International Journal of Innovative Computing, Information and Control
Volume **8**, Number **10(B)**, October **2012**

ICIC International ©2012 ISSN 1349-4198
pp. 7157–7168

GENETIC ALGORITHM BASED ITERATIVE TWO-LEVEL ALGORITHM FOR RESOURCE ALLOCATION PROBLEMS AND APPLICATIONS

Shin-Yeu Lin[1,2] and Che-Yen Chang[1,2]

[1]Department of Electrical Engineering
[2]Green Technology Research Center
Chang Gung University
No. 259, Wen-Hwa 1st Rd., Kwei-Shan, Tao-Yuan 333, Taiwan
shinylin@mail.cgu.edu.tw; tony5367050@yahoo.com.tw

Received August 2011; revised January 2012

Abstract. *This study concerns the resource allocation problem that involves complicating constraints and cannot be solved using correction operations of a genetic algorithm (GA). A GA-based iterative two-level algorithm is developed to solve this problem by decomposing it into master and slave problems, such that the complicating constraint function is treated as the objective function of the slave problem, which can be solved by GA, and the master problem, which includes the complicating constraint, is solved using a bisection method that is based on the optimal objective value determined in the slave problem. An example of the application of the proposed algorithm is the minimum time slot assignment problem (MTSAP) associated with a radio frequency identification (RFID) system. The GA that is utilized to solve the slave problem of the MTSAP has special features. The proposed algorithm is tested by applying it to the MTSAPs of four reader networks and many runs are performed for each MTSAP. The obtained solution to each MTSAP is optimal. The proposed algorithm is compared with a simulated annealing (SA) method. The comparison reveals that the proposed algorithm outperforms the SA method in terms of the optimality of the obtained solutions and computing speed.*

Keywords: Resource allocation, Iterative two-level algorithm, RFID, Reader collision, Minimum time slot assignment problem, Genetic algorithm

1. **Introduction.**

The *genetic algorithm* (GA) has been extensively and successfully used to solve various types of *combinatorial optimization problem*. For *unconstrained* combinatorial optimization problems, the use of a GA is rather straightforward if a suitable representation scheme can be designed. However, for *constrained* combinatorial optimization problems, *correction operations* are typically required to make the *infeasible* offspring or mutated chromosomes *feasible* following the *crossover* and *mutation* operations [1]. Since no systematic method is available for designing the correction operation, which is mostly problem-dependent, the fact that the problem involving *complicating constraints* that cannot be applied using correction operations in the GA is unsurprising.

This study considers a *resource allocation problem*, which is a class of constrained combinatorial optimization problems and is an NP-hard optimization problem, of the following form

$$\begin{array}{rl} \min & N \\ \text{subject to} & e(x, N) = 0 \\ & u(x, N) \le 0 \\ & N_{\min} \le N \le N_{\max} \end{array} \tag{1}$$

where N is an integer variable that is the number of resources; $N_{\min}$ and $N_{\max}$ are the lower and upper bounds of N, respectively; x is the integer vector of resource allocation variables; $e(x, N) = 0$ and $u(x, N) \leq 0$ represent the *equality* and *inequality constraints* that are *easy* and *hard* to handle using correction operations, respectively. Restated, the *complicating constraint* $u(x, N) \leq 0$ prohibits the direct application of GA to resource allocation problem shown in (1). Accordingly, the purpose of this work is to develop a GA-based *iterative two-level* algorithm to resolve the difficulty caused by the complicating constraint in the considered resource allocation problem.

The considered resource allocation problem can be interpreted as the problem of allocating minimum number of resources to meet the required system constraints [2]. The many applications of such problems include allocating minimum number of computing resources at nodes that do not have any resource in a *grid computing network* to deliver the required *computing service reliability* [3,4] or assigning the minimum total number of *time slots* to *readers* in the *radio frequency identification* (RFID) reader network such that each reader can complete the reading of all *tags* in its *interrogation zone* without causing any *reader collision* [5,6]. In the former application, the terms associated with the resource allocation problem (1) are as follows. Allocating computing resources to nodes that do not have any resource is the easy-to-handle equality constraint. Satisfying the computing service reliability requirement is the hard-to-handle complicating constraint. The objective function is to minimize the number of computing resources to be allocated. The latter application is also called the *minimum time slot assignment problem* (MTSAP) of the RFID reader network. In this application, the resources are time slots; the constraint that each reader should read all tags in its interrogation zone is the equality constraint; the constraint that readers must not collide with each other is the complicating constraint, and the objective function is to minimize the total number of time slots to be assigned.

