Engineering Science and Technology, an International Journal xxx (xxxx) xxx

HOSTED BY

Contents lists available at ScienceDirect

Engineering Science and Technology, an International Journal

journal homepage: www.elsevier.com/locate/jestch

Experimental analysis of a statistical multiploid genetic algorithm for dynamic environments

Emrullah Gazioğlu [a,*], A.Sima Etaner-Uyar [b]

[a] Istanbul Technical University, Computer and Informatics Department, Istanbul, Turkey
[b] Fatih Sultan Mehmet Vakif University, Computer Engineering Department, Istanbul, Turkey

ARTICLE INFO

Article history:
Received 28 January 2022
Revised 22 April 2022
Accepted 5 May 2022
Available online xxxx

Keywords:
optimization
genetic algorithm
evolutionary computation
dynamic environments
estimation of distribution algorithms
Bayesian Optimization Algorithm

ABSTRACT

Dynamic environments are still a big challenge for optimization algorithms. In this paper, a Genetic Algorithm using both Multiploid representation and the Bayesian Decision method is proposed. By Multiploid representation, an implicit memory scheme is introduced to transfer useful information to the next generations. In this representation, there are more than one genotypes and only one phenotype. The phenotype values are determined based on the corresponding genotypes values. To determine phenotype values, the well-known Bayesian Optimization Algorithm (BOA) has been injected into our algorithm to create a Bayes Network by using the previous population to exploit interactions between variables. With this algorithm, we have solved the well-known Dynamic Knapsack Problem (DKP) with 100, 250, and 500 items. Also, we have compared our algorithm with the most recent algorithm in the literature by using the DKP with 100 items. Experiments have shown that the proposed algorithm is efficient and faster than the peer algorithms in the manner of tracking moving optima without using an explicit memory scheme. In conclusion, using relationships between variables within the optimization algorithms is useful when concerning dynamic environments.

1. Introduction

Combinatorial optimization problems such as University Timetabling Problem, Traveling Salesman Problem (TSP) [4,13], and social networks have a dynamic structure due to their nature. Since the optimum point to be found over time in dynamic environments will change, the intuitive approaches can only be successful if they are well adapted to these environments.

The environmental change might occur on both sides (constraints and/or objective function) of the optimization algorithms. The simplest way to address the change is to restart the algorithm. However, the new optimal solution may not be so far away from the previous one. For this reason, the restarting idea is not useful. Instead, already gained information so far could be useful to adapt to the current environment. To accomplish this adaptation, some dynamic environment criteria should be considered [6]: *(i)*: frequency of change, *(ii)*: severity of the change, *(iii)*: cycle length/cycle accuracy, *(iv)*: predictability of change.

Genetic Algorithms (GAs) are very popular optimization algorithms that are inspired by the biological evolution of the species in nature. Despite the huge success of GAs in the literature, they lose their genetic diversity in changing environments. There are many reasons for that but in this study, we will focus on two of them: *(i)*: losing useful solutions, and *(ii)*: not being able to use relationships between problem variables. To address the reason *(i)*, an implicit memory scheme is developed by implementing a multiploid chromosome structure. To address the reason *(ii)*, a Bayesian Network (BN) is used to exploit relationships between problem variables (a.k.a. genes in a chromosome).

Epistasis means the interaction of genes in a chromosome in real biological life. More clearly, the effect of one gene is dependent on the presence or absence of another gene(s). In this paper, besides multiploidy, to take advantage of interactions of genes, the Bayesian Optimization Algorithm (BOA) [32], one of the well-known Estimation of Distribution Algorithm (EDA), is injected into the proposed Genetic Algorithm variants.

The dynamic version of the well-known Knapsack Problem (KP) is solved to see its effects on real-world problems. In financial management and industry, lots of real-world problems are related to KP. For example, cargo loading, cutting stock, production scheduling, capital budgeting, project selection, and portfolio management [28] are examples of the KP domain [22].

Consequently, in this paper, a GA with both using a statistical method and an implicit memory scheme is proposed to solve

* Corresponding author.
E-mail addresses: egazioglu@itu.edu.tr (E. Gazioğlu), asuyar@fsm.edu.tr (A. Etaner-Uyar).

https://doi.org/10.1016/j.jestch.2022.101173
2215-0986/

Dynamic Optimization Problems (DOP). The proposed method has been examined on a set of highly correlated Dynamic KPs to monitor its behavior in dynamic environments. Then, its performance has been compared to the most recent peer algorithm in the literature. Results show that the proposed method is quite effective while solving DOPs.

This paper continues as follows: In 2, the literature background is investigated. Then, the proposed method is fully described in 3. In 4, constructing the dynamic test environment and the benchmark problems are explained. Experimental results are shown in 5. In 6, the results are briefly discussed and finally, 7 concludes the paper.

2. Literature Summary

2.1. Dynamic Environments

Among the many of them, the main issue in dynamic environments is keeping track of the changing optimum [23]. Some strategies are proposed for this issue [20]: *(i)* Generating diversity when a change occurs, *(ii)* Maintaining diversity continuously, *(iii)* Using multi-population *(iv)* Using Memory-based techniques.

2.2. Genetic Algorithms for Dynamic Environments

Since the simple GA loses its diversity in a dynamic environment, using a memory scheme to save good solutions or using immigrant schemes might be helpful for its adaptation to the newly formed environment.

2.2.1. Immigrant-based Genetic Algorithms

Random Immigrants (RI) scheme is proposed to maintain diversity at all times in [15]. Then, this scheme is applied to the GA, and Random Immigrant GA (RIGA) is proposed [8]. In this approach, at each generation, a portion of the solution candidates is replaced with new randomly generated solution candidates. This kind of technique is usually problematic during the stationary periods, however, shows good performance where the frequency and severity of the change are relatively high. To address problems for stationary periods case, Elitism-based GA (EIGA) is proposed [41] which replaces a portion of the solution candidates nearly mutated versions of the elite individual of the last generation. However, its performance sharply decreases if the severity of the change is high because the individuals in the population start to become similar to each other. To handle this problem, the Hybrid GA (HIGA) is proposed [43] which merges RIGA and EIGA to maintain a balance between exploration and exploitation.

