Mortal Particles: Particle Swarm Optimization with Life Span

Yong-wei Zhang, Lei Wang, and Qi-di Wu

College of Electronics and Information Engineering
Tongji University, Cao'an Road 4800
201804, Shanghai, P.R. China
yongwzhang@gmail.com

Abstract. Born and death is the nature of lives, but most swarm intelligence algorithm did not reflect this important property. Based on Particle Swarm Optimization, the concept of life span is introduced to control the activity generation of particles. Furthermore, the differential operator is applied to enhance the convergence and precision. The performance of propose algorithm, along with PSO and DE, is tested on benchmark functions. Results show that life span and differential operator greatly improved PSO and with well-balanced exploration and exploitation characteristic.

Keywords: Swarm Intelligence, PSO, Life Span, Differential, Mutation, Optimization, Algorithm.

1 Introduction

A branch of artificial intelligence which deals with the collective behaviour of swarms through complex interaction of individuals without supervision, is referred to as swarm intelligence [1]. During past decades, many swarm-based optimization algorithms have been developed and put into dealing with real world problems. Ant Colony Optimization (ACO) [2,3]mimics the foraging behavior of ant colonies, which uses the pheromone to deliver the landscape information of target problem between ants. Particle Swarm Optimization (PSO) [4] imitates the social behavior of flock of birds or school of fishes, and the particles share their experiences through a component of velocity update formula. Artificial Bees Colony (ABC) [5] utilizes the divided labor of bees to explore and exploit the nectar resources, and the message about food source is represented by the dances of employee bees. The Differential Evolution (DE) [6], even not simulate any real-world system, is still considered as a swarm intelligence algorithm because of the interaction among individuals.

As an important part of swarm intelligence algorithms, PSO was broadly studied and applied to many engineering problems such as controller design [7,8], task assignment problem [9], scheduling [10], etc. Many variants of PSO

Y. Tan et al. (Eds.): ICSI 2011, Part I, LNCS 6728, pp. 138–146, 2011.

are designed as well, such as PSO with inertia weight[11], PSO with linearly decreasing $V_{\max}$[12], PSO with collision-avoiding mechanism [13] etc..

Though PSO claims to be an algorithm that imitates the behavior of birds or fishes, the behavior of particles is more like a bunch of robots than living biomes. Except the initial phase, the particles never born or die. In other words, the particles are immortal. To the author's knowledge, few literatures mentioned the life span of particles, not along the effect the living particles will bring.

Recently, the differential operator, as a high-efficient heuristic, is increasingly introduced into various algorithms. For example, in Shuffled frog-leaping algorithm (SFLA)[14], the new solutions are generated based on the difference between current solution and the best solution of the memeplex. In the latest version of ABC [15], a differential operation is also used. The appropriate utilization of differential operator balances the exploration and exploitation of a search process. For PSO, there is no doubt that the introduction of differential operator will accelerate convergence and improve accuracy.

In our study, we introduce the concept of life span into the PSO scheme (mortal particles). To further improve the algorithm performance, the differential operator is used to generate candidate solutions out of the memory of individual particles. The performance of proposed algorithm, called Mortal Differential PSO (MDPSO), is compare with the original algorithms (immortal particles).

The rest of this paper is organized as follow. Section two briefly introduces the PSO algorithm. In section three, we propose the Mortal PSO scheme. The experiment results are presented in section four. Section five gives discussion and conclusion.

2 Particle Swarm Optimization

Since first introduced by Kennedy and Eberhart [4] in 1995, PSO has developed numerous variants. A detailed description of these variants of PSO can be found at [16]. Our study is focused on original PSO.

A D-dimension unconstrained minimization problem is defined as follow.

$$\min f(\boldsymbol{x}), \boldsymbol{x} = [x_1, \cdots, x_D] \tag{1}$$

where D is the dimension of the problem or the number of parameters to be optimized, $\mathbf{x}$ is a solution of the problem, $x_{i\min} \leq x_i \leq x_{i\max}$.

