2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

A Uncultivated Wolf Pack Algorithm for High-dimensional Functions and Its Application in Parameters Optimization of PID Controller

Wu Husheng
Material Management and Safety Engineering College
Air Force Engineering University
Sian, China
wuhusheng0421@163.com

Zhang Fengming
Material Management and Safety Engineering College
Air Force Engineering University
Sian, China
zfmwenzhang007@163.com

***Abstract*—To solve high-dimensional function optimization problems, many swarm intelligent algorithms have been proposed. Inspired by hunting behavior and distribution mode of uncultivated and barbarous wolf pack, we proposed a method, named uncultivated wolf pack algorithm (UWPA). Experiments are conducted on a suit of high-dimensional benchmark functions with different characteristics. What's more, the compared simulation experiments with other three typical intelligent algorithms, show that UWPA has better convergence and robustness. At last, this algorithm is successfully applied in parameters searching for PID controller.**

Keywords—evolutionary computation; swarm intelligence; wolf pack algorithm (WPA) ; function optimization

I. INTRODUCTION

Warm intelligence (SI) is an artificial intelligence technique based on the study of behavior of simple individuals[1]. There are many marvelous swarm intelligence phenomena in natural world, which can give us endless inspiration. Inspired by swarm phenomenon of animals, people have developed many optimization computation methods to solve complicated problems in recent decades[2-4]. In 1995, inspired by social behaviors and movement dynamics of birds, Kennedy proposed the particle swarm optimization algorithm (PSO) [5]. In 1996, inspired by social division and foraging behavior of ant colonies, Dorigo M proposed the Ant Colony Optimization algorithm (ACO) [6]; In 2002, inspired by foraging behavior of fish schools, Li Xiaolei proposed the Artificial Fish Swarm Algorithm(AFSA)[7]. In 2005, motivated by the specific intelligent behaviors of honey bee swarms, Karaboga proposed the artificial bee colony (ABC) algorithm [8]. Birds, fishes ants and bees do not have any human complex intelligence such as logical reasoning and synthetic judgment, but under the same aim, food, they stand out powerful swarm intelligence through constantly adapting environment and mutual cooperation, which give us many new ideas for complex problem solution.

In order to solve complex optimization problems, researchers have learned from the marvelous collective behavior of wolves [9-11]. In this paper, we reanalyze the characteristics of wolves.

Wolves are gregarious animals and have clear social work division. Generally, for the common wolf pack, there is a lead wolf, some scout wolves and ferocious wolves.

Firstly, the lead wolf, as a leader under the law of the jungle, is always the smartest and most ferocious one. It is responsible for commanding the wolves and constantly make decision by evaluating surrounding situation and perceiving information from the wolf pack. So lead wolf has more hunting experience and possess the best gene. It is also the only male one who have the qualification to possess female wolves and have the reproductive rights, which ensure the best gene get saved and passed to next generation.

Secondly, the lead wolf send some elite wolves to hunt around and look for prey in the probable scope rather than let everyone out. They walk around and independently make decision according to the concentration of smell left by prey, the thicker the smell is becoming, the nearer from the prey. So they always move towards the direction of getting stronger smell.

Thirdly, once a scout wolf find the trace of the prey, it will howl and summon the ferocious wolves to round up the prey. The ferocious wolves will move fast towards the direction of the scout wolf.

Fourthly, there is a rule of prey distribution. After captured the prey, the prey is distributed equitably in an order from the strong to the weak. Although this distribution rule will make some weak wolf dead for lack of food, it make sure the wolves, who have the ability of capturing the prey, get more food so as to keep them strong and can capture more prey successfully in next time.

But for the uncultivated and barbarous wolf pack, there is a difference. It doesn't have a rule of prey distribution from the strong to the weak. After captured the prey, the prey is distributed randomly. Though this mode of distribution can not make the strong wolves much stronger, it also doesn't make the weak wolves much weaker. So, it can maintain the population diversity of wolf pack at the most, which is very important for complex problems in a high-dimensional solution space.

All above highlight great charm of swarm intelligence of uncultivated wolf pack. Inspired by these, uncultivated wolf

This research is supported by National Science Foundation of Shaanxi province (2013JQ8042)

pack algorithm (UWPA) for high-dimensional functions optimization is proposed and successfully applied in parameter optimization for PID controller.

