Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics
San Antonio, TX, USA - October 2009

Optimal RFID Networks Scheduling using Genetic Algorithm and Swarm Intelligence

Chiu, Chui-Yu
Industrial Engineering and Management Department
National Taipei University of Technology
Taipei, Taiwan, R.O.C.
cychiu@ntut.edu.tw

Ke, Cheng-Hsin
Industrial Engineering and Management Department
National Taipei University of Technology
Taipei, Taiwan, R.O.C.
t6378030@ntut.edu.tw

Chen, K. Y.
Industrial Engineering and Management Department
National Taipei University of Technology
Taipei, Taiwan, R.O.C.
kychen@ntut.edu.tw

Abstract—**RFID is an emerging technique for identifying items and all kinds of real world applications. Multi RFID readers are implemented to the product line in many industries and they consist of varied reader resources. But there are some defects with the disposition of the RFID-based application. The phenomenon of incorrect negative reads occurs in a multi-tag and multi-reader environment where a tag that is present is not detected. Collisions occurring between readers cause the faulty or missing reads. The stopgap is to solve the frequency allocation problem for networks of RFID readers. Furthermore, finding the optimal structure of readers and scheduling the readers to reduce the total system transaction time or response time are both challenging problems. In the presence of interdependencies, the optimal scheduling problem to minimize the overall transaction or response time is modeled as a graph partitioning problem (GPP). GPP is a well known NP-complete problem. The more readers exist in the product line, the higher complexity of the problem. Designing a schedule having the maximum parallelism reduces the total transaction time but may not minimize it.**

In this research, we integrate genetic algorithms with binary particle swarm optimization (GA-BPSO) to solve the Multi RFID networks scheduling problem. Simulation results on a real–world problem show that the GA-BPSO algorithm provides robust solution quality and is suitable for scheduling large scale RFID reader networks.

***Keywords*—networks of RFID readers, scheduling, GA-BPSO**

I. LITERATURE REVIEW

A. Networks of RFID Readers

RFID (Radio Frequency Identification) technology plays an important role in asset tracking and management in the near future. We can deploy networks of RFID readers to track the flow of assets through various environments. Deolalikar [42] derived optimal scheduling schemes for networks of RFID readers in four cases of practical importance. The reader model which is about an RFID reader reading a fixed set of tags is modeled as operating in two time regions. A novel optimal scheduling scheme for networks of RFID readers using a symbiotic multi-species particle swarm optimizer is presented in [18]. In this research, readers with overlapping fields are fired at different times so that they do not collide. The frequency allocation problem for networks of RFID readers is to allocate frequencies to different readers. In other words, when two readers lie in each other's interference region, they are given different frequencies [11].

The network of RFID readers is similar to the sensor network. A sensor network is consisted of a large number of sensor nodes that are densely deployed either inside the phenomenon or quite close to it. The position of sensor nodes need not be engineered or predetermined. This allows random deployment in unreachable terrains or disaster relief operations. On the other hand, sensor networks protocols and algorithms must possess self-organizing capabilities. Another unique characteristic of sensor networks is the cooperative effort of sensor nodes. Sensor nodes are fitted with an onboard processor. They use their processing abilities to carry out simple computations locally and only transmit the required and partially processed data instead of sending the raw data to the nodes responsible for the fusion.

The sensor network problem for steady state processes has been solved in the past by different researchers using different methods. Kretsovalis and Mah [2] design a sensor network for maximum estimation accuracy, and a more effective method is that using graph theory to exploit the configuration of the process and sensor network in the design strategy [34][44][46]. However, these algorithms are not sufficiently common and they can't be used to optimize sensor network problems with respect to more than one criterion, simultaneously.

The following are some researches using heuristic algorithms to solve sensor network problems. Wu [46] proposed that maximizing the coverage based on a probabilistic sensor model in mobile sensor networks by using

978-1-4244-2794-9/09/$25.00 ©2009 IEEE

particle swarm optimization (PSO) to reduce the cost. The sensor network optimization problem discussing the relationship between the population size and the stopping criteria is evaluated by using GA in [4][5]. Binary particle swarm optimization (BPSO) is used for optimal scheduling in sensor networks [28]. Sensors are characterized by their transaction times and interdependencies.

B. Graph Partitioning Problems

Graph partitioning problems (GPP) are important that has wide applications the field of computer science, including task scheduling [25][38][39][45] and VLSI (Very Large Scale Integrated circuit) design [27]. The objective of GPP is that partitioning a graph G into K subgraphs such that the number of edges connecting nodes is minimized in different subgraphs. On the contrary, the number of edges connecting nodes is maximized in the same subgraph.

The GPP is NP-complete problems. However, many algorithms have been developed that find good partitions reasonably. Spectral methods [3][6] have been shown to be very effective for partitioning unstructured problems in a variety of applications, but there is very high computational complexity in them. Geometric partition methods [7][15][32] are very fast but they often obtain worse partitions than those of more expensive methods like spectral. Furthermore, geometric methods are applicable only if coordinate information of the graph is available.

