Procedia Computer Science
Volume 29, 2014, Pages 1926–1936
ICCS 2014. 14th International Conference on Computational Science

# A Hybrid Harmony Search Algorithm for Solving Dynamic Optimisation Problems

Ayad Mashaan Turky[1,2*], Salwani Abdullah[1] and Nasser R. Sabar[3]
[1] *Universiti Kebangsaan Malaysia, 43600 Bangi, Selangor, Malaysia.*
[2] *Faculty of Information & Communication Technologies, Swinburne University of Technology, Victoria 3122, Australia.*
[3] *The University of Nottingham Malaysia Campus, Jalan Broga, 43500 Semenyih, Selangor, Malaysia.*

**Abstract**
Many optimisation problems are dynamic in the sense that changes occur during the optimisation process, and therefore are more challenging than the stationary problems. To solve dynamic optimisation problems, the proposed approaches should not only attempt to seek the global optima but be able to also keep track of changes in the track record of landscape solutions. In this research work, one of the most recent new population-based meta-heuristic optimisation technique called a harmony search algorithm for dynamic optimization problems is investigated. This technique mimics the musical process when a musician attempts to find a state of harmony. In order to cope with a dynamic behaviour, the proposed harmony search algorithm was hybridised with a (i) random immigrant, (ii) memory mechanism and (iii) memory based immigrant scheme. The performance of the proposed harmony search is verified by using the well-known dynamic test problem called the Moving Peak Benchmark (MPB) with a variety of peaks. The empirical results demonstrate that the proposed algorithm is able to obtain competitive results, but not the best for most of the cases, when compared to the best known results in the scientific literature published so far.

*Keywords:* Harmony search algorithm, Dynamic optimization problems, Meta-Heuristic

# 1 Introduction

Optimisation problems can usually be categorised as either static or dynamic [1]. In static optimisation problems, related information such as the problem parameters are known in advance. Dynamic optimisation problems however present a great challenge to the research community since the problem parameters are either revealed or changed during the course of the on-going optimisation

* *Corresponding author. E-mail address: ayadalrashid@gmail.com*

 Selection and peer-review under responsibility of the Scientific Programme Committee of ICCS 2014

doi:10.1016/j.procs.2014.05.177

[2, 3]. In the last decade, population based methods have proven to be to be successful in tackling dynamic optimisation problems [4-6] and such achievements have not considered to be surprising as they deal with a population of solutions that are scattered over the whole search space [7]. However, population based methods that were developed to solve static optimisation problems are considered as infeasible options when it comes to handling dynamic optimisation problems. Over the years, it has become evident that in order to cope with problem dynamism, population-based methods have to integrate some mechanisms that would adaptively modify their behaviours to accommodate changes in the problems. One of the most notable example in literature is to increase the population diversity when the changes are detected [8, 9]. A number of population-based methods, such as Genetic Algorithm (GA) [10], Particle Swarm Optimisation Algorithm (PSO) [11, 12] and Differential Evolution (DE) [13] have been employed for dynamic optimisation problems.

The successes of the above population-based methods are the main motivating factors for proposing a new population-based method that is based on Harmony Search Algorithm (HSA) for dynamic optimisation problems. HSA is a recent population stochastic search algorithm that simulates musician rules when playing music [14]. Over the years, HSA have been successfully used to solve several static optimisation problems [15] and it is worth considering its applicability for solving dynamic optimisation problems. The main aim of this research work therefore is studying the application of HSA to solve dynamic optimisation problems. However, like other population based methods the direct application of HSA on dynamic problems would be impractical. Thus the HSA is hybridised with a (i) random immigrant (HSA-I), (ii) memory mechanism (HSA-M) and (iii) memory based immigrant scheme (HSA-MI) in order to maintain the population diversity [10]. The motivation to conduct this study is due to the fact that even though HSA has never been used to solve dynamic optimisation problems but the distinguishing feature of HSA [16, 17] is that it is free from divergence because it uses a stochastic random search. This feature allows HSA to move away from a common point and helps to prevent being trapped in the local optima. Apart from that, HSA is also able to overcome the drawback of the building block in genetic algorithms by taking into account all solutions when generating new solutions instead of only using two parents (which is the usual case in genetic algorithm). The HSA procedure for generating new solutions allows HSA to have the ability in dealing with both discrete and continuous variables.