The rest of the paper is structured as follows. In Section 2, the hunting behavior and distribution rules of uncultivated wolf pack are analyzed, and then UWPA algorithm is described. In Section 3, the compared simulation results between GA, PSO, ASFA and UWPA are presented and discussed, respectively. Section 4 presents the application of the proposed algorithm to parameter optimization for PID controller. Finally, in Section 5 the conclusions are discussed.

II. UNCULTIVATED WOLF PACK ALGORITHM

In what follows, author made detailed description and realization for the behavior and rules of uncultivated wolf pack.

A. Some definitions

If the space of the artificial wolves is an $N\times D$ Euclidean space, N is the number of wolves, D is the number of variables. The position of wolf i is a vector $\boldsymbol{X}_i=(x_{i1}, x_{i2}, \ldots\ldots, x_{iD})$, and x_{id} is the d^{th} variable value of the i^{th} artificial wolf. $Y=f(\boldsymbol{X})$ represents the concentration of prey's smell perceived by artificial wolves, which is also the objective function value. The distance between wolf p and wolf q is definite as their Manhattan distance such as follows.

$$L(p,q)=\sum_{d=1}^{D}\left|x_{pd}-x_{qd}\right| \tag{1}$$

Moreover, because the problems of maximum value and minimal value can convert with each other, only the maximum value problem is discussed in what follows.

B. The description of intelligent behaviors and rules

The cooperation between the lead wolf, scout wolves and ferocious wolves makes nearly perfect predation, while prey distribution randomly maintain the population diversity of wolf pack. The whole predation behavior of uncultivated wolf pack are abstracted three artificial intelligent behaviors, scouting, calling and besieging behavior, and two intelligent rules, winner-take-all generating rule for the lead wolf, the randomly-survive renew rule for the wolf pack.

(1)The winner-take-all generating rule for the lead wolf. The artificial wolf with best objective function value is lead wolf. During each iteration, compare the function value of the lead wolf with the best one of other wolves, if the value of lead wolf is not better, it will be replaced. Then the best wolf become lead wolf. Rather than acting the three intelligent behaviors, the lead wolf directly goes into the next iteration until it is replaced by other better wolf.

(2)Scouting behavior. The best *S_num* artificial wolves except the lead wolf are considered as the scout wolves, they search the solution in predatory space. Y_i is the concentration of prey smell perceived by the scout wolf *i*. Y_{lead} is the concentration of prey smell perceived by the lead wolf.

$$x_{id}^{p}=x_{id}+\sin(2\pi\times p/h)\times step_{a}^{d}\,,\, p=\{1,2\,,\;\ldots,h\} \tag{2}$$

It should be noted that h is different for each wolf because of their different seeking ways. In actual computation, h is randomly selected in $[h_{min}, h_{max}]$ and it must be an integer. Y_{i0} is the concentration of prey smell perceived by the scout wolf i and Y_{ip} represents the one after it took a step towards the p^{th} direction. If $max\ \{Y_{i1}, Y_{i2}, \ldots, Y_{ih}\}> Y_{i0}$, the wolf i step forward and its state X_i is updated. Then repeat the above until $Y_i>Y_{lead}$ or the maximum number of repetition T_{max} is reached.

(3)Calling behavior. The lead wolf will howl and summon *M_num* ferocious wolves to gather towards the prey. Here, the position of the lead wolf is considered as the one of the prey so that the ferocious wolves aggregate towards the position of the lead wolf. $\boldsymbol{step_b}$ is the step length, g_d^k is the position of artificial lead wolf in the d^{th} variable space at the k^{th} iteration. The position of the ferocious wolf i in the k^{th} iterative calculation is updated according to "(3)".

$$x_{id}^{k+1}=x_{id}^{k}+step_{b}^{d}\cdot(g_{d}^{k}-x_{id}^{k})/\left|g_{d}^{k}-x_{id}^{k}\right| \tag{3}$$

This formula consists of two parts, the former is the current position of wolf i, which represents the foundation for prey hunting; the latter represents the aggregate tendency of other wolves towards the lead wolf, which shows the lead wolf's leadership to the wolf pack.

