On the performance of a hybrid genetic algorithm in dynamic environments

Quan Yuan [*], Zhixin Yang

Department of Mathematics, Wayne State University, Detroit, MI 48202, USA

Abstract

The ability to track the optimum of dynamic environments is important in many practical applications. In this paper, the capability of a hybrid genetic algorithm (HGA) to track the optimum in some dynamic environments is investigated for different functional dimensions, update frequencies, and displacement strengths in different types of dynamic environments. Experimental results are reported by using the HGA and some other existing evolutionary algorithms in the literature. The results show that the HGA has better capability to track the dynamic optimum than some other existing algorithms.

Key words: Hybrid genetic algorithm (HGA), dynamic environments, optimization

1 Introduction

Traditionally, research on optimization problems has been focused on stationary cases, i.e., problems are precisely given in advance and remain fixed during the optimization processes. However, in many practical optimization problems, a wide range of uncertainties should be taken into account [1–4]. For example, the fitness evaluation is subject to noise, the design variables are subject to perturbations or changes after the optimal solution has been determined (e.g. due to manufacturing tolerances). The environmental conditions may change

[*] Corresponding author.
Email addresses: `quanyuan@wayne.edu` (Quan Yuan), `zhixin.yang@wayne.edu` (Zhixin Yang).

over time due to some factors such as the stochastic arrival of new tasks, machine faults, climatic changes, or financial issues. These problems are called dynamic optimization problems.

In this paper, we focus on a situation that the fitness function is deterministic at any point in time, but is dependent on time t. Usually, for stationary optimization problems, the aim is to design an optimization algorithm that can quickly and precisely locate the optimum solution(s) in the search space. However, for dynamic optimization problems, the situation is quite different. Since the optimum may change over time, this kind of optimization problems has *dynamically changing optima*. An optimization algorithm used in this dynamic environments should be able to adapt itself during optimization. That is, it should not only find the optima but also track their trace in the solution space as closely as possible rather than requiring a repeated restart of the optimization process.

A genetic algorithm (GA) is a generic population-based meta-heuristic optimization algorithm. Candidate solutions to the optimization problem play the role of individuals (parents) in a population. Some mechanisms inspired by biological evolution: selection, crossover and mutation are used. The fitness function determines the environment within which the solutions "survive". Then new groups of the population (children) are generated after the repeated application of the above operators. GAs have attracted great interests in the past decades. They are often used for solving stationary optimization problems and there are also many successful applications. Since GAs only use fitness as an indicator to operate individuals in population, they have implicit parallelism, robustness, and better ability of global searching than other heuristics [5]. Moreover, GAs are population-based and stochastic algorithms. So they have adaptivity in some way and researchers think they can be applied on dynamic optimization problems after some modification.

Solving dynamic optimization problems using GAs has received increasing interest. Some methods have been proposed in recent years. These methods can be roughly grouped into three classes:

1. Modify or adapt some parameters of GAs [6–8].
2. Memorize the past using information of GAs [9–12], explicit/implicit memory [13–15], or multiple populations [16, 17].
3. Maintain population diversity by, such as, inserting randomly generated individuals [18], niching [19], or reformulating the fitness function considering the age of individuals [20] or the entropy of the population [21].

For more information, we refer to [22], [23], and the references there in.

Hybrid genetic algorithms (HGAs) belong to a type of GA in which a local search is embedded as an auxiliary. The genetic search is in charge of the broad search, while the local search is referred to as the deep search. The results obtained by the HGA are often better than those gained by a GA without a local search [24]. It is naturally thought that a GA used in dynamic environments can be embedded a local search to improve its performance. And the local search can also help the algorithm track the trace of optima to a certain extent. However, to the best of the authors' knowledge, using HGA in dynamic environments is studied by a few researchers. Garrett and Walker's work [25] is the only one the authors can find.

Recently, a new kind of HGA, which can solve a class of stationary optimization problems, has been proposed by Yuan *et al.* [26]. A new strategy which has adaptive ability is applied in this HGA. In this paper, we will investigate the algorithm's performance when it is applied in dynamic environments and compare its performance with some other existing algorithms. The HGA is described in Section 2. New dynamic environments are presented in Section 3. The results and conclusions are presented in Sections 4 and 5, respectively.

