed by their equations

$$F_{Sphere} = \sum_{i=1}^{N} x_i^2 \tag{3}$$

$$F_{Ackley} = -20\,\mathbf{exp}\left(-0.2\sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2}\right) - \mathbf{exp}\left(\frac{1}{N}\sum_{i=1}^{N} \mathbf{cos}(2\boldsymbol{\pi} x_i)\right) + 20 + e \tag{4}$$

$$F_{Griewank} = \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \tag{5}$$

Table 1 Summary of used functions characteristics

Function	Separable	Multimodal	Global value and location
Sphere	Yes	No	0 where $x^* = 0$
Ackley	Yes	Yes	0 where $x^* = 0$
Griewank	Yes	Yes	0 where $x^* = 0$
Restrigin	Yes	Yes	0 where $x^* = 0$
Weierstrass	Yes	Yes	0 where $x^* = 0$
Levy	No	Yes	0 where $x^* = 0$
Rosenbrock	No	No	0 where $x^* = 1$
Trid	No	No	Depends on the number of variables

$$F_{Rastrigin} = \sum_{i=1}^{N} \left(x_i^2 - 10\,cos(2\pi x_i) + 10\right) \tag{6}$$

$$F_{Weierstrass} = \sum_{i=1}^{n} \left(\sum_{k=0}^{k_{max}} \left[a^k cos\left(2\pi b^k (x_i + 0.5)\right)\right]\right) - n \sum_{k=0}^{k_{max}} \left[a^k cos\left(\pi b^k\right)\right] \tag{7}$$

where: $a = 0.50$, b = 3, and $k_{max} = 20$.

$$\mathbf{F_{Levy}} = \mathbf{sin}^2(\boldsymbol{\pi}\,\mathbf{w}_1) + \sum_{\mathbf{i}=1}^{\mathbf{n}-1} (\mathbf{w_i} - 1)^2 \left[1 + 10\,\mathbf{sin}^2(\boldsymbol{\pi}\,\mathbf{w_i} + 1)\right] (\mathbf{w_n} - 1)^2 \left[1 + \mathbf{sin}^2(2\boldsymbol{\pi}\,\mathbf{w_n})\right] \tag{8}$$

where: $w_i = 1 + \frac{x_i - 1}{4}$, for all I = 1, …, n

$$F_{Rosenbrock} = \sum_{i=1}^{N-1} \left(100\left(x_i^2 - x_{i+1}\right)^2 + (x_i - 1)^2\right) \tag{9}$$

$$F_{Trid} = \sum_{i=1}^{n} (x_i - 1)^2 - \sum_{i=2}^{n} x_i x_{i-1} \tag{10}$$

The compared algorithms were tested under different settings of DOPUAVB as follows:

(1) The frequency of change (FOC), which determines how often a problem changes; was varied as 500, 2000 and 8000 fitness evaluations. This is used to test how the number of fitness evaluations might affect the algorithm performance.
(2) The number of inactive variables/the number of variables that have shifted boundaries (NOV); was varied as 1/1, 5/1, 1/5 and 5/5, where the first number represents the number of inactive variables and the second number represents the number of shifted boundaries of variables. This is used to test how the number of the active variables and changeable boundaries might affect the algorithm performance.

Experimental settings are shown in Table 2. Here, Manhattan distance [6] is used to calculate the degree of constraint violation. Note that fitness evaluations that

Table 2 Experimental settings

Parameter	Value
Population size	50
Tournament size	2
Selection pressure	0.90
Elitism percentage	2
Crossover rate	0.90
Mutation rate	0.15
Number of variables	20

are used in problem change detection and mask change detection are included in the budget of all of the algorithms. The algorithms were all coded in Microsoft C++, on a 3.4 GHz/16 GB Intel Core i7 machine running the Windows 7 operating system. Finally, for a fair comparison, all GAs had the same initial population at the beginning of each run with a total of 25 independent runs.