Recently, a number of researches have explored a class of algorithms that have common computational complexity, and obtain excellent (even better than spectral) graph partitions [6][14][40]. The graph G is first coarsened down to a few hundred vertices and a half of this much smaller graph is calculated and then this partition is projected back towards the original graph (finer graph) by refining the partition periodically. Since the finer graph has more degrees of freedom, such refinements usually shorten the edge-cut. These are called multilevel graph partitioning schemes. A multilevel graph partitioning scheme produces high quality partitions in small amount of time [13][14].

It is clear that multilevel graph partitioning algorithms can find high quality partitions for a variety of unstructured graphs from the experiments presented in [6][14][40].

C. Particle Swarm Optimization

Particle swarm optimization (PSO) developed by Dr. Eberhart and Dr. Kennedy [22][23] in 1995 and inspired by social behavior of birds flocking is an evolutionary computation optimization technique.

In standard particle swarm optimization (SPSO), each possible solution is a "bird" in the search scope, called "particle". According to the research results for birds flocking, birds are finding food by flocking (not by each individual). Such as genetic algorithms (GA), SPSO also have a fitness function that takes the particle's position and assigns it to a fitness value. The objective is to achieve that optimizing the fitness function. Each particle i which is represented as $X_i = (x_{i1}, x_{i2}, \ldots, x_{iD})$ has own coordinates and velocity to change the direction of moving in predefined search scope with D-dimensional. All particles move through the search scope by following the current optimum particles.

SPSO updates the current generation of particles (each particle is a candidate solution to the problem) using the information of the best solution obtained by each particle and the entire population for finding an optimal or near-optimal solution to the problem. SPSO is initialized with a group of random particles (possible solutions) called "population", and then searches for the optimal solution by updating generations until the stopping rule meet. Each particle is updated by following two "best" values in every iteration. The particle swarm optimizer keeps track of the overall best value and its location, obtaining thus far by any particle in the population, which is called *gbest* (P_{gd}). Each particle keeps track of the best solution and saves a memory of its previous best position, called *pbest* (P_{id}) [36].

The particle updates its velocity and coordinates after discovery the two best values by following Eq. (1) and (2).

$$V_{id}^{new} = V_{id}^{old} + c_1 \times rand_1() \times (p_{id} - X_{id}) + c_2 \times rand_2() \times (p_{gd} - X_{id}) \quad (1)$$

$$X_{id}^{new} = X_{id}^{old} + V_{id}^{new} \quad (2)$$

V_{id} is the particle velocity, X_{id} is the particle coordinates (possible solution), P_{id} and P_{gd} are respective *pbest* and *gbest*. *rand()* is a random number uniformly distributed in range [0, 1].c_1 and c_2 are learning factors. They are usually equals to 2.0.

The velocity of each particle is restricted to a maximum velocity V_{max}. The velocity is limited to V_{max} on the dimension. Hence, V_{max} is an important parameter which determines the breadth of search among the present coordinates and the target (best so far) coordinates in search region. If V_{max} is too tidy, particle might over-fly through good solutions. On the country, if V_{max} is too small, particle may not prospect well beyond locally good regions.

The maximum velocity V_{max} is regarded as a constraint to control the global exploration ability of a particle swarm [48]. A smaller V_{max} encourages local exploitation, and a larger V_{max} facilitates global exploration.

Shi [48] developed the modified PSO (WPSO) for improving the convergence rate. The main difference from the original PSO is that considering an inertia weight to velocity update, stated as in Eq. (3) and (4):

$$V_{id}^{new} = w \times V_{id}^{old} + c_1 \times rand_1() \times (p_{id} - X_{id}) + c_2 \times rand_2() \times (p_{gd} - X_{id}) \quad (3)$$

$$X_{id}^{new} = X_{id}^{old} + V_{id}^{new} \quad (4)$$

It can be seen that Eq (3) and (4) are identical to Eq (1) and (2) except the addition of the inertia weight w. The computational result demonstrated that WPSO outperforms SPSO. w is decreased linearly from 0.9 to 0.4 as originally developed. Suitable selection of the inertia weight provides balance between global and local searching in predefined scope, and causes less iteration on average to find an approximate optimal solution.

D. Binary Particle Swarm Optimization

Binary particle swarm optimization (BPSO) described in [22] is very similar to the velocity update equation described in Eq. (1). The difference is the position update equation. The position update equation and the sigmoid function of the velocity for the binary model is given by

$$\begin{cases} x_{id}(t) = 1, \text{if } \rho_{\text{id}} < s(v_{id}(t)) \\ x_{id}(t) = 0, \text{otherwise} \end{cases} \tag{5}$$

$$s(v_{id}) = \frac{1}{1+\exp(-v_{id})} \tag{6}$$

where ρ_{id} is a vector of random numbers, drawn from a uniform distribution between [0,1].

