An Intelligent Invasive Weed Optimization: a Q-learning Approach

Abhronil Sengupta [1], Tathagata Chakraborti [1], Amit Konar [1], and Atulya K. Nagar [2]
[1] Department of Electronics and Telecommunication Engineering, Jadavpur University, Kolkata, India
[2] Department of Computer and Math Sciences, Liverpool Hope University, United Kingdom

Abstract - *Invasive Weed Optimization is a recently proposed population based meta-heuristic that mimics the colonizing action of weeds. In this article, an improvement to the classical algorithm has been proposed by introducing a constriction factor in the seed dispersal stage. Temporal Difference Q-Learning has been employed to adapt this parameter for different population members through the successive generations. The proposed memetic approach, named Intelligent Invasive Weed Optimization (IIWO) has been tested extensively on a set of 15 benchmark functions as well as the real world Circular Antenna Array Design problem. The results indicate the efficacy of our proposed approach.*

Keywords: Invasive Weed Optimization (IWO), Memetic Algorithms, Q-Learning

1 Introduction

Invasive Weed Optimization (IWO) [1] is a derivative-free optimization technique that mimics the ecological behavior of weeds. This meta-heuristic algorithm has attracted researchers because of its reduced computational cost and efficiency in tackling real world optimization problems. However, it is not free from the problems of stagnation and pre-convergence. We attempt to improve the performance of the traditional IWO algorithm by incorporating a learning strategy in the weed population to efficiently disperse seeds throughout the problem space during the reproduction phase. Such a memetic learning technique helps in balancing the exploration and exploitation capabilities of the weeds which is necessary for providing precise solutions to global optimization problems.

Coined by Dawkins[2] in 1976, the term "meme" refers to the basic unit of cultural transmission or imitation [1]. Memetic Algorithms (MAs) are population-based meta-heuristic search algorithms that combine the composite benefits of natural and cultural evolution. Natural evolution realized by Evolutionary Algorithm (EA) works on the Darwinian principle of the struggle for existence, and aims at determining the global optima in a given search landscape. Traditional EA usually takes an excessively large time to locate a precise enough solution because of its inability to exploit local information. Cultural evolution, on the other hand, is capable of local refinement. MA captures the power of global search by its evolutionary component and local search by its cultural component.

The early research on MA was confined in manual crafting of dedicated memes for a given problem. A paradigm shift in research to adaptively select a meme from a pool of memes for application to an individual member of the population has been observed during the new millennium. The class of algorithms incorporating the adaptive selection of memes is referred to as *Adaptive MA* (AMA). AMAs "promote both cooperation and competition among various problem-specific memes and favors neighborhood structures containing high quality solutions" to be attained at low computational costs. Usually, the selection of the meme for an individual member of the population is done based on its ability to perform local improvement.

Several variants of AMAs are found in the literature [4-5]. The one we would use in this paper is Roulette-Choice strategy based Hyperheuristic AMA [4]. In the Roulette-choice strategy, a meme M_e is selected with probability relative to the overall improvement. Given that $g(.)$ is a choice function, then the probability of selection of M_e is $g(M_e)/\sum_{i=1}^{n} g(M_i)$ where n is the total number of memes considered.

The AMA to be proposed, named Intelligent Invasive Weed Optimization (IIWO) includes an Invasive Weed Optimization (IWO) algorithm for global search and a Temporal Difference Q-Learning (TDQL) [6-7] for local refinement. A constriction factor has been included in the expression for standard deviation for dispersal of seeds. It is important to mention here that the constriction factors for all members of the population should not be equal for the best performance. A member with a good fitness should search in the local neighbourhood, whereas a poor performing member should participate in the global search. A good member thus should have small constriction factors, while worse members should have relatively large constriction factors. This is realized in the paper with the help of TDQL.

The TDQL works on the principle of reward and penalty. It employs a Q-table to store the reward/penalty given to an individual member of the population. Members are assigned suitable values of their constriction factors from a given meme pool before participation in the evolutionary process. After completion of the evolutionary process, members are rewarded based on their fitness, and the

reward/penalty given to the member depending on the improvement/deterioration in fitness measures of the trial solution is stored in the *Q*-table. The process of evolution and *Q*-table updating thus synergistically helps each other, resulting in an overall improvement in the performance of the AMA.

