Solving the Faculty-Course Assignment Problem using Genetic Algorithm

Feng-Shuo Yu, Tzu-Chia Wang
Assistant Professor
Department of Information Management
Kun Shan University, Tainan City, Taiwan (R.O.C.)

Abstract : In this research, we study the faculty-course assignment problem using Genetic Algorithm. Our research focuses on how to make balance in two perspectives: teaching and learning. For teaching, we study how to make the courses assignment fair based on several constraints; as for learning, we study how to assign qualified teachers to related courses in order to improve students' learning satisfaction. This research is bases on past study and uses weighted values on several controlling variables. The reason for adopting weighted values on controlling variables is that different departments can obtain a satisfied courses assignment result according to their departmental courses assignment requirements. The contributions of this study not only provide fairness for courses assignment, but also a system that is flexible to meet different requirements need.

Keywords *- Genetic algorithm, Faculty course assignment, Learning satisfaction*

I. INTRODUCTION

Before a new term or semester begin, university departments have to make a decision for courses assignment that will best fit each individual faculty member. Faculty members have different areas of expertise that must be allocated among specific courses. The Faculty-Course Assignment Problem (FCAP) for university institutions can be considered as a special case of the more general course timetabling problem, but in essence, FCAP is different from the course timetabling problem in that the main concern of course timetabling problem is how to assign courses in a semester according to available human resources and/or hardware resources such as classrooms and time periods, its main task is the scheduling of resources and time periods in a week. On the other hand, the main concern of FCAP is about how to assign all the courses offered by a department to all available faculty members in order to maintain the fairness and satisfactory level of courses assignment among faculty members, improve the teaching quality for students.

There are several reasons for the need to have a computational methodology for FCAP. Most university institutions depend on human labor to process courses assignment task for each semester. Not only it's a tedious and time-consuming job for human labor to process this task, but also that some unfairness result of the courses assignment could often cause the disharmony among teaching faculty members. Form administrative perspective, different university institutions have different courses assignment regulations; even different departments at the same university institution could have different assignment regulations. The differences depend on institution's teaching requirements or other administration considerations. Furthermore, during the process of courses assignment, once there are teachers who can not fulfill the assigned task due to certain reasons, the courses assignment result is likely to be re-scheduled for him or her, and some minor changes could result in chain reactions and cause major impact for the assignment result.

In literature, several mathematical programming models for FCAP have been presented and traditional operational research methods have been used to solve the FCAP, but due to the complexity of the problem, difficulty of mathematical formulation and its combinatorial nature, a satisfactory result is hard to obtain [14]. The inadequacy of classical methods has drawn the attention of the researchers towards non-classical techniques. Among non-classical techniques, evolutionary computation techniques are the most widely used. Evolutionary computation techniques that have been used to solve similar type of problems includ Genetic Algorithms (GA) [1][3][4], Tabu search algorithm [5][8][9], Ant Colony Optimization (ACO) [6], just to name a few. In this study, we adopt genetic algorithm to solve the FACP due to its wide range of applications and simple implementation. Genetic algorithm is proposed by Professor John H. Holland at Michigan University during the 70s [10]. Genetic algorithms are search algorithms based on the idea of natural selection. The process of evolution itself can be viewed as a search process for the adaptation of the environment. Hence, GA is a mechanism which simulates the natural selection in biology, and it is a highly parallel, self-adapted random search method. Through the evolutionary mechanism, GA gradually removes bad individual chromosomes in the population through generation after generation, preserves good chromosomes. The preserved chromosomes will go through the process of crossover and mutation operations in order to produce a better offspring. The GA runs for several generations of evolution until it achieves some predefined condition. Using genetic algorithm, we first need to design the encoding of solution for the underlying problem, this encoding individual is called chromosome and hence, a chromosome represents a candidate solution for the underlying problem. A chromosome is composed of genes. One gene may contain numerical values or symbols according to the problem nature under consideration. Different permutation of genes represents different characteristic of a chromosome. Each chromosome will be assessed in order to find its fitness for the environment, the higher the fitness value, the better the chromosome quality. During the evolutionary process, the population of chromosomes will go through three phases, namely selection, crossover and mutation. The purpose of selection operation is to decide which individual chromosome can survive to next generation; the crossover operation is a process for parent generation to produce the child generation, during this process, part of information from parent individual will be copied to the child individual; finally, the design of the mutation operation is to prevent the algorithm be trapped into local search. The process of GA can be illustrated as Figure 1.