BPSO is impressionable by sigmoid function saturation, which occurs when the velocity value is either too large or too small. In such cases, the probability of a change in bit value approaches zero to limit the exploration. For a velocity of 0, the sigmoid function returns a probability of 0.5, representing that there is a fifty percent chance for the bit to flip. Figure 1 shows the rage of the sigmoid function.

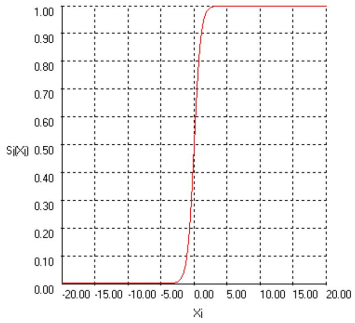

Figure 1. The rage of the sigmoid function.

There are some researches using BPSO. Lee [37] proposed a new model based on BPSO, which finds more precise theoretical values of options with estimates of the implied volatility than GA. Solving a Black-Scholes option pricing by original BPSO and the BPSO combined the concept of mutation in GA. A novel BPSO is proposed in [33]. This algorithm proposes a new definition for the velocity vector of BPSO. It has shown that this algorithm have a better interpretation than BPSO. Chuang [29] proposed a new BPSO combined with GA (IBPSO) to solve the classification problem. Sadri [24] proposed a GBPSO model. GBPSO also implements the concept of mutation in GA to BPSO. The above objective of BPSO combined with GA is to avoid BPSO trapped into the local optimal solution.

E. Genetic Algorithms

Genetic Algorithms (GA) were first introduced by Holland and explored by Goldberg [9]. The mathematical structure is presented in Holland's pioneering book, "Adaptation in Natural and Artificial Systems" [21] published in 1975. In the following subsections, we will discuss about GA.

Ever since 1975, lots of people have participated to the research of GA. Recently, GA have received significant attention regarding their potential as an optimization method for complex problems and have been successfully applied in the domain of industrial engineering [30]. The well-known applications include scheduling and sequencing, digital signal processing [26], transportation, facility layout, and many others.

The main advantage of GA over the traditional optimization method lies in the ability of jumping out of the local optimal solution. The general disadvantage of those traditional optimization algorithms is that the solution is easy being trapped into a local optimal solution if the initial parameter is not properly set. However, this situation less happened to use GA as an optimization method. Another important advantage is that we can avoid a great deal of mathematical operations when dealing with complex nonlinear optimization problem.

It has been proved that GA was useful in various search and optimization problems over the years. GA is based on the survival-of-the-fitness principle which tries to save more genetic information from generation to generation.

The main steps of the simple genetic algorithms are reported in the following:

1. Construct a fitness function from the objective function.
2. A population of n chromosomes is randomly generated. Each of them is represented by a string of digits, zeros and ones. The string is encoded for the design variables x_i, and with fitting accuracy that is determined by the string's length.
3. All the chromosomes of the population are estimated by means of the fitness function $f(x_i)$.
4. **Reproduction**: According to a rule that favors those with higher fitness, chromosomes of the old population are selected and put in the new one. (The better fitness, the bigger chance to be selected)
5. **Crossover**: Two randomly selected strings are mated among those selected in the previous step. A position along one string is randomly selected again and all binary digits following this position are switched with those of the second string (Single-point crossover). Then the two totally new strings move on to the new generation. This operation occurs with a defined probability (or crossover rate) P_c, which represents the number of chromosomes which taking part in the crossover process with respect to the total number of them.
6. **Mutation**: A bit of the strings of the new population is randomly selected by a defined probability P_m (mutation rate), and the value is complemented from 1 to 0 or vice versa. This process explores the new solution in the search space, protecting against the loss of useful genetic information.
7. Use new generated population for a further run of algorithm.
8. If the terminated condition is satisfied, return the best solution in current generation or else go back to step 3.

9. **Decode**：This process is that transforming the best solution which encoded into binary string to decimal value. Assume the parameter of the system θ is encoded into binary string of Z bits, the search range is $[\theta_{\min}, \theta_{\max}]$ and denote the decimal value of the binary representation as X. Then the parameter of the system is decoded linearly as follows:

$$\theta = \frac{\theta_{\max} - \theta_{\min}}{2^Z - 1} X + \theta_{\min} \quad (7)$$

II. METHODOLOGY

A. Framework

PSO has applied into many scopes by researchers and it was a new heuristic algorithm in recent years. It is used the concept of birds searching for food. There are many advantages in PSO, such as having faster search speed, saving a memory of its previous best position and tracking the best solution continuously. BPSO is proposed for solving the discrete problems. The concept of BPSO is similar to PSO. The difference is that they are used respectively for solving different type problems. GA was proposed for long years. It is used the concept of biology. GA is consisted of reproduction, crossover and mutation. In order to create the next generation, offsprings, the chromosomes using some measures of fitness are evaluated during each generation and using mutation to avoid getting worse solutions. However, both PSO and GA also have disadvantages respectively, such as trapped into the local optimal solution and dropped out the optimal solution. The GA-BPSO uses BPSO model mainly and adds the concept of mutation in GA. By combining the advantages of the two heuristic algorithms, we proposed GA-BPSO to solve the problem of the network of RFID readers.