This paper will present the application of the proposed GA-based iterative two-level algorithm to the MTSAP of the RFID reader network: the problem was tackled by Lin and Lin in [5] and Hung et al. in [6] using *simulated annealing* (SA) methods. The SA method that was developed by Lin and Lin [5] is to search for a feasible solution that neighbors the current solution and to determine whether to accept the sought solution based on the criteria applied in the *Metropolis loop* in a typical SA method. Such an SA method may be inefficient in improving the current solution in the vicinity of optimal solution, because the neighboring solution that is defined in the SA method differs from the current solution in only one assigned time slot of one reader, but finding the optimal solution typically involves reassigning the time slots of several readers simultaneously. A similar difficulty may also arise in the SA method that was developed by Hung et al. in [6].

Generating new solutions by simultaneously reassigning time slots of several readers from current solutions is a typical result of the *crossover* and *mutation* operations in a GA. However, a GA can be applied only when the total number of time slots is specified. Therefore, the GA-based iterative two-level algorithm to be developed herein is better suited to the MTSAP. To demonstrate this fact, Section 4 will compare the proposed algorithm with the SA method by performing extensive simulations.

This paper is organized as follows. Section 2 presents the GA-based iterative two-level algorithm. Section 3 presents the MTSAP for an RFID reader network and the application of the proposed GA-based iterative two-level algorithm to the MTSAP. Section 4 tests the proposed algorithm on the MTSAPs for four reader networks and compares it with the simulated annealing (SA) method that was used by Lin and Lin [5]. Section 5 draws conclusions.

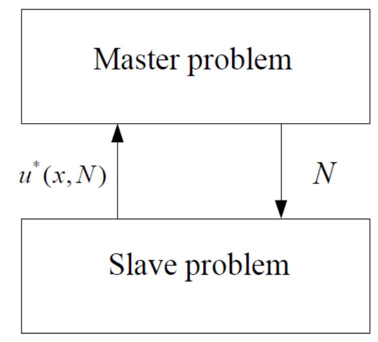

FIGURE 1. Structure of the iterative two-level algorithm

2. GA-Based Iterative Two-Level Algorithm.

To resolve the computational difficulty that is raised by the complicating constraint, $u(x,N) \leq 0$, we decompose the resource allocation problem (1) into master and slave problems first, as shown in (2) and (3), respectively. Then, the proposed iterative two-level algorithm will solve the master and slave problems iteratively as presented in Figure 1. Remark 3.1 in Subsection 3.2 explains why the constraint $u(x,N) \leq 0$ causes complication.

Master problem:

$$\begin{array}{rl} \min & N \\ \text{subject to} & u(x,N) \leq 0 \\ & N_{\min} \leq N \leq N_{\max} \end{array} \tag{2}$$

Slave problem:

$$\begin{array}{rl} \min & u(x,N) \\ \text{subject to} & e(x,N) = 0 \end{array} \tag{3}$$

The master problem formulated in (2) is used to evaluate N. The slave problem formulated in (3) is to minimize the value of $u(x,N)$ for the given N that is obtained from the master problem.

Since the slave problem does not involve the complicating constraint, the slave problem can be solved using GA when N is specified. We let $u^*(x,N)$ be the optimal objective value of (3) for the given N. The basic idea on which the proposed iterative two-level algorithm that is presented in Figure 1 is based is described as follows. For a given N that is obtained from the master problem, the optimal solution $u^*(x,N)$ that is obtained by solving the slave problem indicates whether the value of N is large enough to satisfy the complicating constraint $u(x,N) \leq 0$. Then, based on this $u^*(x,N)$, the method for solving the master problem involves the adjustment of the value of N according to the following rule: increase the value of N if $u^*(x,N) > 0$ and reduce it otherwise; the new N that is determined in the master problem will be passed to the slave problem and so forth. This iterative two-level process will proceed until the optimal N is obtained.

To adjust efficiently the value of N, the master problem is solved using a *bisection* method. The bisection method firstly defines two integer variables $\overline{N}$ and $\underline{N}$. In the slave problem, if $N = \overline{N}$, then $u^*(x,N) \leq 0$; however, if $N = \underline{N}$, then $u^*(x,N) > 0$. Based on the assumption that $u^*(x,N_{\max}) \leq 0$ and $u^*(x,N_{\min}) > 0$, $\overline{N} = N_{\max}$ and $\underline{N} = N_{\min}$ are set initially. The bisection method initially sets $N = \left\lceil \frac{\overline{N}+\underline{N}}{2} \right\rceil$, where $\lceil(\cdot)\rceil$ is the smallest integer that exceeds or equals to $(\cdot)$ and passes this value to the slave problem. The slave problem is thus solved as N is given. Once the slave problem has been solved, $u^*(x,N)$ is passed to the master problem as shown in Figure 1. If $u^*(x,N) \leq 0$, which indicates that the constraint $u(x,N) \leq 0$ in the master problem is satisfied, then the update $\overline{N} := N$ is applied. However, if $u^*(x,N) > 0$, indicating that the constraint $u(x,N) \leq 0$ in the master problem is not satisfied, then the update $\underline{N} := N$ is applied. The bisection method will determine the new $N = \left\lceil \frac{\overline{N}+\underline{N}}{2} \right\rceil$ from the updated $\overline{N}$ and $\underline{N}$ and pass this value to the slave problem and so forth. The iterative two-level algorithm will terminate when $\overline{N} - \underline{N} = 1$, and this $\overline{N}$ is the optimal N. To validate the iterative two-level algorithm, the obtained optimal solution must be proven to be the optimal solution of (1).

