Multi-Processor Tasks with Resource and Timing Constraints Using Particle Swarm Optimization

Tzu-Chiang Chiang †, Po-Yin Chang††, and Yueh-Min Huang†

†Department of Engineering Science, National Cheng Kung University, Tainan, Taiwan, R.O.C
†† Department of Information Management, Hsing-Kuo University of Management, Tainan, Taiwan, R.O.C

Summary
The job-shop scheduling problems have been categorized as NP-complete problems. In our previous work, we use Hopfield Neural Network (HNN) to solve the energy function of the scheduling multi-processor tasks problem. Particle swarm optimization (PSO) is an evolutionary computation technique mimicking the behavior of flying birds and their means of information exchange. However, a pure PSO algorithm approach tends to solve continues linear problems. Therefore, the pure PSO algorithm need to be specially designed or some other methods may be combined to solve the energy function. This paper proposes using the particle swarm optimization to solve the constrained scheduling problem in display system operation. The constrained scheduling problem not only satisfies the resource constraint and the timing constraint. In our work, there are barriers must be overcome in applying energy function to PSO. In particle encoding, we attempt using a one-dimension 0-1 array mapping a three-dimension matrix of a candidate solution for each particle, and then using sigmoid function to produce probability threshold from velocity of each particle for velocity updating. The result showed that the proposed approach is capable of obtaining higher quality solution efficiently in constrained scheduling problems.

Key words:
Job-shop scheduling, Particle swarm optimization, Energy function

1. Introduction

The job-shop scheduling problems have been classified to NP-complete problems. The augmentation of the number of jobs to be processed, the number of operations for each job and the number of flexible machines performing the processes, causes the exponential increase of the time required to obtain an optimal solution. Because of the exponential growth, the exhaustive search for global optimal schedules is very difficult or even impossible. To generate good-quality schedules instead of global optimal schedules, adaptive search approaches have been implemented. In our previous work, 1999 and 2001, to schedule multiprocessor job with resource and timing constraints, an energy function for the HNN was proposed by Huang and Chen [1][2]. Then, in 2001, we integrated fuzzy c-means clustering strategic into a HNN to solve scheduling problems [3]. Recently, stochastic search techniques such as particle swarm optimization have shown the feasibility to solve the job-shop scheduling problems. In 2004, Weijun Xia and Zhiming Wu proposed a new algorithm based on the principle of PSO approach to solve job-shop scheduling [8]. In 2005, Hong Zhang et al. applied PSO to solve resource-constrained project scheduling problem [9]. In this paper, we attempt using discrete binary PSO-based approach with a feasible energy function to solve multi-processor task scheduling problem with timing and resource constraints. The original version of PSO approach which is clever at solves continues linear problems; the pure PSO algorithm need to be specially designed or some other schemes may be combined to solve the energy function. This remainder of the paper is organized as follows: In Section 2, the definition of the energy function of the scheduling problem is presented. In Section 3, the PSO and discrete binary PSO algorithms are reviewed, and then proposed approach applied to the energy function is illustrated. After this, the simulation examples and experimental results are presented in Section 4. Finally, conclusions of this paper are discussed in Section 5.

2. Energy Function of Problem

Job-shop scheduling problems are different from case to case. The scheduling problem domain in this paper has the following definition. Suppose N jobs are considered, each of which can be segmented, and M machines that are capable of performing the operations of all jobs, are also considered. The execution time required by each job is predetermined and can be estimated by calculating the machine cycles. It is supposed that different machines do not possess different segments of a job, and assumed that the job migration between machines is prohibited. Moreover, the deadline constraint for each job is imposed on the proposed system, and a resource to be shared by two jobs simultaneously is not allowed. According to the above assumptions, we attempt to generate legal schedules.

Manuscript revised January 2006.