B. Problem Description and Limitation

Multi RFID readers which consist of the RFID reader network are implemented to the product line in many industries. There are collisions occurring between readers and that cause faulty or missing reads. The RFID reader network problem still exists in nowadays and it is like the sensor network problem. This problem has two objectives: finding the optimal structure of readers and scheduling the readers to reduce the total system transaction time.

1) Problem Description

Given a collection of N RFID readers laid out in some types, we can construct the associated collision graph $G = (V,E)$ where each vertex $v \in V$ corresponds to a reader and each edge $e \in E$ points out that those two readers can be operated in parallel (there are no collisions between these two readers). For instance, a set of 6 RFID readers layout corresponding to the Eq. (8) is given in Figure 2 and the collision graph is given in Figure 3.

$$C=\{(1,3),(1,5),(1,6),(2,5),(2,6),(3,4),(3,5),(5,6)\} \quad (8)$$

Where (R_i, R_j……R_n) are the RFID readers that have conflicts.

The objective function in this research is to find the optimal structure of readers and schedule the readers to reduce the total system transaction time.

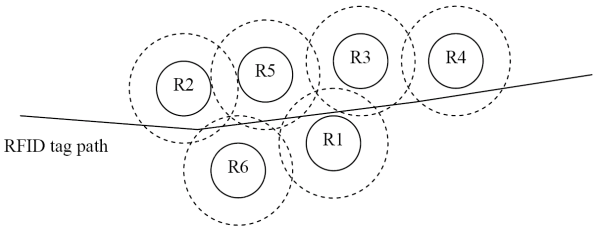

Figure 2. The set of 6 RFID readers sample layout.

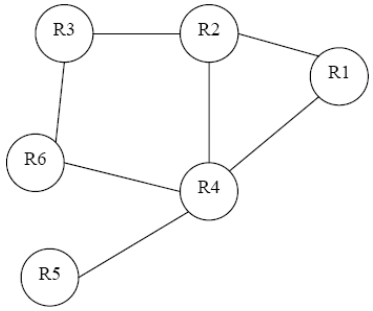

Figure 3. The Collision graph of a set of 6 RFID readers.

2) Problem Limitation

There are two constraints that limit the parallel operation of RFID readers in our model RFID applications. The first is the reader interference or collision. At this scope, this problem seems like the frequency allocation problem, but the allocation is done along the time axis. Interfering readers are allotted non-overlapping periods of time in order to avoid collision between them. Then it may appear that this problem also reduces to the graph partitioning problem. The second constraint is the overall transaction time. It makes clear that partitioning the graph can't entirely solve the time scheduling problem because it also requires minimizing the total transaction time of the RFID reader network. However, the frequency allocation problem is not.

C. The Proposed Heuristic Algorithm

We use the integration of genetic algorithms and binary particle swarm optimization (GA-BPSO) to solve the problem of scheduling RFID reader networks. This section describes the rule being followed, the particle representation used for solving the problem, the function being optimized.

1) Rule

In the traditional graph partitioning algorithm, for optimal scheduling in sensor network problem is the following 4 rules：

1. The partition with the maximum transaction time is given preference. The algorithm attempts to maximize the number of sensors running simultaneously in parallel within the partition with the transaction time. After removing this partition from the graph, it repeats the process iteratively until all sensors are in a partition.
2. Always maximize the number of readers in a partition and this reduces the total number of partitions after partitioning the whole graph.

3. A special weighting value is assigned to each partition to minimize the variation in sensor transaction time within the partition. The weight is defined as the average of the sensors transaction times in the partition. Besides minimizing the range of transaction values in the partition, this also retains sensors with lower transaction times only for further partitioning and reduces total transaction time of the system.

4. A higher preference is given to the sensor with the most conflicts. The reason is that it will cause minimum affects on the number of edges in the graph due to partition of such a sensor into a partition early. It should be noted that the higher the number of edges in the graph, the greater the possibility of finding the optimal partition is.

These four rules result in values that are attached to the partition as it is constructed and is evaluated for possible selection in the algorithm. Again these rules are specific to reduce the total transaction time of the system. These rules are incorporated as shown later.

2) Particle Representation

The particle representation we used in this research is binary and each particle has a dimension equal to the number of readers. Each particle has a binary representation and is a possible partition. For example, "011101" is a particle for 6 readers in the system and a bit "1" implies the presence of that particular reader in the partition which the particle is representing. On the country, a bit "0" implies the absence of that particular reader in the partition which the particle is representing. The partition contains readers 2, 3, 4, 6 in the above example.