Theorem 2.1. *Suppose $\{x|e(x, N) = 0$ in (3)$\}$ is a countably finite nonempty set for every N delivered from the master problem, the optimal solution that is obtained by solving (2) and (3) using the iterative two-level algorithm is an optimal solution of (1).*

Proof: For every N delivered from the master problem, $\{x|e(x, N) = 0 \text{ in } (3)\}$ is a countably finite nonempty set, then (3) has an optimal solution. Let (N^*, x^*) be the optimal solution that is obtained using the iterative two-level algorithm and let $x^*(N)$ be the optimal solution of (3) for the given N. From (3), $x^* = x^*(N^*)$ and $e(x^*, N^*) = 0$. Since N^* is the most recent $\overline{N}$ that is determined in the master problem, and by the definition of $\overline{N}$, $u^*(x^*, N^*) \leq 0$. From (2), N^* must satisfy $N_{\min} \leq N^* \leq N_{\max}$. Based on the termination criterion $N^* - \underline{N} = 1$, and the definition of $\underline{N}$, (x^*, N^*) must be the optimal solution of (1), because no integer N that can be feasible to (1) and satisfy $N - N^* < 0$. The proof is thus completed.

3. Application of GA-Based Iterative Two-Level Algorithm to Minimum Time Slot Assignment Problem (MTSAP) of RFID Reader Network.

3.1. Preliminaries.

In recent years, RFID, which automatically identifies tagged objects, has been extensively utilized in various applications including warehouses, shopping centers, location tracking and healthcare and others [7,8]. RFID devices are of two types. One is the *tag*, which contains a unique identifier and the information about the object, and the other is the *reader*. The area that RFID devices can be identified is called an *interrogation zone*. Within the interrogation zone, multiple tags' sending signals to the same reader simultaneously may cause tag-to-tag collision. Various tag *anti-collision protocols* have been proposed in [9-13]. In many applications, several readers are deployed to ensure complete interrogation coverage and yield a high read rate [14-16]. In these cases, *reader-to-tag collision* may occur when a tag receives signals from more than one reader at the same time. Such a collision can make the tag respond arbitrarily to the readers and cause incorrect interrogation. A *reader-to-reader collision* occurs when a reader, which is in the process of listening to a tag's reply, simultaneously receives strong signals from one or more neighboring readers that are operating at the same frequency. These two forms of collisions are regarded as *reader collisions*, and any pair of readers that may encounter such collisions are called *neighboring* readers. Therefore, to prevent a reader collision, neighboring readers must not be allowed to read tags in the same time slots. Additionally, various readers in the RFID reader network require different number of time slots, because different number of tags are present in the interrogation zones of different readers. Hence, the MTSAP is solved by assigning minimum total number of time slots to readers in the RFID reader network such that each reader can read all tags in its interrogation zone (equality constraint in (1)) without causing any reader collision (inequality constraint in (1)).

3.2. Mathematical formulation of minimum time slot assignment problem (MT SAP).

Assume the reader network, whose network topology is specified, comprises m readers. For example, Figure 2 displays a reader network of four readers. Any pair of readers that interfere with each other if they use the same time slots is connected by a branch.

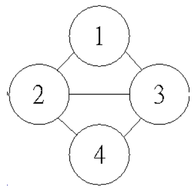

FIGURE 2. Example reader network

Let N be the total number of time slots that are required to read all tags. Let c_j be the number of time slots that are required by reader j to read all of the tags in its interrogation zone. Let the *time slot assignment variable* x_{ij} be defined such that $x_{ij} = 1$ if time slot i is assigned to reader j, and 0 otherwise. The *connection matrix* A of the reader network is defined as $A = [a_{ij}]$, where the (i, j)th entry $a_{ij} = 1$ if readers i and j are connected, and 0 otherwise. For example, the connection matrix A for the reader network that is shown in Figure 2 is