2.2.2. Memory-based Genetic Algorithms

Using memory schemes is another solution method for dynamic environments. Memory scheme can be applied in two ways: *(i) Implicitly* by implementing redundant representation, or *(ii) Explicitly* by implementing extra memory area to save old but good solutions.

Memory/Search GA (MSGA) [41] has two populations: a memory population and a search population. In the beginning, the sizes of the populations are equal. After each generation, the sizes of the populations are updated considering their performance. While the good solutions are saved in the memory population, the search population looks for new solutions in the search space. If a change detected, the algorithm merges the populations, selects a part of the merged population as the first population, and re-initializes the second population. The Memory-enhanced GA (MEGA) and Memory and Random Immigrant GA (MRIGA) are proposed [42]. While in MEGA memory is restarted when a change detected, in MRIGA, the worst solutions in the memory are replaced by random immigrants. Memory-based Immigrants GA (MIGA) is proposed [40]. In MIGA, the memory is re-evaluated at each generation and the best individuals are used for producing immigrants via bitwise mutation.

Most recently, the Environment Reaction GA (ERGA) has been proposed [33]. In ERGA, to enhance the ability to track down the changing optimal point, some memory updating methods and a new reaction policy are investigated.

In [37], the authors proposed the domination GA (domGA) as an implicit memory-using approach. In domGA, solution candidates have one phenotype and two genotypes. A probability vector is used to determine phenotype values based on the corresponding genotype values. In our work, the same approach is applied, however, a different method is used to create a probability vector: while they use a probabilistic learning approach in their work, we let the BN create the probability vector.

2.3. Bayesian Optimization Algorithm

The BOA constructs a BN to evolve a population and sample new individuals [31]. A BN is a directed acyclic graph (DAG) that shows connections between variables (nodes). In BOA, first, randomly a population is formed. Then the population is evaluated to select high-quality solutions via a selection method. Next, a BN is constructed by using selected candidate solutions. After that, by using BN, new candidate solutions are sampled. Last, some or all of the existing population are replaced by new candidate solutions.

2.4. Most Recent Works on Dynamic Optimization

Adaptive Immigrant GA (AIGA) is proposed in [25] which bitwise-mutates the elite solution from the previous generation with a probability of p_m^i which is updated each generation after the evaluation of the effectiveness of immigrants with a given equation for dynamic environments.

Cooperative Co-Evolutionary Algorithms (CCEAs) are adapted to the dynamic environments by introducing the RI scheme, and EI scheme in [2].

The immigrant schemes (e.g., random, elitism-based, hybrid) are applied to the Population-based Incremental Learning (PBIL) and compared to both standard PBIL algorithm and each other to see the effectiveness to solve DOPs in [26].

In [36], the authors used RI and Memory-based Immigrants (MI) in their work for DOPs. In the work, they put immigrants into the population after a change instead of putting every generation.

A BOA-based EA for dynamic environments is proposed in [21]. The proposed method starts with a randomly formed population. In the BOA part, a BN is constructed by using the population, and the next generation is populated via this BN. If a change occurs algorithm retrieves useful solutions from the memory and also retrieves proper network (for BN) translation from the memory and uses it for biasing (modifying) the transition probabilities during BN constructing step.

In [27], the author proposed a PBIL variant with a memory scheme to solve cyclically changing DOPs.

3. Proposed Algorithm

For solving DOPs and adapting to the new environment, we propose Multiploid GA (MGA) which has three components: The first, Bayesian Decision mechanism to determine the phenotype values of the individuals by injecting BOA source code into MGA, the second, multiploid representation of individuals as implicit memory scheme, and finally the third, random and/or Bayesian Immigrant

schemes. To see the effects of the components, eight different variants of the proposed algorithm are implemented by switching on and off random and Bayesian immigrant schemes and setting the number of genotypes to two different values, 2 (diploidy) and 4 (multiploidy):

- Bayesian Diploid GA (BDGA) (No immigrant)
- Bayesian Immigrant Diploid GA (BIDGA) (Bayesian Immigrant only)
- Bayesian Random-**only** Immigrant Diploid GA (BRoIDGA) (Random Immigrant only)
- Bayesian Random Immigrant Diploid GA (BRIDGA) (Both Bayesian and Random Immigrant)
- Bayesian Multiploid GA (BMGA) (No immigrant)
- Bayesian Immigrant Multiploid GA (BIMGA) (Bayesian Immigrant only)
- Bayesian Random-**only** Immigrant Multiploid GA (BRoIMGA) (Random Immigrant only)
- Bayesian Random Immigrant Multiploid GA (BRIMGA) (Both Bayesian and Random Immigrant)

In MGA, the GA part is used to carry out the optimization process, BOA is used for *(i)*: determining phenotype values, and *(ii)*: creating immigrants by sampling new individuals using its own BN. The related pseudo-code can be seen in 1.

3.1. Explanation of MGA

Foremost, MGA employs a multiploid representation, which means that each solution candidate has multiple genotypes but only one phenotype. The goal is, here, to create an implicit memory scheme that will pass on what has been learned so far to future generations. To do this, all genetic operators are only used on genotypes, the phenotype is used only to evaluate fitness value [37].

Algorithm 1: Pseudocode for MGA

Algorithm 1: Pseudocode for MGA

$\eta \leftarrow number\ of\ genotypes$;
set $BOA\ immigrant\ scheme\ (optional)$;
set $Random\ immigrant\ scheme\ (optional)$;
$pop \leftarrow init(popSize, \eta)$;
while *termination condition not met* **do**
  $best \leftarrow eval(pop)$;
  $BN \leftarrow constructBN(pop)$;
  $auXpop \leftarrow sampleBN(BN)$;
  $probVec \leftarrow constructPrbVec(auXpop)$;
  $tournamentSelection(pop)$;
  $uniformCrossover(pop)$;
  $bitwiseMutation(pop)$;
  $geno2pheno(pop)$\{
    if $\sum geno_j^g = \eta$ **then** $pheno_j \leftarrow 1$;
    else if $\sum geno_j^g = 0$ **then** $pheno_j \leftarrow 0$;
    else
      $p \leftarrow rand()$;
      $pheno_j \leftarrow (p < probVec_j)? : 1 : 0$;
  \}
  if $mod(generation, \tau) = 0$ **then**
    $changeEnv()$;
return $best$

The pseudocode of the MGA is shown in 1. Also, its flowchart is shown in 1. First, the number of genotypes is determined (see Section 5.2) and BOA and/or Random immigrant scheme is set optionally.