2 The hybrid genetic algorithm

In this section, we will describe the HGA in [26]. Consider the following optimization problem:

$$\min\ f(\boldsymbol{x})\ \ s.t.\, \boldsymbol{x} \in S \tag{1}$$

where S is an n-dimensional box constraint, i.e., $S = \{\boldsymbol{x} = (x_i)_{n\times 1} \in \mathbb{R}^n | a_i \leq x_i \leq b_i, i = 1, \cdots, n\}$, and $f(\boldsymbol{x})$ is a continuous and multimodal function on S.

Let

$$x_i = \frac{(a_i + b_i)}{2} + \frac{(a_i - b_i)}{2} \sin y_i\, ,\ y_i \in \mathbb{R}\, ,\ i = 1, \ldots, n.$$

Then problem (1) can be transformed into an unconstrained optimization problem

$$\min\ f(\boldsymbol{y})\ \ s.t.\, \boldsymbol{y} \in \mathbb{R}^n \tag{2}$$

For the HGA used in this study, an individual X_i is an object variable $\boldsymbol{y}$.

The following steps of the HGA are proposed:

Step 1 (Initialization) Define the *population size* N and the *maximal generation number*; randomly generate N feasible *individuals* as the *initial population* $\overrightarrow{X}(0) = \{X_1, X_2, \cdots, X_N\}$; set the initial generation number $t_{gen} = 0$.

Step 2 (Local-search) Perform a local search for every individual X_i in $\overrightarrow{X}(t_{gen})$ to obtain the local optimum $X_i^{\#}$.

Step 3 (Evaluation) Evaluate X_i in $\overrightarrow{X}(t_{gen})$ by $f(X_i^{\#})$, X_i itself is not changed.

Step 4 (a) Selection: Choose N pairs of parents.

(b) Reproduction (Crossover or Mutation): for pairs of parents Y_1 and Y_2, who have the same fitness value and have some local optimum $X_i^{\#}$, a mutation step is performed to generate a child; otherwise, a crossover step is performed.

Selection and reproduction will generate a new population.

Step 5 Stop the algorithm if the maximal generation number is reached, or let $t_{gen} = t_{gen} + 1$, and return to Step 2.

We use a traditional optimization method, such as the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm, in Step 2. *Tournament selection* is performed in Step 4(a). In Step 4(b), the method of crossing two parents Y_1 and Y_2 is as below:

$$Y' = \alpha Y_1 + (1-\alpha)Y_2 \tag{3}$$

where α is a random number satisfying $0 \le \alpha \le 1$.

The mutation step should be designed in order to let the individuals have the chance to be better. But excessive mutation will destruct some good individuals. So in the HGA, the mutation step is executed as

$$Y' = Y + \pi \cdot (Y_1 - Y_2) \tag{4}$$

where Y is either one of the two parents. If the norm $\|Y_1 - Y_2\|$, where $\|\cdot\|$ denotes the Euclidian norm, is too small, a random $\boldsymbol{d} = (d_1, \ldots, d_n)$ is generated where each d_i is taken 1 or -1. We take this mutation step because in the later stage of the HGA, some individuals will tend to converge. And the norm $\|Y_1 - Y_2\|$ in (4) will decrease. At this situation, the mutation step will be less destructive to some good individuals.

The mechanism of the crossover and mutation operations are based on avoiding "premature". Yuan et al. in [26] tried to let the population scatter on the feasible set. They considered two individuals are "similar" if a local search can find the same local optimum starting from both of them; in another words, they belong to the same local optimums "neighborhood". If two parents selected by a GA are similar, the mutation step is executed to generate a new child. The GA tries to explore a new area of the feasible domain. Otherwise, if two parents are not similar, a crossover step is performed. The crossover and mutation steps may find a better region that includes a better local optimum, and which can be found by a local search. A further detailed discussion of the HGA in [26] can be found in [27].

The following aspects are the motivations that we obtained by reviewing the steps of the HGA mentioned above.

- From the mechanism of the crossover and mutation operations, we can find that the algorithm is self-adapted and can maintain the population's diversity. This characteristic may be utilized in dynamic environments.

- The HGA will find the global optimum successfully only if the points obtained from the algorithm drop into the neighborhood of the global optimum. Therefore, when the environment has changed, if the global optimum is varied slightly, the HGA could still track it successfully because of the function of the local search.

Based on these motivations, we will make a slight revision of the HGA and apply this HGA in some dynamic environments. The details are mentioned in the next Section.