In Particle Swarm Optimization, the position of a particle represents a solution. And the motion of particle P_i is driven by velocity V_i. Through updating the velocity and the position of particles, the feasible space is searched. The updating formula of velocity and position are as follow.

$$V_i(t+1) = V_i(t) + c_1 rand_1(pbest_i - P_i) + c_2 rand_2(gbest - P_i) \tag{2}$$

$$P_i(t+1) = P_i(t) + V_i(t+1) \tag{3}$$

where $pbest_i$ is the best position previously discovered by particle P_i, $gbest$ is the best position discovered by the entire population. $rand_1$ and $rand_2$ are uniformly

distributed random number in region [0,1]. c_1 and c_2 are accelerating constants that reflect the weighting of stochastic acceleration term that pull particles to *pbest* and *gbest* position[16]. The range of P_i and V_i is limited in $[X_{\min}, X_{\max}]$ and $[V_{\min}, V_{\max}]$. When parameter exceeds the upper or lower bounds, we simply set parameter as the value of bounds. Some researcher use different random numbers in every dimension, as defined in (4)-(5).

$$V_i^d(t+1) = V_i^d(t) + c_1 rand_1^d(pbest_i^d - P_i^d) + c_2^d rand_2^d(gbest^d - P_i^d) \quad (4)$$

$$P_i^d(t+1) = P_i^d(t) + V_i^d(t+1) \quad (5)$$

where superscript d is the dimension index. According to (4), every component of velocity V_i needs newly generated random numbers weighting the portion of individual and global terms. The difference between (2)-(3) and (4)-(5) is reported in [17]. For clear discussion, we use (2)-(3) in our study.

3 Mortal Particle Swarm Optimization

3.1 Mortal Particles

Death, as one kind of updating mechanisms in biomes, is an indispensible part of evolution. To sustain the population with limited resources, the old and weak individuals have to make room for the young and fitter ones. In the context of swarm intelligence, the elimination of old solutions plays the same role with death in the real world. However, most swarm intelligence algorithms did not take the life span of solutions into account. From the beginning to the end of a search process, a solution is only changed, not been replaced. If the change stops, i.e., the solution trapped in local extreme, then the premature convergence is unavoidable unless some other operations are introduced.

Consider (2), when a solution is trapped, it means *gbest* and $pbest_i$ are remain unchanged. This solution can easily replicate itself and occupy the whole population. Because the *gbest* and $pbest_i$ will attract the rest particles to them, and eventually to *gbest*[16]. Since all the particles are immortal, the algorithm will end up with a population that has same individuals. To help the solution escape local, we use 'life span' to control the number of activity generations of a particle. For problem (1), the life span of a particle is decided by following formula.

$$life_i = \exp\left(\frac{\min_{i=1}^{N}(f_i) - f_i}{\sum_{i=1}^{N}(f_i - \min_{i=1}^{N}(f_i))/N}\right) \quad (6)$$

where f_i is the evaluation of object function, N is the number of particles.

The life span of each particle is within (0,1]. The life of a particle will reduce Δ with every position update. When $life_i \leq 0$, the $pbest_i$ particle will be eliminated and reinitialized within the search space. The $pbest_i$ is reinitialized as well to erase the memory of the particle, otherwise the new particle will be attracted to $pbest_i$ again.

In some cases, the evaluation of some particle are far bigger than the rests, thus leads to a large denominator in (6), and eventually shorten the life span of all particles. In that case, the following formula can be used instead.

$$life_i = \exp\left(\frac{\min_{i=1}^{N}(f_i) - f_i}{\text{median}(f_i - \min_{i=1}^{N}(f_i))/N}\right) \tag{7}$$

where $\text{median}(f_i - \min_{i=1}^{N}(f_i))$ is median number.

