A Quantum Particle Swarm Optimization

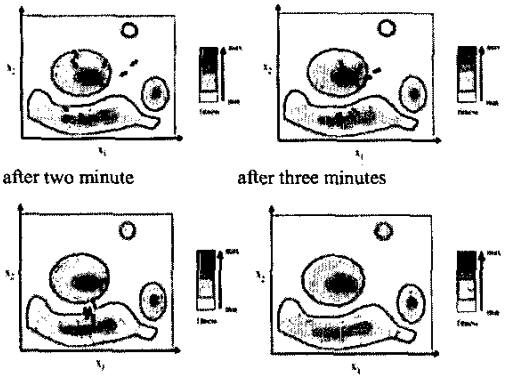
Shuyuan Yang, Min Wang
Dept. of Electrical and Computer Engineering
Xidian Univ, Xi'an,Shannxi
China, 710071
Email: syyang@xidian.edu.cn

Licheng Jiao
Institute of Intelligent Information Processing
Xidian Univ, Xi'an,Shannxi
China, 710071
Email: ysy20021978@etang.com

***Abstract*-The particle swarm optimization algorithm is a new methodology in evolutionary computation. It has been found to be extremely effective in solving a wide range of engineering problems, however, it is of low efficiency in dealing with the discrete problems. In this paper, a new discrete particle swarm optimization algorithm based on quantum individual is proposed. It is simpler and more powerful than the algorithms available. The simulations experiments and its application in the CDMA also prove its high efficiency.**

I. INTRODUCTION

We often see the birds, fish assemble in groups and look for food together. This kind of social behavior help them find food and escape being captured. Once an individual finds food, it informs the others by the chemical signal or physical touch, which helps the birds fly towards the food source. By observing a flock of birds, one can find that each bird fly at random initially,but they organize to a community gradually and fly towards a same place as time goes on, then the small groups assemble into a big one. It may disperse for pieces of small groups again,but finally the flock gathers to a same place-food source.We can see that the information shares in the whole colony, and exchanges with each other among the individuals[1,2]. Figure 1 descibes the movement of a swarm.



after two minute after three minutes

after four minutes after five minutes

Figure 1 The movement process of particles

Particle swarm optimization (PSO) is an evolutionary computation technique developed by Dr. Eberhart and Dr. Kennedy in 1995, which is inspired by social behavior of bird flocking and fish schooling[3][4].It is somelike genetic algorithm(GA) for it also begins with a random population and searches for optima by updating the population. While PSO has no evolution operators such as the crossover and mutation. In PSO, the potential solutions, called particles, "flown" through the whole space by following the current optimum particles.Each particle keeps track of the best solution it has experienced so far(pbest).The other "best" solution being tracked is the best individual in a particle's neighbors (lbest). When a particle takes all the individuals as its topological neighbors, then the best value is a global best(gbest). Each particle changes its velocity toward the pbest and lbest at each iteration. Acceleration is weighted by a random number toward pbest and lbest locations[4]-[6].

Comparing with GA, PSO's advantages lies on its easy implementation and few parameters to adjust. Now PSO has been applied in many fields successfully such as function optimization, training of artificial neural network, fuzzy system control ect[7-11].

PSO is essentially a swarm intelligent(SI) method. Swarm Intelligence is a kind of model which simulates the social behavior of insect, where "swarm" means a group of items which can communicate with each other in a direct or indirect way. Although the research on SI is still at primary stage, and there are a lot of difficulties, it has a success development, such as Ant Colony Optimization-ACO, PSO, Cluster's algorithm etc.

II. PARTICLE SWARM OPTIMIZATION

PSO is initially used in the continuous optimization, but now it has been expanded to discrete variable. Though its strict convergence has not been proved, it performs better than GA remarkably. Therefore, it has already found lots of applications in recent years.

In a basic PSO, a swarm comprises a group of particles, firstly many particles are distributed in the searching space (each particle has a speed vector and position vector to represent a possible solution), then the particles fly rapidly over and search the space, and record the best individuals they have met. At each step, they change their positions according to the best individuals to reach a new position. In this way, the whole population evolves towards the optimum step by step.