Next, the genotypes of the solution candidates are generated randomly in the beginning, and phenotype values are assigned randomly *-since a BN is not constructed yet-* unless corresponding genotype values are the same. Then the fitness values are calculated, and by using the best $k\%$ of them, a BN is constructed. The BN is then used to populate an aux-population (short for auxiliary-population) which has the same size as MGA has. Last, similar to PBIL does, a probability vector is built using the aux-population. If at least one of the corresponding genotype values are different, the probability vector is employed to determine the phenotype value for the particular gene.

The standard tournament selection (size of four) is applied in the MGA's selection phase: Randomly chosen four solution candidates are compared and the fittest one is passed on to the next generation. This operation is performed until the size of the next generation is satisfied. Next, the BOA and/or Random immigrant schemes are performed if they are set. Simply, for the BOA immigrant scheme, a portion of the solution candidates in the aux-population are moved to the main population randomly, and for the Random immigrant scheme, a portion of the randomly-chosen solution candidates from the main population is switched with the solution candidates are generated at random to maintain diversity. Without recalculating its fitness value, the elite solution candidate from the last iteration is passed on to the current generation (elitism). In addition, the immigrant steps are carried out in this step according to the algorithm variants mentioned in Section 3.

The uniform crossover approach is employed for crossover operation. It begins by randomly generating a mask vector in binary and then selecting two individuals, say i_1, and i_2, at each iteration. For each variable j (genes in chromosome), if the mask vector's corresponding value is 1, i_1's 1st genotype's j^{th} variable is swapped with i_2's 2nd genotype's j^{th} variable with a probability of p_c. This procedure is performed on each pair of genotypes in each solution candidate separately.

The simple bitwise mutation is employed as a mutation tool. The genotypes of each solution candidate are inverted with a probability of p_m in the bitwise mutation process.

The phenotype construction stage follows the genetic operators. If the genotype values are the same, as seen in 1, this value is also used for the phenotype as well. Otherwise, the probability vector of the BOA is used to determine if it will be one or zero.

4. Constructing Dynamic Test Environments

4.1. Dynamic Benchmark Test Environment Generator

The XOR Generator proposed in [39] is a dynamic environment generator with different difficulty degrees for any binary encoded stationary problem. Assume $\vec{X}$ is a binary encoded solution candidate for a problem, then the fitness value of that solution candidate is calculated for time t as shown in (1).

$$f\left(\vec{X}, t\right) = f\left(\vec{X} \oplus \vec{M}_k\right) \tag{1}$$

where $\oplus$ is the XOR operator and $\vec{M}_k$ is a masking vector for the k^{th} environment. In the beginning, mask $\vec{M}$ is initialized with zeros. After that, every τ generations, it is updated as shown in (2).

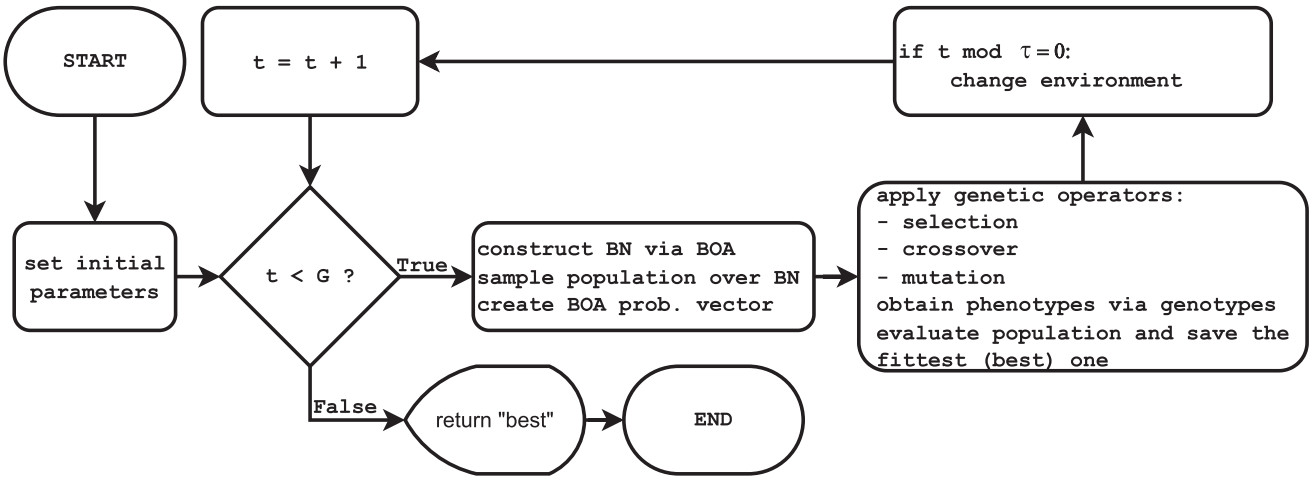

Fig. 1. Flowchart of MGA.

$$\vec{M}_k = \vec{M}_{k-1} \oplus \vec{T}_k \tag{2}$$

where $\vec{T}_k$ is a binary template.

Besides the above generator (*random environment*), there are also *cyclic* and *cyclic with noise* environments which are proposed in [44]. In order to construct a cyclic environment, we first construct K binary encoded templates[1] $\vec{T}(0), \ldots, \vec{T}(K-1)$ randomly but exclusively: Assume that each template is a row of a matrix, then the number of total ones in a column of the matrix must be 1. The first mask $\vec{M}(0)$ is composed of zeros, then the rest of the XOR masks are constructed as shown in (3).

$$\vec{M}(i+1) = \vec{M}(i) \oplus \vec{T}(i\%K), \qquad i = 0, \ldots..2K-1. \tag{3}$$

With the above formula, after K environmental changes, the $\vec{M}(K)$ will be all ones and then the K base states will be reused to construct next K masks till to return to the environment $\vec{M}(0) = \vec{0}$.

By using the above cyclic environment generator, Yang and Yao also introduced the cyclic environment with noise [44] via introducing noise to the next mask by using bitwise flipping with a small probability, called p_n.