3) Fitness Function

"Fitness" is a mathematical quantity evaluated uses the implied load imbalance and communication costs. If the i^{th} reader is adjacent to the i+1 reader for each i in the system, then "11100011" would more fit than "10101011" (which has 6 inter-partition edges), but less fit then "11100001" (which is a more balanced partition). A predefined cost function should be formulated for evaluating the performance of an individual. The cost function considers four parameters. f is the fitness function computed as the reciprocal of the summation of C_i as follows:

$$C_i = w_1 \times (N) + w_2 \times (T) + w_3 \times (W) + w_4 \times (I) - p \times (N) \quad (9)$$

$$f = \frac{1}{\sum C_i} \quad (10)$$

where N is the number of readers and T is the transaction time of the partition and can be calculated as:

$$T = \max(t_i) \quad (11)$$

where t_i is the transaction time of the i^{th} reader that constructing the partition. W is the weight attached to the partition and can be calculated as:

$$W = \frac{1}{N} \sum t_i \quad (12)$$

It is the average of the transaction times of the readers constructing the partition. The weight is necessary to avoid that there are two solutions each consisting the same dominating reader in terms of transaction time and having equal number of readers. The first two N and T will be equal for both the partitions in the situation. However, a better solution will be the partition having the readers with high transaction times when compared to other.

I is the summation of all the possible collision which the members of the partition have with the readers still remaining to be partitioned in the graph. Intuitively, this makes sense since removal of the partition leaves a lot of scope for further formation of partitions in remaining nodes and would not cause much loss of edges in the graph.

$$\begin{cases} p = 1000, \text{if any of the dependencies are violated} \\ p = 0, \text{otherwise.} \end{cases} \quad (13)$$

N is multiplied by p in the cost function and also represents the number of readers that violate dependency in the term. The swarm converges to a partition containing all the readers which is based on minimization of the fitness function to achieve optimal results.

w_1, w_2, w_3, w_4 are the weights given to each term of the fitness function separately and $w_2 \geq w_1 \geq w_4 \geq w_3$ representing the importance of each term.

From above discussion, it can be seen that the design of a cost function is very important for any optimization algorithm. For GA-BPSO, it gives the direction to the particles in the search space.

4) Scheduling the network of RFID Readers By GA-BPSO

Because the particle representation is binary, we use the BPSO concept to evolve the partitions. The dependencies and the transaction times are inputs to GA-BPSO. The GA-BPSO will find the optimal maximum partition. The readers constructing the partition are removed. The dependencies and the transaction times are updated, too. The new dependencies and the transaction times are put into GA-BPSO again. This is repeated iteratively till all the readers are grouped. Assuming that there is no specific sequence required, the groups can be scheduled one after another. Figure 4 shows the flowchart of GA-BPSO implemented for finding the optimal groups of the readers, which can be run in parallel.

The procedure of GA-BPSO is as follows and shown as Figure 5:

Step1. Set up parameters including the population size (the number of particles), Maximal velocity (V_{max}), inertial weight (w), and two learning factors (c_1 and c_2). The two random variables, $rand_1$ and $rand_2$ are in [0, 1].

Step2. Initialize each particle with initial position (X_{id}) randomly and the velocity (V_{id}) in the range of maximal speed (V_{max}).

Step3. Calculate every particle's fitness value using Eq. (10). Record the particle of better fitness value.

Step4. Update the local best position (P_{id}) and global best

position (P_{gd}). If P_{id} and P_{gd} change, go to the step 6, or go to the next step.

Step5. Due to the P_{id} and P_{gd} do not change, use mutation to stir the particle for jumping out the local solution. The following is the update equation:

$$X_{id}^{new} = \begin{cases} 1, \text{if } X_{id}^{old} = 0 \\ 0, \text{otherwise} \end{cases} \quad (14)$$

Step6. According to the updated P_{id} and P_{gd}, use Eq. (3), (5) and (6) to update the velocity and position for every particle.

Step7. Calculate every particle's fitness value by implementing the X_{id}^{new}, then replace the particle with worst fitness value in step 3 by using elitist policy.

Step8. Use the following three Equations to update the velocity and position for every particle by implementing the updated P_{id} and P_{gd}.

Step9. Stop if the specified number of generations is satisfied or go back to step 3.