If $Y_i>Y_{lead}$, the ferocious wolf i become lead wolf and $Y_{lead}=Y_i$, then the wolf i take the calling behavior; If $Y_i<Y_{lead}$, the ferocious wolf i keep on aggregating towards the lead wolf with a fast speed until $L(i, l)<L_{near}$, the wolf take besieging behavior. $L(i, l)$ is the distance between the wolf i and the lead wolf l, L_{near} is the distance determinant coefficient as a judging condition, which determine whether wolf i change state from aggregating towards the lead wolf to besieging behavior.

Calling behavior shows information transferring and sharing mechanism in wolf pack and blends the ideas of social cognition. Other wolves are all in response to the best wolf, the lead wolf, which fully shows that the algorithm is intelligent and sociality.

(4)Besieging behavior. After large-steps running towards the lead wolf, the wolves are close to the prey, then all wolves except the lead wolf will take besieging behavior for capturing prey. Now, the position of lead wolf is considered as the position of prey. In particular, G_d^k represents the position of prey in the d^{th} variable space at the k^{th} iteration. The position of wolf i is updated according to "(4)".

$$x_{id}^{k+1}=x_{id}^{k}+\lambda\cdot step_{c}^{d}\cdot\left|G_{d}^{k}-x_{id}^{k}\right| \tag{4}$$

λ is a random number uniformly distributed at the interval [-1, 1], $\boldsymbol{step_c}$ is the step length of wolf i when it takes besieging behavior. Y_{i0} is the concentration of prey smell perceived by the wolf *i and* Y_{ik} represents the one after it took this behavior. If $Y_{i0}<Y_{ik}$, the position $\boldsymbol{X_i}$ is updated, otherwise it not be updated.

There are $\boldsymbol{step_a}$, $\boldsymbol{step_b}$, $\boldsymbol{step_c}$ in the three intelligent behaviors, and the three steps length in d^{th} variable space have the relationship follows.

$$step_{a}^{d}=step_{b}^{d}/2=2\cdot step_{c}^{d}=S \tag{5}$$

S is step coefficient and represents the fineness degree of artificial wolf searching for prey in resolution space.

(5)The randomly-survive renew rule for the wolf pack. In order to maintain the population diversity, the prey is distributed randomly, which will result in some weak wolves dead randomly. The algorithm will generate R wolves while randomly delete R wolves. Specifically, with the help of the lead wolf, in the d^{th} variable space, the position of i^{th} one of R wolves is defined as follows.

$$x_{id} = g_d \cdot rand \quad , i = \{1, 2, \dots , R\} \tag{6}$$

g_d is the position of artificial lead wolf in the d^{th} variable space, *rand* is a random number uniformly distributed at the interval [-0.1, 0.1].

The larger the value of R, the better for sustaining wolf's diversity and making the algorithm have the ability of opening up new resolution space. But if R is too large, the algorithm will nearly be a random search approach. Because the number and scale of prey captured by wolves are different in natural word, which will lead to different number of weak wolf dead. R is an integer and randomly selected at the interval $[n/(2*\beta), n/\beta]$. β is the population renewal proportional coefficient.

C. *Algorithm description*

As described in the previous section, WPA has three artificial intelligent behaviors and two intelligent rules. There are scouting behavior, calling behavior and besieging behavior, winner-take-all rule for generating lead wolf, the randomly-survive renew rule for wolves.

Firstly, the scouting behavior accelerating the possibility that WPA can fully traverse the solution space; Secondly, the winner-take-all rule for generating lead wolf and the calling behavior make the wolves move towards the lead wolf whose position is the nearest to the prey and most likely capturing prey. The winner-take-all rule and calling behavior also make wolves arrive at the neighborhood of the global optimum only after a few iterations elapsed, since the step of wolves in calling behavior is the largest one. Thirdly, with a small step $\boldsymbol{step_c}$, besieging behavior makes UWPA algorithm have the ability of opening up new solution space and carefully searching the global optima in good solution area. Fourthly, with the help of randomly-survive renew rule for the wolves, the algorithm can get several new wolves whose positions are near the best wolf, lead wolf, which allows more latitude of search space to anchor the global optimum while keep population diversity in each iteration.

All the above make UWPA possess superior performance in accuracy and robustness, Which will be seen in section Ⅲ.

Having discussed all the components of UWPA, the important computation steps are detailed below.