4.2. Dynamic Knapsack Problem

Dynamic Knapsack Problem (DKP), proposed in [29], is the dynamic form of the 0/1 KP,and its aim is to collect as many items as possible to maximize profit while ensuring that the knapsack's total weight does not surpass the capacity allotted C [35]. The KP is an NP-hard combinatorial optimization problem that can be seen in real-world scenarios such as choosing investments or resource allocation. [22,10,5,30,38,7,24]. 4 shows the formula of the 0/1 KP.

$$\begin{aligned} &max \qquad f(x) = \sum_{i=1}^{n} p_i * x_i \\ &subject\ to \sum_{i=1}^{n} w_i x_i \leqslant C \qquad x_i \in \{0,1\}, \end{aligned} \tag{4}$$

where x_i denotes the decision value in binary format, indicating if the item i is collected or not, p_i denotes the item i's profit value, w_i denotes the item i's weight, and C is the capacity.

[1] Each template should contain $\rho \times l = l/K$ ones.

Because the correlation between profits and weights influences the DKP's complexity, strongly correlated datasets are formed for this work as it is explained in [29].

5. Experimental Studies

5.1. Design

Before testing the MGA, first, some key parameters are set. For a fair comparison, these parameters are set to exact same values as used in [33]. The frequency of change, τ, is set to 20 and 40. The severity of the change, ρ, is set to 0.1, 0.2, 0.5, 1.0. At this point, it should be indicated that the results given for point 1.0 are not very realistic since in the real world there is no such an oscillation state. However, still, the reason we tested our algorithm for point 1.0 is to conform to the literature. The probability of noise p_n is set to 0.05 for the noisy cyclic environment.Also, each algorithm has 100 fitness function calculations per generation. And the number of generations, is calculated as follows: $G = \tau \times T$, where $T = 60$. Thus, we achieve the number of fitness calculations per generation specified in [33].

Besides the above adjustments, the other parameters' settings can be seen in 1. These settings are commonly accepted values in the literature by experiments, except p_n, which is based on [39].

A total of 72 test environments are generated by using three DKP datasets, four different ρ values, two different τ values, and three different environment schemes. Then, these 72 environments are solved by variants of the proposed method mentioned in Section 3.50 independent runs (N) with the same set of seeds were executed for each test. The best-of-generation for each run is saved [44], and total performance is calculated as seen in 5.

$$\overline{F}_{BOG} = \frac{1}{G}\sum_{i=1}^{G}\left(\frac{1}{N}\sum_{j=1}^{N} F_{BOG_{ij}}\right)$$

where G denotes the number of generations, N denotes the number of total runs, $F_{BOG_{ij}}$ is the i^{th} generation's best-of-fitness value of the j^{th} run, and finally, $\overline{F}_{BOG}$ denotes total offline performance.

5.2. Preliminary tests for parameter tuning

Before running the actual test, first, the best number of genotypes is investigated. To get a reliable result, the p_m is set to 0.01, and the Bayesian Immigrant rate, β, and Random Immigrant rate,

Table 1
Parameter settings of MGA.

parameter	value
number of generation (G)	$\tau \times T$
population size	100
mutation probability (p_m)	0.1
crossover probability (p_c)	1.0
probability of noise (p_n)	0.05
tournament size for selection	4

Γ, are set to 0.0. Also, for the test, 25 independent runs were executed for the DKP-500 problem. As depicted in 2, 4 is the best option for the number of genotypes.

After fixing the number of genotypes to 4, next, the algorithm is tested to determine the best mutation probability value on the DKP-500 problem. For the sake of reliability, again, β and Γ are set to 0.0. The results can be seen in 3. Results show that using 0.1 is the best option for the probability of mutation, p_m.

Finally, the last two tests had applied for determining the BOA immigrant rate, β, and the random immigrant rate, Γ. These two tests had applied for both Diploid and Multiploid algorithms (in total four tests). Based on the results, it is concluded that setting both values to 0.1 will give the best result.

5.3. Experimental results

After fixing crucial parameters of the proposed algorithm, three DKP are solved with eighth variants of the MGA mentioned earlier. The corresponding results are shown in 4 for $\tau = 20$ (Results for $\tau = 40$ is not given in this paper due to the page limitation).

When we check 4, we can come up with some conclusions:

While looking up to results from top to bottom, it can be seen that the general performances of the algorithms are decreases, naturally, since the problems are getting harder.

While looking up to results from left to right, it is seen that both performances and the behaviors are almost the same for cyclic and random environments. However, due to the noise effect, for the cyclic with noise environments, the algorithms performed close to each other except for minor differences.

Diploid vs. Multiploid: Although BIDGA and BRIDGA perform slightly better than the other diploid ones, still worse than the multiploid ones and we have already concluded that in 2 earlier.

When we look at only the multiploid ones, we see that the performances of the algorithms with the Bayesian Immigrant scheme are getting decreases while the severity of the problem is increases and vice versa. This is because the Bayesian Immigrant scheme tends to remain in the local optimum by intensification.

To compare the results to the ERGA's results which are provided in [33], we chose only BMGA among the eight variants of the MGA. Before interpreting the results given in 5, we remind you, again, that the results given for point 1.0 are not very realistic. When checking the results in this context, except for two scenarios, BMGA gave much better results than ERGA. From here, it is understood that BMGA's implicit memory system and Bayesian Decision method are more effective for DOPs than ERGA's open memory approach.

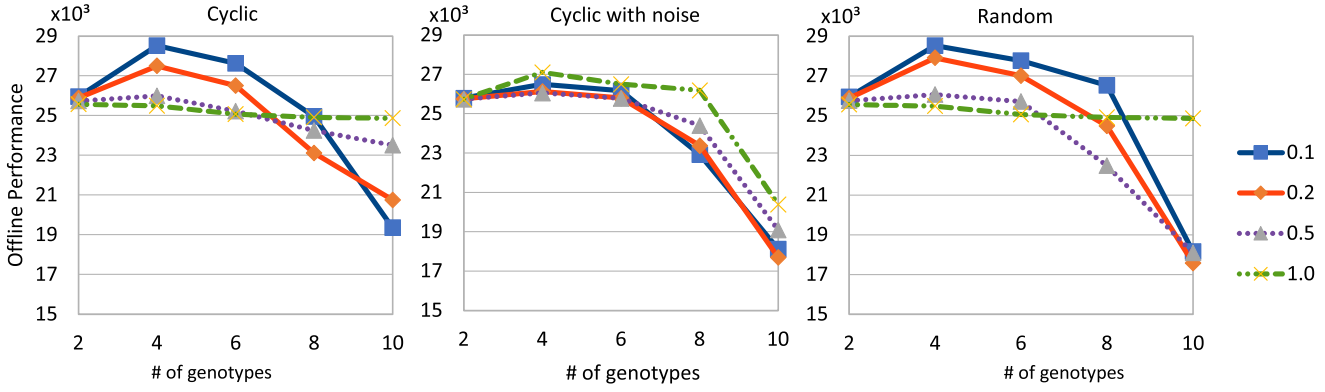

Fig. 2. Sensitivity test for determining the number of genotypes.