The rest of the paper is organized as follows. Sections 2 and 3 provide an overview of the Classical IWO algorithm and Differential *Q*-Learning. Our proposed approach has been described in Section 4. Extensive experimental results comparing the IIWO algorithm with IWO as well as other popular meta-heuristic algorithms namely Particle Swarm Optimization (PSO) [8] and Differential Evolution (DE) [9-10] have been presented in Section 5. Comparative results have been presented on a set of 15 benchmark functions as well as the Circular Antenna Array Design problem.

2 An Outline of Iwo Algorithm

2.1 Generation of Initial Population

IWO starts with a population of *NP D*-dimensional parameter vectors or weeds representing the candidate solutions. We shall denote subsequent generations in IWO by $G = 0, 1, \ldots, G_{max.}$ We represent the *i*-th vector of the population at the current generation as:

$$\vec{X}_{i,G} = [x_{1,i,G}, x_{2,i,G}, x_{3,i,G}, \ldots, x_{D,i,G}]$$

The initial population (at G = 0) should cover the entire search space as much as possible by uniformly randomizing individuals within the search space constrained by the prescribed minimum and maximum bounds:

$$\vec{X}_{min} = \{x_{1,min}, x_{2,min}, \ldots, x_{D,min}\}$$

and $\vec{X}_{max} = \{x_{1,max}, x_{2,max}, \ldots, x_{D,max}\}$

So we may initialize the *j*-th component of the *i*-th vector as

$$x_{j,i,0} = x_{j,min} + rand_{i,j}(0,1).\left(x_{j,max} - x_{j,min}\right) \quad (1)$$

where $rand_{i,j}(0,1)$ is a uniformly distributed random number lying between 0 and 1 and is instantiated independently for each component of the *i*-th vector.

2.2 Reproduction

The plants will produce seeds depending on their relative fitness which will be spread out over the problem space. Each seed, in turn, will grow into a flowering plant. Thus, if S_{max} and S_{min} denote the number of seeds produced by plants with best and worst fitness respectively then seed count of plants will increase linearly from S_{min} to S_{max} depending on their corresponding fitness values. The number of seeds produced by the *i*-th weed $\vec{X}_{i,G}$ is therefore given by,

$$s_{i,G} = \left\lfloor \frac{F_{max,G} - f\left(\vec{X}_{i,G}\right)}{F_{max,G} - F_{min,G}} . (S_{max} - S_{min}) \right\rfloor \quad (2)$$

where $F_{max,G}$ and $F_{min,G}$ are the maximum and minimum fitness values at the *G*-th generation of the weed colony.

2.3 Dispersal of Seeds through Search Space

The produced seeds are randomly distributed over the *D* dimensional search space by random numbers drawn from a normal distribution with zero mean but with a varying variance. However, the standard deviation (SD), σ, of the normal distribution decreases over the generations from an initial value, σ_{max}, to a value, σ_{min}, and is determined by the following equation,

$$\sigma = \left(\frac{G_{max} - G}{G_{max}}\right)^n . (\sigma_{max} - \sigma_{min}) + \sigma_{min} \quad (3)$$

where σ is the SD at the current generation and G_{max} is the maximum number of iterations while *n* is the non linear modulation index. This is the adaptation property of the algorithm.

2.4 Competitive Exclusion

If a plant does not reproduce it will become extinct. Hence this leads to the requirement of a competitive exclusion in order to eliminate plants with low fitness values. This is done to limit the maximum number of plants in the colony. Initially fast reproduction of plants take place and all the plants are included in the colony. The fitter plants reproduce more than the undesirable ones. The elimination mechanism is activated when the population exceeds a stipulated NP_{max}. The plants and produced seeds are ranked together as a colony and plants with lower fitness values are eliminated to limit the population count to NP_{max}. This is the selection property of the algorithm. The above steps are repeated until maximum number of iterations is reached..