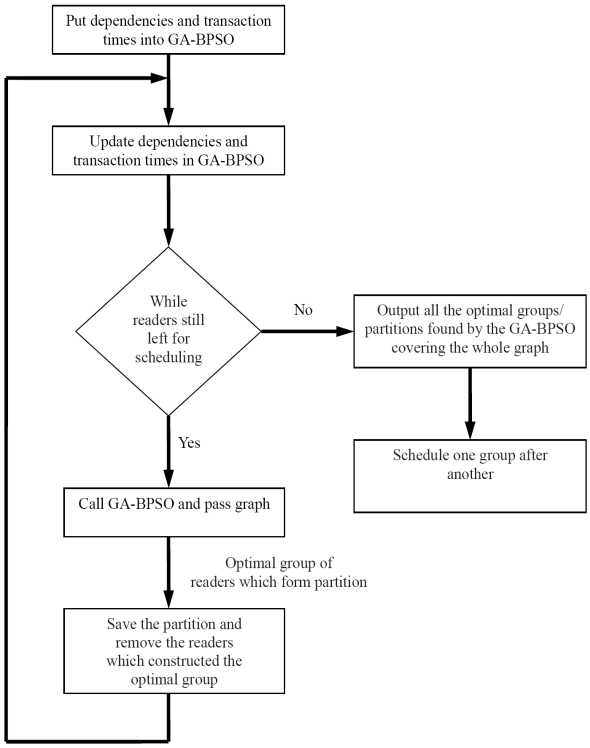

Figure 4. Implementation of GA-BPSO.

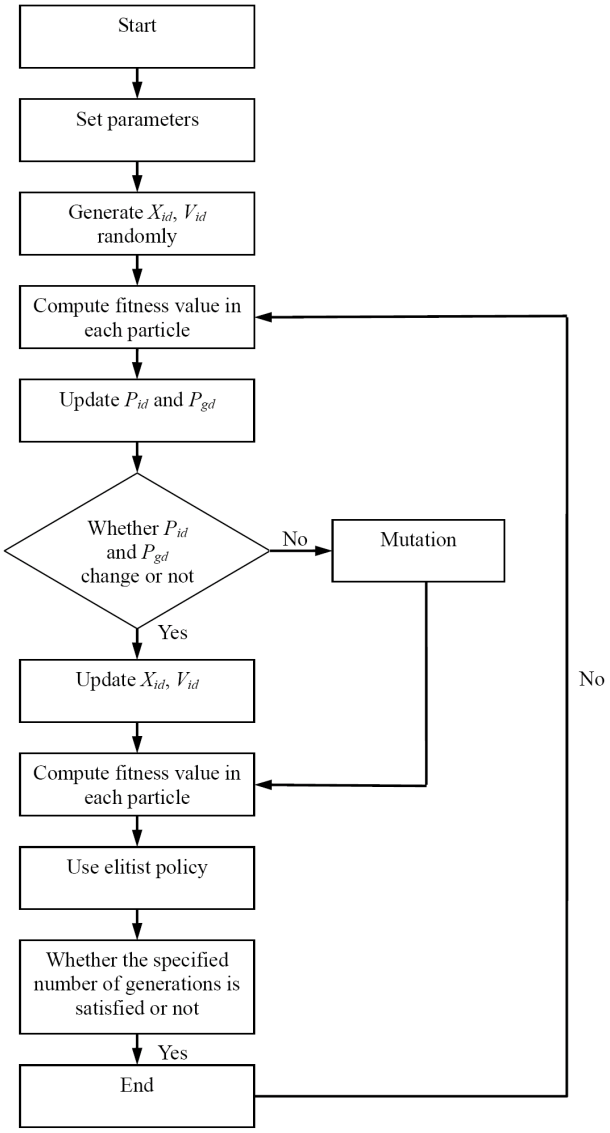

Figure 5. Flowchart of GA-BPSO.

III. SIMULATION TEST AND ANALYSIS

In this section, a RFID reader network application, including 10 readers is scheduled to validate the capability of the proposed method. Table I list the processing times for this 10 readers networks and the collision graph are shown as in Figure 6. The simulation results that obtained by our algorithm are listed in Table II.

It should be noted that the GA-BPSO algorithm can constantly find an optimal schedule results. In fact, with an increasing in the number of the readers (hence the degree of the graph), the problem of finding best solution becomes intractable. However, for the larger scale reader networks, our

proposed method is able to find the optimal schedule results robustly and consistently.

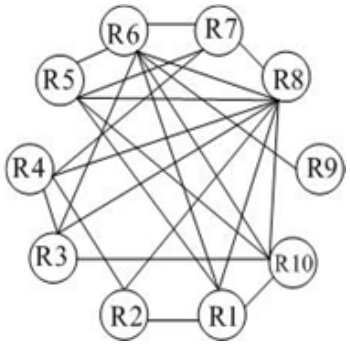

Figure 6. Collision graph of 10 RFID readers

Table I. Processing times of 10 readers

Reader	**1**	**2**	**3**	**4**	**5**
Time	14.7606	3.9499	3.0008	14.7618	9.4537
Reader	**6**	**7**	**8**	**9**	**10**
Time	14.6680	15.9297	9.9700	9.2377	9.2131