In order to demonstrate the applicability of the proposed HSA in dealing with dynamic optimisation problems, the well-known Moving Peaks Benchmark problem (MPB) [18] is considered in this work [19, 20] [21].

# 2 Problem Description: Moving peak benchmark

Moving Peak Benchmark (MPB) is a well-known dynamic optimisation problem and has been widely studied in literature [5, 22]. In MPB, the fitness landscape dynamically changes. Solution landscape in MPB comprises a set of peaks. Each peak has its own *height*, *width* and *location*. Hence, each peak is determined based on the value of its *height*, *width* and *location*. The values of these parameters keep changing as the solving progresses, thus there is a change in the location of the global optima. For the D-dimensional landscape, the fitness of each peak is defined as a maximisation function as in Eq. 1.

$$F(\vec{x},t) = \max_{i=1,\dots,p}\left(\frac{H_i(t)}{1+W_i(t)\sum_{j=1}^{5}(x_j - X_{ij}(t))^2}\right) \tag{1}$$

where $H_i$ *(t)* and $W_i$ *(t)* are the height and width of peak *i* at time *t*, respectively, and $X_{ij}$ is the $j^{th}$ element of the location of peak *i* at time *t*. *P* represents the number of peaks.

During the solving process, the position of each peak is shifted to a random direction by a vector $\vec{v}_i$ of a distance *s* (*s* also refers to a shift length which determines the severity of the dynamics problem). The movement of a single peak is performed as in Eq. 2.

$$\vec{v}_i(t) = \frac{s}{|\vec{r} + \vec{v}_i(t-1)|}((1-\lambda)\vec{r} + \lambda \vec{v}_i(t-1)) \tag{2}$$

where the shift vector $\vec{v}_i(t)$ is a linear combination of a random vector $\vec{r}$ and the previous shift vector $\vec{v}_i$ *(t-1)* and is normalised to the shift length *s*. The parameter $\lambda$ is set to 0, which implies that the movements of the peak are uncorrelated. Precisely, a change of a single peak can be defined as in Eq. 3 to Eq. 5.

$$H_i(t) = H_i(t-1) + height\_severity * \sigma \tag{3}$$

$$W_i(t) = W_i(t-1) + width\_severity * \sigma \tag{4}$$

$$\vec{X}_i(t) = \vec{X}_i(t)(t-1) + \vec{v}_i(t) \tag{5}$$

where $\sigma$ is a normal distributed random number with a zero mean and variation of 1. The MPB parameters are presented in Table 1. The *U* in Table 1 refers to a change frequency. Initially, the parameter values of all the peaks are randomly generated with the given boundaries as shown in Table 1. Thus, the change occurs when the height and width of the peak randomly shifts within the given boundaries.

| ***Parameters*** | ***Description*** | ***Value*** |
|---|---|---|
| *P* | Number of peaks | 10 |
| *U* | Change frequency | 5000 |
| *Height severity* | Height severity | 7.0 |
| *Width severity* | Width severity | 1.0 |
| *Peak shape* | Peak shape | Cone |
| *s* | Shift length | 1.0 |
| *D* | Number of dimensions | 5 |
| $\lambda$ | Correlation coefficient | 0 |
| *S* | Each dimension boundaries | [0,100] |
| *H* | Peak Height | [30.0,70.0] |
| *W* | Peak Width | [1,12] |

**Table 1:** Standard MPB parameter setting [18]

# 3 Proposed Method

This section describes the basic harmony search algorithm for dynamic optimisation problems. The mechanisms that have been used to maintain the population diversity and their hybridisation with the harmony search algorithm (coded as hybrid harmony search) are also presented.