3 Dynamic environments

The dynamic optimization problem can be described as:

$$\min\ f(\boldsymbol{x}, t) \quad \boldsymbol{x} \in S,\ t \in T \tag{5}$$

The problem depends on both $\boldsymbol{x}$ and an additional parameter $t \in T$ (the *time*). Generally, the objective function might be different after each function evaluation. In this paper, we assume that the functional change is observable and the objective function remains constant within specific time intervals $[t_k, t_k + \Delta t_k)$ $(k = 0, 1, 2, \ldots)$. Moreover, the dynamics of the objective function and the dynamics of the genetic algorithm are synchronized by identifying t with the number of evaluations of the algorithm and by keeping f constant. Furthermore, $\Delta t_k =: \Delta g$ (Δg denotes the number of the evaluations between the time intervals) is also assumed to be constant, such that the objective function changes every Δg evaluations in case of a genetic algorithm.

An often used benchmark problem in some literature to test GA in dynamic environments (such as [28]) is

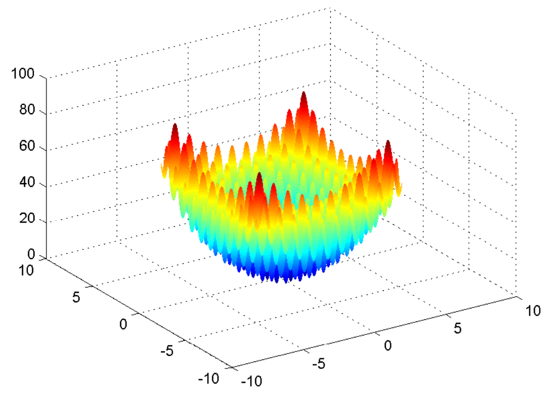
$$f(\boldsymbol{x}, t) = \sum_{i=1}^{n} (x_i - \delta_i(t))^2.$$

This benchmark is too simple for the HGA described in this paper. Instead, three dynamical environments are derived from the model:

$$f(\boldsymbol{x}) = 10n + \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i)], \tag{6}$$

The figure of (6) when $n = 2$ on $[-5.12,\ 5.12] \times [-5.12,\ 5.12]$ is shown in Figure 1. The global optimum is $(0, 0, \ldots, 0)^T$ and the optimum value is 0.

Fig. 1. The figure of (6) when $n = 2$ on $[-5.12, 5.12] \times [-5.12, 5.12]$



The dynamical environments are generated by translating the base function along three different types of dynamics (linear, circular, and random) based on

$$f(\boldsymbol{x},\, t) = 10n + \sum_{i=1}^{n} [(x_i - \delta_i(t))^2 - 10\cos(2\pi(x_i - \delta_i(t))] \tag{7}$$

where $t \in \mathbb{N}$ denotes the time counter; $\boldsymbol{\delta}(t) = (\delta_1(t), \ldots, \delta_n(t))$ denotes the dynamical offset. The latter depends on three parameters, namely the *update type* (linear, circular, and random), the *update frequency* Δg (i.e., the number of evaluations between each update of offset vector), and the *severity* s (i.e., a factor that determines the amount of the displacement per object variable).

The three different types of dynamics are defined as follows:

- *Linear dynamics*: $\delta_i(0) = 0,\ i \in \{1, \ldots, n\}$

$$\delta_i(t+1) = \begin{cases} \delta_i(t) + s, & (t+1) \bmod \Delta g = 0 \\ \delta_i(t), & \text{else} \end{cases} \tag{8}$$

- *Circular dynamics*:

$$\delta_i(0) = \begin{cases} 0, & i \text{ odd} \\ s, & i \text{ even} \end{cases} \tag{9}$$

$$\delta_i(t+1) = \begin{cases} \delta_i(t) + s \cdot c(i,t), & (t+1) \bmod \Delta g = 0 \\ \delta_i(t), & \text{else} \end{cases} \tag{10}$$

where

$$c(i,t) = \begin{cases} \sin\left(\frac{2\pi\lfloor t/\Delta g\rfloor}{\gamma}\right), & i \text{ odd} \\ \cos\left(\frac{2\pi\lfloor t/\Delta g\rfloor}{\gamma}\right), & i \text{ even} \end{cases} \tag{11}$$

Here, the expression $\lfloor t/\Delta g \rfloor$ is the number of applications of the dynamics within evaluations $\{0, 1, \ldots, t\}$ and γ represents the number of applications of the dynamics required to cycle the values of the offset vector. Following [28], $\gamma = 25$ is used in the experiments performed here.