4.1 Comparison Based on the Quality of Solutions

To compare the quality of solutions, a variation of the Best-of-Generation measure was used, where best-of-generation values were averaged over all generations [7]. However, in DOPUAVBs, due to the change in the feasible boundaries, solving techniques might have infeasible generations, so we consider only feasible generations in these calculations. To do this, we propose a new variation of the Best-of-Generation measure, which is the average best-of-feasible-generation (ABOFG) and it is calculated as follows:

$$\overline{F}_{BOFG} = \frac{1}{F_i} \sum_{i=1}^{G} F_{BOFG_{ij}} \left(\frac{1}{N} \sum_{j=1}^{N} F_{BOFG_{ij}} \right), \text{where generation } i \text{ is feasible,} \quad (11)$$

where $\overline{F}_{BOFG}$ is the mean best-of-feasible-generation fitness, G is the number of generations, N is the total number of runs and $F_{BOFG_{ij}}$ is the best-of-feasible-generation fitness of generation i of run j of an algorithm on a problem [8]. As solved functions have different scales for their objective functions values, a normalized score is used so as to be able to sum obtained values of different functions to analyze the performance of compared algorithms. Note that lower values are better and the lowest are shown as bold and shaded entries. Table 3 shows the results of normalized ABOFGs for the compared techniques in regards to the number of variables (NOV).

From Table 3, it is first clearly observed that GAUAV performed better, especially when the number of unknown variables increased (5/1 and 5/5). Second, GAUB slightly performed better than SGA when the number of shifted boundaries increased (1/5 and 5/5). Third, GAUAVB outperformed all GAs in most cases.

Table 3 Normalized ABOFGs of compared algorithms in regards NOV

NOV	SGA	GAUAV	GAUB	GAUAVB
1/1	0.7444	**0.1743**	0.8111	0.2220
1/5	0.8412	0.3795	0.5727	**0.1730**
5/1	0.8898	**0.0893**	0.8226	0.1795
5/5	0.8559	0.2848	0.7030	**0.0828**
Average	0.8328	0.2320	0.7273	**0.1643**

Table 4 Normalized ABOFGs of compared techniques in regards to FOC

FOC	SGA	GAUAV	GAUB	GAUAVB
500	0.7889	**0.2540**	0.6733	0.2671
2000	0.8447	0.2264	0.7273	**0.1587**
8000	0.8648	0.2156	0.7814	**0.0672**
Average	0.8328	0.2320	0.7273	**0.1643**

Presumably, GAUAV and GAUAVB performed better as they prevented inactive variables from being mutated, as this helps GAs to effectively converge to better solutions.

From Table 4, it is clearly observed that GAUAV performed better than other GAs except GAUAVB. However, GAUAVB outperformed other GAs, especially when the frequency of changes (FOC) increased.

The Wilcoxon signed rank test [9] was also used to statistically judge the difference between paired scores, this was done because as obtained values of compared algorithms are not normally distributed, a non-parametric statistical test is used. As a null hypothesis, it is assumed that there is no significant difference between obtained values of two samples, whereas the alternative hypothesis is that there is a significant difference at the 5% significance level. Based on the obtained results, one of three signs (+, −, and ≈) is assigned when the first algorithm was significantly better, worse than, and no significance different with the second, respectively. Here, GAUAVB was paired to be compared with other GA-based variations to see how it effectively solved DOPUAVBs. In Table 5, Wilcoxon tests were applied on the total number of changes in regards to the number of variables; in this paper, there are 8 problems, and each has 10 types of changes, and each change has 3 frequency of changes, with a total of 240 values.

From Table 5, it is clear that GAUAVB was statistically better than other GAs in most cases, especially when the problem was more complex 5/5. The performance of GAUAV was statistically better for 5/1 test problems, as the number of inactive variables increases with limited changes in dynamic boundaries.

In Table 6, Wilcoxon test were again applied on the number of changes. In regards to the frequency of changes; in this paper, there are 8 problems, each has 10 changes, and each change has 4 variations of the number of variables, which gives a total of 320 values.

From Table 6, it is clear that GAUAVB was statistically better than the other GAs in most cases, especially when frequency of change increases.