3 Differential Q-Learning

In classical *Q*-learning, all possible states of an agent and its possible actions in a given state are deterministically known. In other words, for a given agent A, let $S_1, S_2, \ldots, S_n$, be *n*- possible states, where each state has *m* possible actions $a_1, a_2, \ldots, a_m$. At a particular state-action pair, the specific reward that the agent acquires is known as *immediate reward*. Let $r(S_i, a_j)$ be the immediate reward that the agent A acquires by executing an action a_j at state S_i. The agent selects its next state from its current states by using a policy. The policy attempts to maximize the cumulative reward that the agent could acquire in subsequent transition of states from its next state.

Step 1 For each state S and action a, initialize $Q(S,a) = 0$.
Step 2 Observe the current state S_i
Step 3 REPEAT
Select $a_j \in \{a_1, a_2, \dots, a_m\}$ and execute it.
Receive an immediate reward $r(S_i, a_j)$.
Observe the new state $S_k \leftarrow \delta(S_i, a_j)$.
Update the table entry $Q(S_i, a_j)$ by
$$Q(S_i, a_j) \leftarrow (1-\alpha).\, Q(S_i, a_j) + \alpha.\,(r(S_i, a_j) + \gamma \max_{a'} Q(\delta(S_i, a_j), a')).$$
$S_i \leftarrow S_k$
FOR EVER

Algorithm 1. Differential Q-Learning Algorithm.

Let the agent be in state S_i and is expecting to select the next best state. Then the Q-value at state S_i due to action of a_j is given by,

$$Q(S_i, a_j) = r(S_i, a_j) + \gamma \max_{a'} Q(\delta(S_i, a_j), a') \quad (4)$$

where $0<\gamma<1$ and $\delta(S_i,a_j)$ denotes the next state due to the selection of action a_j at state S_i. Let the next state selected be S_k. Then $Q(\delta(S_i,a_j), a') = Q(S_k, a')$. Consequently selection of a' that maximizes $Q(S_k, a')$ and in turn $Q(S_i, a_j)$ is an interesting problem.

The classical Q-learning algorithm for deterministic state transitions starts with a randomly selected initial state. An action 'a' from a list of actions $a_1, a_2, \dots, a_m$ is selected, and the agent because of this action receives an *immediate reward* r, and moves to the new state following the δ-transition rule given in a table. The Q-value of the previous state due to the action of the agent is updated following the Q-learning equation. Now, the next state is considered as the initial state and the steps of action selection, receiving immediate reward, transition to next state and Q-update are repeated forever.

Differential Q-learning is a modified version of Q learning. The Q-table update policy in Differential Q-learning is different from classical Q-learning. It has the ability to remember the effect of past Q value of a particular state-action pair while updating the corresponding Q value. The modified Q update equation is given by

$$Q(S_i, a_j) \leftarrow (1-\alpha).\, Q(S_i, a_j) + \alpha.\,(r(S_i, a_j) + \gamma \max_{a'} Q(\delta(S_i, a_j), a')) \quad (5)$$

The formula has the effect, that the Q-value $Q(S_i, a_j)$ is incremented, when the action a_j led to a state $\delta(S_i,a_j)$ in which there exists an action a', such that the best possible Q-value $Q(\delta(S_i,a_j), a')$ in the next time step plus the achieved reward $r(S_i,a_j)$ is greater than the current value of $Q(S_i, a_j)$. This is exactly the desired behaviour, because in such a situation, the old estimate of $Q(S_i, a_j)$ was too pessimistic. The learning rate α determines the extent to which the newly acquired information will override the old information. A setting of α= 0 makes the agent stop learning, while α=1 would make the agent consider only the most recent information.

The discount factor γ determines the importance of future rewards. A factor of 0 will make the agent "opportunist" by only considering current rewards, while a factor approaching 1 will make it strive for a long-term high reward. If the discount factor is greater than or equal to 1, the Q values may diverge.

