A Study of the Impact of Different Data Densities on the Search Operator Rates in the Matrix-based EAs Model for Examination Room Assignment Problem

27641

Li-Pei, Wong[1] **A. T. Khader**[2]

[1]*Faculty of Information Science and Technology (FIST),*
Multimedia University (Melaka Campus), Jalan Ayer Keroh Lama, 75450 Bukit Beruang, Melaka, Malaysia.
Tel: +60-6-2523623, Fax: +60-6-2318840, E-mail: lpwong@mmu.edu.my

[2]*Artificial Intelligence Lab, School of Computer Sciences,*
Universiti Sains Malaysia, 11800 Minden, Pulau Pinang, Malaysia.
Tel: +60-4-6533888 x 3646, Fax: +60-4-6573335, E-mail: tajudin@cs.usm.my

Abstract

Evolutionary Algorithms (EAs) is an umbrella phrase used to illustrate a computer-based problem solving systems, which utilizes a computational model of evolutionary processes as key fundamentals in their design and implementation. In these algorithms, encoding and reproduction mechanisms are used to solve some difficult problems based on the principle of evolution – survival of the fittest. The search operators such as crossover, mutation and reproduction are applied to evolve the solutions based on certain probability values. However, it is usually very difficult to determine or estimate an optimal set of search operator rates for a problem.

This paper intends to investigate the relationship between the data density and the search operator rates for examination room assignment problem. For this, we have suggested the Matrix-based EAs Model to solve the examination room assignment problem. In this particular problem, we allocate examinations into a number of rooms for a particular slot. Thus, data density in this paper refers to the total number of candidates for all the examinations per the maximum capacity of all available rooms in a particular slot. We believe that for better performance, the probability values of search operators need to be adjusted inline with different data densities.

Keywords

Examination Room Allocation, Evolutionary Algorithms, Matrix Representation, EAs Search Operator Rates.

Introduction

Evolutionary Algorithms (EAs) are invented based on the observation of natural evolution. In these algorithms, encoding scheme (representation), selection method, reproduction mechanisms and evaluation functions are used as a robust tool to evolve the solutions for some difficult problems based on the principle of natural evolution – survival of the fittest [1]. It can also be viewed as a phenomenon of utilizing the collective learning process of a population of individuals.

In general, EAs share a common framework. Firstly, there must be a group of chromosomes as the basic element in a population. These chromosomes must be represented properly by using some of the encoding scheme such as bit string, matrix, tree etc [2, 3, 4]. Next, we need to initialize a random population of chromosomes at the initial stage. From a pool of chromosomes, parent selection is done based on chromosomes' fitness. The selected chromosomes will then perform genetic operators such as crossover (mating process), mutation or reproduction in order to search the solutions with respect to the task that needs to be solved. Along the process of searching the solutions, it is directed by some objective functions that measure the fitness or quality of the chromosomes in a population. These steps are repeated until an appropriate solution is obtained.

There are three common search operators in most of the EAs: crossover, mutation and reproduction. We usually employ a stochastic transition rules in determining which operators to be performed on the selected chromosomes. Hence, the occurrences of genetic operators depend on probability values. However, in most of the cases, it is hard to determine or estimate an optimal set of search operator rates in improving the performance.

The aim of this paper is to study the significance relationship between data density and the genetic operator rates for examination room assignment problem. Let us denote by P_C as the probability of applying crossover, P_M as the probability of applying mutation as P_R as the probability of applying reproduction. We have applied the Matrix-based EAs Model [5] in solving the examination room assignment problem. In this particular model, we employ a matrix to represent a chromosome. A chromosome is a candidate solution for the problem. This matrix consists of rows and columns. Each row represents a room whereas each column represents an examination in a particular time slot. The matrix is assigned with a set of numbers. These numbers represent the total candidates allocated in a room. Thus, data density refers to the total number of candidates for all the examinations per the maximum capacity of all available rooms in a particular slot.

Examination room assignment problem is an instance of resource allocation problem. This problem is a NP-complete problem [6, 7]. It is even more complicated when it involves a many-to-many relationship between examinations and rooms. A many-to-many relationship reflects a situation that an examination could be split into multiple rooms or a room could be shared by multiple examinations. We may acquire more information about university examination timetabling problem via [8]. They provide a comprehensive survey regarding the university examination timetabling in Britain. If we solve this problem with EAs, it has been identified that an optimal set of parameter setting is hard to find. The setting usually depends on data density. We believe that this study will aid us in determining or estimating the EAs parameter setting of P_C, P_M and P_R.