To solve this problem, the energy function of the problem regarding all constraints is derived [1][2]. The energy function is modified and reduced in this work. A state variable V_{ijk} is defined as representing whether the job i is executed on machine j at a certain time k or not. Moreover, the state V_{ijk} = 1 denotes that the job i is run on machine j at the time k; otherwise, V_{ijk} = 0. Because a machine j can operate only one job i at any certain time k, the energy term can be formulated as the following:

$$\sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{k=1}^{T}\sum_{\substack{i1=1\\ i1\neq i}}^{N} V_{ijk}V_{i1jk} \tag{1}$$

V_{ijk} are defined as above; i (i = 1,…,N) represents the total number of jobs to be scheduled; j (1 to M) represents the total number of machines to be assigned; k represents a specific time from 1 to T, the latest deadline of the job. The same notations are used hereinafter. The minimum value of this term is zero, which occurs when either V_{ijk} or V_{i1jk} equals zero. As mentioned earlier, if a job is assigned on a dedicated machine, then all of its segments must be executed on the same machine. According this constraint, the energy term is defined as follows:

$$\sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{k=1}^{T}\sum_{\substack{j1=1\\ j1\neq j}}^{M}\sum_{k1=1}^{T} V_{ijk}V_{i1jk1} \tag{2}$$

Since a job i can be processed on either machine j or machine j_1, at any time, the minimum value of this term is zero, which occurs when either V_{ijk} or V_{ijk1} equals as for the resource constraint, two jobs are not allowed to utilize the same resource instance simultaneously. Besides, the resource is non-preemptive so that the energy term can he defined as follows zero.

As for the resource constraint, two jobs are not allowed to utilize the same resource instance simultaneously. Besides, the resource is non-preemptive so that the energy term can he defined as following:

$$\sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{k=1}^{T}\sum_{\substack{i1=1\\ i1\neq i}}^{N}\sum_{p=1}^{P}\sum_{f=1}^{F}\sum_{k1=1}^{T} V_{ijkpf}R_{is}V_{i1jkpfk1}R_{is} \tag{3}$$

where F denotes the quantity of available resource instances, R_{is} and R_{i1s} are the elements of the resource requested matrix for job i and $i1$, respectively. The value R_{is} = 1 means that job i requires resource s while R_{i1s} = 1 implies that job $i1$ requests resource s. When two distinct jobs are scheduled to be processed on different machines j and j_1 at the same time k (say V_{ijk} =1 and V_{i1jk} =l), machines j and j_1 cannot share the same resource at the time k. Hence, either R_{is} or R_{i1s} is zero. This observation implies that the energy term becomes zero if the resource constraint is satisfied. Correspondingly, the total energy function with all constraints can be induced as Eq.(4)

$$E=\frac{C_1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{k=1}^{T}\sum_{\substack{i1=1\\ i1\neq i}}^{N} V_{ijk}V_{i1jk}+\frac{C_2}{2}\sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{k=1}^{T}\sum_{\substack{j1=1\\ j1\neq j}}^{M}\sum_{k1=1}^{T} V_{ijk}V_{i1jk1}$$
$$+\frac{C_3}{2}\sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{k=1}^{T}\sum_{\substack{i1=1\\ i1\neq i}}^{N}\sum_{\substack{j1=1\\ j1\neq j}}^{M}\sum_{s=1}^{F} V_{ijk}R_{is}V_{i1j1k}R_{i1s} \tag{4}$$

where c_1, c_2 and c_3 refer to weighting factors and are supposed to be positive constants in our work. This study concentrates mainly on scheduling problems with constraint satisfaction. In the following section, PSO algorithm is introduced to solve the constraint satisfaction of the scheduling problems.

3. Experimental Consideration

Particle Swarm Optimization (PSO), one of the modern heuristic algorithms, is a population-based stochastic optimization technique developed by Eberhart and Kennedy in 1995[6], and was developed through simulation of a simplified social system. It has been found to be potent when it solves continuous nonlinear optimization problems. In our previous work, the energy function is set in discrete space. In this paper, we introduce a method of converting energy function to PSO.

3.1 Standard PSO Algorithm

PSO is an optimization tool providing a population-based search procedure in which individuals, called particles, change their positions, or states, with time. Particles in a PSO system fly around in a multidimensional search space. During the flying process, each particle modifies its position according to earned experience and the experience of nearby particles, and makes use of the best position met by it and other neighboring particles. The PSO method generates high-quality solution within shorter calculation

time and the more stable convergence feature than other stochastic methods [4]. Let x_i xi and v_i represent the coordinates and the corresponding flight speed of the particle i in a search space, respectively. The particle-updating mechanism for particle flying can be formulated as

$$v_i = w \cdot v_i + c_1 * rand_1 * (x_{igbest} - x_i) + c_2 * rand_2 * (X_{ipbest} - x_i) \quad (5)$$

$$x_i = x_i + v_i \quad (6)$$

w : inertia weight factor
c1, c2: acceleration constant
$rand_1, rand_2$: uniform random value in the range [0,1]
x_{igbest} : the best particle among all individuals in the population
X_{ipbest} : the best history position of particle x_i .