Table II. Schedule result of 10 readers

Time step	**Readers**	**Processing time**	**Total time**
1	1 5 6 8 10	14.7606	14.7606
2	4 7	15.9297	30.6903
3	9	9.2377	39.928
4	2	3.9499	43.8779
5	3	3.0008	46.8787

IV. CONCLUSIONS AND FUTURE WORK

This paper is devoted to giving a new strategy for scheduling reader networks in RFID-based ubiquitous computing environment. A symbiotic mechanism based algorithm, symbiotic multi-species optimizer, is proposed to search through space for an optimization problem. Simulation results on both mathematical benchmark functions and a real–world problem (i.e., the optimal scheduling for RFID reader networks) show that the GA-BPSO algorithm offers more robust and consistent performance in term of both solution quality and convergence rate. In fact our proposed method is suitable for scheduling large scale RFID reader networks.

REFERENCE

[1] A. George and J. W.-H. Liu, *Computer Solution of Large Sparse Positive Definite Systems*, Englewood Cliffs: Prentice-Hall, 1981.

[2] A. Kretsovalis and R. S. H. Mah, "Effect of redundancy on estimation accuracy in process data reconciliation," *Chemical Engineering Science*, vol. 42, no. 9, 1987, pp. 2115-2121.

[3] A. Pothen, H. D. Simon and K.-P. Liou, "Partitioning sparse matrices with eigenvectors of graphs," *SIAM Journal of Matrix Analysis and Applications*, vol. 11, no. 3, 1990, pp. 430–452.

[4] A. L. Buczak, H. Darabi and M.A. Jafari, "Research on genetic algorithm convergence for sensor network optimization problem," Information Sciences, vol. 5, no. 1, 2000, pp. 1035-1039.

[5] A. L. Buczak, H. Wang, H. Darabi and M.A. Jafari, "Genetic convergence research for sensor network optimization," *Information Sciences*, vol.133, no. 3-4, 2001, pp. 267-282.

[6] B. Hendrickson and R. Leland, "A multilevel algorithm for partitioning graphs," *Technical Report*, SAND93-1301, 1993.

[7] B. Nour-Omid, A. Raefsky and G. Lyzenga, "Solving finite element equations on concurrent computers," *American Society of Mechanical Engineers*, Applied Mechanics Division, AMD 86, 1987, pp. 209-227.

[8] D. B. West, *Introduction to Graph Theory*, Prentice Hall, second edition, 2000.

[9] D. E. Goldberg, Genetic Algorithms in Search, *Optimization and Machine Learning*, MA: Addison-Wesley, 1989.

[10] E. J. Larry, *Proceedings of the Sixth International Conference on Genetic Algorithms*, CA: Morgan Kaufmann Publishers, 1995.

[11] E. Malesinska, *Graph-theoretical models for frequency assignment problems*, Ph.D. thesis, Technischen University, Berlin, 1997.

[12] Forrest and Stephanie, Proceedings of the Fifth International Conference on Genetic Algorithms, CA: Morgan, 1993.

[13] G. Karypis and V. Kumar. "METIS: Unstructured graph partitioning and sparse matrix ordering system," *Technical report*, 1995.

[14] G. Karypis and V. Kumar. "A fast and high quality multilevel scheme for partitioning irregular graphs," *Technical Report*, TR 95-035, 1995.

[15] G. L. Miller, S.-H. Teng, W. Thurston and S. A. Vavasis, "Automatic mesh partitioning in sparse matrix computations: Graph theory issues and algorithms," *an IMA workshop volume*, 1993.

[16] Grefenstette and J. John. *Proceedings of an International Conference on Genetic Algorithms and Their Applications*, NJ: Lawrence Erlbaum Associates, 1985.

[17] Grefenstette and J. John, *Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms*, Hillsdale, NJ Lawrence Erlbaum Associates, 1987.

[18] H. Chen, Y. Zhu, K. Hu and B. Niu, "Application of a multi-species optimizer in ubiquitous computing for RFID networks scheduling," *Proceedings - Third International Conference on Natural Computation,* ICNC 2007 2, no. 4304589, 2007, pp. 420-425.

[19] J. A. Miller, W. D. Potter, R. V. Gandham and C. N. Lapena, "An evaluation of local improvement operators for genetic algorithm," *IEEE Trans*, vol. 23, 1993, pp. 1340-1351.

[20] J. C. Potts, T. D. Giddens and S. B. Yadav, "The Development and Evaluation of an Improved Genetic Algorithm Based on Migration and Artificial Selection," *IEEE Trans*, vol. 24, no. 1, 1994, pp. 73-86.

[21] J. H. Holland, *Adaptation in Natural and Artificial Systems*, MA: University of Michigan Press, 1975.

[22] J. Kennedy and R. C. Eberhart and Y. Shi, *Swarm Intelligence*, Morgan Kaufmann, 2002.