3.2 Differential Operator

With mortal particle, the algorithm can escape local now. But the convergence speed is still a problem. If the algorithm reinitializes the solutions again and again to explore wider region, it becomes random search. In addition, the life span did not necessarily speed up the convergence. In this context, we introduce differential operator to improve the convergence.

$$Pn_i^d = \begin{cases} pbest_i^d + (pbest_{r1}^d - pbest_{r2}^d), & \text{if } rand < p \\ pbest_i^d, & \text{else} \end{cases} \tag{8}$$

$$\begin{array}{c} r1,\ r2 \in [1, \cdots, N] \\ r1 \neq r2 \end{array} \tag{9}$$

where Pn_i^d is the d^{th} variable of i^{th} new particle, p is the mutation probability, $r1$ and $r2$ are random integers. From our observation, $p = 0.15$ to 0.25 is suitable for most problems.

The P_i and $pbest_i$ is updated according to following.

$$f_{\text{obj}}(Pn_i(t)) < f_{\text{obj}}(pbest_i) \longrightarrow P_i(t+1) = Pn_i(t),\ pbest_i = Pn_i(t) \tag{10}$$

where $f_{\text{obj}}(\cdot)$ is the object function.

The pseudo code of MDPSO see Algorithm 1. Where FE and $FE_{\max}$ denote the number of function evaluation and the maximum of it.

4 Validation Experiments

4.1 Parameter Setting

To verify the proposed algorithm, numerical experiments are conducted. The performance of MDPSO is compared with PSO and DE. The DE parameters are set according to [18]. The rest parameters are set as follow. Accelerating constants $c1 = c2 = 2$, population size $N = 20$, mutation probability $p = 0.15$, maximum number of function evaluation $FE_{\max}$=2e5, velocity V_i is limited by (11).

$$\begin{array}{c} V_{\max} = (X_{\max} - X_{\min})/10 \\ V_{\min} = -(X_{\max} - X_{\min})/10 \end{array} \tag{11}$$

Algorithm 1. Mortal Differential PSO

```
Object function f(x), x = (x_1, ..., x_D)^D
Generate N particles P_i(i = 2, 3, ... N), evaluate particles, record pbest_i and gbest
Compute life value of particles by (7)
while FE < FE_max do
  for each particle P_i do
    Update V_i and P_i by (2)-(3), FE = FE + 1, evaluate P_i : f_i = f_obj(P_i)
    if f_i < f_obj(pbest_i) then
      pbest_i = P_i
    else
      life_i = life_i − Δ
    end if
    if life_i < 0 then
      Reinitialize P_i and evaluate it, FE = FE + 1
    end if
  end for
  Generate new particles by (8)-(9) and evaluate them, FE = FE + N
  Update P_i and pbest_i according to (10)
  Update particle life by (7) and record gbest
end while
Post process results and visualization
```

4.2 Benchmark Functions

The benchmark functions are usually used as standard test bed for optimization algorithms. To avoid the unintentional attraction of zeros, the benchmark functions are shifted with a displacement $\boldsymbol{d} = (d_1, d_2, \cdots d_D)^D$ according to following formula.

$$\begin{aligned} \boldsymbol{z} &= \boldsymbol{x} - \boldsymbol{d} \\ f &= f_{\text{obj}}(\boldsymbol{z}) \end{aligned} \tag{12}$$

For example, the Non-continuous Rastrigin function has numerous local minima, which located at small flat local regions. The flat region has no differential information and the function is un-derivable within its search space, which makes this function very difficult for most search algorithms. The function's shifted version landscape is shown at Fig. 1. The benchmark functions used in our experiment are listed at Table 1. To test the overall performance of proposed algorithm, we try to cover all kinds of test functions. As the characteristic showed at Table 1, four functions are unimodal and four are multimodal. At the same time, four functions are separable and four are non-separable.