## 3.1 Harmony Search Algorithm (HSA)

The HSA is one of the newest stochastic population-based meta-heuristic optimisation algorithms proposed by Geem at al. [14]. HSA mimics the musical process where musicians attempt to find a state of harmony through the improvisation process. The improvisation process tries to find a better harmony by playing existing harmony, refining the current one or generating a new harmony. The latest harmony will then be evaluated by aesthetic standards, either to accept or to discard it. This process is similar to the optimisation process where the solution for the considered problem is refined step by step in order to find a better one which is assessed by the objective function. The process of HSA comprises five steps which are [14]:

- **Step 1**: **Initialize HSA parameters**. This step is concerned with stetting the main parameters of the HSA which are: Harmony memory size (HMS), Harmony memory consideration rate (HMCR). Pitch adjustment rate (PAR) and the Maximum number of generations (MNI).
- **Step 2: Initialize the harmony memory (HM).** HM contains a set of solution and its size is equal to the HMS. In this step, HSA randomly creates a set of solutions and then add them to the HM.
- **Step 3**: **Improvises a new solution**. This step generates (improvises) a new solution from scratch according to HMCR and the PAR values where decision variables of the new solution either are selected from HM or randomly created.
- **Step 4: Update HM.** This step compares the fitness value of the new generated solution with the worse one in HM. The worse solution in HM will be replaced by the new one if the new one has a better fitness value.
- **Step 5: The termination condition**. This step decides whether to terminate HSA if the maximum number of iterations is reached or starts a new iteration (go to Step 2).

## 3.2 Population Diversity Mechanisms

This section presents three different mechanisms that are embedded within the HSA with an aim to maintain the population diversity. The common feature between these mechanisms is that all of them store a pool of solutions and these solutions will be used during the course of optimization in maintaining the HSA diversity by replacing some of the HM solutions. However, the differences between them are the way they generate the pool of solutions, types of solution to be kept, and the updating strategies (details are discussed below). Since each mechanism has its own strengths and weaknesses, it is believed that different mechanisms are needed to cope with the environment changes that occur. Thus the strengths of several mechanisms can be combined under one framework in order to appreciate their effectiveness.

i. **Random Immigrant Mechanism.** This mechanism has been widely referred to in the read literature to maintain the population diversity within the evolutionary algorithms [10]. The idea is quite simple as at each of the iteration a subset of solutions is generated at random and is used to replace the worst solutions in the harmony memory. Hence, the number of solutions to be replaced affects the performance of the search process. A smaller number may be enough to diversify the search while a larger number may cause too much diversification which may lead the search to jump on to a different area. However, there is no universal size for the number of replaced solutions, *rs*. In this work, the number of solutions were fixed to be replaced at every iteration as *rs*=HMS*0.2 (as in [19]).

**ii. Memory Based Mechanism.** This mechanism keeps a subset of best solutions [18]. These solutions will be re-inserted in the harmony memory once changes are detected (in contrast to a random immigrant where solutions are randomly generated from scratch). Although, a random immigrant can ensure a high population diversity, it is not suitable for cyclic changes because this will direct the search process into a different area rather than go back to the previous search space [18]. In this work, an explicit memory is used to store the best solution of the current HM. The size of the memory, *ms* is calculated as *ms*=0.1*HM (as in [19]). Once the change in the environment is detected, solutions that are stored in the memory will replace the bad solutions in the HM with a size that equals to *ms*.

**iii. Memory Based Immigrant Mechanism.** It can be seen that a random immigrant is good in ensuring high population diversity, whilst, a memory based mechanism is efficient in directing the search into the previous search space. The selection of which mechanism to be used usually depends on the changes because different changes may require different mechanisms. In this work, the idea of hybridising a random immigrant and a memory based mechanism in order to maintain the harmony memory (population) diversity is utilised. The hybridised mechanism works as follows: at every improvisation step, a set of solutions, *s*, is selected from the memory, where *s*=0.1*HM [19]. The selected solutions are mutated with a probability *pm*=0.01 [19]. Then, the mutated solutions will replace the bad solutions in harmony memory with size equalling to *s*.