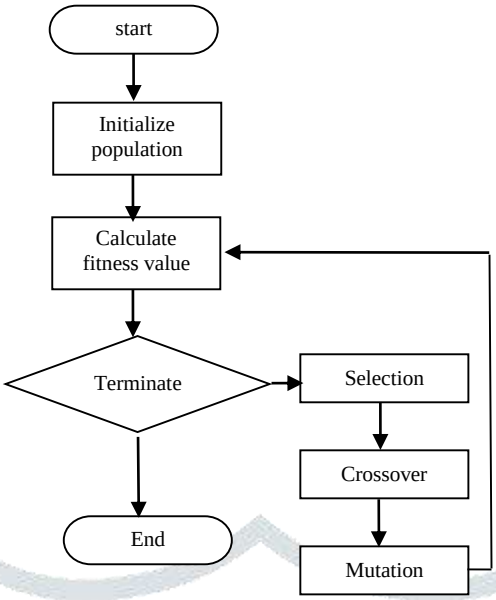


Figure 1. Genetic algorithm

The paper is structure as follows. In section 2, we describe the faculty courses assignment problem and related research approaches for solving it. In section 3, the design of the model for FCAP and the constraints under consideration for our study is presented. In section 4, we illustrate the design of the encoding scheme for chromosome, method for fitness calculation as well as the selection, crossover and mutation operations in GA. In section 5, we present a novel experiment for our design and finally make some conclusion in section 6.

II. FACULTY-COURSE ASSIGNMENT PROBLEM

Faculty-Course assignment problem (FCAP) is essentially different from university course timetabling. The FCAP concerns about how to assign courses offered by department to all available teachers subject to some constrains. In literature, several mathematical programming models of FCAP have been presented and traditional operational research (OR) methods have been used trying to solve the FCAP.

Andrew and Collins [2] proposed a linear programming model for FCAP, the model constraints ensure that the needed number of sections for each course are been taught, and the number of sections assigned to faculty member do not exceed the teaching load for that member. Some authors used integer programming (IP) formulation [12][16][17] or mixed integer goal programming model [7] to solve the FCAP. Hultberg et al. [11] tackled this problem from the viewpoint of classical transportation problem, and represented the FCAP as mixed integer program. In his model, they only considered one simple condition: how to minimize the number of different courses assigned to teachers. Hultberg research transforms the FCAP to cardinality partitioning optimization problem and shows that FCAP is NP-hard in the strong sense. Besides integer programming model, several authors tried to use different methodology to approach the FACP. Ozdemir et al. [13] uses 0-1 nonlinear multi-objective programming to solve the FCAP. The way Ozdemir used is to treat every teacher's requirements as a need-to-be satisfied objection and use them as constrains for the problem. By considering both the requirements for teacher and administrator, Ozdemir's work use Analytic Hierarchy Process (AHP) to simplified the multi-objective function as single function and solve their model efficiently. Parthiban et al. [14] also approaches FCAP using AHP and solve a model based on teacher's preference. Partovi and Arinze [15] present a knowledge-based system (KBS) approach to the FCAP. A KBS is a computer hardware and software system that mimics the decision-making processes of a human. It uses encoded "knowledge" acquired from a decision-maker or written sources that is captured and represented in several possible ways.

III. MODEL DESCRIPTION

For FCAP, we investigate the problem of how to assign courses to teachers in university institutions. For this, we need to take into account about teachers' preference, teachers' qualification, administration tasks, etc. Each requirement has its importance for the problem under consideration. Therefore, similar to solve the courses timetabling problem, we divide FCAP constraints into two categories: one category is called hard constraint (HC), the other category is called soft constraint (SC). HC is a property that the solution of the problem can not violate such as resource restriction, school regulations, or teachers assignment confliction; SC is a property that the problem needs to satisfy as much as possible. Most of the soft constraints are due to the consideration of human factor.