PSO is of a simple rule. Before a change, each particle has three choices: 1. insist on oneself 2. move towards the optimum it has met3.move towards the best the population has met. PSO reaches a balance among these three choices. The algorithm is described as follows:

$$\vec{p}(k+1) = \vec{p}(k) + \vec{v}(k) \quad (1)$$

$$\vec{v}(k+1)=c_1\times\vec{v}(k)+c_2\times r(0,1)\times(\vec{p}_{selfbest}(k)-\vec{p}(k))+c_3\times r(0,1)\times(\vec{p}_{groupbest}(k)-\vec{p}(k)) \quad (2)$$

where $\vec{p}$ represent a particle with speed vector $\vec{v}$;$r(0,1)$ is a random number in [0,1]; $p_{groupbest}$ and $p_{selfbest}$ are the best individula the population and particle has met; c_1, c_2, c_3 are called the coefficient of inertia,society study and cognitive respectively. They represent the degree of belief on itself, its experience and its neighbors.Figure 2 describes the PSO algorithm. The standard PSO has too much reliance on $p_{selfbest}$ and $p_{groupbest}$,which may limits its searching. Moreover, the population size and parameters are difficult to determine,so the algorithm presents a slow convergence in the later stage. Accordingly, some improved PSO have appeared[12-15],such as adjusting population size and PSO's coefficient adaptively,or combining PSO with other algorithms. For example, a goal function and an admissible error are defined according to the population's energy and relative momentum, every n steps they are estimated again. In another example, "hopeful"convergence standard is defined, and the population is

able to update its position according to the goal function and its last position.

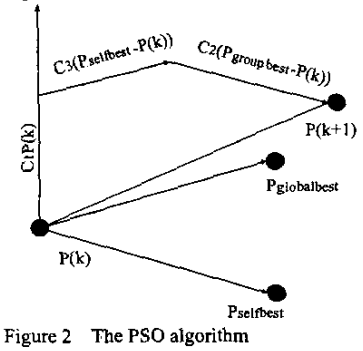


Figure 2 The PSO algorithm

One can find the flowchart of a standard PSO in figure 3.

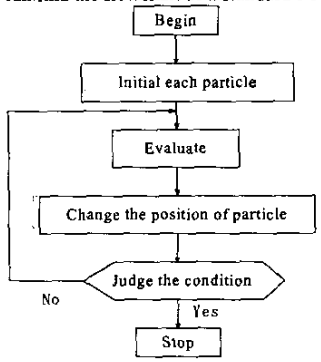


Figure 3 The flowchart of standard PSO

III. A DISCRETE QUANTUM PSO

A. The discrete PSO algorithm

There have been some discrete PSOs. For example, in the algorithm proposed by Eberhart[16], each gene's taking 0 or 1 in a particle is effected by many factors, which can be expressed as:

$$P(p_i^{k+1}=1)=f(p_i^k,v_i^k,p_{selfbest},p_{groupbest}) \quad (i=1,\cdots m) \tag{3}$$

where m is the length of particles; $f(.)$ is a function which satisfies $0<f(x)<1$ for any $x\in(-\infty,+\infty)$. Obviously, there are many functions that can satisfy this condition. Sigmoid function commonly used in neural network is one of them. The discrete PSO algorithm that uses sigmoid function can be expressed as:

$$\vec{v}_i^{k+1}=c_1\times\vec{v}_i^{k}+c_2\times r(0,1)\times(\vec{p}_{selfbest}^{k}-\vec{p}_i^{k})+c_3\times r(0,1)\times(\vec{p}_{groupbest}^{k}-\vec{p}_i^{k}) \tag{4}$$

$$sigmoid(v_i^k)=1/[1+\exp(-v_i^k)] \tag{5}$$

$$\text{if } \rho_i^k < sigmoid(v_i^{k+1}) \text{ then } p_i^k=1 \text{ else } p_i^k=0 \tag{6}$$

where $\rho_i^k\in[0,1]$ is a random number. Though it has been proved: PSO can also be used in discrete optimization as a common optimization method, it is not so effective as in successive optimization. An outstanding advantage is its computational complexity, and the sigmoid function only plays a role of mapping, so it hasn't much meaning.

B. The quantum discrete PSO

As a foundation of modern science and technology, the development of Physics brings a deep understanding of nature. At the same time of exploring the "biologic evolution mechanism", people are also inspired by the idea of "simulating things". The two ideas permeate into each other and produce a lot of successful theories, and the simulation anneal algorithm-SAA is one of them[7]. In the advanced algorithm, we define a particle $Q(t)$ based on the quantum bit(qubit). A random observation is uses to replace the sigmoid function, and the best chromosome's guidance is also used to draw close to the optimum step by step, so we call this new kind of discrete PSO the quantum particle swarm optimization-QPSO.