Table 5 Wilcoxon signed test the compared techniques in regards to NOV

NOV	Comparison	Better	Worse	Significance
1/1	GAUAVB-to-SGA	197	43	+
	GAUAVB-to-GAUAV	123	117	≈
	GAUAVB-to-GAUB	208	32	+
1/5	GAUAVB-to-SGA	215	25	+
	GAUAVB-to-GAUAV	150	90	+
	GAUAVB-to-GAUB	198	42	+
5/1	GAUAVB-to-SGA	225	15	+
	GAUAVB-to-GAUAV	93	147	–
	GAUAVB-to-GAUB	220	20	+
5/5	GAUAVB-to-SGA	232	8	+
	GAUAVB-to-GAUAV	161	79	+
	GAUAVB-to-GAUB	228	12	+

Table 6 Wilcoxon signed test the compared techniques in regards to FOC

FOC	Comparison	Better	Worse	Significance
500	GAUAVB-to-SGA	268	52	+
	GAUAVB-to-GAUAV	149	171	≈
	GAUAVB-to-GAUB	255	65	+
2000	GAUAVB-to-SGA	288	32	+
	GAUAVB-to-GAUAV	177	143	+
	GAUAVB-to-GAUB	285	35	+
8000	GAUAVB-to-SGA	313	7	+
	GAUAVB-to-GAUAV	201	119	+
	GAUAVB-to-GAUB	314	6	+

Finally, in order to statistically compare and rank the algorithms altogether, the non-parametric Friedman test, which is similar to the ANOVA parametric, is used with a confidence level of 95% ($\alpha = 0.05$) was used [10, 11]. The null hypothesis was that there is no significant differences among compared algorithms. The computational value of the *p*-value was less than 0.00001. Consequently, there were significant differences among the compared algorithms. Finally, Table 7 shows Freidman test ranks; it supports above mentioned observations.

Table 7 Freidman test average ranks for compared techniques

Algorithm	SGA	GAUAV	GAUB	GAUAVB
Average rank	3.34	1.86	3.14	**1.66**

Table 8 AFPs of compared techniques in regards to the NOV

NOV	SGA (%)	GAUAV (%)	GAUB (%)	GAUAVB (%)
1/1	77.50	77.39	89.74	**90.26**
1/5	45.64	46.63	67.15	**68.83**
5/1	86.13	86.78	92.23	**93.03**
5/5	38.33	40.39	64.96	**67.31**
Average	61.90	62.80	78.52	**79.86**

Table 9 AFPs of compared techniques in regards to the FOC

FOC	SGA (%)	GAUAV (%)	GAUB (%)	GAUAVB (%)
500	58.96	60.29	71.82	**73**
2000	62.90	62.91	79.53	**80.24**
8000	63.96	64.19	83.74	**84.61**
Average	61.94	62.46	78.36	**79.28**

4.2 Comparison Based on Feasibility

In this section, the behaviors of the used algorithms are compared, based on the feasibility of the population. To do this, the average feasibility (AFP) was calculated for each algorithm. AFP indicates how an algorithm can guide its population into the changeable feasible region. Tables 8 and 9 summarize the obtained AFPs, higher values are better and the best are shown as bold and shaded entries.

From Tables 8 and 9, it is clearly observed that GAUAVB achieved higher AFPs, compared with other GAs. GAUAV also slightly achieved better AFPs than SGA when the number of shifted boundaries increased (1/5 and 5/5). It is clear that GAUB and GAUAVB achieved higher AFPs, as they guided the infeasible solution (s) towards the feasible area, by assigning a constraint violation value that guided the selection process. Also, it is clear that the founding feasible area while solving DOPUAVB is getting more complex and harder when NOV increases, especially when the number of changed boundaries increase (Table 7), and the FOC decreases (Table 8).