3.1Constraints

The soft and hard constraints we considered in this study are listed below:

Hard Constraints:

HC1 Only one teacher can be assigned to same course.

HC2 Every teacher needs to be assigned at least 3 courses, and at most 5 courses.

HC3 Part-time teacher can be assigned at most three courses.

HC4 Teacher who takes administration tasks should be assigned at most two courses.

HC5 All required courses must assign teacher.

Soft Constraints:

SC1 The area that a course belongs to should match teacher's expertise as much as possible

SC2 The courses assignment result should match teacher credit-hour requirement

SC3 The number of different courses assigned to a teacher should be as small as possible.

SC4 The assignment result should satisfy teacher's preference as much as possible

SC1 is for the purpose that qualified teacher will be assigned course that he/she is specialized; since each institution has regulations concerning how many courses can be offered by a teacher, so we need SC2 in order not to violate those rules. For SC3, this is due to that most teachers prefer to be assigned same type of courses in order to reduce the course preparation time. Finally, the SC4 says that teacher's teaching preference should be satisfied as much as possible.

For those soft constraints we mentioned above, we need to collect some required data before hand for the continuing work. Those required data include "teacher preference table", "course area table", "teacher specialty table".

Teacher preference table contains the course teaching preference for all faculty members of the department. Each teacher needs to fill in their preference for those courses they would like to teach. The scale ranges from 1 to 10 with 10 indicates the most preferable. Each teacher needs to provide at least 10 courses preference in this table.

Course area table is a course structure table which indicates the areas a course belongs to. A course can belong to one area, or belong to several areas. We need this table for the calculation of SC1 later.

Teacher specialty table contains information about teacher's qualified teaching specialty. It is filled by teachers themselves and its purpose is for the calculation of SC1. These three tables are shown below.

Table 1. Teacher preference table

Teacher / Course	T1	T2	T3	…
C1	5		10	
C2		1		
C3	2		6	
C4	2		6	

Table 2. Course area table

Course / Area	C1	C2	C3	…
Area A		v	v	
Area B		v		
Area B	v			
Area D			v	

Table 3. Teacher specialty table

	Area A	Area B	Area C	Area D
T1	v		v	
T2		v	v	v
T3	v		v	
T4		v		v

IV. DESIGN METHOD

In this study, we are trying to use genetic algorithm to solve the FCAP. In following sections, we illustrate the design for each step in GA. These steps include the encoding of chromosome, the definition of fitness function, the selection, crossover, and mutation operators that are been used in this study.

4.1Chromosome Encoding

According to genetic algorithm, we first define the encoding of chromosome for this study. One chromosome represents a solution for the problem under consideration; hence a chromosome for the FCAP represents a solution for the courses assignment result. The design of chromosome has direct impact for the effectiveness and efficiency of GA. From the literature review of timetabling problem, several procedures that use genetic algorithm often need to go through gene repair process and this can affect the efficiency of GA dramatically. For the structure of chromosome that represents a solution of courses assignment, we define it as follows: Let chromosome X represents a result of courses assignment, X consists of gene groups where n represents the number of courses given for one semester. Each gene group contains m genes where m is the number of teachers. Therefore, each gene group represents the assignment result for that course. The structure of chromosome can be illustrated in Figure 2.

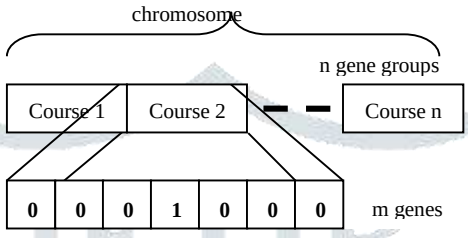


Figure 2. The encoding of chromosome

Let $C=\{C_1,C_2,\ldots,C_n\}$ represents the set of all courses, $T=\{T_1,T_2,\ldots T_m\}$ represents the collection of all teachers. Define decision variable x_{ij} as:

x_{ij} =1 if teacher T_j is assigned course C_i,

x_{ij} =0 if not; where $1\le j\le m$, $1\le i\le n$.

then, the result of courses assignment can be represented as a matrix $X_{n\times m}$. That is, the chromosome for our study can be represented by a matrix. This representation is different from the traditional ways which use an array of number or character for chromosome encoding. Beside the matrix representation of the chromosome, we also transform the teacher preference table as a preference matrix $F_{n\times m}$. Each row in preference matrix represents the preference of all teachers for that course, and each column in preference matrix represents the preference of one particular teacher with respect to all courses.