4 IIWO: The Proposed Approach

The modified algorithm is based on the concept that fitter individuals should be involved in local search while the remaining plants should search the problem space globally at a particular generation. The classical IWO algorithm neglects this fact by assuming the same standard deviation σ for all the weeds in the seed dispersal stage. Although σ is made to decay through the successive generations yet there is no provision for σ to attain low values for fitter individuals at a particular generation to enable the local search procedure. Local search is initiated only when the generation count has increased to a large value to ensure a low value of σ. Thus in classical IWO, all the weeds undergo a gradual behavioral transformation from an explorative to an exploitive one. In our proposed algorithm, we state that fitter individuals should behave in an exploitative manner through successive generations from the initialization of the weed colony and not wait for the standard deviation to reduce to low values. Following this concept we introduce a constriction factor, η in equation (3) as follows,

$$\sigma = \eta.\left(\left(\frac{G_{max} - G}{G_{max}}\right)^{n}.(\sigma_{max} - \sigma_{min}) + \sigma_{min}\right) \quad (6)$$

where $\eta \in (0,1]$. The proper choice of parameter η for different population members will help balance the explorative and exploitive capabilities of the individuals resulting in local refinement.

The proposed approach employs a synergy of IWO and TDQL to realize an Adaptive Memetic Algorithm for achieving superior performance in global optimization problems. After each evolutionary step, the performance of the members is evaluated based on their fitness. High performing members are rewarded with positive immediate reward, whereas low performing members are penalized. The reward/penalty given to a member is stored in the Q-table

using the TDQL learning rule. A meme pool for parameter η is maintained in order to select the control parameters for individual members of the population. The adaptive selection of memes is performed by a hyperheuristic choice-metric based selection from the meme pool. The process of selection of η from the meme pool, followed by one step of IWO and reward/penalty updating in the *Q*-table is continued until the condition for convergence of the AMA is satisfied.

The proposed AMA algorithm accesses the *Q*-table to select the appropriate constriction factors of the individual members before evolution, and updates the *Q*-table after one evolution. The row indices of the *Q*-table represent states of the population members obtained from the last iteration of the IWO algorithm, in order of their fitness. The column indices which represent the actions performed by the members at a particular state correspond to uniform quantized values of the control parameter in the range (0, 1]. For example, let the parameter under consideration be η with possible quantized values $\{\eta_1, \eta_2, \dots, \eta_{10}\}$. Then $Q(S_i, \eta_j)$ represents the total reward given to a member at state $S_i$for selecting $\eta = \eta_j$. The Roulette-Choice strategy is used to select a particular value of η from the meme pool $\{\eta_1, \eta_2, \dots, \eta_{10}\}$ using the $Q(S_i, \eta_j), j = 1,2, \dots,10$ for the individual member located at state S_i.

The adaptation of $Q(S_i, \eta_j)$ is done through a reward/penalty mechanism as used in classical TDQL. If a member of the population, residing at state S_i on selecting $\eta = \eta_j$ moves to a new state S_k by the evolutionary algorithm, and such state transition causes an improvement in fitness measure, then $Q(S_i, \eta_j)$ is given a positive reward following the TDQL algorithm. If the state transition results in no improvement in fitness measure, then a penalty is given to the selected $Q(S_i, \eta_j)$. The penalty is introduced by a decrease in *Q*-value. Principles used in designing the AMA are introduced below.

4.1 Initialization

The algorithm employs a population of *NP D*-dimensional parameter vectors representing the candidate solutions. The initial population (at $G = 0$) should cover the entire search space as much as possible by uniformly randomizing individuals within the search space constrained by the prescribed minimum and maximum bounds. Thus the *j*-th component of the *i*-th population member is initialized according to (1) as mentioned in section 2.

The entries for the *Q*-table are initialized as small values. If the maximum *Q*-value attainable is 100, then we initialize the *Q*-values of all cells in the *Q*-table as 1.