Step1 Initialization. Initialize the follow parameters, the initial position of artificial wolf i ($\boldsymbol{X}_i$), the number of the wolves (N), the maximum number of iterations(k_{max}), the step coefficient (S), the distance determinant coefficient (L_{near}), the maximum number of repetition in scouting behavior(T_{max}), the population renewal proportional coefficient(β);

Step2 The wolf with best function value is considered as lead wolf. In practical computation, S_num=M_num=n-1, which means that wolves except for lead wolf act different behavior as different status. So, here except lead wolf, refer to "(2)", the rest n-1 wolves firstly act as the artificial scout wolves to take scouting behavior until Y_i>Y_{lead} or the maximum number of repetition T_{max} is reached, and then go to step3;

Step3 Except the lead wolf, the rest n-1 wolves secondly act as the artificial ferocious wolves and gather towards the lead wolf according to (3), Y_i is the concentration of prey smell perceived by wolf i, if $Y_i \geq Y_{lead}$, go to step2; otherwise the wolf i continue running until $L(i, l) \leq L_{near}$, then go to step 4;

Step4 The position of artificial wolves who take besieging behavior is updated according to "(4)";

Step5 Update the position of lead wolf under the winner-take-all generate rule and update the randomly-survive renew rule for wolves according to "(6)";

Step6 If the program reaches the precision requirement or the maximum number of iterations, the position and function value of lead wolf, the problem optimal solution, will be outputted, otherwise go to step2.

III. RESULTS ON FUNCTIONS OPTIMIZATION

The ingredients of the UWPA method have been described in Section Ⅱ. In this section, experiments are given to validate UWPA and compare UWPA with GA, PSO and ASFA.

A. *Validation*

There are many benchmark functions for validating new algorithms. Here, we have chosen the well-known Rosenbrock function .

$$f(x,y) = 100(y - x^2)^2 + (1-x)^2 \, , \; x, y \in [-2.048, 2.048] \tag{7}$$

and the Eggcrate function

$$g(x,y) = x^2 + y^2 + 25\left(\sin^2 x + \sin^2 y\right) , \; x, y \in [-2\pi, 2\pi] \tag{8}$$

Global minimum value for Rosenbrock function is 0 and optimum solution is (x, y) = (1, 1). But it is difficult to converge to the global optimum of this function. As shown in Fig.1, the global optimum is inside a long, narrow, parabolic-shaped flat valley.

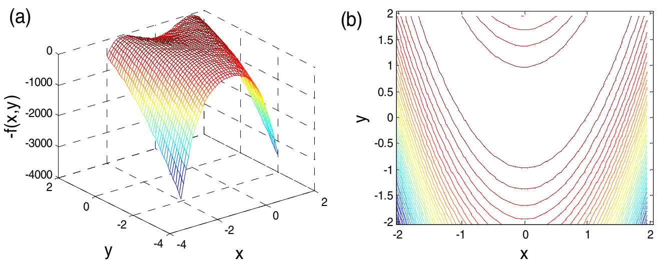

Fig. 1. Rosenbrock function: (a) surface plot and (b) contour lines

Global minimum value for Eggcrate function is also 0 and

and optimum solution is $(x, y) = (1, 1)$. But it is also difficult to converge to the global optimum. As shown in Fig. 2, Eggcrate is a multimodal function so that there are many local extrema near the global optimum.

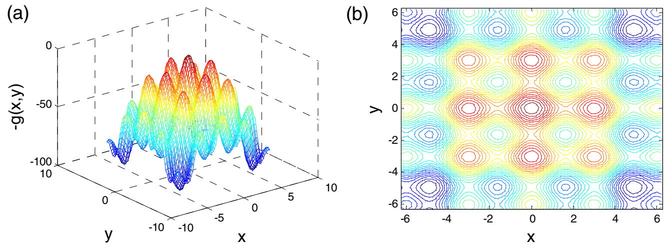

Fig. 2. Rosenbrock function: (a) surface plot and (b) contour line s

So these problem are suitable to test the performance of the algorithms. For Rosenbrock function, the distribution of 100 artificial wolves during the consecutive 40 time steps are shown in Fig. 3 where we can see that the artificial wolves converge at the global optimum (1, 1). For Eggcrate function, a snapshot of the last iterations is shown in Fig. 4 again, all wolves move towards the global optimum (0, 0). And finally at iteration 20 nearly all of wolves are on the best position, which is the global minimum of the problem. All these show the effectiveness of this algorithm.