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

Define *same-time-slot utilization matrix* $Y(N)$ for a given N as follows. $Y(N) = [y_{jk}]$, where the (j, k)th entry $y_{jk} = L$ if L time slots are simultaneously used by readers j and k, and L is a non-negative integer. The "$\wedge$" operation is defined for matrices A and $Y(N)$ as $A \wedge Y(N) = [a_{jk} \wedge y_{jk}]$, where $a_{jk} \wedge y_{jk} = y_{jk}$ if $a_{jk} = 1$, and 0 otherwise. Using the above notation, the MTSAP can be formulated as the minimization of the total number of time slots that will enable each reader to complete the reading of all tags in its interrogation zone without causing any reader collision:

$$\begin{aligned} \min \quad & N \\ \text{subject to} \quad & c_j = \sum_{i=1}^{N} x_{ij}, \ j = 1, \cdots, m \\ & \tfrac{1}{2} \sum_{(j,k)} A \wedge Y(N) \leq 0 \\ & N_{\min} \leq N \leq N_{\max} \end{aligned} \tag{4}$$

where $c_j = \sum\limits_{i=1}^{N} x_{ij}$ is the number of time slots that are required by reader j; $\frac{1}{2} \sum\limits_{(j,k)} A \wedge Y(N)$ represents the total number of collisions, which is a function of N, and $\frac{1}{2} \sum\limits_{(j,k)} A \wedge Y(N) \leq 0$ represents the constraint for no reader collision; $\frac{1}{2} \sum\limits_{(j,k)} A \wedge Y \leq 0$ is the complicating constraint of the MTSAP, and is equivalent to $\frac{1}{2} \sum\limits_{(j,k)} A \wedge Y = 0$, because $\frac{1}{2} \sum\limits_{(j,k)} A \wedge Y$ cannot be less than 0; $N_{\max} = \sum\limits_{j=1}^{m} c_j$, because if $N = \sum\limits_{j=1}^{m} c_j$, then each time slot can be assigned only to one reader, causing no reader collision; $N_{\min} = \max\limits_{j \in \{1, \cdots, m\}} c_j$, which is the smallest number, N, that $c_j = \sum\limits_{i=1}^{N} x_{ij}$, $j = 1, \cdots, m$ can possibly hold. The variables in (4) are N and x_{ij} for all i and j.

Remark 3.1. $\frac{1}{2}\sum_{(j,k)} A \wedge Y(N) \le 0$, *which is the complicating constraint of the MTSAP (4), is explained as follows. For a given* N *and time slot assignment pattern* x_{ij}, $i = 1, \cdots, N$, $j = 1, \cdots, m$, *suppose that* $\frac{1}{2}\sum_{(j,k)} A \wedge Y(N) > 0$; *reshuffling the time slot assignment* x_{ij}, $i = 1, \cdots, N$, $j = 1, \cdots, m$, *to make* $\frac{1}{2}\sum_{(j,k)} A \wedge Y(N) \le 0$ *is an optimization problem, which can not be solved by applying the correction operations in the GA.*

3.3. **Application of GA-based iterative two-level algorithm.** The MTSAP has exactly the same form as the considered resource allocation problem (1) that was presented in Section 1. Therefore, the MTSAP can be decomposed into master and slave problems described by (5) and (6), respectively, to which the proposed GA based iterative two-level algorithm can be applied.

Master Problem:

$$\begin{aligned} \min \quad & N \\ \text{subject to} \quad & \tfrac{1}{2}\sum_{(j,k)} A \wedge Y(N) \le 0 \\ & N_{\min} \le N \le N_{\max} \end{aligned} \tag{5}$$

Slave Problem:

$$\begin{aligned} \min \quad & \tfrac{1}{2}\sum_{(j,k)} A \wedge Y(N) \\ \text{subject to} \quad & c_j = \sum_{i=1}^{N} x_{ij},\ j = 1, \cdots, m \end{aligned} \tag{6}$$

Since the form of the master problem (5) is exactly the same as that of (2), the bisection method that is used to solve (2) can be used to solve (5) based on the optimal objective value $\frac{1}{2}(\sum_{(j,k)} A \wedge Y(N))^*$ of the slave problem (6). Therefore, the validation of solving (5) and (6) using iterative two-level algorithm can be justified in the following.

Lemma 3.1. *Optimal solution obtained by solving (5) and (6) using iterative two-level algorithm is an optimal solution of (4).*

Proof: The N that is obtained by solving the master problem must satisfy $N \ge N_{\min}$; then by the definition of $N_{\min} = \max_{j\in\{1,\cdots,m\}} c_j$, the set $\{x = [x_1, \cdots, x_m]^T | c_j = \sum_{i=1}^{N} x_{ij}, j = 1, \cdots, m\}$ must be a countably finite nonempty set for the given N. Hence, this lemma directly follows from Theorem 2.1.

3.3.1. *Genetic algorithm (GA) for solving save problem.* Although the slave problem (6) does not include the complicating constraint, it does include an equality constraint. Two critical components of GA for solving any specific constrained optimization problem are *representation scheme* and *correction operations*; the design of these two components for solving the slave problem (6) is the special features of the employed GA.

3.3.1.1. Representation scheme. The representation scheme in GA for (6) is rather straightforward, because the time slot assignment variable, x_{ij}, is an integer-either 0 or 1- and N is given. A *chromosome* is defined as a string of mN symbols, 0 and 1, as shown in Figure 3, in which the N symbols from symbol $(j-1)N+1$ to symbol jN represent the assignment of time slots to reader j.