## 3.3 Hybridised Harmony Search Algorithm with a Diversity Mechanism

To cope with the dynamic changes, the proposed algorithm needs to keep track of the changes [10] for example by maintaining the population diversity during the search process. This is needed because the changes in the problem may change the current local optima into global optima and vice versa [2]. In addition, it is also shown in the literature that the developed algorithms for stationary problems cannot be directly used to solve dynamic problems [2]. Therefore, in order to handle this problem, the harmony search algorithm has been hybridised with three population diversity mechanisms (as presented in Section 3.2) i.e., (i) HSA with random immigrant, HSA-I, (ii) HSA with memory mechanism, HSA-M, and (iii) HSA with memory based immigrant mechanism.

# 4 Experimental Setup

In this section, the parameter settings of HSA and the problem description (moving peak benchmark) are provided and the results of hybrid HSA with the three mechanisms as well as the comparisons with state of the art are discussed.

## 4.1 HSA Parameters

A preliminary test was conducted to determine the appropriate values by taking into account the best results and the computational time. The problem of five peaks is used to determine the HSA parameter values. The parameter values of HSA are presented in Table 2. Please note that, to assure a fair comparison with the state-of-the-art, the number of improvisations or the terminal condition is fixed as in [19].

| *Parameters* | *Description* | *Tested range* | *Suggested value* |
|---|---|---|---|
| HMS | Harmony memory size HMS= 1 to 100 | 10-200 | 100 |
| HMCR | Harmony memory consideration rate (0 < HMCR < 1) | 0.1-0.99 | 0.6 |
| RCR | Random consideration rate | - | RCR=1-HMCR |
| PAR | Pitch adjustment rate (0 < PAR < 1) | 0.1-0.99 | 0.3 |
| NI | Number of improvisations or iterations | - | 500000 function evaluations |

**Table 2:** HSA parameter values

The discussion on the obtained results is divided into two sections (i) comparison between hybrid HSA with different diversity mechanisms, and (ii) comparison with the state-of-the-art. The experiments were run 50 times with different seed numbers. The quality of the result represents the offline error that is calculated based on Eq. 7 as suggested by [2]:

$$\mu = \frac{1}{k}\sum_{k=1}^{k}(h_k - f_k) \tag{7}$$

where $h_k$ is the optimum value of the $k^{th}$ environment. $f_k$ is the best solution obtained before the $k^{th}$ environmental changes. $\mu$ is the average of all differences between $h_k$ and $f_k$ over the environmental changes. $K$ is the total number of environment changes. For each run, there are 100 environment changes ($K$=100), which result in $K \times U$ = 100×5000 fitness evaluations. All the results reported are based on the average of over 50 independent runs with different random seeds.

## 4.1.1. Comparison between Hybrid HSA with Different Diversity Mechanisms

The results of the hybrid HSA with the three different mechanisms, denoted as HSA-I, HSA-M, and HSA-MI, are presented in this section. Note that the details on these hybrids HSA are presented in Section 4. Table 3 presents the offline error and the standard deviation (std) over 50 runs. In order to assess the capability of HSA-I, HSA-M, and HSA-MI when dealing with different problem sizes (different number of peaks), each of them were tested by using a different number of peaks. As highlighted in the literature, the size of the tested peaks varies between 1 and 200. From the results, we can deduce that, in terms of the offline error, HSA-MI outperforms HSA-I and HSA-M on all cases. Considering the standard deviation (std), HSA-MI obtains better results than HSA-I and HSA-M on both the 10 out of 11cases. This is mainly due to the combination of immigrant and memory-based mechanisms that able to complement each other. Such results are also consistent with the reviewed literature [10] that states that the combination of these two mechanisms with genetic algorithm yields better results than just combining the genetic algorithm in isolation with each diversity mechanism.