4.2 Adaptive Selection of Memes

We employ Fitness proportionate selection, also known as Roulette-Wheel selection, for the selection of potentially useful memes. A basic advantage of this selection mechanism is that diversity of the meme population can be maintained. Although fitter memes would enjoy much higher probability of selection, yet the memes with poorer fitness do manage to survive and may contribute some components as evolution continues. Mathematically, the selection commences by the selection of a random number in the range [0, 1] for each population member. Let us consider the selection from the η meme pool for a member of state S_i. The next step involves the selection of $\eta_j$such that the cummulative probability of selction of $\eta = \eta_1$ through η_{j-1} is greater than *r*. Symbolically,

$$\sum_{m=1}^{j-1} p(S_i, \eta = \eta_m) < r \leq \sum_{m=j}^{10} p(S_i, \eta = \eta_m) \qquad (7)$$

The probability of selection of $\eta = \eta_j$ from the meme pool $\{\eta_1, \eta_2, \dots, \eta_{10}\}$ is given by

$$p(S_i, \eta = \eta_j) = \frac{Q(S_i, \eta_j)}{\sum_{k=1}^{10} Q(S_i, \eta_k)} \qquad (8)$$

4.3 Invasive Weed Optimization

The IWO algorithm used here employs reproduction, seed dispersal and competitive exclusion as introduced in Section 3. The basic difference of the current realization is the selection of constriction factor η from the meme pool adaptively by step 4.2 before invoking the IWO process.

4.4 State Assignment

The population members are now ranked in increasing order of fitness and assigned corresponding states.

4.5 Updating the Q-table

Let a member at state S_i on selection of η_j moves to a new state S_k. The update equation for $Q(S_i, \eta_j)$ is given by,

$$Q(S_i, \eta_j) \leftarrow (1 - \alpha).\, Q(S_i, \eta_j) + \alpha.\, (r(S_i, \eta_j) + \gamma \max_{\eta'} Q(\delta(S_k, \eta_j), \eta')) \qquad (9)$$

The choice of the reward function is critical to the proper operation of the *Q*-learning mechanism. In case the seeds produced by a particular weed experience greater fitness in comparison to the parent weed then $r(S_i, \eta_j)$ is set equal to the absolute difference of fitness of the parent weed and the fittest seed. Otherwise a penalty of $-K$ is applied, however small.

Step 1 Set the generation number *G=0* and randomly initialize a population of *NP* individuals, $P_G = \{\vec{X}_{1,G}, \vec{X}_{2,G}, \dots, \vec{X}_{NP,G}\}$ with $\vec{X}_{i,G} = [x_{1,i,G}, x_{2,i,G}, x_{3,i,G}, \dots, x_{D,i,G}]$ with *i*= [*1, 2,……., NP*].
Initialize the *Q*-table: $Q(S_i, \eta_j) = 1 \; \forall i = [1, \dots, NP] and j = [1, \dots, 10]$

Step 2 Evaluate the population.

Step 3 WHILE stopping criterion is not reached, DO

Step 3.1 Initialize $r(S_i, \eta_j) = 0 \; \forall i = [1, \dots, NP] and j = [1, \dots, 10]$.

Step 3.2 /*Adaptive Selection of memes*/

FOR *i=1 to NP*

Select $\eta = \eta_k$ by Roulette-Wheel Selection.

END FOR

Step 3.3/*Reproduction*/

FOR *i=1 to NP*

Determine the number of seeds produced by the i-*th* population member at generation *G*,

$$s_{i,G} = \left\lfloor \frac{F_{max,G} - f(\vec{X}_{i,G})}{F_{max,G} - F_{min,G}} . (S_{max} - S_{min}) \right\rfloor$$

END FOR

Step 3.4/*Seed Dispersal*/

FOR *i=1 to NP*

FOR *j=1 to* $s_{i,G}$

Generate a population of $s_{i,G}$ seeds by,

$\vec{V}_{i,j,G} = [v_{1,i,j,G}, v_{2,i,j,G}, v_{3,i,j,G}, \dots, v_{D,i,j,G}]$

where $v_{k,i,j,G} = x_{k,i,G} + randn(0, \sigma)$ with *k*=[*1,2,…, D*]

and $\sigma = \eta . \left(\frac{G_{max} - G}{G_{max}}\right)^n (\sigma_{max} - \sigma_{min}) + \sigma_{min}$

END FOR

END FOR

Step 3.5/*Competitive Exclusion*/

Evaluate the colony of seeds and weeds.