In the quantum theory, the minimum unit that carries information is a qubit, which can be in any superposition of state 0 and 1. We define such a quantum particle vector:

$$Q(t)=[q^1(t),q^2(t),\cdots q^N(t)]\ (q^j(t)=[q_1^j(t),q_2^j(t),\cdots q_m^j(t)])$$

where $0\le q_i^j(t)\le 1$ $(i=1,\ldots m, j=1,\ldots N)$; m is the particle's length and N is the population size; $q_i^j(t)$ represents the probability of the i-th bit of the j-th particle being 0 at the t-th generation. Then how to get a discrete particle vector from a quantum particle vector? The answer is to perform a random observation which can be described as: For each $q_i^j(t)$ $(i=1,\ldots m, j=1,\ldots N)$, generate a random number, if it is greater than $q_i^j(t)$, then $p_i^j(t)=1$; otherwise $p_i^j(t)=0$ ($p_i^j(t)$ is the corresponding discrete particle of the quantum particle $q_i^j(t)$), then the QPSO algorithm can be described as:

$$Q_{groupbest}(k)=\alpha\times p_{groupbest}(k)+\beta\times(1-p_{groupbest}(k)) \tag{7}$$

$$Q_{selfbest}(k)=\alpha\times p_{selfbest}(k)+\beta\times(1-p_{selfbest}(k)) \tag{8}$$

where $\alpha+\beta=1, 0<\alpha,\beta<1$ are called the control parameters which represent the control degree of Q. The smaller of α, the bigger of the appear probability of the desired item

$$Q(k+1)=c_1\times Q(k)+c_2\times Q_{selfbest}(k)+c_3\times Q_{groupbest}(k) \tag{9}$$

where $c_1+c_2+c_3=1, 0<c_1,c_2,c_3<1$ represent the degree of the belief on oneself, local maximum and global maximum respectively. The procedure of QPSO is described in the above procedure.The flowchart is shown in figure 4.

```
begin
    t =0
    initialize  Q(t)  and  P(t)
    evaluate  P(t)
    store  the  best  solution  among  P(t)
    while  ( not  termination -condition )  do
    begin
       t=t + 1
       change Q(t)  using the PSO algorithm:
            q_selfbest(t)=α p_selfbest + β(1 − p_selfbest)
            q_globalbest(t)=α p_globalbest + β(1 − p_globalbest)
            q(t+1)=c1q(t) + c2q_selfbest(t)+c3q_globalbest(t)
       observing  Q(t)  to get P(t):
            if   rand>q_i^j(t)    (∀i ∈ N[1,N],j ∈ N[1,m])
               p_i^j(t) = 1
            else p_i^j(t) = 0
       evaluate  P(t)
       store  the  best  solution among  P(t)
    end
end
```

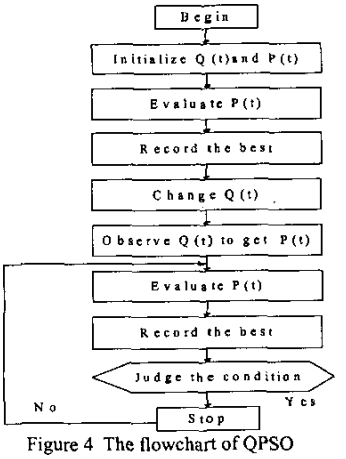


Figure 4 The flowchart of QPSO

IV. SIMULATIONS

A. Knapsack problem

Knapsack problem is a typical combinational optimization, which can be described as follows: Suppose m items with values $C_i>0$ and weights $W_i>0$ ($i=1,\ldots,m$), and a knapsack with maximum weight V. Which items are choosen to get a maximum total values. In the following experiment, m=100, Q is a quantum particle of length m, P is a binary string with 0 and 1of the same length, where "1" means packed in, "0" means not. We observe the effect of α,β and c_1, c_2, c_3 on QPSO, and compare it with other algorithms. The population size is 6 and the maximum iteration is 500. In GA, the crossover and mutation probability are 0.7, 0.1. QA is an algorithm which only lacks "using PSO algorithm to change $Q(t)$"in relative to QPSO. We take average result of 50 times, and get the result in figure 5 ($\alpha=0.3, \beta=0.7$, c_1=c_2=0.1 and figure 6 ($\alpha=0.3, \beta=0.7$, c_1=0.2,c_2=0.2,c_3=0.6).