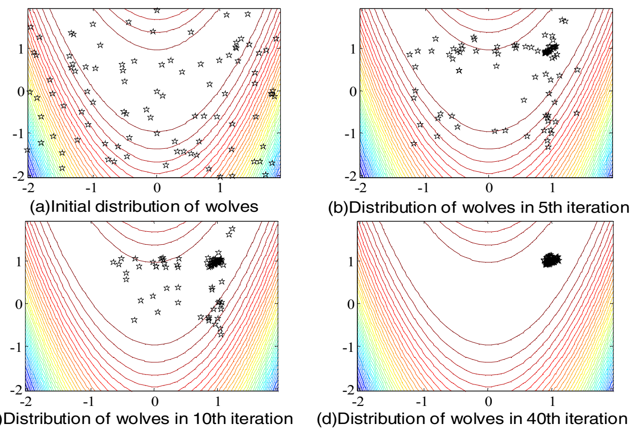

Fig. 3. Distribution of artificial wolves in solution space for Rosenbrock

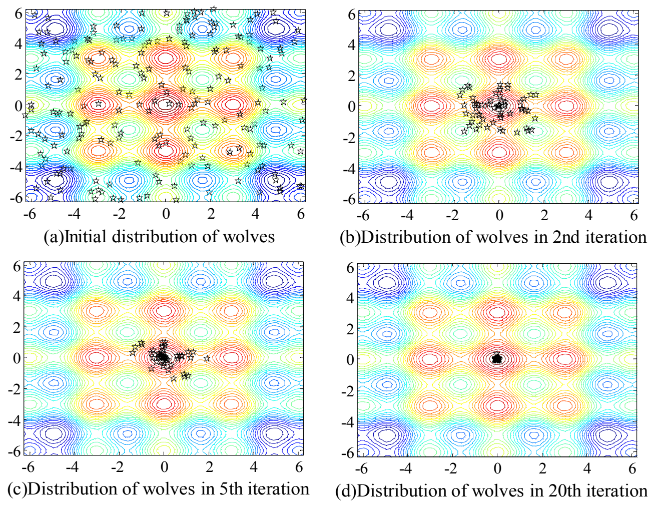

Fig. 4. Distribution of artificial wolves in solution space for Eggcrate

B. *UWPA vs PSO, ASFA and GA*

In this section, we compared PSO, AFSA, GA and UWPA algorithms on six high-dimensional functions.

1) Bohachevsky3 Function *(D=2)*

$$f(\vec{x}) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1 + 4\pi x_2) + 0.3 \tag{9}$$

2) Bridge Function *(D=2)*

$$f(\vec{x}) = \frac{\sin\sqrt{x_1^2 + x_2^2}}{\sqrt{x_1^2 + x_2^2}} + \exp\left(\frac{\cos 2\pi x_1 + \cos 2\pi x_2}{2}\right) - 0.7129 \tag{10}$$

3) Ackley Function *(D=50)*

$$f(\vec{x}) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos 2\pi x_i\right) + 20 + e \tag{11}$$

Bohachevsky3, Bridge and Ackley are multimodal and non-separable functions, which is difficult to be solved, since there are many local extrema near the global extremum [12]. Their $\vec{x}$ are in interval of [-100,100], [-1.5,1.5] and [-32,32], and the global values are 0, 3.0054 and 0 respectively.

4) Griewank Function *(D=100)*

$$f(\vec{x}) = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \tag{12}$$

Griewank function, a multimodal and non-separable function, $\vec{x}$ is in the interval of [-600, 600]. The global minimum value for this function is 0 and the corresponding global optimum solution is $(x_1, x_2, \dots, x_m) = (0, 0, \dots, 0)$.

5) Sumsquares *Function (D=150)*

$$f(\vec{x}) = \sum_{i=1}^{D} i x_i^2 \tag{13}$$

6) Sphere Function *(D=200)*

$$f(\vec{x}) = \sum_{i=1}^{D} x_i^2 \tag{14}$$

Sphere and Sumsquares are continuous, convex, unimodal and separable functions. They are all high-dimensional functions for their 200 and 150 parameters respectively. $\vec{x}$ of Sphere and Sumsquares are respectively in the interval of [-100, 100] and [-10, 10], and the global minimum values of them are all 0 and optimum solution is $(x_1, x_2, \dots, x_m) = (0,0, \dots, 0)$.