| *Number of Peaks* | *HSA-I* | *HSA-M* | *HSA-MI* |
|---|---|---|---|
| 1 | 0.30<br>±0.27 | 0.23<br>±0.21 | **0.15**<br>±0.17 |
| 2 | 0.32<br>±0.31 | 0.30<br>±0.31 | **0.23**<br>±0.29 |
| 5 | 0.92<br>±0.70 | 0.81<br>±0.90 | **0.66**<br>±0.19 |

| | | | |
|---|---|---|---|
| 7 | 0.81 ±0.13 | 0.82 ±0.10 | **0.70** ±0.22 |
| 10 | 3.00 ±1.94 | 2.81 ±2.02 | **0.90** ±1.19 |
| 20 | 2.16 ±1.14 | 2.23 ±1.34 | **1.51** ±1.01 |
| 30 | 2.00 ±1.07 | 2.06 ±0.77 | **1.52** ±0.76 |
| 40 | 2.09 ±1.13 | 2.58 ±1.16 | **1.53** ±0.81 |
| 50 | 2.32 ±1.11 | 2.05 ±0.92 | **1.57** ±0.67 |
| 100 | 2.00 ±0.70 | 1.92 ±0.94 | **1.39** ±0.74 |
| 200 | 1.92 ±0.90 | 2.02 ±0.87 | **1.17** ±0.51 |
| *Note: Bold fonts indicate the best results* | | | |

**Table 3:** Offline error of hybrid HSA with the three different mechanisms

The results were further analysed by conducting a Wilcoxon test to examine if there was any significant difference between the proposed algorithm with the significance interval 95% ($\alpha = 0.05$). A pair comparison was executed as follows:

- HSA-MI vs. HSA-I
- HSA-MI vs. HSA-M

Table 4 shows the *p*-values for the MPB. The presented *p*-values show enough evidence to conclude that there is a significant difference between the algorithms in comparison, in which only 1 and 2 cases are not significant for the "HSA-MI vs. HSA-I" and "HSA-MI vs. HSA-M", respectively.

| ***HSA-MI vs.*** | ***HSA-I*** | ***HSA-M*** |
|---|---|---|
| ***Instances*** | ***p-value*** | ***p-value*** |
| 1 | 0.001 | 0.013 |
| 2 | 0.007 | **0.237** |
| 5 | **0.119** | **0.559** |
| 7 | 0.022 | 0.001 |
| 10 | 0.000 | 0.000 |
| 20 | 0.005 | 0.003 |
| 30 | 0.033 | 0.000 |
| 40 | 0.004 | 0.000 |
| 50 | 0.000 | 0.004 |
| 100 | 0.000 | 0.002 |
| 200 | 0.000 | 0.000 |
| ***Average*** | 0.017 | 0.074 |
| *Note: Bold fonts indicate HSA-MI is not significantly better.* | | |

**Table 4:** *p*-values of Wilcoxon test for MPB

## 4.1.2. Comparison with state-of-the-art

The comparison between these hybridisation approaches has shown that the HSA-MI is the best algorithm. A further investigation on the performance (offline error ± standard deviation (std)) of the HSA-MI was conducted by comparing it with the state-of-the-art approaches. The algorithms in comparison are presented in Table 5.

| # | *Symbol* | *References* |
|---|---|---|
| 1 | *CPSO* | [19] |
| 2 | *mCPSO* | [20] |
| 3 | *mQSO* | [20] |
| 4 | *mCPSO** | [20] |
| 5 | *mQSO** | [20] |
| 6 | *SOS+LS* | [23] |
| 7 | *CDE* | [24] |
| 8 | *DynPopDE* | [25] |

**Table 5:** Acronyms of compared methods

The results of the comparison are presented in Table 6. The best results are presented in bold. The overall comparison shows that the approach used is able to obtain seven new best results out of eleven tested datasets. The approaches used in this study can be considered to be more reliable when compared with other approaches (except with CPSO) on all datasets. The higher the number of peaks would normally cause the problem to be more complex in solving. However, this complexity does not degrade the performance of the HSA-MI. It is proven that where HSA-MI is still able to obtain the better results with the number of peaks equalling to 200.