By the design of encoding of chromosome, it is clear that HC1 and HC5 hard constraints are automatically satisfied. During the initialization process, any chromosome which violates HC2, HC3 and HC4 will be discarded and regenerated by GA in order to make sure that each chromosome passes the hard constraints.

4.2 Fitness Function

The fitness function plays an important role in GA process because the responsibility of fitness function is to provide the measurement of fitness for a chromosome during GA process. By using fitness function, the more fit the chromosome does, the chance it will survive during nature selection will be enhanced. This can be translated as that the chromosome is more fit for the constraints of the problem. The fitness value is closely related to the cost in that for chromosome with larger cost, we expect that it has lower fitness value. Based on the problem we are considering, the fitness function is composed of two parts, one is called total average preference (TAP) and the other is called cost function. Let X be a chromosome, its fitness value is defined as:

$$fitness(X)=\frac{TAP(X)}{1+\cos t(X)}$$

From the definition of above mathematical formula, we can see that a courses assignment result which satisfies most teachers' preference will have a fitness value that approaches to the value 1. This is because for that chromosome, we will have $TAP(X)\to 1$ and $\cos t(X)\to 0$. In the following subsections, we state the detail design for $TAP(X)$ and $\cos t(X)$.

4.2.1 Total Average Preference (TAP)

The total average preference represents an average of preference of all teachers for a courses assignment result. The calculation of TAP is based on the multiplication of the courses assignment result and transpose of the teacher preference matrix $F_{n\times m}$. That is, let $I=\{1,\ldots,n\}$ be the collection of courses, $J=\{1,\ldots,m\}$ be the collection of teachers, and $P=\{P_1,\ldots P_n\}$ be the collection of preference where n is the number of courses, m is the number of teachers. Then, TAP is calculated as follows:

$$TAP(X)=\frac{\sum_{i\in I}\sum_{j\in J}x_{ij}p_{ji}}{\max(P)\cdot n}=\frac{tr(X_{n\times m}F^{T}_{n\times m})}{\max(P)\cdot n}$$

where $F^{T}_{n\times m}$ denotes the transpose of the preference matrix $F_{n\times m}$, $tr(A)$ denotes the trace of matrix A and $\max(P)$ denotes the maximum value of the set P. According to this definition, a perfectly satisfied courses assignment result will have value close to 1.

4.2.2 Cost Function

Cost function plays very important role in genetic algorithm in that it decides how good a chromosome is, and the GA can base on this value for the subsequence operations. On the other hand, cost function is also related to the calculation of the fitness value. For a giving chromosome X, we define its cost as follows:

$$\text{Cost}(X)=\sum_{i=1}^{3}w_i\times d_i(X)$$

where w_i is the weighted value of $d_i(X)$.

In cost function, $d_1(X)$ denotes the value of the cost when a chromosome violates SC1. SC1 says that the area a course belongs to should match teacher's expertise as much as possible. This is to ensure that the courses assignment result will really assign course to a qualified teacher in order to maximize the expertise matching value. For a giving chromosome X, the function d_1 is defined as follows:

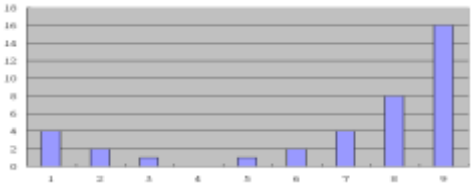
$$d_1(X)=\sum_{j\in J}\sum_{i\in I}x_{ij}(a_i-T_{ji})$$

where a_i denotes the number of expertise areas that course i required, T_{ji} denotes the number of expertise areas that teacher j have for course i. The data for a_i and T_{ji} come from table 2 and table 3. The design consideration for $d_1(X)$ is to calculate whether the assignment has assigned qualified teachers to related courses according their expertise. For the purpose of the design, we expect this value to be small in order to reduce the cost.