- *Random dynamics*: $\delta_i(0) = 0 \, \forall i \in \{1, \ldots, n\}$

$$\delta_i(t+1) = \begin{cases} \delta_i(t) + s \cdot N_i(0,1), & (t+1) \bmod \Delta g = 0 \\ \delta_i(t), & \text{else} \end{cases} \tag{12}$$

  $N_i(0,1)$ denotes a normally distributed random variable with expectation zero and variance one.

In the three cases mentioned above, the update frequency Δg determines whether an update of $\boldsymbol{\delta}$ will happen or not. The parameter s determines the severity of the update in each dimension.

In order to let the HGA more suitable for dynamic environments, a slight revision is made. When a change is detected (notice that we have assumed the change is observable), in (4) we temporarily replace π as a random number and $Y_1 - Y_2$ a random vector in one generation to make the individuals scatter on the solution set, then restore them.

4 Experimental results

We used the HGA on the benchmark mentioned in Section 3 and compared the results with some algorithms in the literature. The first is a variation of standard genetic algorithm with a memory module ([12]). It is denoted as **SGA/M**. Another algorithm introduces a number of random immigrants when a change occurs [18]. In this paper, 25 random immigrants are migrated into the current population, as suggested in [1]. This implementation of random immigrants is accompanied by a memory module to enhance its performance. It will be referred to as **RI25/M**. The third algorithm we used is the hyper-mutation algorithm [6] in which a memory module is also included. This implementation will be referred to as **HM/M**. The last algorithm, referred to as **ERS**, is proposed in [25]. It unified genetic algorithm and a hill-climb search in dynamic optimization and can be regarded as another HGA.

In the experiments, each test is run 20 times. We use the parameters same as [12, 17] which are convenient for comparison. For all algorithms we use a population size of 100, a crossover rate of 0.6, a mutation rate of 0.2. In addition, we use **SBX** crossover and mutation with distribution index of 0.7. Binary tournament selection is adopted in recombination and replace schemes. In addition, it is also used in the selection of individuals for reevaluation. A single elitism is also used to preserve the performance of the best individual at a given generation. A maximum fitness evaluations is set to 500 000 for all implementations. Each dimension in decision space is bound between -50 and 50.

There are several performance indices that have been suggested to measure the performance of dynamic evolutionary algorithms. In this paper, we use *off-line error variation* [29] as means to quantify the performance of the proposed algorithm.

Off-line error variation index [29] is the most commonly used performance index in GAs in dynamic environments, and it is obtained as the average of the error between the true optimal fitness and the best fitness at each evaluation. It is mathematically expressed as:

$$\overline{e_{\text{off-line}}} = \frac{1}{T}\sum_{i=1}^{T}(f_{\text{true}} - f_{\text{best}}^{i}) \tag{13}$$

where i is the evaluation counter; T is the total number of evaluations considered; f_{true} is the true optimum solution that is updated whenever a change occurs; and finally f_{best}^{i} is the best individual out of the evaluations starting from the most recent occurrence of the change until the current evaluation.

Firstly, we fix the severity s in (8)-(10), and (12) as 0.1, the update frequency $\Delta g = 5\,000$, and test different functional dimensions of (7). Secondly, we set the functional dimension as 15, the same update frequency, and test all algorithms on different severities (0.01, 0.1, and 0.5). Finally, we set the functional dimension as 15, severity $s = 0.1$, and test all algorithms on different update frequencies. Tables 1, 2, and 3 show off-line error variations after 50 000 evaluations of the HGA and other algorithms in the dynamic environments with different moving types (linear, circular, and random). The best result in a comparison is emphasized as bold font. In all these cases, the HGA can achieve much better performance than other algorithms mentioned in this paper.

5 Conclusions

An HGA proposed in [26] has been tested by different functional dimensions, update frequencies, and displacement strengths in different types of dynamics. Compared with some other existing evolutionary algorithms for dynamic environments, the HGA has been illustrated its better capability to track the dynamic optimum based on the results. This HGA can be an alternative for dynamic optimization problems. A more detailed investigation of the working principles of the HGA and how to apply this HGA to other kinds of dynamic

Table 1
Off-line error variation after 50 000 evaluations in different dimensions (Severity $s = 0.1$, update frequency $\Delta g = 5\,000$)