[23] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," *IEEE International Conference on Neural Networks - Conference Proceedings*, vol. 4, 1995, pp. 1942-1948.

[24] J. Sadri, C.Y. Suen, "A genetic binary particle swarm optimization model," *IEEE Congress on Evolutionary Computation*, CEC 2006, no. 1688373, 2006, pp. 656-663.

[25] J. Shield, "Partitioning concurrent VLSI simulation programs onto a multiprocessor by simulated annealing," *IEE Proceedings E: Computers and Digital Techniques*, vol. 134, no. 1, 1987, pp. 24-30.

[26] K. F. Man, K. S. Tang and S. Kwong, *Genetic Algorithm*, Springer, 1999.

[27] K. Shahookar and P. Mazumder, "VLSI placement techniques," *ACM Computing Surveys*, vol. 23, no. 2, 1991.

[28] L.A. Osadciw and K. Veeramachaneni, *Optimal Scheduling in Sensor Networks Using Swarm Intelligence*, CISS, Princeton, NewJersy, 2004.

[29] L.-Y. Chuang, H.-W. Chang, C.-J. Tu, C.-H. Yang, "Improved binary PSO for feature selection using gene expression data," *Computational Biology and Chemistry*, vol. 32, no. 1, 2008, pp. 29-37.

[30] M. Gen and R. Cheng, *Genetic algorithms and engineering design*, New York: Wiley, 1997.

[31] M. Srinivas and L. M. Patnaik, "Genetic Algorithms: A Survey," *Computer*, vol. 276, 1994, pp. 17-26.

[32] M. T. Heath and P. Raghavan, "A Cartesian nested dissection algorithm," *SIAM Journal on Matrix Analysis and Applications*, 1994.

[33] M. A. Khanesar, M. Teshnehlab, M.A. Shoorehdeli, "A novel binary particle swarm optimization," *Mediterranean Conference on Control and Automation*, MED, no. 4433821, 2007.

[34] P.-M. Zhang and G.. Rong, "Sensor network design for linear processes," *System Engineering Theory and Practice*, vol. 21, no. 12, 2001, pp. 36.

[35] R. Belew and L. Booker, *Proceedings of the Fourth International Conference on Genetic Algorithms*, CA: Morgan Kaufmann, 1991.

[36] R. C. Eberhart and J. Kennedy, *A new optimizer using particle swarm theory, 6th International Symposium on Micro Machine and Human Science*, Nagoya Japan, 1995, pp. 39-43.

[37] S. Lee, J. Lee, D. Shim and M. Jeon, "Binary particle swarm optimization for black-scholes option pricing," Computer Science, 4692 LNAI (PART 1), 2007, pp. 85-92.

[38] S. Selvakumar and C. Siva Ram Murthy, "An efficient algorithm for mapping parallel programs onto multicomputers," *Microprocessing and Microprogramming*, vol. 36, no. 2, 1993, pp. 83-92.

[39] S.H. Bokhari, "On the mapping problem," *IEEE Transactions on Computers*, vol. 30, no. 3, 1981, pp. 207-214.

[40] T. Bui and C. Jones. "A heuristic for reducing fill in sparse matrix factorization," *Parallel Processing for Scientific Computing*, 1993, pp. 445–452.

[41] T. Y. Chen and C. J. Chen, "Improvements of Simple Genetic Algorithm in Structural Design," *International Journal for Numerical Methods in Engineering*, vol. 40, 1997, pp. 1323-1334.

[42] V. Deolalikar, J. Recker, M. Mesarina and S. Pradhan, "Optimal scheduling for networks of RFID readers," *Computer Science*, 3823 LNCS, 2005, pp. 1025-1035.

[43] V. Kumar, A. Grama, A. Gupta and G.. Karypis, *Introduction to Parallel Computing: Design and Analysis of Algorithms*, Redwood City: Benjamin/ Cummings Publishing Company, 1994.

[44] V. Vaclavek and M. Loucka, "Selection of measurements necessary to achieve multicomponent mass balances in chemical plants," *Chemical Engineering Science*, vol. 31, no. 12, 1976, pp. 1199-1205.

[45] V. M. Lo, "Heuristic algorithms for task assignment in distributed systems," *IEEE Transactions on Computers*, vol. 37, no. 11, 1988, pp. 1384-1397.

[46] X. Wu, S. Lei, W. Jin, J. Cho and S. Lee, "Energy-efficient deployment of mobile sensor networks by PSO," *Computer Science*, 3842 LNCS, 2006, pp. 373-382.

[47] Y. Ali, and S. Narasimhan, "Sensor network design for maximizing reliability," *AIChE Journal*, vol. 42, no. 9, 1996, pp. 2563-2575.

[48] Y. Shi and R. C. Eberhart, "A modified particle swarm optimizer," *Proceedings of the IEEE International Conference on Evolutionary Computation*, 1998, pp. 69-73.