The parameter determined the resolution, or fitness, with which regions are to be searched between the max present position and the target position. When value is too high, max particles might fly past good solutions. Contrarily, particles may not explore sufficiently beyond local solutions when is too small. Experiences with PSO indicated that was often set at 10-20% of the dynamic range of the variable on each dimension. The constants c1 and c2 can describe the weighting of the stochastic acceleration terms that pull each particle xi toward x and x positions. Low values allow i_{pbest} and i_{gbest} particles to roam far from the target regions before being tugged back. On the other, high values result in abrupt movement toward, or past, target regions. Hence, the acceleration constants c1 and c2 were often set to be 2.0 according to past experiences.

$$w = w_{\max} - \frac{w_{\max} - w_{\min}}{iter_{\max}} \times iter \quad (7)$$

where $w_{\max}$ and $w_{\min}$ are both random numbers called initial weight and final weight respectively. In addition, $iter_{\max}$ is the maximum number of iterations, and $iter$ is the current iteration. The standard PSO algorithm was described as following:

PSO algorithm
Begin
Initialize
Particles to multidimensional scope with randomly
While the maximum of iterations is not reached
For each particle
Using Fitness function calculate fitness value
If the fitness value is better than the best fitness value according to past experiences
Set current value as the new pbest
End For
Choose the particle with the best fitness value of all the particles as the gbest
For each particle
Calculate particle velocity, V, according Eq. (5)
Update particle position, present, according Eq. (6)
End for
End While
End Begin

The output state in the energy function is 0 or 1 only, thus it is not suitable to apply the original version of PSO that is clever at operating the real values. A clever method for establishing a discrete binary version of the PSO (BPSO) was presented by Kennedy and Eberhart [5]. The dimensional value of BPSO must stay in values of 0 or 1 only. The velocity V_i will determine a probability threshold. If V_i is higher, the individual will be more likely to choose 1; lower values will favor the 0 choice; such a threshold needs to stay in the range [0.0, 1.0]. One familiar sigmoid function in neural network achieves this goal; the function is defined as the following:

$$s(V_i) = \frac{1}{1 + e^{-Vi}} \quad (8)$$

The function scales the Velocities V_i between 0 and 1 and has a value that makes it agreeable to be used as a probability threshold. Finally we update the dimension d of the particle X_i shown in the following:

$$X_i \begin{cases} 1, & \text{If rand}() < \text{s}(V_i) \\ 0, & \text{otherwise } X_i = 0. \end{cases} \quad (9)$$

Particles velocities on each dimension are limited to a maximum velocity $V_{\max}$. If the sum of accelerations may cause the velocity on that dimension to surpass $V_{\max}$, which is a parameter specified by the user, then the velocity on that dimension is limited to $V_{\max}$.

3.2 Particle Encoding

To establish a “solution mapping” and “generate solution” mechanisms are critical issues in applying PSO for solving a specified domain problem. If these two issues are dissolved successfully, it is possible to find a good-quality solution. In section 2, the energy functions of our previous

work are formulated with timing and resource constraints. In this work, the purpose of energy function is to evaluate the energy value of candidate solution of each particle, and then chose best candidate solution into next iteration according to the energy value. However, energy function clever on dissolving discrete matrix which is a 0-1 form; thus how to encode the particle for mapping a multi-dimension discrete matrix is an important issue in applying energy function to PSO. In our work, we use an one-dimension array to denote a three-dimension discrete matrix of candidate solutions. In a swarm, a particle P_i is represented by a s dimension and can be defined as $P_i = [p_1, p_2, ..., p_s]$. The s denotes the size of the three-dimension matrix of candidate solution, Eg. job=4, resource = 2, time=2, the s dimensional vector of each particle is calculated by Job*Resource*Time. In this case, the s = 16, the three-dimension matrix of a candidate solution can be illustrated as Fig 1:

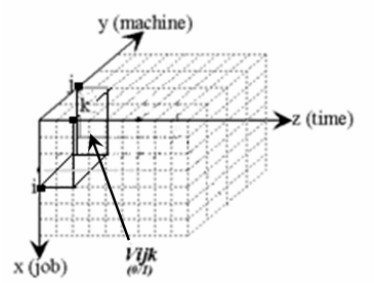

Fig. 1 The three-dimension matrix of a candidate solution

In Fig. 1, the "x" axis denotes the "job" variable and I represents a job with a range from 1 to N, the total number of jobs to be scheduled; where the "y" axis denotes the machines variable and any point j on the axis represents a dedicated machine identified from 1 to M, the total number of machines to be operated; where the z axis denotes the time variable and k represents a specific time which should be less than or equal to T, the deadline of the job. Thus, a state variable V_{ijk} is defined as representing whether the job i is executed on machine j at a certain time k. However, it is difficult to operate and understand the 0-1 permutations in three-dimensional matrix, thus we transform all permutations of a candidate solution into a two-dimensional matrix, mapping one-dimension array as Fig. 2:

Job	Resource	Time	P_i
1	1	1	0
1	1	2	0
1	2	1	1
1	2	2	1
2	1	1	1
2	1	2	0
2	2	1	1
2	2	2	0
3	1	1	1
3	1	2	1
3	2	1	1
3	2	2	0
4	1	1	1
4	1	2	0
4	2	1	0
4	2	2	0

s = 16

Fig. 2 three-dimension matrix is mapping to one-dimension array

In Fig.2, it illustrated representation of the solution; the three-dimension matrix is transformed into two-dimension matrix and then maps a candidate solution via one-dimension array. The job, resource and time fields of two-dimension matrix denote the axes of x, y and z in Fig.1, and all permutations are listed in two-dimension matrix. The field P_i denotes the code of a particle and is mapping the state value of the three-dimension matrix. Therefore, a candidate solution is encoded as a particle successfully. In order to calculate the energy value for each particle, these two simple equations are derived for operating the state value of Job_i in one-dimension array. The equations are illustrated as the following:

$$Lower\ Bound = (s / Total\ Job * Job_i) - (s / Total\ Job-1) \quad (10)$$

$$Upper\ Bound = (s / Total\ Job * Job_i) \quad (11)$$

For example, the candidate solution of a particle is computed by Eq. (1), initial iteration Job_i =1, the lower and upper bound of Job_i is computed by Eq.(10) and (11), the example is illustrated as following Fig3:

	Job	Resource	Time	P_i	
Lower Bound	1	1	1	0	1
Job1	1	1	2	0	
	1	2	1	1	
Upper Bound	1	2	2	1	4
	2	1	1	1	5
Job2	2	1	2	0	
	2	2	1	1	
	2	2	2	0	8
	3	1	1	1	9
Job3	3	1	2	1	
	3	2	1	1	
	3	2	2	0	12
	4	1	1	1	13
Job4	4	1	2	0	
	4	2	1	0	
	4	2	2	0	16

Fig. 3 the upper and lower bound for each job

In the Fig. 3, the lower and upper bound are calculated by Eq. (10), (11). Then energy value of a candidate solution is calculated by Eq. (1) in this simplify numerical demonstration.

While Job_i =1 then

Lower Bound = (s /Total Job * Job_i) – (s /Total Job-1)

= (16/4 * 1)-(16/4-1) = 1

Upper Bound = (s /Total Job * Job_i) = (16/4*1) = 4

For

Job_i =2, Lower Bound = 5, Upper Bound = 8

Job_i =3, Lower Bound = 9, Upper Bound = 12

Job_i =4, Lower Bound = 13, Upper Bound = 16

As above, the lower and upper bound of job1, 2, 3, 4 are calculated (10), (11). In order to simplify the calculate process, thus only list the permutations of Job_i =1 and the energy value is calculated according to Eq.(1) shown as following:

V_{111}*V_{211}	0	*	1	0
V_{111}*V_{311}	0	*	1	0
V_{111}*V_{411}	0	*	1	0
V_{112}*V_{212}	0	*	0	0
V_{112}*V_{311}	0	*	1	0
V_{112}*V_{411}	0	*	0	0
V_{113}*V_{213}	1	*	1	0
V_{113}*V_{313}	1	*	1	0
V_{113}*V_{413}	1	*	0	1
V_{114}*V_{214}	1	*	0	1
V_{114}*V_{314}	1	*	0	1
V_{114}*V_{414}	1	*	2	0

Fig. 4 three-dimension matrix is mapping to one-dimension array

Finally, the energy-based PSO algorithm is illustrated following:

Step1. Initialization
The initial candidate solutions of particles in the swarm are generated randomly according to timing and resource constraints, and calculate energy value for each particle by eq.(4), then set the pBest and gBest of swarm.