3.3.1.2. Fitness. Since (6) is a minimization problem, the mechanism of GA is maximizing a positive *fitness*, and so the fitness of a chromosome is defined as $\sum_{j=1}^{m} c_j - \frac{1}{2}\sum_{(j,k)} A \wedge Y(N)$,

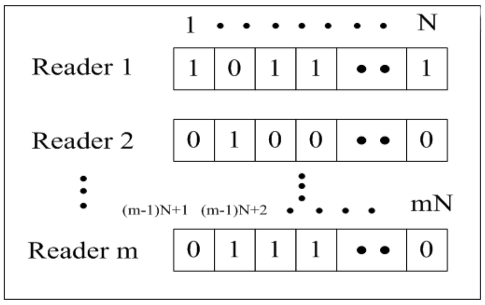

FIGURE 3. Representation of a chromosome

which is positive. Notably, the maximization of the defined fitness is equivalent to the minimization of the total number of collisions.

3.3.1.3. Correction operations. We define a *legal* chromosome as the chromosome that satisfies the constraint in (6). Accordingly, for a legal chromosome, the number of 1s in the set of N symbols that are associated with reader j should be c_j for $j = 1, \cdots, m$. If the chromosome is not legal, then it cannot be a feasible solution of (6).

Since the *crossover* operation in the GA cannot guarantee that the generated *offspring* are legal, correction operations must be applied to each resulting offspring to make it legal. A similar situation is associated with the *mutation* operation. The details of the correction operations are as follows.

Suppose an offspring is not legal; the following condition must hold for at least one reader j such that the number of 1s in the N symbols that correspond to reader j does not equal c_j. Restated, if the N symbols ranging from $(j-1)N+1$ to jN include k 1s, then either $k < c_j$ or $k > c_j$. To legalize the illegal chromosome, the following is performed. In the case $k > c_j$, $(k - c_j)$ 1s are *randomly* selected from the k 1s in the symbols from $(j-1)N+1$ to jN and all are set to 0. Similar procedures apply in the case $k < c_j$.

3.3.1.4. Genetic algorithm for solving slave problem. The aforementioned representation scheme, fitness and correction operations can be used in a typical GA [1] that includes (i) a *roulette tournament selection scheme* for selecting and keeping the chromosomes with better fitness, (ii) a *single-point crossover scheme* that is applied to the parents that are selected from the population pool with probability p_c, followed by correction operations that are applied to the resulting offspring, and (iii) a *mutation scheme* that mutates each symbol in a chromosome with probability p_m, and then applies correction operations to each mutated chromosome. The flow chart in Figure 4 summarizes the GA. The convergence criterion that is applied herein is that the fitness of the best-so-far chromosome equals $\sum_{j=1}^{m} c_j$, which implies no reader collision, or the number of iterations exceeds a preselected number $I_{\max}$.

4. **Test Results and Comparisons.**

To test the proposed algorithm, four reader networks of various sizes (reader networks 1, 2, 3 and 4), presented in Figures 5-8, respectively, are used. Reader networks 1, 2, 3 and 4 comprise 10, 20, 37 and 51 readers, respectively.

In each reader network, neighboring readers are connected by a branch; the number in the circle is the reader index, and the number in the parenthesis represents the number of time slots that are required by that reader. Hence, the values of $(N_{\max}, N_{\min})$ of the four reader networks are (49, 8), (102, 8), (192, 9) and (264, 8) corresponding to reader networks 1, 2, 3 and 4, respectively. The parameters that are used in the GA shown in