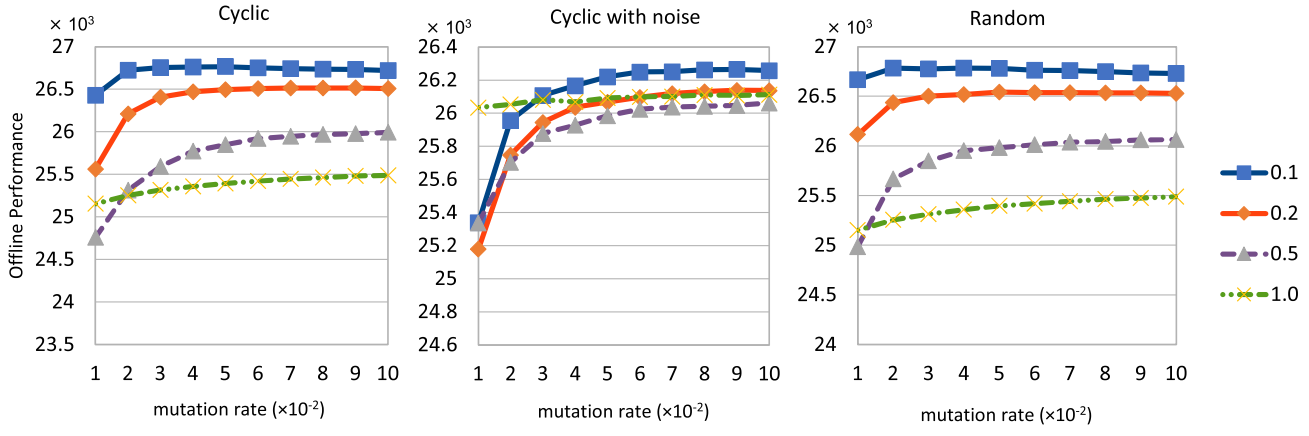

Fig. 3. Sensitivity test for determining probability of mutation.

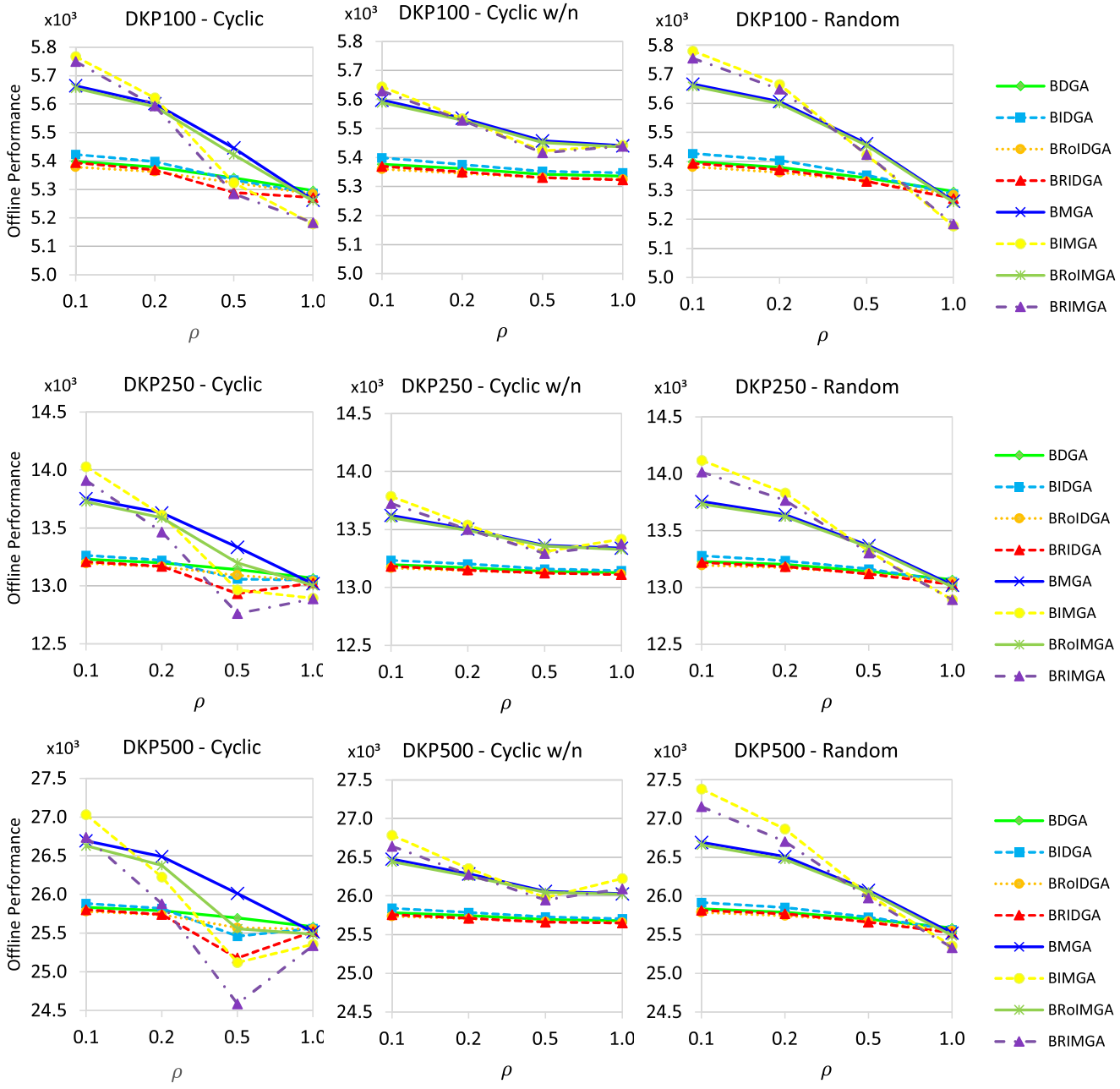

Fig. 4. Offline performances of the algorithms on DKP 100, 250 and 500 items under random, cyclic and cyclic with noise environments with $\tau = 20$ and $\rho = 0.1, 0.2, 0.5$ and 1.0.