Step2. Generating new velocity
New velocities for all the dimensions in each particle are generated by eq. (8).

Step3. Updating pBest & gBest Particle
The new position for each particle is generated by eq. (9); pBest & gBest are calculated and updated.

Step4. Stopping criteria
Energy values are calculated for each particle, if the stopping criteria are met, otherwise jumps to Step2.

4. Experiment Result

Three sets of resource and timing constraints are applied for the simulations. The constants of the energy function, c_1, c_2, and c_3, are all given to 1 in this work. Each population of individual of the PSO algorithm is initialized randomly. The resource requested matrix and the timing constraints matrices for three cases are shown in Table 1 and Table 2.

Table 1: Resource Requested Matrix

Case1

	R1	R2	R3
Job1	1	0	0
Job2	0	0	1
Job3	0	1	0
Job4	1	0	0

Case2

	R1	R2	R3	R4
Job1	1	0	0	0
Job2	0	0	1	0
Job3	0	1	0	0
Job4	1	0	0	1
Job5	1	0	0	1

Case3

	R1	R2	R3	R4
Job1	1	0	0	0
Job2	0	0	1	0
Job3	0	1	0	0
Job4	1	0	0	1
Job5	1	0	0	1
Job6	0	1	0	0
Job7	0	0	0	1
Job8	0	0	1	0
Job9	0	0	0	0
Job10	0	0	1	0

Table 2: Timing Constraints Matrix

Case1

	Time Required	Time Limit
Job1	4	6
Job2	3	4
Job3	3	6
Job4	2	3

Case2

	Time Required	Time Limit
Job1	2	3
Job2	3	8

Job3	**3**	**4**
Job4	**4**	**8**
Job5	**2**	**5**

Case3

	Time Required	Time Limit
Job1	**5**	**10**
Job2	**3**	**5**
Job3	**3**	**9**
Job4	**2**	**5**
Job5	**3**	**9**
Job6	**2**	**6**
Job7	**3**	**10**
Job8	**2**	**5**
Job9	**3**	**9**
Job10	**4**	**10**

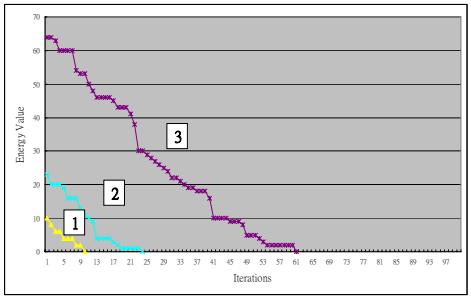

Fig. 5 The Energy curve of the iteration in the population. (a)Case1 (b) Case 2 (c) Case 3.

Fig. 5 displays the energy curve of the best member in the population for 3 cases during iterations. The simulated scheduling results are graphically represented by the Gantt charts and are shown in Fig 6. To estimate the quality of the scheduling results, we can calculate the makespan, the sum of the maximum completion time of each job. The energy function used by the latter is simplified in this work, the computation time can he reduced.

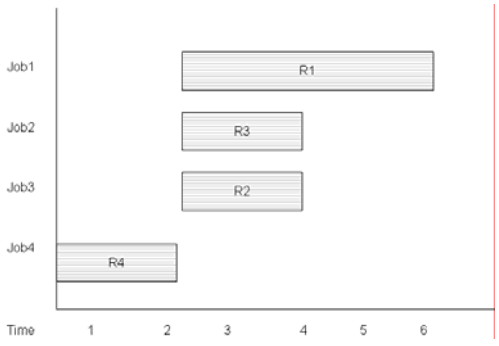

Fig. 6 the good-quality solution in case (1)