This paper is organized as follows: Firstly, it describes the examination room assignment problem in general. The following section gives a basic description about the Matrix-based EA Model. Then, it describes the hypotheses that have been identified. The subsequent section presents our experiment design towards the study of the relationship between data density and search operator rates. Our experimental findings have also been discussed. Finally, we conclude the whole paper with a short summary.

Examination Room Assignment Problem

An examination room assignment problem is a problem in which events (examinations or subjects) have to be arranged into a number of rooms/venues, subjected to a set of constraints. There are two basic entities in this problem: examination and room. Hence, our aim is to allocate examinations into a number of rooms for a particular slot. This allocation plan must also fulfill the hard [1] and soft [2] constraints listed below:

- Provide only one place to every student for a particular examination (hard constraint).
- Allocate some examinations to a specific room (hard constraint).
- Threat subjects with equivalent code as one entity (hard constraint).
- Optimize the utilization of room by decreasing the empty seat in a particular room (soft constraint).
- Minimize the cases where just a handful of students are allocated over several rooms (soft constraint).
- Minimize the cases where examinations with different duration are allocated in one room (soft constraint).
- Ensure the rooms are near to one another for the split examinations (soft constraint).

In this particular problem, two possible relationships may exist between examination and room: one-to-one relationship and many-to-many relationship. In the one-to-one relationship, a room may accommodate only one examination. Sometimes, it is difficult to accomplish this requirement since there is a possibility where an academic institution lacks of any large examination hall that is able to accommodate all the candidates. Thus, we need to split a large examination to several rooms or allow a room to be shared by several examinations. This situation is recognized as a many-to-many relationship. Considering all these factors simultaneously in producing a feasible allocation plan is a difficult task.

[1] Constraints that must not be violated at any circumstances.
[2] Objectives that are highly desirable but these may be ignored, if necessary to produce a feasible assignment timetable.

Matrix-based EAs Model

In this Matrix-based EAs Model, we adapt the algorithm that is stated in Koza [p.29, 9] (see Figure 2). Firstly, we employ a matrix to encode the chromosome in a population. This matrix consists of rows and columns. Each row corresponds to a room whereas each column corresponds an examination in a particular time slot. The matrix is assigned with the number of candidates allocated in a room. Refer to Figure 1 for a conceptual representation of a chromosome. That figure illustrates the examination with code AKW101 is split over R1 and R4. At the same time, R1 accommodates candidates that take the examination AKW101, AKW102 and MAA102.

Courses

Rooms	AKW101	AKW102	CAT102	CAS102	MAA102	HTU201
R1	70	60	0	0	30	0
R2	0	20	0	0	0	0
R3	0	20	0	0	0	60
R4	80	0	0	0	0	0
R5	0	0	150	0	0	0
R6	0	0	0	200	0	60
R7	0	0	0	0	450	60

Figure 1 – Conceptual Representation of a Chromosome

According to the Figure 2, we need to create a population with a number of chromosomes. These chromosomes are then evaluated by a fitness function. We perform a summation technique to design the fitness function and it consists of four different evaluations. Each evaluation in the fitness function corresponds to a soft constraint that needs to be optimized. This fitness function eventually gives a penalty value as the final outcome. Based on the penalty value, we select the promising chromosomes for further transformation. For this, we employ the Roulette Wheel selection. The selected chromosomes will either be applied reproduction, crossover or mutation. Only one operator is chosen at a time. In other words, $P_R + P_C + P_M = 1$. The transformed chromosomes are then inserted to the new population. We repeat these steps until the individuals in the new population reach a limit. This is a complete cycle for one generation. We usually reiterate the system until designated result is obtained or the fitness value converges to a point.