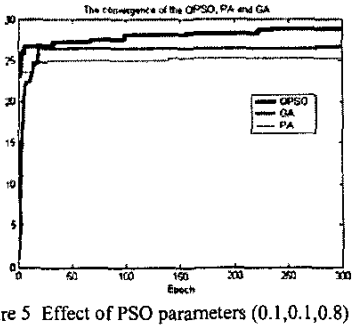


Figure 5 Effect of PSO parameters (0.1,0.1,0.8)

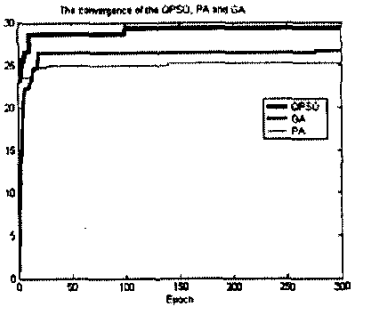


Figure 6 Effect of PSO parameters (0.2,0.2,0.6)

In figure 7, α,β are changed.

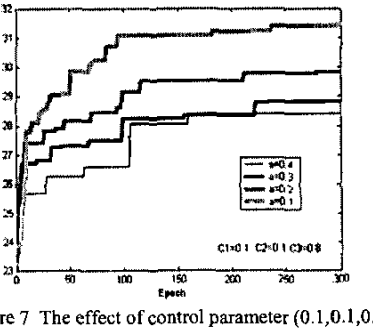


Figure 7 The effect of control parameter (0.1,0.1,0.8)

B. Function optimization

We take a optimization of a 2-dimension function in as an example to investigate the performance of QPSO.

$$f(x_1,x_2)=\frac{1}{\sqrt{1+x_1^2+x_2^2}}-4x_1^2x_2e^{\frac{-\sqrt{x_1^2+x_2^2}}{2}} \quad (10)$$

The problem of searching for the minimum can be described as: searching for the (x_1^{max},x_2^{max}) with the maximum function in $x\in[-10,10], y\in[-10,10]$:

$$f(x_1^{max},x_2^{max})\geq f(x_1,x_2) \quad (11)$$

Its function curve is shown in figure 8. Similarly, the convergence curve of the four algorithms are shown in figure 9.

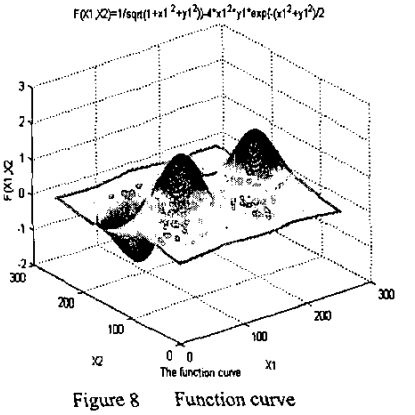


Figure 8 Function curve

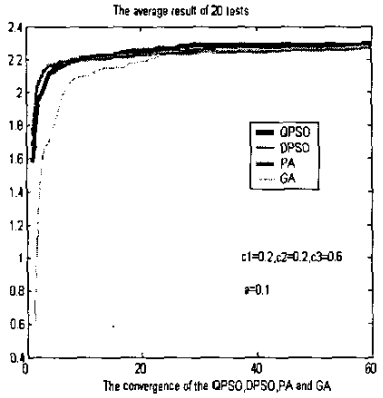


Figure 9 The convergence curve

We take the function of F1-F6 to have a further investigation of QPSO.

Table I. F1-F6

F	Representation	Range
F1	$f(x_1,x_2)=100(x_1-x_2^2)^2+(x_2-1)^2$	$x_1,x_2\in[-5,5]$
F2	$f(x_1,x_2)=4x_1^2-2.1x_1^4+\frac{x_1^6}{3}-x_1x_2+4x_2^2$	$x_1,x_2\in[-5,5]$
F3	$f(x_1,x_2)=\sum_{i=1}^{5} i\cos[(i+1)x_1+i]\sum_{i=1}^{5} i\cos[(i+1)x_2+i]$	$x_1,x_2\in[-10,10]$
F4	$f(x_1,x_2)=[x_2-\frac{5.1}{4\pi^2}x_1^2+\frac{5}{\pi}x_1-6]^2+10(1-\frac{1}{8\pi})\cos x_1+10$	$x_1\in[-5,10]$ $x_2\in[0,15]$
F5	$f(x_1,x_2)=-0.5+\frac{\sin^2(\sqrt{x_1^2+x_2^2})-0.5}{[1+0.01(\sqrt{x_1^2+x_2^2})^2]^2}$	$x_1,x_2\in[-100,100]$
F6	$f(x)=\frac{1}{2}\sum_{i=1}^{10}(x_j^4-16x_j^2+5x_j)$	$x_j\in[-10,10]$

In table 1, F1 is the function of Rosenborock、F2 is a 6-hump function、F3 is a 2-dimension Shubert function、F4 is a Brainin function. Taking the average of 50 tests, we get the result in table 2,where DPSO means the DPSO using Sigmoid function and "O" and "B"means float coding and binary coding respectively.