5.4. Non-parametric statistical tests

To apply a statistical test for multiple comparisons, for each test case, the "1 $\times$ *k multiple comparison test*" [9] is used, where, k indicates the number of approaches.

In the beginning, by applying the Friedman test [12], an algorithm is highlighted as a controller via selecting the lowest (best) rank as the control algorithm. Then, a set of post hoc procedures [11,18,16,19,17,34] are applied to mark the significantly different algorithm. This operation returns an $n \times m$ matrix, and, here, n is the number of algorithms except the highlighted one and m is the number of post hoc procedures. Each post hoc method returns a set of p-values for the control algorithm versus the rest which enables us to decide which algorithms are significantly different than the highlighted algorithm by checking whether p-values $< \alpha$ (level of significance: 0.05). The results are shown in 2.

The first thing that catches your eye in the table is that Diploid algorithms are more successful at $\rho = 1.0$ point, except in cyclic with noisy environments. This is because the system is oscillating and the Bayesian decision-making stage is called more in Multiploid algorithms than Diploid algorithms, and consequently it affects the phenotype representation more. Therefore, Diploid algorithms are more likely to revert to the previous environment while in the oscillating state. As proof of this, when we look at the results for cyclic with noise environments, it can be seen that the Bayesian decision-making stage is more invoked in Multiploid algorithms and corrects genes distorted by noise better than Diploid algorithms.

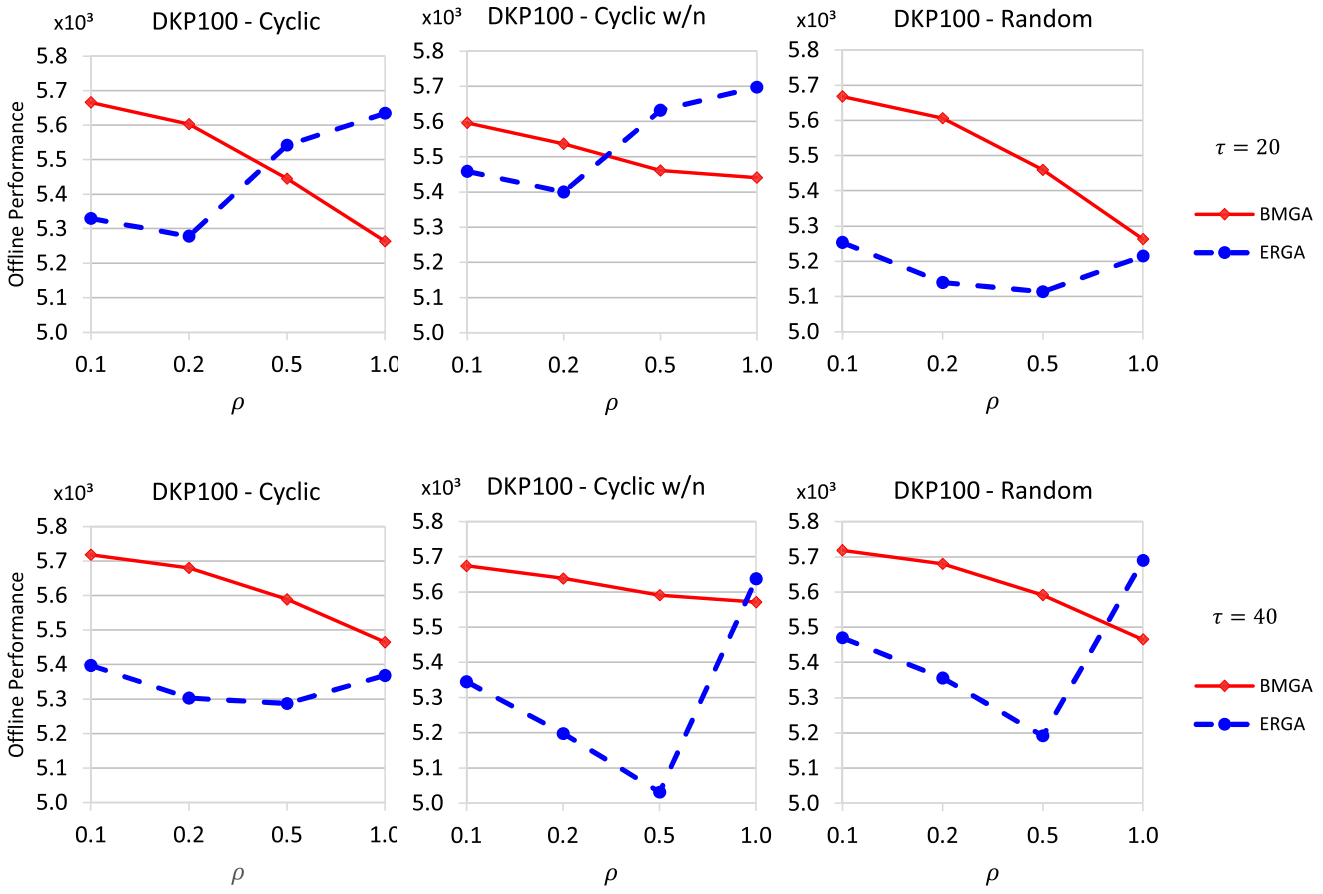

Fig. 5. Offline performances of the ERGA and BMGA on DKP-100 under random, cyclic and cyclic with noise environments with $\tau = 20$ and 40 and $\rho = 0.1, 0.2, 0.5$ and 1.0.

Table 2
Non-parametric statistical test results.