FOR *i=1 to NP*

FOR *j=1 to* $s_{i,G}$

IF $f(\vec{V}_{i,j,G}) < f(\vec{X}_{i,G})$

$temp = | f(\vec{V}_{i,j,G}) - f(\vec{X}_{i,G}) |$

IF $temp > r(S_i, \eta_k)$

$r(S_i, \eta_k) = temp.$

END IF

ELSE $r(S_i, \eta_k) = -K.$

END IF

END FOR

END FOR

IF $NP + \sum_{i=1}^{i=NP} s_{i,G} > NP_{max}$

Form new weed colony with the first NP_{max}weeds arranged in order of fitness.

END IF

Step 3.6 /*State Assignment*/

Rank individuals in order of fitness and assign corresponding states.

Step 3.7/*Update of the *Q*-table*/

FOR *i=1 to NP*

FOR *j=1 to 10*

IF $r(S_i, \eta_j) \neq 0$

$Q(S_i, \eta_j) \leftarrow (1 - \alpha) . Q(S_i, \eta_j) + \alpha . (r(S_i, \eta_j) + \gamma \max_{\eta'} Q(\delta(S_k, \eta_j), \eta'))$

END IF

END FOR

END FOR

Step 3.8/*Increment the generation count*/ *G=G+1*

Step 4 END WHILE

Algorithm 2. The Proposed IIWO Algorithm.

The next step involves the determination of the factor $\max_{\eta'} Q(\delta(S_k, \eta_j), \eta')$. A particular weed may enter the next generation along with multiple seeds or it may be completely eliminated. In case of multiple state acquisition in the next generation the factor is set equal to the maximum of $\max_{\eta'} Q(\delta(S_k, \eta_j), \eta')$ for all S_k s. Otherwise it is set equal to 0 in case of plant exclusion.

The sections B-E are repeated till maximum number of iterations is reached.

5 Experiments and Results

5.1 Experimental Setup

We evaluate the performance of our proposed IIWO algorithm on a test-suite of 15 benchmark functions with varying degrees of complexity. The functions have been chosen from the benchmarks proposed in the CEC 2005 conference. Among them, the first five functions are unimodal while the remaining are multimodal. Due to lack of space we provide the results on the first 15 representative benchmarks. Details of the benchmark functions can be found in [12]. Results have been presented for 30 dimensions of all the benchmark functions. Each of the algorithms was run for a specified number of function evaluations: *D**1e+05 where *D* is the dimension of the problem. The mean value and standard deviation (within parenthesis) of the error in fitness value over 25 independent runs of each algorithm are presented in table 2.

Since all the algorithms start with the same initial population over each problem instance, we have used paired t-tests to compare the means of the results produced by the best and second-best algorithm (with respect to their final accuracies) for each benchmark. We have also reported the statistical significance level of the difference of means of the two algorithms in the respective columns of Table 2 and 3. The best performance has been highlighted in each row. The † sign indicates the t value of 49 degrees of freedom is significant at a 5% tolerance level of significance by 2 tailed test. The ‡ sign indicates that it is non-significant.

Comparisons have also been presented for the real world Circular Antenna Array Design problem [11]. The mean and standard deviation results have been presented after 1.5e+05 function evaluations. The optimization problem is briefly outlined below.

The array factor of a circular antenna array of N antenna elements placed on a circle of radius *r* in the *x-y* plane is given by:

$$AF(\phi) = \sum_{i=1}^{N} I_n \exp\left(jkr\left(\cos\left(\phi - \phi_{ang}^{n}\right) - \cos\left(\phi_0 - \phi_{ang}^{n}\right)\right) + \beta_n\right)$$

where $\phi_{ang}^{n} = 2\pi(n-1)/N$ is the angular position of the n[th] element in the x-y plane,

$kr = Nd$ where k is the wave-number, d is the angular spacing between elements and r is the radius of the circle defined by the antenna array,

ϕ_0 is the direction of maximum radiation,

ϕ is the angle of incidence of the plane wave,

I_n is the current excitation and

β_n is the phase excitation of the n[th] element.