| | ***Number of Peaks*** | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ***Algorithm*** | 1 | 2 | 5 | 7 | 10 | 20 | 30 | 40 | 50 | 100 | 200 |
| ***HSA-MI*** | 0.15 ±0.17 | 0.23 ±0.29 | **0.66** ±0.19 | **0.70** ±0.22 | **0.90** ±1.19 | **1.51** ±1.01 | **1.52** ±0.76 | 1.53 ±0.81 | 1.57 ±0.67 | **1.39** ±0.74 | **1.17** ±0.51 |
| ***CPSO*** | **0.14** ±0.11 | **0.20** ±0.19 | 0.72 ±0.30 | 0.93 ±0.30 | 1.05 ±0.24 | 1.59 ±0.22 | 1.58 ±0.17 | **1.51** ±0.12 | **1.54** ±0.12 | 1.41 ±0.08 | 1.24 ±0.06 |
| ***mCPSO*** | 4.93 ±0.17 | 3.36 ±0.26 | 2.07 ±0.08 | 2.11 ±0.11 | 2.08 ±0.07 | 2.64 ±0.07 | 2.63 ±0.08 | 2.67 ±0.07 | 2.65 ±0.06 | 2.49 ±0.04 | 2.44 ±0.04 |
| ***mQSO*** | 5.07 ±0.17 | 3.47 ±0.23 | 1.81 ±0.07 | 1.77 ±0.07 | 1.80 ±0.06 | 2.42 ±0.07 | 2.48 ±0.07 | 2.55 ±0.07 | 2.50 ±0.06 | 2.36 ±0.04 | 2.26 ±0.03 |
| ***mCPSO**** | 4.93 ±0.17 | 3.36 ±0.26 | 2.07 ±0.11 | 2.11 ±0.11 | 2.05 ±0.07 | 2.95 ±0.08 | 3.38 ±0.11 | 3.69 ±0.11 | 3.68 ±0.11 | 4.07 ±0.09 | 3.97 ±0.08 |
| ***mQSO**** | 5.07 ±0.17 | 3.47 ±0.23 | 1.81 ±0.07 | 1.77 ±0.07 | 1.75 ±0.06 | 2.74 ±0.07 | 3.27 ±0.11 | 3.60 ±0.08 | 3.65 ±0.11 | 3.93 ±0.08 | 3.86 ±0.07 |
| ***SOS+LS*** | - | - | - | - | 3.41 | - | - | - | 2.62 | - | 1.88 |
| ***CDE*** | - | - | - | - | 0.92 ±0.07 | - | - | - | - | - | - |
| ***DynPopDE*** | - | - | 1.03 ±0.13 | - | 1.39 ±0.07 | - | - | - | 2.10 ±0.06 | 234 ±0.05 | 2.44 ±0.05 |

**Table 6:** offline error (± standard deviation (std)) of algorithms on the MPB problems with a different number of peaks

The results obtained were further analysed by conducting a series of multi comparison statistical tests, (Friedman and Iman-Davenport) with a significant interval of 95% (α = 0.05) to check whether there was a significant difference between HSA-MI and the compared methods (CPSO, mCPSO, mQSO, mCPSO* and mQSO*) [26]. Note that, only those methods that have been tested on all cases are considered in this test. For the statistical analysis, the Friedman's test was applied, followed by Holm and Hochberg tests as post-hoc methods (if significant differences are detected) to obtain the adjusted *p*-values for each comparison between the control algorithm (the best-performing one) and the rest. The *p*-value computed by the Friedman's test is 0.000, which is below the significant interval of 95% (α = 0.05). This value shows that there is a significant difference among the observed results. Table 7 summarises the ranking obtained by the Friedman's test that shows HSA-MI is ranked as first. The post-hoc methods (Holm's and Hochberg's test) were also run with HSA-MI as a control algorithm. Table 8 shows the adjusted *p*-values which reveals that HSA-MI is better than (mCPSO, mQSO, mCPSO* and mQSO*) with α = 0.05. Although the statistical test shows that HSA-MI is not better than CPSO, however, the results in Table 6 demonstrate that HSA-MI is able to obtain less offline error for 7 out of the 11 instances as compared to CPSO (obtain 4 best results).