Table II. The result of 50 tests

	QPSO	DPSO	PA	GA (B)	GA (O)	Optimum
F1	1.532 $\times10^{-14}$	1.357 $\times10^{-13}$	2.642 $\times10^{-6}$	2.914 $\times10^{-7}$	1.215 $\times10^{-14}$	0
F2	-1.0128	0.5629	0.1332	-0.0043	-1.0316	-1.0316
F3	-185.88	-135.44	-40.943	-144.38	-186.70	-186.70
F4	0.3977	0.9482	1.0240	0.1559	0.3977	0.3977
F5	-0.9976	-0.8401	-0.0692	-0.8877	-0.9834	-1
F6	-391.25	-327.66	-216.08	-227.08	-390.25	-391.66

C. *Application in CDMA*

The development of the Code Division Multiple Access-CDMA is like a raging firein recent years, at present it has become the mainstream of the third generation of mobile communication. Multi-user detection-MUD is an efficient way to smooth away the interference of users and decline of channels. *S.Verdu* studied the MUD in Gauss channel and advanced an optimum MUD(OMUD) under a premise of equal transmit probability of all users. For synchronous CDMA, OMUD means searching for a likelihood function $J(b)$ in all 2^k solutions. For asynchronous CDMA, OMUD can be fulfilled by matched filters and *Viterbi* algorithm.

$$\tilde{b}=\arg\max\{\exp[-\frac{1}{2\sigma^2}\times(\int_0^T[y(t)-\sum_{k=0}^{K}b_kA_kS_k(t)]^2dt]\} \quad (12)$$

$$y_k=\int_0^T y(t)s_k(t)dt=A_kb_k+\sum_{j\neq k}A_jb_j\rho_{j,k}+n_k \quad (13)$$

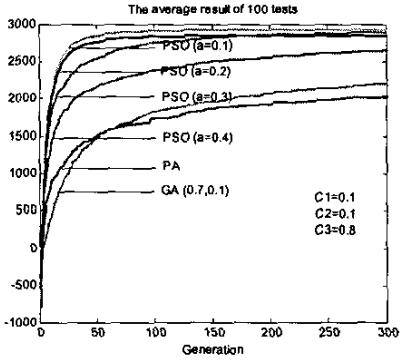
$$n_k=\sigma\int n(t)s_k(t)dt \quad (14)$$

where y is the output signal of the matched filter; $b_k\in\{-1,1\}$, A_k and $S_k(t)$ are the transmit bit, characteristic sequence and amplitude of signal of the k-th user; $n(t)$ is Gaussian noise and σ is its variant. The likelihood function $J(b)$ can be represented as:

$$J(b)=2b^TAy-b^TARAb=2b^TAy-b^THb \quad (H=A\times R\times A) \quad (15)$$

$$\rho_{j,k}=\langle S_j,S_k\rangle=\int_0^T S_j(t)S_k(t)dt \quad (16)$$

where A=diag$[A_1,.....,A_K]^T$ is the amplitude matrix of the transmit signal; b=$[b_1,.....,b_K]^T$ is users' transmit sequencet; $R=[\rho_{j,k}]$ is users' correlation matrix. Obviously, OMUD is a combinational optimization problem. We use QPSO to solve it. In the following tests, we compare it with other algorithms.