Here we shall try to suppress side-lobes, minimize beamwidth and achieve null control at desired directions by varying the current and phase excitations of the antenna elements. For a symmetrical excitation of the circular antenna array objective function as:

$$OF = \left|AR(\varphi_{sll}, \vec{I}, \vec{\beta}, \varphi_0)\right| / \left|AR(\varphi_{max}, \vec{I}, \vec{\beta}, \varphi_0)\right| + 1/DIR(\varphi_0, \vec{I}, \vec{\beta}) + |\varphi_0 - \varphi_{des}| + \sum_{k=1}^{num} \left|AR(\varphi_k, \vec{I}, \vec{\beta}, \varphi_0)\right|$$

where φ_{sll} is the angle at which maximum sidelobe level is attained, φ_{des} is the desired maxima, *num* is the number of null control directions and φ_k specifies the k[th] null control direction.

The first component attempts to suppress the sidelobes. Nowadays directivity has become a very useful figure of merit for comparing array patterns. The second component attempts to maximize directivity of the array pattern and the third component strives to drive the maxima of the array pattern close to the desired maxima. The fourth component penalizes the objective function if sufficient null control is not achieved.

5.2 Other Competitive Algorithms

Differential Evolution and Particle Swarm Optimization has recently gained wide popularity as a fast and efficient optimization algorithm over continuous search spaces. We compare the performance of IIWO with classical IWO, DE and PSO. The parameter settings are given in the next page.

5.3 Simulation Results

The results obtained for the 15 benchmark problems as well as the real world optimization problem are tabulated below.

Table 1. Parameter Settings

PARAMETER	VALUE
Pop_size	50
Inertia weight	0.25-0.4
C_1, C_2	2
F	0.5
Cr	0.9
σ_{max}	10% of search range
σ_{min}	1% of search range

Table 2. Results for 30D Benchmark Problems

F.	IIWO	IWO	DE	PSO
1	**8.537e-01†** **(2.673e-01)**	4.977e+01 (9.453e-00)	7.229e+04 (2.599e+03)	1.523e+03 (3.743e+02)
2	**1.653e+00†** **(4.373e-01)**	8.251e+01 (9.487e+00)	7.176e+04 (6.289e+03)	8.578e+03 (2.788e+02)
3	**5.124e+05†** **(8.763e+04)**	2.899e+06 (5.11e+05)	6.286e+08 (5.271e+07)	8.176e+06 (2.389e+05)
4	**2.075e+00†** **(1.745e-02)**	1.667e+02 (1.013e+01)	4.389e+02 (1.865e+01)	4.391e+03 (5.283e+02)
5	2.481e+03 (8.351e+02)	**5.419e+01†** **(1.032e+01)**	2.747e+04 (2.577e+03)	1.011e+04 (3.733e+02)
6	**2.961e+02†** **(8.927e+01)**	3.766e+04 (1.198e+04)	3.281e+10 (2.744e+09)	5.789e+08 (7.639e+07)
7	**7.989e-02†** **(2.322e-03)**	1.836e+00 (1.921e-01)	2.836e+02 (4.899e+01)	3.137e+03 (5.533e+02)
8	**2.016e+01†** **(5.814e-05)**	2.094e+01 (1.344e-04)	2.115e+01 (4.436e-02)	2.291e+01 (2.487e-01)
9	1.185e+02 (4.013e+01)	**5.962e+01†** **(6.392e+01)**	7.321e+02 (2.987e+01)	7.491e+01 (3.987e+01)
10	1.173e+02 (1.332e+01)	**8.673e+01†** **(2.587e+01)**	5.287e+02 (4.731e+01)	1.928e+02 (2.677e+01)
11	**1.384e+01‡** **(6.037e+00)**	1.437e+01 (1.345e+00)	9.663e+01 (1.393e+00)	2.349e+01 (1.024e01)
12	**5.196e+04†** **(9.723e+03)**	9.148e+05 (7.285e+04)	9.825e+05 (1.281e+05)	1.064e+05 (3.112e+05)
13	**3.052e+00†** **(1.021e+00)**	1.265e+01 (1.626e+00)	5.973e+02 (1.385e+02)	6.979e+00 (2.562e+00)
14	**1.126e+01†** **(2.311e-01)**	1.135e+01 (3.156e-01)	1.453e+01 (1.121e-01)	1.217e+01 (1.452e+00)
15	**4.013e+02‡** **(6.724e+01)**	4.038e+02‡ (5.982e+01)	8.832e+02 (2.281e+01)	6.747e+02 (1.043e+02)