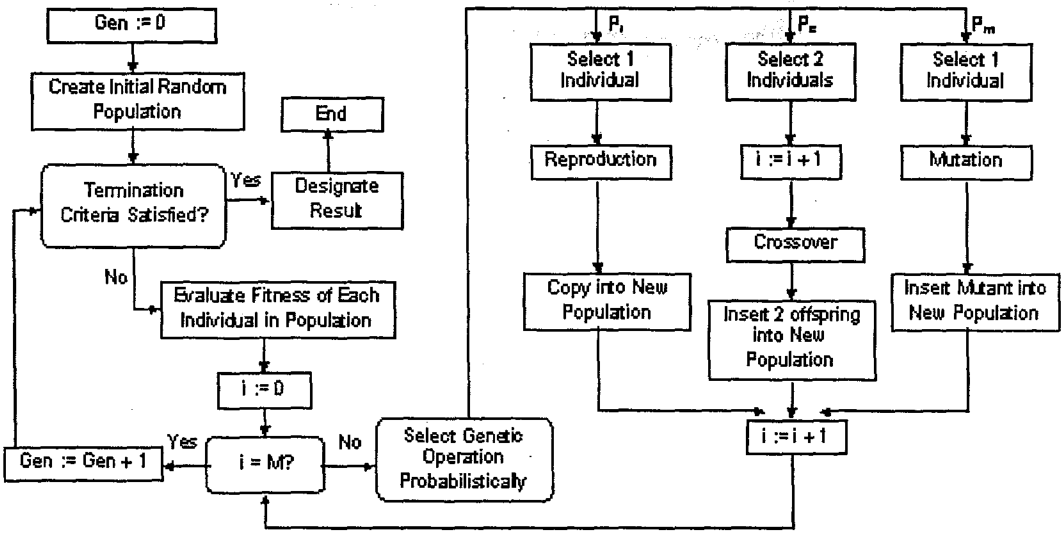

Figure 2 – EAs-based Flowchart

Hypotheses

The aim of this paper is to study the relationship between search operator rates and the data density. Before we mention about the identified hypotheses, we would like to explain more on the phrase of "data density" by giving the example as below.

Data Density

In the Matrix-based EAs Model, we assign the matrix with a set of numbers. These numbers represent the total candidates that sit for an examination in a particular room. We may have several examinations in a particular slot. For instance, we may have ten examinations to be conducted in a slot. Let's assume there are a total of 2000 candidates sitting for those ten examinations. Also, let's assume the capacity of available rooms is 2500. With these two pieces of information, we emerge the data density for that slot as total candidate per room capacity, which is 2000 / 2500 = 0.80. We believe that this data density has a great impact on the setting of search operator rate in the Matrix-based EAs Model.

Data Density Versus Search Operator Rates

Let's assume we have four types of scenarios, each of them characterizes a slot with different data densities, which is high, intermediate, low and extremely low respectively. The room capacity remains unchanged for those four slots.

For a slot with a high data density, we anticipate a high crossover rate (P_C) will produce individuals with relatively high penalty point. This indirectly will decrease the overall fitness of a population. On the other hand, we believe that a high reproduction rate (P_R) and mutation rate (P_M) will produce satisfactory results. This makes sense, as the slot with high data density will only have limited seats to perform optimization. For instance, assume that we have 2500 empty seats and there are 2400 candidates to be allocated. The data density for that slot is 0.96, which is quite high. Thus, the system will only has 2500 – 2400 = 100 empty seats to perform optimization. These similar setting (low P_C, high P_R and P_M) also applies to a slot with extremely low data density. We believe that the slot with limited empty seats will require relatively high P_R and P_M to fine-tune the results. High P_C tends to disrupt the system.

For a slot with an intermediate data density, we expected a high P_C would help in improving the overall fitness of a population. P_R and P_M should be decreased. This is because a slot with intermediate data density will has adequate empty seats to perform optimization. For instance, assume that we have 2500 empty seats and there are 1800 candidates to be allocated. The data density for that slot is 0.75, which is relatively moderate. Thus, the system will has 2500 – 1800 = 700 empty seats to perform optimization. This similar setting (low P_R and P_M, high P_C) also applies to a slot with low data density.

Experimental Design

We have proposed an experiment design in investigating the relationship between the data density and search operator rates. We would like to provide the details of the data selection and how we conduct the experiments.

Data Selection

We use the examination data at Universiti Sains Malaysia[3] (USM). At that academic year, there are 39 slots of data with different total number of candidates. We select four

[3] Examination data for Semester 1, Academic Session 2001/2002.

groups of data, named as G1, G2, G3 and G4. These four groups of data represent slots with high, intermediate, low and extremely low data density respectively. The room capacity is 2540. G1 has 2520 candidates, G2 has 1635 candidates, G3 has 325 candidates and G4 has 31 candidates. Thus, the data density for G1, G2, G3 and G4 is 0.99, 0.64, 0.13 and 0.01 respectively.