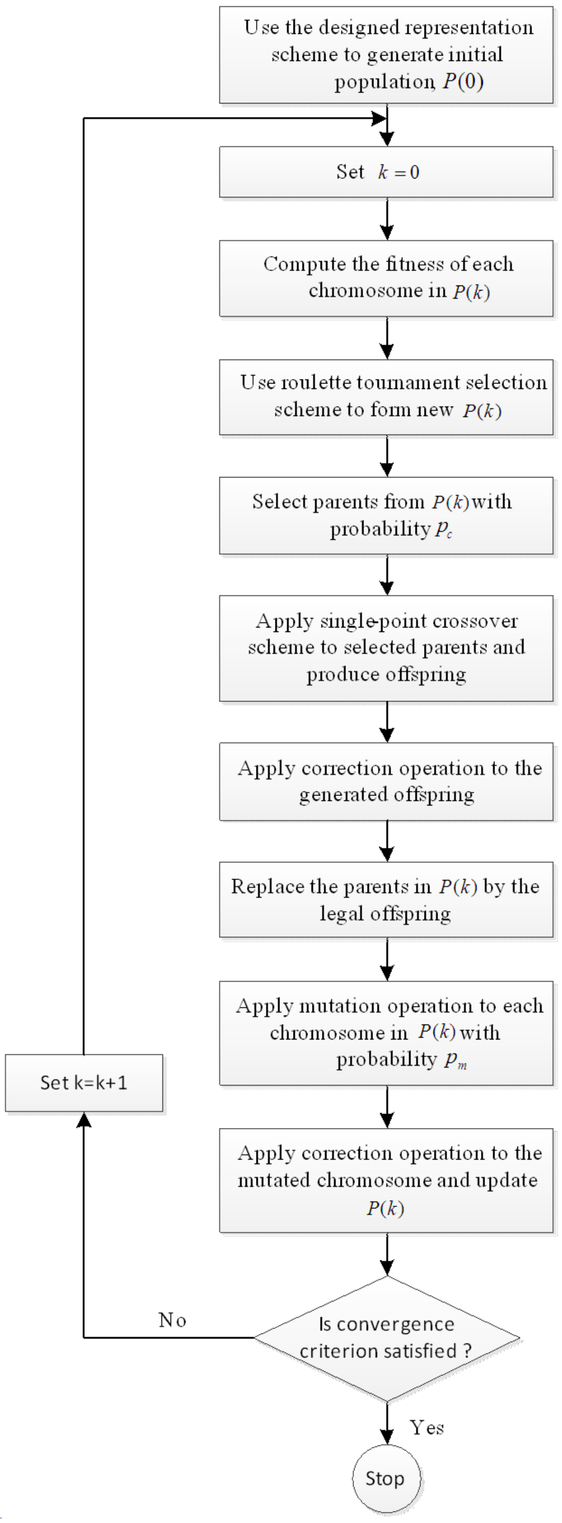
Use the designed representation scheme to generate initial population $P(0)$
Set $k=0$
Compute the fitness of each chromosome in $P(k)$
Use roulette tournament selection scheme to form new $P(k)$
Select parents from $P(k)$ with probability P_c
Apply single-point crossover scheme to selected parents and produce offspring
Apply correction operation to the generated offspring
Replace the parents in $P(k)$ by the legal offspring
Apply mutation operation to each chromosome in $P(k)$ with probability p_m
Apply correction operation to the mutated chromosome and update $P(k)$
Is convergence criterion satisfied ?
No
Yes
Set k=k+1
Stop

FIGURE 4. Flow chart of GA for solving slave problem

Figure 4 are set to be $p_c = 0.7$, $p_m = 0.02$, $I_{\max} = 5000$ and $|P(k)| = 300$ for any k, where $|(\cdot)|$ denotes the cardinality of the set $(\cdot)$.

The GA-based iterative two-level algorithm is used to solve the MTSAPs of the above four reader networks. Because of the random nature of the operations in GA, the proposed algorithm is applied five times to each MTSAP of the four reader networks. The average numbers of two-level iterations for the MTSAPs of reader networks 1, 2, 3 and 4 are 2, 3, 8 and 7, respectively. The average total numbers of time slots for reader networks 1, 2, 3 and 4 are 19, 20, 25 and 22, respectively. These results reveal the *stable numerical property* of the proposed algorithm, because the five obtained total numbers of time slots are the same in all five runs of each MTSAP. These obtained total numbers of time slots are optimal N for the corresponding MTSAP, which fact is easily determined from the reader networks that are presented in Figures 5-8. For example, for reader network 1 shown in Figure 5, $\max_i \sum_{j \in C_i} c_j = 19$, where C_i represents the ith *clique* of the reader network, which is defined as a subset of readers in the reader network such that every pair of readers in this subset is connected. Since different readers in a clique cannot use the same time slot, the number of time slots that are required by the MTSAP for reader network 1 cannot be less than 19. Therefore, $N = 19$ obtained by the proposed algorithm must be the optimal total number of time slots required for the MTSAP of reader network 1. Similarly, total numbers of time slots obtained for the remaining three MTSAPs of reader networks 2, 3 and 4 can also be proven to be optimal. Therefore, for the MTSAPs of the four reader networks that are presented in Figures 5-8, the optimal solution is obtained in every run.

For the purpose of comparisons, the MTSAPs of the four reader networks are solved using the *simulated annealing* (SA) method that was proposed by Lin and Lin [5]. Owing to the random nature of the search operations in the SA method, five runs are conducted for each MTSAP as for the proposed algorithm. The average total numbers of time slots obtained by the SA method for the MTSAPs of reader networks 1, 2, 3 and 4 are 20.2, 22.8, 28.8 and 26.2, respectively. The average total numbers of time slots obtained using the proposed algorithm and the SA method are denoted by $\hat{N}_{our}$ and $\hat{N}_{SA}$, respectively, and the percentage $\frac{\hat{N}_{SA}-\hat{N}_{our}}{\hat{N}_{our}} \times 100\%$ is calculated for the MTSAPs of the four reader