4.3 Simulation Results

The initial and final locations of 20 particles of 2-D Non-continuous Rastrigin function in MDPSO are shown at Fig. 2. The final locations are around the optimal solution [1.28,1.28], but distances between particles are still fairly large. This explains why MDPSO can escape local. To further demonstrate the

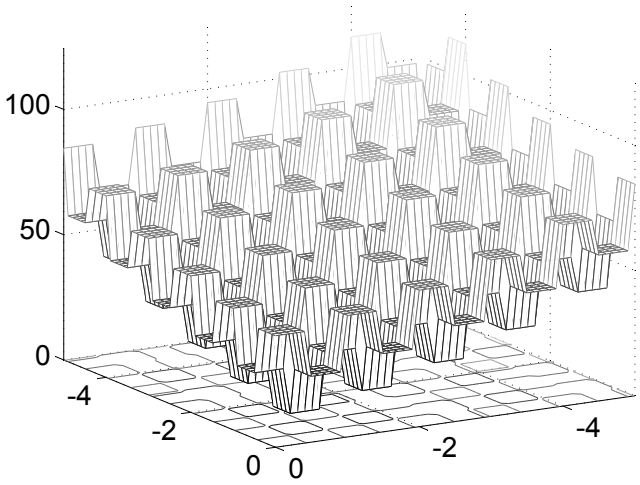


Fig. 1. The 3-D landscape of Shifted Non-continuous Rastrigin function

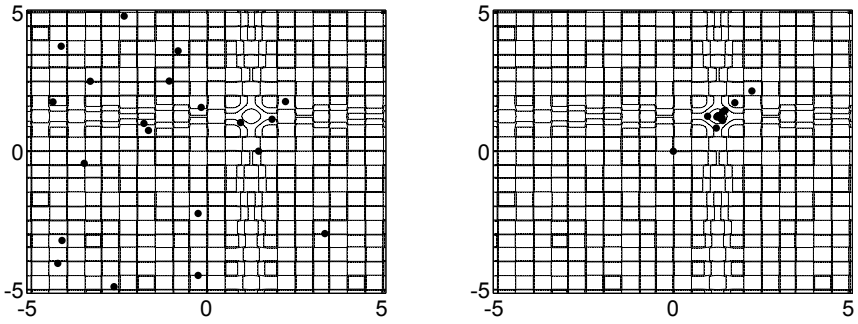
Table 1. Benchmark Functions. D: Dimension, F: Feasible Bounds, C: Characteristic, U: Unimodal, M: Multimodal, S: Separable, N: Non-Separable, R: Regular, I: Irregular.

Name	Equation	D	F	C	**d**
Sphere	$f_i = \sum_{i=1}^{D} x_i^2$	30	[-100, 100]	USR	25^D
Schwefel 2.21	$f_2 = \max_{i=1}^{D} \lvert x_i \rvert$	30	[-100, 100]	USI	25^D
Rosenbrock	$f_3 = \sum_{i=1}^{D} \left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right]$	30	[-30, 30]	UNR	0^D
Schwefel 2.22	$f_4 = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_i \rvert$	30	[-10, 10]	UNI	2.5^D
Rastrigin	$f_5 = \sum_{i=1}^{D} \left[x_i^D - 10\cos(2\pi x_i) + 10\right]$	30	[-5.12, 5.12]	MSR	1.28^D
Non-continuous Rastrigin	$f_6 = \sum_{i=1}^{D} \left[y_i^D - 10\cos(2\pi y_i) + 10\right]$ $y_i = \begin{cases} x_i & \lvert x_i \rvert < \frac{1}{2} \\ \frac{\text{round}(2x_i)}{2} & \lvert x_i \rvert \geq \frac{1}{2} \end{cases}$	30	[-5.12, 5.12]	MSI	1.28^D
Griewank	$f_7 = \frac{1}{4000} \sum_{i=1}^{D} x_i^D - \prod_{i=1}^{D} \cos(\frac{x_i}{\sqrt{i}}) + 1$	30	[-600, 600]	MNR	150^D
Ackley	$f_8 = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right)$ $-\exp\left(\frac{1}{D}\sum_{i=1}^{D} \cos(2\pi x_i)\right) + 20 + \text{e}$	30	[-32.768, 32,768]	MNR	8.192^D

effectiveness of proposed algorithm, 50 runs are conducted for each algorithm, and the results are summarized at Table 2. For each benchmark function, the best, median, worst, mean, standard deviate (Std) and success (Scr) rate of 50 runs are listed. The success rate is determined by (13)-(14).