5. Conclusion

The job-shop scheduling problems have been categorized as NP-complete problems. It is a complex optimization problems, the exponential growth of time required to obtain an optimal solution. In our previous work, we used HNN to solve the energy function of the scheduling multi-processor task problem. In this work, we attempt using PSO algorithm to solve the energy function of HNN. However, some barriers must be overcome in applying energy function to PSO. Fist, the multi-dimension 0-1 matrix of a candidate solution is mapping in an one-dimension 0-1 array; the candidate solution is encoded in each particle successfully. Next, the discrete state value 0 or 1 is throughout the state scope of energy function, thus original version of PSO that applied on real value is not suitable to solve the energy function. The sigmoid function is utilized to produce probability threshold from velocity of each particle, then generating new candidate solution. In the three simulated cases, the proposed scheme converges rapidly. Results show that the energy function is applying PSO successfully. The energy-based PSO is a competent method to solve the scheduling multi-processor task problem. In future work, we attempt to derive our energy function for describing the complex problem in various industries, and make efforts to improve our PSO approach

References

[1] Yueh-Min Huang and Ruey-Maw Chen "Scheduling Multiprocessor Job with Resource and Timing Constraints Using Hopfield Neural Networks", *IEEE Transactions on Sysrems*, pp. 490-502, 1999.

[2] Ruey-Maw Chen and Yueh-Min Huang, "Competitive Neural Network UI Solve Scheduling Problems", *Neurocompuring*, pp. 177-l96,2001.

[3] Ruey-Maw Chen and Yueh-Min Huang, "Multiprocessor Task Assignment with Fuzzy Hopfield Neural Network Clustering Technique", *Neural Computing and Applications*, pp. 12-21, 2001.

[4] Y. Shi and R. C. Eberhart, "Empirical Study of Particle Swarm Optimization," *Proceedings of the 1999 Congress on Evolutionary Computation*, pp. 1945-1950, 1999.

[5] J. Kennedy, R. Eberhurt, "A discrete binary version of the particle swarm algorithm," *Proc. 1997 Conf. Systems*, 1997.

[6] Eberhart, R. C., Kennedy, J., "A new optimizer using particle swarm theory," *Proc. Sixth Intl. Symposium on Micro Machine and Human Science*, pp. 39-43, 1995

[7] T. O. Ting, M. V. C. Rao., C.K. Loo, "A Novel Approach for Unit Commitment Problem via an Effective Hybrid Particle Swarm Optimization", *IEEE Transactions on Power Systems*, pp. 411-418, 2006.

[8] Weijun Xia, Zhiming Wu, "An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems", *Computer h& Industrial Engineering*, pp. 409-425, 2005.

[9] Hong Zhang, Xiadong Li, Heng Li and Fulai Huang, "Particle swarm optimization-based schemes for resource-constrained project scheduling", *Automation in Construction*, pp. 393-404, 2004.

Tzu-Chiang Chiang received his B.S. degree in Engineering Science from the National Cheng Kung University, Taiwan, R.O.C., in 1987. He received his M.S. degree in computer science from University of Southern California, Los Angeles, USA in 1992. He worked in Computer Center at National Chung Cheng University, Taiwan from 1993 to 2002. Since 2002, he has been a lecturer in Department of Information Management of Hisng-Kuo University of Management, Taiwan, R.O.C. He is currently a Ph.D. student in Department of Engineering Science of National Cheng Kung University, Taiwan, R.O.C. His current research interests include security and routing protocol issues in wireless ad hoc networks and multi-objective optimization with genetic algorithms, multimedia communications.

Po-Yin Chang received the B.S. degree in Information Management from Husing-Kao University of Management, Taiwan, R.O.C., in 2002. He received his MBA degree in Graduate Institute of Commerce Automation and Management from National Taipei University of Technology in 2005. He is a lecturer in Department of Applied Internet Science of Hisng-Kuo University of Management, Taiwan, R.O.C. His current research interests include scheduling, optimization in wireless ad hoc networks, web-services and above issues using particle swarm optimization, genetic algorithms

Yueh-Min Huang was born in Taiwan, R.O.C., in 1960. He received the B.S. degree in Engineering Science from the National Cheng Kung University, Taiwan, R.O.C., in 1982, and both the M.S. and Ph.D. degrees in electrical engineering from the University of Arizona, Tucson, AZ, in 1988 and 1991, respectively. He has been with National Cheng Kung University since 1991, and is currently a professor of the Department of Engineering Science. His research interests include wireless ad hoc networks, distributed multimedia systems, data mining, and real-time systems. Dr. Huang is a member of IEEE Computer Society, the American Association for Artificial Intelligence, and the Chinese Fuzzy Systems Association.