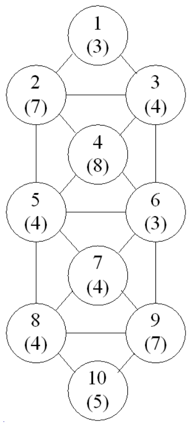

Figure 5. Reader network 1

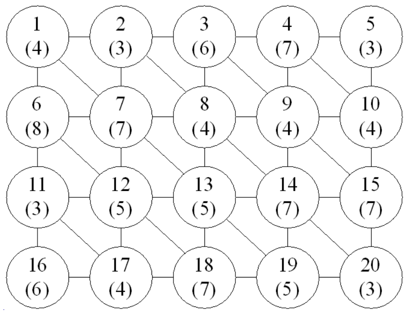

Figure 6. Reader network 2

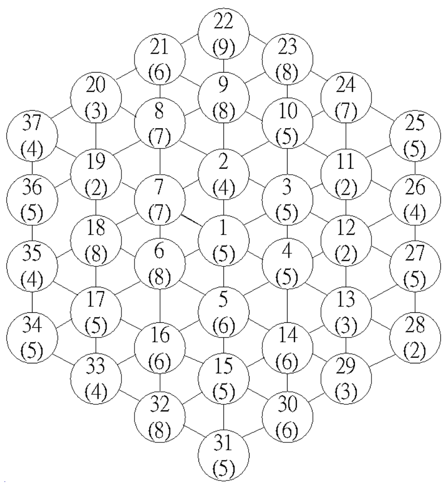

Figure 7. Reader network 3

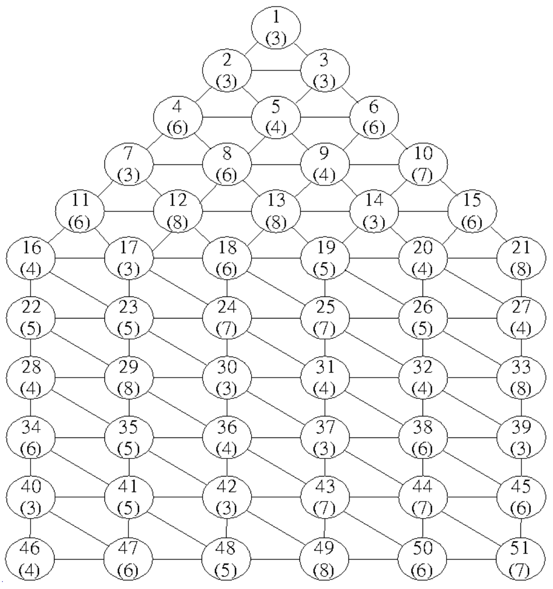

Figure 8. Reader network 4

Table 1. Comparisons of proposed algorithm with SA method

Reader Network	Number of Readers (m)	N^*	$\hat{N}_{our}$	$\hat{N}_{SA}$	$\frac{\hat{N}_{SA}-\hat{N}_{our}}{\hat{N}_{our}} \times 100\%$	CPU time (seconds)		Speed up factor: $\frac{\bar{t}_{SA}}{\bar{t}_{our}}$
						$\bar{t}_{our}$	$\bar{t}_{SA}$	
1	10	19	19	20.2	6.3%	0.9	4	4.44
2	20	20	20	22.8	14%	24.5	117.8	4.81
3	37	25	25	28.8	15.2%	114.5	2184	19.07
4	51	22	22	26.2	19%	522.8	10117	19.35

networks as the *degree of superiority* of the proposed algorithm over the SA method. Table 1 presents the size of each reader network, N^*, $\hat{N}_{our}$, $\hat{N}_{SA}$ and $\frac{\hat{N}_{SA}-\hat{N}_{our}}{\hat{N}_{our}} \times 100\%$, where N^* is the optimal total number of time slots for the MTSAP of the corresponding reader network. As stated above, $\hat{N}_{our} = N^*$. According to the column $\frac{\hat{N}_{SA}-\hat{N}_{our}}{\hat{N}_{our}} \times 100\%$

in Table 1, the proposed algorithm outperforms the SA method, and as the size of the reader network increases, the performance of the proposed algorithm is even better. The average CPU times consumed by the proposed algorithm and the SA method for the four MTSAPs are denoted by $\bar{t}_{our}$ and $\bar{t}_{SA}$, respectively. The data demonstrate that $\bar{t}_{our}$ is much shorter than $\bar{t}_{SA}$ for all MTASPs. Moreover, the final column indicates that the speed-up factor, $\frac{\bar{t}_{SA}}{\bar{t}_{our}}$, of the proposed algorithm with respect to the SA method is around 20 when the network is large.