Moving Type	Dimension	SEA/Mem	RI25/Mem	HM/Mem	ERS	HGA
Linear	5	16.25	18.92	22.51	2.85	**2.10**
	10	61.21	65.70	87.24	39.51	**4.48**
	50	528.32	557.37	738.44	1015.1	**25.43**
	100	1273.2	1422.7	1815.4	2757.5	**48.31**
	200	2376.6	3349.8	3473.7	7772.9	**88.90**
Circular	5	26.22	23.65	25.01	3.27	**0.73**
	10	107.37	96.96	103.61	32.94	**1.95**
	50	1034.6	931.70	1019.2	904.96	**20.39**
	100	2141.8	2135.9	2446.4	2685.8	**23.07**
	200	4096.8	4792.2	4688.1	7183.0	**46.87**
Random	5	22.10	20.42	22.18	3.65	**2.39**
	10	86.17	94.61	99.41	33.52	**5.87**
	50	775.13	818.55	806.51	923.08	**49.98**
	100	1478.6	1570.6	1855.4	2574.5	**97.71**
	200	3384.7	3154.7	3676.5	6942.4	**236.46**

optimization problems (such as online optimization) will be the subjects of further work.

Acknowledgments

The authors are very grateful to the anonymous referee and Professor Qing-Long Han for their comments and suggestions that lead to a significant improvement of the paper.

Table 2
Off-line error variation after 50 000 evaluations in different severities (Dimension $n = 15$, update frequency $\Delta g = 5\,000$)

Moving Type	Severity	SEA/Mem	RI25/Mem	HM/Mem	ERS	HGA
Linear	0.01	93.00	102.61	173.89	61.77	**1.63**
	0.1	98.18	114.26	177.60	85.00	**6.83**
	0.5	116.79	179.30	177.07	114.12	**22.48**
Circular	0.01	111.20	101.06	173.58	104.59	**0.09**
	0.1	194.73	191.84	215.23	99.91	**2.27**
	0.5	343.03	307.68	312.17	113.23	**10.94**
Random	0.01	112.40	137.16	176.21	81.97	**8.99**
	0.1	177.56	173.75	202.77	88.94	**10.94**
	0.5	235.01	223.54	244.05	138.72	**17.08**

Table 3
Off-line error variation after 50 000 evaluations in different update frequencies (Dimension $n = 15$, severity $s = 0.1$)

Moving Type	Update frequency	SEA/Mem	RI25/Mem	HM/Mem	ERS	HGA
Linear	1 000	111.93	106.91	151.39	44.24	**85.15**
	2 500	75.26	83.73	105.28	28.67	**19.29**
	10 000	72.13	56.70	88.43	25.72	**0.33**
Circular	1 000	132.45	135.09	179.36	42.23	**44.40**
	2 500	92.49	96.32	115.54	34.56	**9.81**
	10 000	73.27	84.02	85.90	33.40	**1.31**
Random	1 000	111.78	119.19	186.77	43.72	**15.36**
	2 500	95.51	97.36	114.20	29.81	**10.69**
	10 000	86.70	86.75	92.98	26.45	**9.73**

References

[1] J. Branke, *Evolutionary Optimization in Dynamic Environments.* Norwell, MA: Kluwer, 2001.

[2] R. W. Morrison, *Designing Evolutionary Algorithms for Dynamic*

Environments. Berlin, Germany: Springer-Verlag, 2004.

[3] K. Y. Chan, T. S. Dillon, and C. K. Kwong, Modeling of a Liquid Epoxy Molding Process Using a Particle Swarm Optimization-Based Fuzzy Regression Approach. *IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS*. 7(1), pp.148-158, 2011.

[4] K. Y. Chan, T. S. Dillon, and E. Chang, A intelligent particle swarm optimization for short-term traffic flow forecasting using on-road sensor systems. *IEEE Transactions on Industrial Electronics*. 2012. DOI: 10.1109/TIE.2012.2213556.

[5] J. Koza, Genetic programming: A paradigm for genetically breeding populations of computer programs to solve problems. Ph.D. dissertation, Stanford University, Stanford, CA, 1990.

[6] H.G. Cobb and J.J. Grefenstette. Genetic algorithms for tracking changing environments. In *Proc. of 5th Int. Conf. on Genetic Algorithms*, pp. 523-530, 1993.

[7] J.J. Grefenstette. Evolvability in dynamic fitness landscapes: A genetic algorithm approach. In *IEEE Congress on Evolutionary Computation*, pp. 2031-2038, 1999.

[8] K. Krishnakumar. Micro-genetic algorithms for stationary and non-stationary function optimization. In *Proc. SPIE Conf. Intell. Control and Adaptive Syst.*, Orlando, FL, 1989, pp. 289-296.