For each experiment in this section, 50 independent runs were conducted with different random seeds. To evaluate the algorithm's efficiency and effectiveness, we adopted several criteria, such as the best solutions (*Best*), the worst value (*Worst*), and the mean value (*Mean*), standard deviations (*StdDev*) and successful rate of the results (*SR*). *SR* provides very useful information about how stable an algorithm is [13]. To calculate the success rate, an error accuracy level $\varepsilon = 10^{-6}$ must be set ($\varepsilon = 10^{-6}$ also used in [14]). Thus, it is considered that a run of the algorithm reaches a solution F which meets (15). F^* is the ideal solution value.

$$\begin{cases} \left|F - F^*\right| / F^* < \varepsilon, & F^* \neq 0 \\ \left|F - F^*\right| < \varepsilon, & F^* = 0 \end{cases} \tag{15}$$

The SR is a percentage value that is calculated as:

$$SR = \frac{\#successful \quad runs}{\#runs} \tag{16}$$

In order to fully compare the performance of different algorithms, we take the simulation under the same situation. So the values of the common parameters used in each algorithm such as population size and evaluation number were chosen to be the same. Population size was 100 and the maximum evaluation number was 2000 for all algorithms on all functions. Besides, other main parameters are setted as TABLE Ⅰ.

TABLE I. THE PARAMETERS OF FOUR ALGORITHMS

Method	Main parameters
UWPA	The step coefficient S=0.10, distance determinant coefficient L_{near}=0. 08, the maximum number of repetition in scouting T_{max}=8, population renewal coefficient $\beta=3$
PSO	The inertia weight w=0. 7298, the learning factor c_1=c_2=1. 4946 , individual speed limitation is [-0. 5, 0. 5]
AFSA	The maximum number of tentative times try_num=100, the cognitive distance v=1, the factor of congestion degree $delta$=0. 618, the length of step s=0. 1
GA	The crossover probability p_c=0. 7, the mutation probability p_m=0. 01, taking the roulette selection operator and a strategy of self-adapting multi-point cross and random mutation

We compared GA, PSO, AFSA and UWPA algorithms on four functions, the results of *Best*, *Worst*, *Mean*, *StdDev*, and *SR* are given in TABLE Ⅱ. The best results for each case are highlighted in boldface.

In the experiments, there are 6 high-dimensional functions with variables ranging from 2 to 200. It can be drawn that the efficiency of UWPA becomes much clearer as the number of variables increases. UWPA performs statistically better than the three other state-of-the-art algorithms on high-dimensional functions. Griewank function is a good example for promising UWPA, as it traps other algorithms into local optima or premature at a bad solution in such a great search space, while UWPA successfully avoids falling into the deep local optimum which is far from the global optimum, and get the global optimum 0 in 50 runs computation.

As is also shown in TABLE Ⅱ, *SR* shows the robustness of every algorithms, and it means how consistently the algorithm achieves the threshold during all runs performed in the experiments. UWPA achieves 100% success rate for high-dimensional functions with different characteristics and variable numbers, which shows its good robustness.

Nowadays, high-dimensional problems have been a focus in evolutionary computing domain, since many recent real-world problems (bio-computing, data mining, design, etc.) involve optimization of a large number of variables [15]. It is convincing that UWPA has extensive application in science research and engineering practices.