| # | Algorithm | Ranking |
|---|---|---|
| 1 | HSA-MI | 1.36 |
| 2 | CPSO | 1.63 |
| 3 | mQSO | 3.63 |
| 4 | mCPSO | 4.36 |
| 5 | mQSO* | 4.63 |
| 6 | mCPSO* | 5.36 |

**Table 7:** Average ranking of Friedman test

| # | Algorithm | Unadjusted P | P Holm | P Hochberg |
|---|---|---|---|---|
| 1 | mCPSO* | 5.300E-7 | 2.660E-6 | 2.66E-6 |
| 2 | mQSO* | 4.085E-6 | 1.634E-4 | 1.634E-4 |
| 3 | mCPSO | 1.694E-4 | 5.083E-4 | 5.083E-4 |
| 4 | mQSO | 4.385E-3 | 8.771E-3 | 8.771E-3 |
| 5 | CPSO | 7.324E-1 | 7.324E-1 | 7.324E-1 |

**Table 8:** Adjusted *p*-values of the compared methods

The main competitor for the MPB in this case is CPSO (the algorithm of [19]). The performance of HSA-MI over CPSO when dealing with different shift severities (*s*) was further tested. Note that in this experiment, the shift severity was set between 0.0 and 6.0. The results given in Table 9 demonstrate that HSA-MI is able to obtain better results than the CPSO (as presented in bold).

| Shift Severities(s) | HSA-MI | CPSO |
|---|---|---|
| 0.0 | **0.64** ±0.27 | 0.80 ±0.21 |
| 1.0 | **0.90** ±1.19 | 1.05 ±0.24 |
| 2.0 | 1.32±0.32 | 1.17 ±0.22 |
| 3.0 | 1.47 ±0.31 | 1.36 ±0.28 |
| 4.0 | **1.33** ±0.35 | 1.38 ±0.29 |
| 5.0 | **1.37** ±0.39 | 1.58 ±0.32 |
| 6.0 | **1.42** ±0.33 | 1.53 ±0.29 |

**Table 9:** Comparison on offline error with different shift severities

# 5 Conclusion

The overall goal of the work presented in this paper is to investigate the performance of the hybrid harmony search algorithm in maintaining the population diversity in addressing dynamic optimisation problems, particularly moving peak benchmark. In this work, three kinds of population diversity mechanisms are presented i.e. the random immigrant, memory mechanism and memory based immigrant mechanism. Initial experiments show that the memory based immigrant mechanism outperformed the random immigrant and memory mechanisms (in isolation) in maintaining the population diversity, and was able to outperform other available approaches on seven out of the eleven datasets. In conclusion, this approach which is considered simple yet effective has managed to produce a number of better results. This indicates the importance of the population-based approaches to maintain the population diversity especially when dealing with dynamic optimisation problems since the changes occur during the optimisation course, thus the algorithm should be able to keep track of these changes.

# References


[1] E. G. Talbi, *From design to implementation*: Wiley Online Library, 2009.

[2] J. Branke and H. Schmeck, "Designing evolutionary algorithms for dynamic optimization problems," *Advances in Evolutionary Computing,* pp. 239-262, 2003.

[3] E. G. Talbi, "A taxonomy of hybrid metaheuristics," *Journal of Heuristics,* vol. 8, pp. 541-564, 2002.

[4] Y. Jin and J. Branke, "Evolutionary optimization in uncertain environments-a survey," *Evolutionary Computation, IEEE Transactions on,* vol. 9, pp. 303-317, 2005.

[5] C. Cruz, J. R. González, and D. A. Pelta, "Optimization in dynamic environments: a survey on problems, methods and measures," *Soft Computing-A Fusion of Foundations, Methodologies and Applications,* vol. 15, pp. 1427-1448, 2011.