Remark 4.1. *The proposed algorithm can yield the optimal solution of the MTSAP very rapidly for the small reader network as shown in Table 1. Although the computation time increases with the size of the reader network, and NP-hard optimization problems are being solved herein, the consumed CPU time for large reader network can still be regarded as very fast. However, for real-time applications, heuristic algorithms for large reader networks may have to be studied. Although this issue is beyond the scope of this paper, the proposed algorithm can yield the optimal solution in a reasonable computation time and thus can be used to evaluate the performance of any newly developed heuristic algorithm.*

5. **Conclusions.**

This work presented a GA-based iterative two-level algorithm for solving the resource allocation problem with complicating constraints. The considered problem is an NP-hard optimization problem. The proposed algorithm is used to solve the MTSAPs of RFID reader networks and many simulations using various reader networks are conducted to demonstrate the numerical stability of the proposed algorithm and its effectiveness in finding optimal solutions. Comparisons are also made with the SA method for all of the tested MTASPs. The results reveal that the performance and computing speed of the proposed algorithm are better than those of the SA method to an extent that increases with the size of the reader network.

Acknowledgment. The authors wish to thank the associate editor and the anonymous reviewers for their helpful and constructive comments. This research work was supported by the NSC of Taiwan under grant #NSC100-2221-E-182-035. The financial aid of graduate students involved in this work is supported by CGURP of Chang Gung University. Ted Knoy is appreciated for his editorial assistance.

REFERENCES

[1] S. M. Sait and H. Youssef, *Iterative Computer Algorithms with Applications in Engineering: Solving Combinatorial Optimization Problem*, IEEE Comput. Soc., Los Alamitos, CA, USA, 1999.

[2] T. Ibaraki and N. Katoh, *Resource Allocation Problems: Algorithmic Approaches*, MIT Press, Boston, 1988.

[3] Y. S. Dai and X. L. Wang, Optimal resource allocation on grid systems for maximizing service reliability using a genetic algorithm, *Reliab. Eng. Syst. Saf.*, vol.91, no.9, pp.1071-1082, 2006.

[4] Y. S. Dai, M. Xie and X. Wang, A heuristic algorithm for reliability modeling and analysis of grid systems, *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol.37, no.2, pp.189-200, 2007.

[5] C.-F. Lin and F. Y.-S. Lin, A simulated annealing algorithm for RFID reader networks, *Proc. of the 2007 IEEE WCNC*, pp.1669-1672, 2007.

[6] S.-W. Hung, Y.-F. Huang and J. Sum, Simulated annealing for the reader collision problem in RFID systems, *Proc. of the 3rd International Symposium on Health Information Management*, Tainan, 2008.

[7] E. Bottani and A. Rizzi, Economical assessment of the impact of RFID technology and EPC system on the fast-moving consumer goods supply chain, *Int. J. Production Economics*, vol.112, no.2, pp.548-569, 2008.

[8] A. Oztekin, F. M. Pajouh, D. Delen and L. K. Swim, An RFID network design methodology for asset tracking in healthcare, *Decision Support Systems*, vol.49, no.1, pp.100-109, 2010.

[9] S. Lee, S. Joo and C. Lee, An enhanced dynamic framed slotted ALOHA algorithm for RFID tag identification, *Proc. of MouiQuitous*, pp.166-172, 2005.

[10] J. Myung, W. Lee and J. Srivastava, Adaptive binary splitting for efficient RFID tag anti-collision, *IEEE Commun. Lett.*, vol.10, no.3, pp.144-146, 2006.

[11] G. Khandelwal, A. Yener and M. Chen, OPT: Optimal protocol tree for efficient tag identification in dense RFID systems, *Proc. of IEEE Int. Conf. Commun.*, pp.128-133, 2006.

[12] J. Park, M. Y. Chung and T.-J. Lee, Identification of RFID tags in framed slotted ALOHA with robust estimation and binary selection, *IEEE Commun. Lett.*, vol.11, no.5, pp.452-454, 2007.

[13] J.-B. Eom, T.-J. Lee, R. Rietman and A. Yener, An efficient framed slotted ALOHA algorithm with pilot frame and binary selection for anti-collision of RFID tags, *IEEE Commun. Lett.*, vol.12, no.11, pp.861-863, 2008.

[14] J. Waldrp, D. W. Engels and S. E. Sarma, Colorwave: A MAC for RFID reader networks, *Proc. of IEEE Conf. Wireless Commun. Netw.*, pp.1701-1704, 2003.

[15] J. Waldrp, D. W. Engels and S. E. Sarma, Colorwave: An anti-collision algorithm for the reader collision problems, *Proc. of IEEE Conf. Wireless Commun. Netw.*, pp.1206-1210, 2003.

[16] J. Ho, D. W. Engels and S. E. Sarma, HiQ: A hierarchical Q-learning algorithm to solve the reader collision problems, *Proc. of Int. Symp. Appl. Internet Workshop*, pp.88-91, 2006.