[9] D.E. Goldberg and R.E. Smith. Nonstationary function optimization using genetic algorithms with dominance and diploidy. In *Proc. 2nd Inter. Conf. on Genetic Algorithms*, pp. 59-68, 1987.

[10] D. Dasgupta and D.R. McGregor. Nonstationary function optimization using structured genetic algorithms. In *Parallel Problem Solving from Nature*, pp. 145-154. Elsevier, 1992.

[11] N. Mori, H. Kita, and Y. Nishikawa. Adaptation to a changing environment by maens of the feedback thermmodynamical genetic algorithm. *Trans. Inst. Syst., Control, Inf. Eng.*, vol. 12, no. 4, pp. 240-249, 1999.

[12] J. Branke. Memory enhanced evolutionary algorithms for changing optimization problems. In *Proc. Congr. Evol. Comput.*, Washington DC, 1999, 1875-1882.

[13] R.E. Smith. Diploid genetic algorithms for search in time varying environments. In *Proc. Annu. Southeast Reg. Conf. ACM*, Birmingham, AL, 1987, pp. 175-179.

[14] K.P. Ng and K.C. Wong. A new dipoid scheme and dominance change mechanism for non-stationary function optimization. In *Proc. Int. Conf. Genetic Algorithms*, Pittsburgh, PA, 1995, pp. 159-166.

[15] M. Wineberg and F. Oppacher. Enhancing the GA's ability to cope with dynamic environments. In *Proceedings of Genetic and Evolutionary Computation Conference*. pp. 3-10. Morgan Kaufmann, 2000.

[16] R.K. Ursem. Multinational GAs: Multimodal optimization techniques in dynamic environments. In *Proceedings of Genetic and Evolutionary Computation Conference*. pp. 19-26. Morgan Kaufmann, 2000.

[17] J. Branke, T. Kaus̆sler, C. Schimidt, and H. Schmeck. A multi-population approach to dynamic optimization problems. In *Adaptive Computing in Design and Manufacturing*, pp. 299-307. Springer, 2000.

[18] J.J. Grefenstette. Genetic algorithms for changing environments. In *Parallel Problem Solving from Nature*, 2, pp. 137-144. Springer, 1992.

[19] W. Cedeno and V.R. Vemuri. On the use of niching for dynamic landscapes. In *International Conference on Evolutionary Computation*, pp. 361-366. IEEE, 1997.

[20] A. Ghosh, S. Tsutsui, and H. Tanaka. Function optimization in nonstationary environment using steady state genetic algorithms with aging individuals. In *Proceedings of IEEE Congress on Evolutionary Computation*, pp.666-671, 1998.

[21] N. Mori, H. Kita, and Y. Nishikawa. Adaptation to a changing environment by maens of the feedback thermmodynamical genetic algorithm. In *Proc. Parallel Problem Solving from Nature*, pp. 149-158, 1998.

[22] Y. Jin and J. Branke. Evolutionary optimization in uncertain environments–A survey. *IEEE Transactions on Evolutionary Computation*. vol. 9, no. 3, pp. 303-317. June 2005.

[23] Y.G. Woldesenbet and G.G. Yen. Dynamic evolutionary algorithm with variable relocation. *IEEE Transactions on Evolutionary Computation*. vol. 13, no. 3, June 2009, pp. 500-513.

[24] M. Gen and R. Cheng, *Genetic Algorithms and Engineering Optimization*. John Wiley & Sons, Inc., 2000.

[25] S.M. Gerratt, J.H. Walker. Genetic algorithms: Combining evolutionary and “non”-evolutionary methods in tracking dynamic global optima. *Proc of the Genetic and Evolutionary Computation Conf.* San Francisco: Morgan Kaufmann Publishers, 2002. pp.359-366.

[26] Q. Yuan, Z. He, and H. Leng. A hybrid genetic algorithm for a class of global optimization problems with box constraints. *Applied Mathematics and Computation*. 197, pp.924-929, 2008.

[27] Q. Yuan, F. Qian, and W. Du. A hybrid genetic algorithm with the Baldwin effect. *Information Sciences*. 180, pp.640-652, 2010.

[28] T. Bäck. On the behavior of evolutionary algorithms in dynamic environments. Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on , vol., no., pp.446-451, 4-9 May 1998

[29] K. de Jong. An analysis of the behavior of a class of genetic adaptive systems. Ph.D. dissertation, Univeristy of Michigan, Ann Arbor, MI, 1975.