5 Conclusions and Future Work

Motivated by the literature [11, 12], in this paper we proposed a new type of dynamic optimization problem: single objective unconstrained dynamic optimization problems with unknown active variables and boundaries (DOPUAVBs). In such problems, both the active variables and boundaries of the variables change as time passes. Therefore, DOPUAVB consists of two sub-problems: DOP with

unknown active variables (DOPUAV) and DOP with unknown boundaries (DOPUB).

Moreover, we proposed three genetic algorithms (GA)-based techniques to solve DOPUAVBs. These techniques are GAUAV (GA that deals with unknown active variables), GAUB (GA that deals with unknown changeable boundaries) and GAUAVB (GA that simultaneously deals with unknown active variables and dynamic boundaries). These techniques were compared with each another, as well as with a simple GA (SGA). Based on the quality of obtained solutions and the average of feasibility, as well as statistical tests, results showed that the proposed GAUAVB, that simultaneously considered both sub-problems, was superior to others. This is because GAUAVB had the ability to detect active variables, while also keeping track of feasible boundaries during the course of a search process. Hence it effectively solved DOPUAVBs. The advantages of the proposed technique is using the detected information of the population during the search process to solve the dynamic problem, e.g. the detected feasible boundaries and active variables. However, the disadvantage of GAUAVB is needing for existing of detected feasible boundaries and this would be difficult if the change in boundaries is rapid and has much shift rate.

There are several possible directions for future work. One direction is comparing our proposed algorithms with previously used GAs for DOPs, such as random immigration (RIGAs) and hyper-mutation (HyperM). Regarding sub-problems, we intend to solve each of them in more effective ways. For example, designing an algorithm that would implicitly detect active variables by keeping track of active variables, rather than using the mask process, as it consumes fitness evaluations. This is because the number of fitness evaluations it uses is 2 N, where N is the number of variables.

References

1. AbdAllah, A. F. M., Essam, D. L., & Sarker, R.A. (2014). Solving dynamic optimisation problem with variable dimensions. In *SEAL 2014*. Dunedin, New Zealand: Springer International Publishing.
2. Coello Coello, C. A., & Mezura Montes, E. (2002). Constraint-handling in genetic algorithms through the use of dominance-based tournament selection. *Advanced Engineering Informatics, 16*(3), 193–203.
3. Li, C., Yang, S., Nguyen, T. T., Yu, E. L., Yao, X., Jin, Y., et al. (2009). Benchmark generator for CEC' 2009 competition on dynamic optimization.
4. Surjanovic, S., & Bingham, D. (2015). Virtual library of simulation experiments: Test functions and Datasets. January 2015 [cited 2016 April 20]. Retreived from: http://www.sfu.ca/~ssurjano.
5. Adorio, E. P., & Diliman, U. P. MVF—Multivariate test functions library in C for unconstrained global optimization.
6. Padhye, N., Deb, K., & Mittal, P. An efficient and exclusively-feasible constrained handling strategy for evolutionary algorithms.

7. Morrison, R. W. (2003) Performance measurement in dynamic environments. In *GECCO workshop on evolutionary algorithms for dynamic optimization problems* (pp. 5–8).
8. Yang, S., Nguyen, T. T., & Li, C. (2013). Evolutionary dynamic optimization: Test and evaluation environments. In S. Yang & X. Yao (Eds.), *Evolutionary computation for dynamic optimization problems* (pp. 3–37). Berlin Heidelberg: Springer.
9. Corder, G. W., & Foreman, D. I. (2009) *Nonparametric statistics for non-statisticians: A step-by-step approach*. Wiley.
10. García, S., Molina, D., Lozano, M., & Herrera, F. (2009). A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: A case study on the CEC' 2005 Special Session on Real Parameter Optimization. *Journal of Heuristics, 15* (6), 617–644.
11. Cruz, C., González, J. R., & Pelta, D. A. (2011). Optimization in dynamic environments: a survey on problems, methods and measures. *Soft Computing, 15*(7), 1427–1448.
12. Nguyen, T. T., Yangb, S., & Branke, J. (2012). Evolutionary dynamic optimization: A survey of the state of the art. *Swarm and Evolutionary Computation, 6,* 1–24.