Friedman	**DKP 100**				**DKP 250**				**DKP 500**			
Cyclic, $\rho \Rightarrow$	0.1	0.2	0.5	1.0	0.1	0.2	0.5	1.0	0.1	0.2	0.5	1.0
τ **= 20**	BIMGA BRIMGA	BMGA BIMGA	BRoIMGA BMGA	BDGA BIDGA	BIMGA	BIMGA BMGA	BMGA	BDGA BIDGA	BIMGA	BMGA	BMGA	BDGA BIDGA
τ **= 40**	BIDGA BRIMGA	BMGA BRIMGA	BRoIMGA BMGA	BDGA BIDGA	BIMGA BRIMGA	BIMGA BRIMGA	BRoIMGA BMGA	BRIDGA	BIMGA BRIMGA	BIMGA BRIMGA	BMGA BRoIMGA	BRIDGA
Cyclic w/n, $\rho \Rightarrow$	0.1	0.2	0.5	1.0	0.1	0.2	0.5	1.0	0.1	0.2	0.5	1.0
τ **= 20**	BIMGA BRIMGA	BRoIMGA BMGA	BMGA BDGA	BRIMGA BIMGA	BIMGA BRIMGA	BIMGA BMGA	BRoIMGA BMGA	BIMGA BRIMGA	BRIMGA BMGA	BIMGA BMGA	BRoIMGA BMGA	BIMGA
τ **= 40**	BMGA BRIMGA	BRoIMGA BMGA	BMGA BRoIMGA	BMGA BRIMGA	BIMGA BRIMGA	BRIMGA	BMGA BRIMGA	BIMGA BRIMGA	BIMGA BRIMGA	BRIMGA	BRIMGA	BRIMGA BIMGA
Random, $\rho \Rightarrow$	0.1	0.2	0.5	1.0	0.1	0.2	0.5	1.0	0.1	0.2	0.5	1.0
τ **= 20**	BIMGA BRIMGA	BIMGA BRIMGA	BMGA BRoIMGA	BDGA BIDGA	BIMGA	BIMGA BRIMGA	BMGA BRoIMGA	BDGA BIDGA	BIMGA	BIMGA	BMGA BRoIMGA	BDGA BIDGA
τ **= 40**	BRIMGA	BRIMGA	BMGA BRoIMGA	BRIDGA	BMGA BIMGA	BMGA BRIMGA	BMGA BRIMGA	BRIDGA	BRIMGA BIMGA	BIMGA BRIMGA	BRIMGA	BRIDGA

6. Result and Discussion

To see the effectiveness of the mechanisms implemented, we created eight variants of the algorithm by switching on and off these particular mechanisms, namely, random immigrant scheme (on/off), Bayesian immigrant scheme (on/off), diploid or multiploid.

With these eight variants, three DKPs with 100, 250, and 500 items are solved. To observe the performances of the algorithms, a dynamic environment generator is used. By observing the test results, several conclusions can be drawn.

First, the proposed algorithm's structure improved simple GA's performance for dynamic environments. Second, GAs with RI perform better than the GAs with Bayesian Immigrant ones in most cases, since Bayesian Immigrant causes local optima.

After comparing these eight variants to each other, one of them is (BMGA) selected by its performance, for comparing to a GA-based algorithm (ERGA) proposed by a recent study by solving the DKP-100 problem [33]. Results show that BMGA has a huge advantage over the ERGA. Further, knowing that the ERGA's performance is better than other well-known state-of-the-art algorithms

(RIGA, EIGA, HIGA, MIGA, MEGA, MRIGA, MSGA), it is easy to understand that BMGA is a good choice to solve DOPs.

7. Conclusion and Future Works

In this paper, a GA-based optimization algorithm is developed for dynamic environments. First, multiploid representation is introduced into GA to get an implicit memory scheme: In the proposed algorithm, each candidate solution has four genotypes and one phenotype. Thus, we can reuse old but good solutions. For comparison, also a diploid version of the algorithm is implemented. Second, a well-known EDA member, the BOA is embedded into the structure to exploit interactions between variables (members of the candidate solutions) by constructing a BN. This BN is used for two purposes: *(i):* Determining phenotype values of the candidate solutions according to their corresponding genotype values, and, *(ii):* sampling new candidate solutions as a whole for introducing the Bayesian Immigrant scheme. Last, a random immigrant scheme is implemented to maintain diversity.

Since the proposed method works only on the binary representation domain, in future work, we plan to inject the Real-Coded BOA [1] into the proposed method for real-coded and integer representations (which are the limitations of the proposed method) in order to solve any other DOPs, such as Dynamic Graph Coloring Problem [3] and/or continuous optimization problems [45,14].