$d_2(X)$ denotes the value of the cost when a chromosome violates SC2 which means that the number of courses assigns to every teacher should be on an average basis as much as possible. That is, instead of some teachers been assigned more courses, we would like to ensure every teacher's teaching load should maintain as average as possible. In order to calculate the cost of this soft constraint, for each teacher, we first calculate the total teaching hours assigned to this teacher, and then the value obtained will be used as an input for the function called cost increment (CI) function. And finally, $d_2(X)$ is defined as the sum of each teacher's CI:

$$d_2(X)=\sum_{\text{i}}CI(\sum_{\text{j}}x_{ij}c_i)$$

where c_i denotes the semester hours for course $\boldsymbol{i}$, $CI(n)$ denotes the cost increment function mentioned above, $\boldsymbol{n}$ is an integer. The value of $CI(n)$ can be illustrated in Figure 3. In Figure 3, the x-axis represents the number of courses, the y-axis represents the cost increment value for corresponding number of courses. From the definition, we see that when a teacher been assigned 12 semester hours, that is, 4 courses (here, we assume that each course has 3 semester hours), $C(4)=0$; while as the semester hours assigned is 27 which means 9 courses, $C(9)=16$, and hence the cost is large in this case.



$d_3(X)$ denotes the value of the cost when a chromosome violates SC3 which means that the number of different courses assigned to a teacher should be as small as possible. The definition of $d_3(X)$ is as follows:

$$d_3(X)=\sum_{j}CI(\#(\bigcup_{i}G(x_{ij}))-[\frac{k}{m}])$$

where $G(x_{ij})$ represents the course set that the course i belongs to, $\#(S)$ denotes the number of different elements in set S, k denotes the number of different courses, m is the number of teacher, the notation $[x]$ denotes the greatest integer less than the

number x. The course set is a collection of courses with the same name, for example, the math department offers the calculus course for the whole university, but since there are so many students who need to take this course, hence, the math department will offer several sections for the same course. Table 4 illustrates an example.

Table 4. Course set example

		T1	T2	...
1	Calculus_001	v		
2	Calculus_002	v		
3	Calculus_003		v	
4	Calculus_004	v		

Although these courses are the same, but in reality it is counted as 4 courses. In order to make them different, the course is encoded with different number. Therefore, the course set for calculus is {Calculus_001,… , Calculus_004}. The calculation of $G(x_{ij})$ for this example is: for T1, even thought s/he is assigned 3 calculus courses, but since they are the same, we have $\bigcup_i G(x_{i1}) = \{Calculus\}$ and hence

$$\#\left(\bigcup_i G(x_{i1})\right) = 1$$

For other teachers, the calculation is the same, and $d_3(X)$ is the sum of this calculation for each teacher. It is evident that the value of $d_3(X)$ represents the number of different courses assigned to teachers and we expect this value the smaller the better.

4.3 Selection

For this study, we adopt the roulette wheel selection as selection operator in order to decide whether a chromosome will survive or not. In addition to this selection, during the process, the GA incorporates the elitist selection to ensure the best individual will survive to next generation.

4.4 Crossover

In this study, we use the traditional single-point crossover operator but with a little difference in traditional implementation. The classical single-point crossover randomly selects the cutting point for crossover operation, but due to the design of our chromosome for courses assignment, we randomly selection the cutting point from the courses boundary in order to avoid the repairing process which is common in other GA applications. Figure 4 illustrates the chromosome before and after crossover operation.

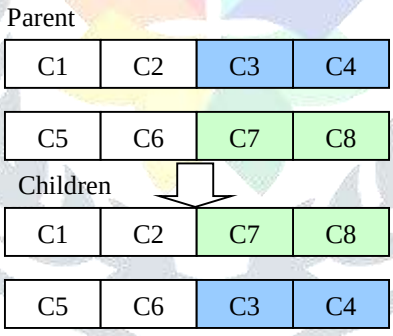


Figure 4. Before and after crossover

4.5 Mutation

The design purpose of mutation operation in GA is to avoid premature converge of the algorithm, which means to prevent the algorithm fall into local maximum or local minimum during GA process. In this study, the mutation rate is set to 0.01 and the position for mutation operation is randomly selected from the chromosome. After selecting a position, the algorithm will randomly decide whether the value in the selected position should indeed be mutated. The operation is illustrated in Figure 6.