$$\text{Scr} = N_{\text{success}}/\text{Maxrun} \times 100\% \tag{13}$$

$$|f_{\text{best}} - f_*| < \varepsilon \tag{14}$$



(a) Initial Locations (b) Final Locations

Fig. 2. Initial and final 2-D locations of 20 particles in MDPSO

where N_{success} is the number of success runs. If f_{best} satisfy (14), a success run is recorded. In our experiment, the tolerance is truncated at 1.00e-8, which means all values that less than 1.00e-8 are treated as zero. From Table 2 we can conclude that MDPOS outperforms PSO for all eight test functions and outperforms DE for three multimodal functions (Rastrigin, non-continuous Rastrigin and Griewank) and one unimodal function (Schwefel 2.21). DE outperforms MDPSO for two unimodal non-separable functions (Rosenbrock and Schwefel 2.22). Both MDPSO and DE have same performance for Sphere and Ackley function.

Table 2. Comparison of Three Algorithms

	Function	f_1	f_2	f_3	f_4	f_6	f_6	f_7	f_8
	ε	1.00e-8	0.5	25.0	1e-8	1e-8	1e-8	1e-8	1e-8
PSO	Best	0.07	0.78	29.04	0.16	33.91	38.00	0.22	0.10
	Median	0.19	1.41	74.15	0.27	63.73	69.00	0.40	0.22
	Worst	0.38	3.29	368.86	0.46	94.62	126.00	0.64	1.71
	Mean	0.19	1.60	87.93	0.27	63.88	71.88	0.41	0.53
	Std	0.07	0.57	70.88	0.07	14.61	16.96	0.09	0.50
	Scr	0%	0%	0%	0%	0%	0%	0%	0%
DE	Best	0.00	0.06	0.04	0.00	15.69	32.67	0.0	0.0
	Median	0.00	2.24	17.40	0.00	83.48	81.43	0.0	0.0
	Worst	0.00	20.66	69.43	0.00	133.97	118.62	7.40e-3	0.0
	Mean	0.00	3.44	26.12	0.00	84.25	81.25	1.48e-4	0.0
	Std	0.00	4.02	21.63	0.00	27.56	22.97	0.001	0.0
	Scr	100%	16%	78%	100%	0%	0%	98%	100%
MDPSO	Best	0.00	0.17	4.80e-3	0.00	0.00	0.00	0.00	0.00
	Median	0.00	0.28	20.78	0.00	0.00	0.00	0.00	0.00
	Worst	0.00	0.41	81.90	12.50	5.59e-7	0.00	0.00	0.00
	Mean	0.00	0.29	33.19	0.25	1.16e-8	0.00	0.00	0.00
	Std	0.00	0.05	27.92	1.75	7.82e-8	0.00	0.00	0.00
	Scr	100%	100%	72%	98%	96%	100%	100%	100%

Clearly MDPSO performed well when deals with multimodal functions. Especially for Rastrigin and non-continuous Rastrigin function, MDPSO is far more superior. On the other hand, the advantage of DE is not obvious. In general, the proposed algorithm is more efficient than PSO and DE.

5 Discussion and Conclusion

From the experiment results we can conclude that the life span and differential operator significantly improved the performance of PSO and showed great potential on dealing with multimodal problems. With life span, some old particles are eliminated, so is its memory, *pbest*. With the elimination of old particles, the population diversity is increased. Technically the population will not converge to a same local extreme (like the original PSO always do), hence the algorithm will not trap. Theoretically, as long as the algorithm keeps running, it will find the global minima eventually.

Though the algorithm will not trap in local, it is not practical if it takes long time to converge. A differential operator can speed up convergence and increase the accuracy. In the early stage of searching process, the *pbest* records only the information of scattered local regions, and the average distance of particles is large. The new generated particle can search broader region. In the later stage, with the decreasing distance of particles, the differential factor becomes smaller. This drives the algorithm to higher precision. The process is self-adaptive, the exploration and exploitation are well balanced.