TABLE II. STATISTICAL RESULTS OF 50 RUNS OBTAINED BY GA, PSO, AFSA AND UWPA ALGORITHMS

Function	Global extremum	D	Algorithms	*Best*	*Worst*	*Mean*	*StdDev*	*SR*/%
Bohachevsky3	$f_{\min}(\vec{x})=0$	2	GA	4.07e-9	2.34e-8	3.78e-9	1.09e-8	**100%**
			PSO	8.19e-12	3.19e-9	5.68e-10	6.70e-10	**100%**
			AFSA	3.66e-11	7.11e-8	9.32e-9	4.27e-9	**100%**
			UWPA	**0**	**0**	**0**	**0**	**100%**
Bridge	$f_{\max}(\vec{x})=3.0054$	2	GA	3.0054	2.9463	2.9981	0.0136	10%
			PSO	3.0054	2.9736	3.0029	0.0092	36%
			AFSA	3.0054	3.0041	3.0052	5.12e-5	52%
			UWPA	**3.0054**	**3.0054**	**3.0054**	**6.24e-8**	**100%**
Ackley	$f_{\min}(\vec{x})=0$	50	GA	11. 4570	12. 6095	12. 1612	0. 2719	0
			PSO	0. 0469	1. 7401	0. 6846	0. 6344	0
			AFSA	20. 1600	20. 6009	20. 4229	0. 1009	0
			UWPA	**8. 88e-16**	**8. 88e-16**	**8. 88e-16**	**0**	**100**
Griewank	$f_{\min}(\vec{x})=0$	100	GA	317. 4525	399. 6376	363. 4174	17. 2922	0
			PSO	0. 0029	0. 0082	0. 0052	0. 0011	0
			AFSA	2. 05e+3	2. 55e+3	2. 33e+3	109. 6821	0
			UWPA	**0**	**0**	**0**	**0**	**100**
Sumsquares	$f_{\min}(\vec{x})=0$	150	GA	5. 93e+4	7. 15+4	6. 63e+4	2. 88e+3	0
			PSO	39. 7098	91. 1145	55. 9050	10. 4165	0
			AFSA	1. 43e+5	1. 79e+5	1. 64e+5	9. 58e+3	0
			UWPA	**2. 08e-170**	**3. 77e-168**	**5. 13e-169**	**0**	**100**
Sphere	$f_{\min}(\vec{x})=0$	200	GA	1. 56e+5	1. 81e+5	1. 71e+5	5. 78e+3	0
			PSO	1. 0361	1. 5520	1. 2883	0. 1206	0
			AFSA	5. 12e+5	5. 79e+5	5. 51e+5	1. 63e+4	0
			UWPA	**8. 69e-172**	**4. 75e-170**	**2. 11e-170**	**0**	**100**

IV. RESULTS ON FUNCTIONS OPTIMIZATION

Proportional-Integral-Derivative (PID) controllers are still used extensively in many control fields because of their robust performance and simplicity. Three parameters of PID controller must be determined and tuned to obtain a satisfactory closed-loop performance.

So, the key for the performance of PID controller is parameters selection [16]. The unstable system is selected as controlled object, its transfer function is shown in (17).

$$G(s) = \frac{s+2}{s^4 + 8s^3 + 4s^2 - s + 0.4} \tag{17}$$

The input signal is unit-step signal. The parameters of PID controller is optimized by uncultivated wolf pack algorithm, and its Simulink model is shown in Fig.5.

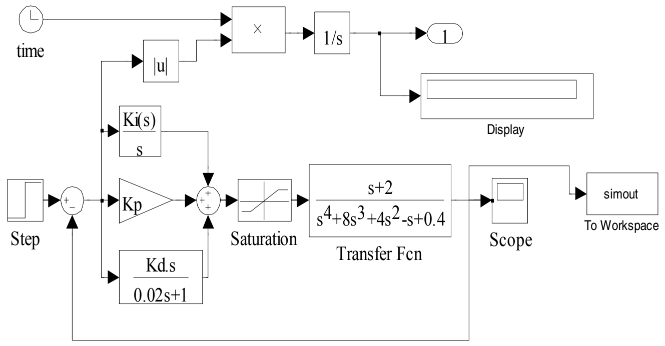

Fig. 5. The Simulink model of PID controller

The parameters of algorithms are the same as ones listed in TABLE Ⅰ. K_p is proportion coefficient, K_i is integral time constant, K_d is differential time constant and K_p, K_i, K_d is searched at the interval [0, 300]. Integral of time-weighted absolute value of the error (ITAE) is the evaluation function according to (18).

$$J = \int_0^{\infty} t\left|e(t)\right| dt \tag{18}$$

The reciprocal of J is considered as objective function. There is a comparison between UWPA, PSO and GA. After optimization, the optimized result of PID parameters is shown in TABLE Ⅲ. And t is regulating time, σ is exceeding quantity.

TABLE III. THE OPTIMIZED RESULTS OF PID PARAMETERS

Method	K_p	K_i	K_d	J	t	σ
UWPA	139. 136	0. 157	165. 071	1. 122	3. 182	0. 96%
PSO	33. 646	0. 166	38. 799	1. 058	3. 257	5. 91%
GA	234. 035	0. 256	269. 389	1. 209	3. 579	5. 42%

By analyzing the TABLE Ⅲ, naturally uncultivated wolf pack algorithm gets the best exceeding quantity. With the help of UWPA, PID controller can get suitable parameters of K_p, K_i, K_d, which make the objective system be controlled well. That validate UWPA again.