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] C.W. Ahn, R.S. Ramakrishna, D.E. Goldberg, Real-coded bayesian optimization algorithm: Bringing the strength of boa into the continuous world, in: Genetic and Evolutionary Computation Conference, Springer, 2004, pp. 840–851.
[2] C.K. Au, H.F. Leung, Cooperative coevolutionary algorithms for dynamic optimization: an experimental study, Evolutionary Intelligence 7 (2014) 201–218.
[3] Barba, L., Cardinal, J., Korman, M., Langerman, S., Renssen, A.v., Roeloffzen, M., Verdonschot, S., 2017. Dynamic graph coloring, in: Workshop on Algorithms and Data Structures, Springer. pp. 97–108..
[4] A. Benyamin, S.G. Farhad, B. Saeid, Discrete farmland fertility optimization algorithm with metropolis acceptance criterion for traveling salesman problems, International Journal of Intelligent Systems 36 (2021) 1270–1303.
[5] D. Blado, A. Toriello, Relaxation analysis for the dynamic knapsack problem with stochastic item sizes, SIAM Journal on Optimization 29 (2019) 1–30.
[6] J. Branke, Evolutionary optimization in dynamic environments, volume 3, Springer Science & Business Media, 2012.
[7] C. Changdar, G. Mahapatra, R.K. Pal, An improved genetic algorithm based approach to solve constrained knapsack problem in fuzzy environment, Expert Systems with Applications 42 (2015) 2276–2286.
[8] H.G. Cobb, J.J. Grefenstette, Genetic algorithms for tracking changing environments Technical Report, Naval Research Lab, Washington DC, 1993.
[9] J. Derrac, S. García, D. Molina, F. Herrera, A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms, Swarm and Evolutionary Computation 1 (2011) 3–18.
[10] Y. Feng, G.G. Wang, L. Wang, Solving randomized time-varying knapsack problems by a novel global firefly algorithm, Engineering with Computers 34 (2018) 621–635.
[11] H. Finner, On a monotonicity problem in step-down multiple test procedures, Journal of the American Statistical Association 88 (1993) 920–923.
[12] M. Friedman, A comparison of alternative tests of significance for the problem of m rankings, The Annals of Mathematical Statistics 11 (1940) 86–92.
[13] F.S. Gharehchopogh, B. Abdollahzadeh, An efficient harris hawk optimization algorithm for solving the travelling salesman problem, Cluster Computing (2021) 1–25.
[14] M.J. Goldanloo, F.S. Gharehchopogh, A hybrid obl-based firefly algorithm with symbiotic organisms search algorithm for solving continuous optimization problems, The Journal of Supercomputing (2021) 1–34.
[15] J.J. Grefenstette et al., Genetic algorithms for changing environments, in: PPSN, 1992, pp. 137–144.
[16] Y. Hochberg, A sharper bonferroni procedure for multiple tests of significance, Biometrika 75 (1988) 800–802.
[17] B.S. Holland, M.D. Copenhaver, An improved sequentially rejective bonferroni test procedure, Biometrics (1987) 417–423.
[18] S. Holm, A simple sequentially rejective multiple test procedure, Scandinavian journal of statistics (1979) 65–70.
[19] G. Hommel, A stagewise rejective multiple test procedure based on a modified bonferroni test, Biometrika 75 (1988) 383–386.
[20] Y. Jin, J. Branke, Evolutionary optimization in uncertain environments-a survey, IEEE Transactions on evolutionary computation 9 (2005) 303–317.
[21] M. Kaedi, N. Ghasem-Aghaee, C.W. Ahn, Biasing the transition of bayesian optimization algorithm between markov chain states in dynamic environments, Information Sciences 334 (2016) 44–64.
[22] H. Kellerer, U. Pferschy, D. Pisinger, Multidimensional knapsack problems, in: Knapsack problems, Springer, 2004, pp. 235–283.
[23] E.C. Li, X.Q. Ma, Dynamic multi-objective optimization algorithm based on prediction strategy, Journal of Discrete Mathematical Sciences and Cryptography 21 (2018) 411–415.
[24] L.F.M. López, N.G. Blas, A.A. Albert, Multidimensional knapsack problem optimization using a binary particle swarm model with genetic operations, Soft Computing 22 (2018) 2567–2582.
[25] Mavrovouniotis, M., Yang, S., 2013. Genetic algorithms with adaptive immigrants for dynamic environments, in: 2013 IEEE Congress on Evolutionary Computation, IEEE. pp. 2130–2137..
[26] M. Mavrovouniotis, S. Yang, Population-based incremental learning with immigrants schemes in changing environments, in: 2015 IEEE Symposium Series on Computational Intelligence, IEEE, 2015, pp. 1444–1451.
[27] M. Mavrovouniotis, S. Yang, Direct memory schemes for population-based incremental learning in cyclically changing environments, in: European Conference on the Applications of Evolutionary Computation, Springer, 2016, pp. 233–247.
[28] R. Mendes, A. Paiva, R.S. Peruchi, P.P. Balestrassi, R.C. Leme, M. Silva, Multiobjective portfolio optimization of arma–garch time series based on experimental designs, Computers & Operations Research 66 (2016) 434–444.
[29] Z. Michalewicz, J. Arabas, Genetic algorithms for the 0/1 knapsack problem, in: International Symposium on Methodologies for Intelligent Systems, Springer, 1994, pp. 134–143.
[30] F.B. Ozsoydan, Artificial search agents with cognitive intelligence for binary optimization problems, Computers & Industrial Engineering 136 (2019) 18–30.
[31] M. Pelikan, Bayesian optimization algorithm: From single level to hierarchy, University of Illinois at Urbana-Champaign Urbana, IL, 2002.
[32] M. Pelikan, D.E. Goldberg, K. Sastry, Bayesian optimization algorithm, decision graphs, and occam's razor, in: Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation, Morgan Kaufmann Publishers Inc., 2001, pp. 519–526.
[33] S. Qian, Y. Liu, Y. Ye, G. Xu, An enhanced genetic algorithm for constrained knapsack problems in dynamic environments, Natural Computing (2019) 1–20.
[34] D.M. Rom, A sequentially rejective test procedure based on a modified bonferroni inequality, Biometrika 77 (1990) 663–665.
[35] V. Roostapour, A. Neumann, F. Neumann, On the performance of baseline evolutionary algorithms on the dynamic knapsack problem, in: International Conference on Parallel Problem Solving from Nature, Springer, 2018, pp. 158–169.
[36] R. Tinós, S. Yang, Artificially inducing environmental changes in evolutionary dynamic optimization, in: International Conference on Parallel Problem Solving from Nature, Springer, 2016, pp. 225–236.
[37] A.Ş. Uyar, A.E. Harmanci, A new population based adaptive domination change mechanism for diploid genetic algorithms in dynamic environments, Soft Computing 9 (2005) 803–814.
[38] H. Wu, S. Qian, Y. Liu, D. Wang, B. Guo, An immune-based response particle swarm optimizer for knapsack problems in dynamic environments, Soft Computing 24 (2020) 15409–15425.
[39] Yang, S., 2004. Constructing dynamic test environments for genetic algorithms based on problem difficulty, in: Evolutionary Computation, 2004. CEC2004. Congress on, IEEE. pp. 1262–1269..
[40] S. Yang, Memory-based immigrants for genetic algorithms in dynamic environments, in: Proceedings of the 7th annual conference on Genetic and evolutionary computation, ACM, 2005, pp. 1115–1122.
[41] S. Yang, Genetic algorithms with elitism-based immigrants for changing optimization problems, in: Workshops on Applications of Evolutionary Computation, Springer, 2007, pp. 627–636.
[42] S. Yang, Genetic algorithms with memory-and elitism-based immigrants in dynamic environments, Evolutionary Computation 16 (2008) 385–416.
[43] S. Yang, R. Tinós, A hybrid immigrants scheme for genetic algorithms in dynamic environments, International Journal of Automation and Computing 4 (2007) 243–254.
[44] S. Yang, X. Yao, Population-based incremental learning with associative memory for dynamic environments, IEEE Transactions on Evolutionary Computation 12 (2008) 542–561.
[45] H.R.R. Zaman, F.S. Gharehchopogh, An improved particle swarm optimization with backtracking search optimization algorithm for solving continuous optimization problems, Engineering with Computers (2021) 1–35.