The implementation of life span and differential operator dose not depend on algorithms, which means the proposed improvement can be applied to other algorithms as well. PSO is just an example, how to improve other algorithms by these heuristics is still an open issue.

Acknowledgement. The research is supported by Ph.D. Programs Foundation of Ministry of Education of China (No. 20100072110038), National Natural Science Foundation of China (NSFC, No. 70871091, 61075064, 61034004), Foundation of the Ministry of Education of China for Returned Scholars and Program for New Century Excellent Talents in University of Ministry of Education of China.

References

1. Akay, B., Karaboga, D.: A modified Artificial Bee Colony algorithm for real-parameter optimization. Inform. Science (in press) (2010), Corrected Proof doi:10.1016/j.ins.2010.07.015
2. Dorigo, M., Maniezzo, V., Colorni, A.: The Ant System: Optimization by a colony of Cooperating agents. IEEE Trans. Syst. Man. Cybern. Part B. Cybern. 26, 29–41 (1996)
3. Dorigo, M., Birattari, M., Stutzle, T.: Ant colony optimization. IEEE Comput. Intell. Mag. 1, 28–39 (2006)

4. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: IEEE International Conference on Neural Networks, pp. 1942–1948 (1995)
5. Karaboga, D.: An idea based on honey bee swarm for numerical optimization. Technical report, Computer Engineering Department, Engineering Faculty, Erciyes University (2005)
6. Storn, R., Price, K.: Differential Evolution - A simple and efficient adaptive scheme for global optimization over continuous spaces. Technical report, International Computer Science Institute (1995)
7. Zamani, M., Sadati, N., Ghartemani, M.K.: Design of an H PID controller using Particle Swarm Optimization. Int. J. Contr. Autom. Syst. 7, 273–280 (2009)
8. Zhang, Y., Qiao, F., Lu, J., Wang, L., Wu, Q.: Performance Criteria Research on PSO-PID Control Systems. In: 2010 International Conference on Intelligent Computing and Cognitive Informatics (ICICCI), pp. 316–320 (2010)
9. Salman, A., Ahmad, I., Al-Madani, S.: Particle swarm optimization for task assignment problem. Microprocess. Microsy. 26, 363–371 (2002)
10. Bo, L., Ling, W., Yi-Hui, J.: An Effective PSO-Based Memetic Algorithm for Flow Shop Scheduling. IEEE Trans. Syst. Man. Cybern. Part B. Cybern. 37, 18–27 (2007)
11. Shi, Y., Eberhart, R.: A modified particle swarm optimizer. In: Proc. of the IEEE Int'l Conf. of Evolutionary Computation, pp. 69–73. IEEE Press, Piscataway (1998)
12. Fan, H.Y., Shi, Y.: Study on Vmax of particle swarm optimization. In: Workshop Particle Swarm Optimization (2001)
13. Blackwell, T.M., Bentley, P.: Don't push me! Collision-avoiding swarms. In: Proceedings of the 2002 Congress on Evolutionary Computation (CEC 2002), pp. 1691–1696 (2002)
14. Eusuff, M.M., Pasha, K.L.F.: Shuffled frog-leaping algorithm: a memetic metaheuristic for discrete optimization. Eng. Optimiz. 38, 129–154 (2006)
15. Karaboga, D., Basturk, B.: A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J. Global. Optim. 39, 459–471 (2007)
16. Liang, J.J., Qin, A.K., Suganthan, P.N., Baskar, S.: Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. Ieee T. Evolut. Comput. 10, 281–295 (2006)
17. Wilke, D.N.: Analysis of the particle swarm optimization algorithm. Dept. Mechanical and Aeronautical Eng., Univ. of Pretoria, Pretoria, South Africa, (2005)
18. Pedersen, M.E.H.: Good Parameters for Differential Evolution. Technical report, Hvass Computer Science Laboratories (2010)