[6] M. Hadwan, M. Ayob, N. R. Sabar, and R. Qu, "A harmony search algorithm for nurse rostering problems," *Information Sciences,* vol. 233, pp. 126-140, 2013.

[7] S. Yang, Y. S. Ong, and Y. Jin, *Evolutionary computation in dynamic and uncertain environments*: Springer Verlag, 2007.

[8] H. G. Cobb and J. J. Grefenstette, "Genetic algorithms for tracking changing environments," DTIC Document1993.

[9] H. G. Cobb, "An investigation into the use of hypermutation as an adaptive operator in genetic algorithms having continuous, time-dependent nonstationary environments," DTIC Document1990.

[10] S. Yang, "Genetic algorithms with memory-and elitism-based immigrants in dynamic environments," *Evolutionary Computation,* vol. 16, pp. 385-416, 2008.

[11] D. Parrott and X. Li, "Locating and tracking multiple dynamic optima by a particle swarm model using speciation," *Evolutionary Computation, IEEE Transactions on,* vol. 10, pp. 440-458, 2006.

[12] W. Du and B. Li, "Multi-strategy ensemble particle swarm optimization for dynamic optimization," *Information sciences,* vol. 178, pp. 3096-3109, 2008.

[13] R. Mendes and A. S. Mohais, "DynDE: a differential evolution for dynamic optimization problems," in *The 2005 IEEE Congress on Evolutionary Computation, 2005.*, 2005, pp. 2808-2815 Vol. 3.

[14] Z. W. Geem and J. H. Kim, "A new heuristic optimization algorithm: harmony search," *Simulation,* vol. 76, pp. 60-68, 2001.

[15] Z. W. Geem, *Recent advances in harmony search algorithm* vol. 270: Springer Verlag, 2010.

[16] B. Alatas, "Chaotic harmony search algorithms," *Applied Mathematics and Computation,* vol. 216, pp. 2687-2699, 2010.

[17] S. Das, A. Mukhopadhyay, A. Roy, A. Abraham, and B. K. Panigrahi, "Exploratory power of the harmony search algorithm: analysis and improvements for global numerical optimization," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on,* vol. 41, pp. 89-106, 2011.

[18] J. Branke, "Memory enhanced evolutionary algorithms for changing optimization problems," in *Evolutionary Computation Congress CEC 99.*, 1999, p. 1882 Vol. 3.

[19] S. Yang and C. Li, "A clustering particle swarm optimizer for locating and tracking multiple optima in dynamic environments," *Evolutionary Computation, IEEE Transactions on,* vol. 14, pp. 959-974, 2010.

[20] T. Blackwell and J. Branke, "Multiswarms, exclusion, and anti-convergence in dynamic environments," *Evolutionary Computation, IEEE Transactions on,* vol. 10, pp. 459-472, 2006.

[21] K. Trojanowski and S. T. Wierzchon, "Immune-based algorithms for dynamic optimization," *Information sciences,* vol. 179, pp. 1495-1515, 2009.

[22] A. M. Turky and S. Abdullah, "A multi-population harmony search algorithm with external archive for dynamic optimization problems," *Information Sciences,2014,* vol. http://dx.doi.org/10.1016/j.ins.2014.02.084.

[23] D. Ayvaz, H. R. Topcuoglu, and F. Gurgen, "Performance evaluation of evolutionary heuristics in dynamic environments," *Applied Intelligence,* pp. 1-15, 2011.

[24] M. C. du Plessis and A. P. Engelbrecht, "Using competitive population evaluation in a differential evolution algorithm for dynamic environments," *European Journal of Operational Research,* 2011.

[25] M. C. du Plessis and A. P. Engelbrecht, "Differential evolution for dynamic environments with unknown numbers of optima," *Journal of Global Optimization,* pp. 1-27, 2012.

[26] S. García, A. Fernández, J. Luengo, and F. Herrera, "Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power," *Information sciences,* vol. 180, pp. 2044-2064, 2010.