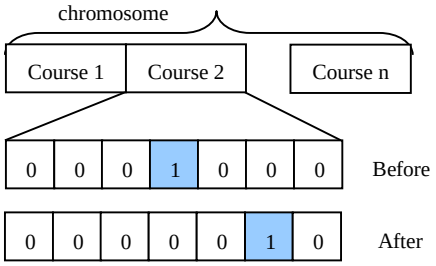


Figure 5. Before and after mutation

V. EXPERIMENT RESULTS

In this section, we present a novel simulation which we assume there is a department with 6 teachers and offering 20 courses. The generation of evolution is set to 3000 and the population size is set to 60 chromosomes. The mutation rate is set to 0.01. The experimental results are shown in Table 5, 6, 7, 8. The gray areas indicate the courses assignment result after running the genetic algorithm and we can see the courses assignment when choosing different weighting values for those controlling parameters for soft constraints.

5.1 Comparison of Different Weighting

In Table 5, the weighting of SC1 is set to 0.8 and the other two are set to 0.1. The number of violations for SC1 is 2, SC2 is 4, and SC3 is 6. The result indicates that the courses assignment result actually follows the rule that the assignment should match teachers' expertise as much as possible. In Table 6, we set the weighting of SC2 to 0.8 and the other two are set to 0.1. We can see from the result that the number of violation for SC1 is 6, SC2 is 2 and for SC3 is 6. Also, it can be observed from Table 6 that the number of courses each teacher needs to teach are almost equally distributed. This indicates that SC2 is mostly satisfied. In Table 7, we intentionally set the weighting of SC3 to 0.8 and the other two to 0.1 in order to see how well the algorithm run regarding to the SC 3. In Table 7, the result shows that the number of violations for SC1 is 6, SC2 is 6 and SC3 is 3. It is clear to see that different courses sets are assigned to the same teacher. In Table 8, the weighting for SC1~SC3 are set to 0.33. In this scenario, we wish to get a courses assignment result which is distributed equally with respect to the design of soft constraints in our study. Table 8 shows that the courses are distributed average among teachers, course sets are assigned to the same teacher, and most courses are assigned to teachers who have the expertise in the course area. According to these experiments, we conclude that the GA can effectively produce a courses assignment result which meets different requirements by changing the individual weighting of soft constraints.

5.2 Efficiency for GA

For efficiency of using genetic algorithm to solve this type of combinatorial optimization problem, SC1, SC2, SC3 are set to 0.33 as comparison basis. Figure 6 shows the experimental results by running the GA simulation 40 times and then taking the average for fitness values for each generation. For this novel experiment, the best fitness value is 0.4900705 for the case (0.33, 0.33, 0.33) and we see for Figure 6 that GA approaches 0.456124779 after 2000 generation. Figure 7 shows the error of fitness value with respect to the best value, and we see that the error from running GA is less than 0.05 when compare with the best fitness value. The computation time for running 10000 generations is less than 1 minutes, and it is far better than running a baseline exhaustive search which takes more than 50 minutes and the best result is still not yet to be found.

VI. EXPERIMENT RESULTS

In this study, we present a flexible mechanism to solve the faculty courses assignment problem using genetic algorithm. By using different weighting on soft constraints, we have shown that this prototype system can provide a courses assignment result which maintains the fairness of courses assignment, satisfying teachers' preference and matching teachers' expertise. In the future, we will continue to enhance the prototype system by considering the teaching assessment as one of the factors to incorporate into the system such that the courses assignment result will reflect teachers' teaching quality. Furthermore, based on our experiments, we have observed that when running GA, most of the time the fitness value for the best chromosome is kept constant, which indicates that it is very easy for GA to fall into local extreme. Therefore, it is worthy to investigate how to provide a more efficiency mechanism by using different crossover or mutation operator to improve this common phenomenon when using evolutionary computation for problem solving.