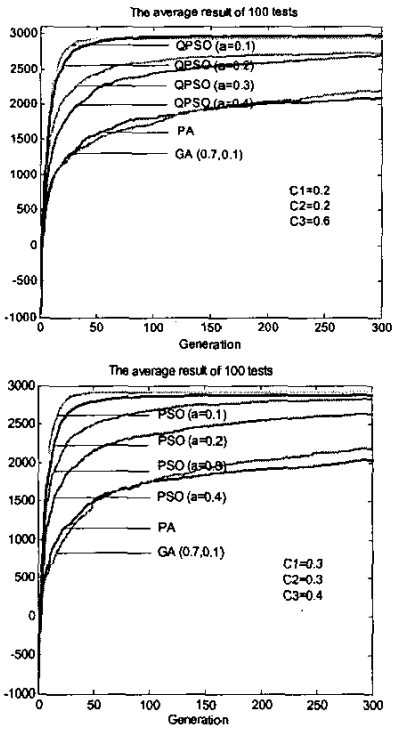


Figure 10-12 The convergence curves

V. CONCLUSIONS

A novel particle swarm optimization algorithm based on the quantum individual-QPSO is proposed to solve the discrete optimization problems. It is simple and reliable, and it can converge rapidly. It breaks a new path for building a fast and easy discrete PSO by its smart quantum representation. Simulation experiments also prove its effciency and good performance.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous receives for their helpful suggestions, and also thank for Min Wang, who helped in editing the original manuscripts, and who has given many helpful suggestions on writing..

REFERENCES

[1] Kennedy, J. The particle swarm: social adaptation of knowledge. *IEEE International Conference on Evolutionary Computation* pp. 303-308. IEEE service center, Piscataway, NJ, Indianapolis, IN, 1997.

[2] Kennedy, J. Minds and cultures: particle swarm implications. *Socially Intelligent Agents: Papers from the 1997 AAAI Fall Symposium* pp. 67-72. AAAI Press, Menlo Park, CA, 1997.USA, 2001, pp. 289-294.

[3] Kennedy, J. and Eberhart, R. C. Particle swarm optimization. *Proc. IEEE int'l conf. on neural networks* Vol. IV, pp. 1942-1948. IEEE service center, Piscataway, NJ, 1995.

[4] Eberhart, R. C. and Kennedy, J. A new optimizer using particle swarm theory. *Proceedings of the sixth international symposium on micro machine and human science* pp. 39-43. IEEE service center, Piscataway, NJ, Nagoya, Japan, 1995.

[5] Clerc, M. and Kennedy, J. The particle swarm-explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, Vol. 6, No. 1, pp. 58-73. 2002.

[6] Eberhart, R. C. and Shi, Y. Comparison between genetic algorithms and particle swarm optimization. *Evolutionary programming vii: proc. 7th ann. conf. on evolutionary conf.*, Springer-Verlag, Berlin, San Diego, CA., 1998.

[7] Abido,.M. A. Particle swarm optimization for multimachine power system stabilizer design. *Power Engineering Society Summer Meeting* Vol. 3, pp. 1346-2001. 2001.

[8] Eberhart, R. C. and Shi, Y. Evolving artificial neural networks. *Proc. 1998 Int'l Conf. on neural networks and brain* pp. PL5-PL13. Beijing, P. R. China, 1998.

[9] Eberhart, R. C. and Hu, X. Human tremor analysis using particle swarm optimization. *Proc. Congress on evolutionary computeration 1999* pp. 1927-1930. IEEE service center, Piscataway, NJ., Washington D.C., 1999.

[10] He, Z., Wei, C., Yang, L., Gao, X., Yao, S., Eberhart, R. C., and Shi, Y. Extracting rules from fuzzy neural network by particle swarm optimization. *IEEE International Conference on Evolutionary Computation* Anchorage, Alaska, USA, 1998.

[11] Hu, X. and Eberhart, R. C. Tracking dynamic systems with PSO: where's the cheese? *Proceedings of the workshop on particle swarm optimization* Purdue school of engineering and technology, Indianapolis, IN, 2001.

[12] Eberhart, R. C. and Shi, Y. Comparing inertia weigthts and constriction factors in particle swarm optimization. *Proc. CEC 2000* pp. 84-88. San Diego, CA, 2000.

[13] Hu, X. and Eberhart, R. C. Adaptive particle swarm optimization: detection and response to dynamic systems. *IEEE Congress on Evolutionary Computation, 2002* Honolulu, Hawaii USA, 2002.

[14] Hu, X. Multiobjective optimization using dynamic neighborhood particle swarm optimization.

[15] Shi, Y. and Eberhart, R. C. Parameter selection in particle swarm optimization. *Evolutionary Programming VII: Proc. EP 98* pp. 591-600. Springer-Verlag, New York, 1998.

[16] Kennedy, J., and Eberhart, R. C. (1997). A discrete binary version of the particle swarm algorithm. In Proceedings of the 1997 Conference on Systems, Man, and Cybernetics, 4104-4109. IEEE Service Center, Piscataway, NJ.