Table 3. Objective Function Values for the Circular Antenna Array Design Problem

IIWO	IWO	DE	PSO
-20.7013† **(1.312e-01)**	-16.4178 (4.293e+00)	-13.9306 (1.041e-01)	-5.4852 (3.543e-00)

6 Conclusions

In this paper we present a novel approach to improved global optimization by using a synergy of Invasive Weed Optimization and Temporal Difference *Q*-Learning to adaptively select memes (constriction factors) from the meme pool. To the best of our knowledge, such Machine Learning techniques have not been used previously to incorporate learning strategies in Evolutionary Algorithms. Experimental results conducted on a wide variety of benchmark functions as well as a real world optimization problem justifies our claim to the robustness and efficiency of the proposed approach.

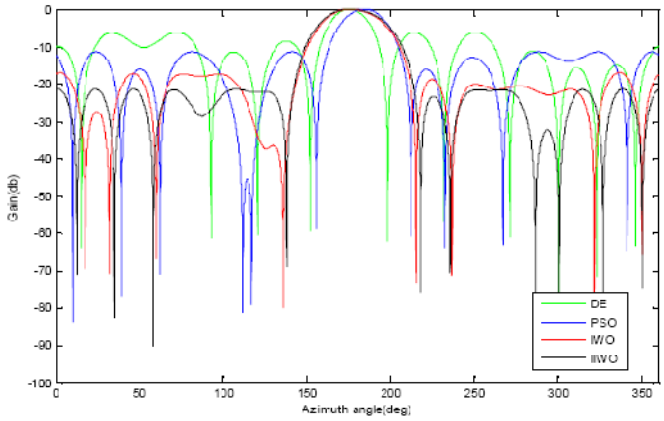

Figure 1. Power radiation pattern.

7 References

[1] Mehrabian, A. R. and Lucas, C. 2006. A novel numerical optimization algorithm inspired from weed colonization. *Ecological Informatics* 1 (2006), 355–366.

[2] Dawkins R. 1976. The Selfish Gene. *Oxford University Press* (1976).

[3] Moscato, P. On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms. In *Caltech Concurrent Computation Program* (report 826).

[4] Ong, Y.-S., Lim, M.H., Zhu, N. and Wong, K.-W. 2006. Classification of Adaptive Memetic Algorithms: A Comparative Study. In *IEEE Trans. on Systems, Man and Cybernetics* 36, 1 (Feb. 2006).

[5] Kendall, G., Cowling, P. and Soubeiga, E. 2002. Choice function and random hyperheuristics. In *Proceedings of the 4th Asia-Pacific Conferrence on simulated Evolution and Learning* (Singapore, Nov. 2002), 667-671.

[6] Watkins, C. 1989. Learning from delayed rewards. *PhD dissertation* (King's College, Cambridge, England, 1989).

[7] Watkins, C. and Dayan P. 1992. Q-learning. *Machine Learning,* 8, (1992), 279- 292.

[8] Kennedy, J. and Eberhart, R.C. 1995. Particle swarm optimization. In *Proceedings of IEEE International conference on Neural Networks* (1995), 1942-1948,

[9] Konar, A. and Das, S. 2006. Recent advances in evolutionary search and optimization algorithms. In *Proceedings of NGMS 2006* (BESU, Shibpur, Howrah, India, January 11-13, 2006).

[10] Storn, R. and Price, K. V. 1997. Differential Evolution–a simple and efficient heuristic for global optimization over continuous spaces. *J. Global Optimization* 11, 4 (1997), 341–359.

[11] Gurel, L. and Ergul, O. 2008. Design and simulation of circular arrays of trapezoidal-tooth logperiodic antennas via genetic optimization. *Progress In Electromagnetics Research* PIER 85 (2008), 243 - 260.

[12] P.N. Suganthan, N. Hansen , J.J. Liang, K. Deb, Y. P. Chen A. Auger and S. Tiwari, "Problem Definitions and Evalution Criteria for the CEC 2005 Special Session on Real-Parameter Optimization," *Technical Report*, Nanyang Technological University, Singapore.