V. CONCLUSION

Inspired by the intelligent behavior of wolves, uncultivated wolf pack algorithm (UWPA) is presented for locating the global optima of high-dimensional functions. Compared UWPA with PSO, ASFA and GA on high-dimensional functions such as Sphere (D=200), Sumsquares (D=150), Ackley (D=50) and Griewank (D=100), UWPA possesses superior performance in term of accuracy, convergence speed, stability and robustness. What's more, UWPA also achieves comparative performance on parameters optimization of PID controller.

After all, UWPA is a new attempt, in future, different improvements should be made on the UWPA algorithm and tests can be made on more different test functions. Meanwhile, practical applications in areas of classification, parameters optimization, engineering process control, would also be worth further studying.

ACKNOWLEDGMENT

The authors would like to thank three anonymous referees for their careful reading of the paper and for their helpful comments. This research is partially supported by National Science Foundation of Shaanxi province (2013JQ8042)

REFERENCES

[1] E. Cuevas, M. Cienfuegos, D. Zaldivar, "A swarm optimization algorithm inspired in the behavior of the social-spider," Expert Systems with Applications, vol. 40, no. 16, pp. 6374-6384, November 2013.

[2] G. G. Wang, L. H. Guo, H. Q. Wang, "Incorporating mutation scheme into krill herd algorithm for global numerical optimization, " Neural Computing and Applications, vol. 128 , pp.363-370. March, 2014

[3] K. Passino. "Biomimicry of bacterial foraging for distributed optimization and control," IEEE Control Systems Magazine, vol. 22, pp. 52–67, January 2002.

[4] M. M. Eusuff, K. E. Lansey. "Optimization of water distribution network design using the shuffled frog leaping algorithm," Journal of Water Resources Planning and Management, vol. 129, no. 3, pp. 210-225, May 2003

[5] J. Kennedy and R. Eberhart, "Particle swarm optimization", in Proc. IEEE International Conference on Neural Networks, Perth, Australia, 1995, pp. 1942-1948.

[6] M. Dorigo, "Optimization, learning and natural algorithms," Ph. D. dissertation, Dipartimento di Elettronica, Politecnico di, Italie, 1992.

[7] C. G. Yang, X. Y. Tu, J. Chen. "A lgorithm of Marriage in Honey Bees Optimization Based on the Wolf Pack Search," 2007 International Conference on Intelligent Pervasive Computing, pp. 462-467, 2007.

[8] C. G. Liu, X. H. Yan, C. Y. Liu, et al. "The wolf colony algorithm and its application. Chinese Journal of Electronics", vol. 20, no. 2,pp. 212-216, February 2011.

[9] T. Rui, S. Fong, X. Yang, et al. "Wolf search algorithm with ephemeral memory," 2012 Seventh International Conference on Digital Information Management,pp.165- 172, 2012.

[10] L. X. Li, Z. J. Shao, J. X. Qian, "An optimizing method based on autonomous animals: fish-swarm algorithm", Systems Engineering Theory and practice, vol. 22, no. 11, pp. 32-38. November. 2002.

[11] D. Karaboga, "An idea based on honeybee swarm for numerical optimization", Technical Report TR06, Erciyes University, Engineering Faculty, Computer Engineering Department, 2005.

[12] R. Enayatifar, M. Yousefi, A. H. Abdullah, "LAHS: A novel harmony search algorithm based on learning automata", Commun Nonlinear Sci Numer Simulat, vol. 18, no. 12, pp. 3481-3497. December. 2013.

[13] S. K. S. Fan, Z. Erwie, "A hybrid simplex search and particle swarm optimization for unconstrained optimization", European Journal of Operational Research, vol. 181, no. 2, pp. 527-548, February. 2007.

[14] C. Pilar, B. Francisco, J. A. Becerra, et al, "Evolutionary algorithm characterization in real parameter optimization problems", Applied Soft Computing, vol. 13, no. 4, pp. 1902-1921, April. 2013.

[15] Y. F. Ren, Y. Wu, "An efficient algorithm for high-dimensional function optimization", Soft Computing, vol. 17, no. 6, pp. 995-1004, Junuary. 2013.

[16] K. J. Astrom, T. Hagglund, "The future of PID control, " Control Engineering Practice, vol. 9, no. 11, pp. 1163-1175, November. 2001.