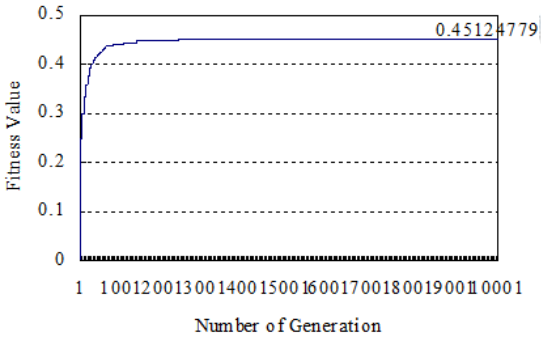


Figure 6. Fitness values for generations

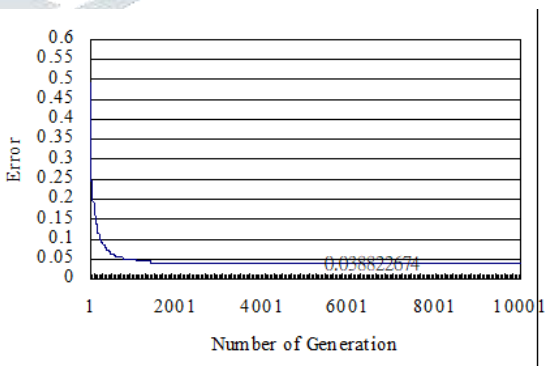


Figure 7. Error for generations

REFERENCES

[1] Abramson, D. and Abela, J. 1991 A parallel genetic algorithm for solving the school timetabling problem. Technical report, Division of Information Technology, C.S.I.R.O.

[2] Andrew, G. M. and Collins R. 1971 matching faculty to courses, College Univ. 46, 83-89.

[3] Colorni, A., Dorigo, M., and Maniezzo V. 1990 Genetic algorithms nd highly constrained problems: The time-table case. In G. Goos and J. Hartmanis, editors, Parallel Problem Solving from Nature, pp. 55-59, Springer-Verlag.

[4] Colorni, A., Dorigo, M., and Maniezzo V. 1992 A genetic algorithm to solve the timetable problem, Tech rep. 90-060 revised, Politecnico di Milano, Italy.

[5] Costa, D. 1994 A Tabu Search Algorithm for Computing an Operational Time Table, European Journal of Operational Research, 76, pp. 99-110.

[6] Dowsland, K.A., and Thompson, J.M. 2005 "Ant colony optimization for the examination scheduling problem", journal of the Operational Research Society, pp.426-438.

[7] Harwood, G. B. and Lawless, R. W. 1975 Optimizing organizational goals in assigning faculty teaching schedules, Decis. Sci. 6, 513-524.

[8] Hertz, A. 1991 Tabu Search for large scale timetabling problems, European Journal of Operational Research, 54, pp. 39-47.

[9] Hertz, A. 1992 Finding a feasible course schedule using tabu search, Discrete Applied Mathematics, 35(3), pp. 255-270.

[10] Holland, J. H. 1975 Adaptation in Natural and Artificial Systems, The University of Michigan Press.

[11] Hultberg, T. and Cardoso, D. 1997 The teacher assignment problem: A special case of the fixed charge transportation problem, European journal of operational research, vol.101, pp. 463-473.

[12] McClure, R. H. and Wells, C. E. 1984 A mathematical programming model for faculty course assignments. Decis. Sci. 15, 409-420.

[13] Ozdemir, M. S., and Gasimov, R. N. 2004 The analytic hierarchy process and multiobjective 0-1 faculty course assignment, European Journal of Operational Research, 157, pp. 398-408.

[14] Parthiban, P., Ganesh, K., Narayanan, S. 2004 Dhanalakshmi, R., Preferences Based Decision-making Model (PDM) for Faculty Course Assignment Problem, In Proceedings of Engineering Management Conference, IEEE International, pp. 1338-1341.

[15] Partovi, F. Y. and Arinze B. 1995 A Knowledge Based Approach to the Faculty-course Assignment Problem, Socio-Econ. Plann. Sci. 29, 245-256.

[16] Shih W. and Sullivan, J. A. 1977 Dynamic course scheduling for college faculty via zero-one programming, Decis. Sci. 8, 711-721.

[17] Tillet, P. I. 1975 An operations research approach to the assignment of teachers to courses, Socio-Econ. Plann. Sci. 9, 101-104.