Experiment Details

Table 1 – Experiment Design Details

Category	Experiment	P_R	P_C	P_M
Category 1	Exp 1	0.00	0.50	0.50
	Exp 2	0.50	0.25	0.25
	Exp 3	0.90	0.05	0.05
Category 2	Exp 4	0.50	0.00	0.50
	Exp 5	0.25	0.50	0.25
	Exp 6	0.05	0.90	0.05
Category 3	Exp 7	0.50	0.50	0.00
	Exp 8	0.25	0.25	0.50
	Exp 9	0.05	0.05	0.90

We group the experiments into three major categories. Each category examines the behavior of search operators with different probability value. Category 1, Category 2 and Category 3 examines the reproduction, crossover and mutation respectively. There are three experiments included in each category. For instance, in Category 1:

- First experiment – the reproduction is prohibited to perform whereas the other two operators have equally remaining chance to perform.
- Second experiment – the reproduction has half the chance to perform whereas the other two operators have equally remaining chance to perform.
- Third experiment – the reproduction has highly dominated the chance to perform whereas the other two operators have equally remaining chance to perform.

Similar setting also will be applied to Category 2 and Category 3, which examine the behavior of crossover and mutation respectively.

We create a population of 100 individuals. The system is then run for 10000 generations. The details of the experimental design are shown in Table 1. Because of randomness of the EA, a single run may not guarantee the accuracy of the results. Therefore, we perform each experiment for ten times. The average value will be calculated accordingly.

Experimental Result

We will discuss our results based on two different perspectives: quality of the merit individual and performance of the system. For the first part, we will acquire the average penalty value for the best individual taken from ten experiments. This value reflects the average quality of the best individual in ten experiments. For the latter part, we will study the system performance by acquiring the average penalty value for each individual in a population. These values are then presented via a graph of type average penalty value versus generation. This graph reflects the changes in penalty value of individuals in a population. A good EAs-based system tends to show reduction in penalty value along the evolution process.

Quality of the Merit Individual

Table 2, 3 and 4 show results for Category 1, Category 2 and Category 3 respectively. Note that the results for G4 are identical for these three categories.

In Category 1 (see Table 2), which examines the reproduction behavior, a setting of relatively high P_R tends to produce relatively good results for G1 (refer to Exp 2 for G1). However, this will not work on G2 and G3 (refer to Exp 3 for G2 and G3).

Table 2 – Average Penalty Point of the Merit Individual for Category 1

Category	Exp	G1	G2	G3	G4
Category 1	Exp 1	1301.2	94257.6	609902.2	817301.0
	Exp 2	1064.0	93017.8	609205.0	817301.0
	Exp 3	1276.6	101924.0	614969.0	817301.0

Table 3 – Average Penalty Point of the Merit Individual for Category 2

Category	Exp	G1	G2	G3	G4
Category 2	Exp 4	1464.8	150582.8	617727.0	817301.0
	Exp 5	1261.4	92609.6	609534.8	817301.0
	Exp 6	1466.0	92328.2	609926.6	817301.0

Table 3 – Average Penalty Point of the Merit Individual for Category 3

Category	Exp	G1	G2	G3	G4
Category 3	Exp 7	1468.8	92436.2	609778.2	817301.0
	Exp 8	1241.6	98597.6	609626.2	817301.0
	Exp 9	1212.8	116510.4	619336.8	817301.0

In Category 2 (see Table 3), which examines the crossover behavior, a setting of relatively high P_C tends not to produce relatively good results for G1 (refer to Exp 6 for G1). However, this works on G2 and G3 (refer to Exp 6 for G2 and Exp 5 for G3).

The behavior of mutation is similar to the behavior of reproduction. A setting of relatively high P_M tends to produce relatively good results for G1. However, this will not work on G2 and G3 (refer to Exp 9 for G1, G2 and G3 in Table 4).

Performance of the System

This section describes the system performance for each category.

Figure 3, 4, 5 and 6 illustrate the impact of reproduction (Category 1) on the system performance for G1, G2, G3 and G4 respectively. Figure 3 and 4 show similar characteristic, both graphs show that Exp 1 has the highest average penalty value followed by Exp 2 and Exp 3. Thus, a setting of high P_R tends to improve the average fitness of a population over generations for G1 and G4. Notice also in Figure 3, the graph for Exp 1 and Exp 2 show major fluctuation along the EA process when the P_R is low. On the other hand, Figure 4 and 5 show reduction in Exp 1, Exp 2 and Exp 3. The reduction will then stop and show no more changes after some times. Exp 2 shows relatively good result as it stagnates faster than Exp 1 and Exp 3. Thus, a relatively low P_R works well in G2 and G3.

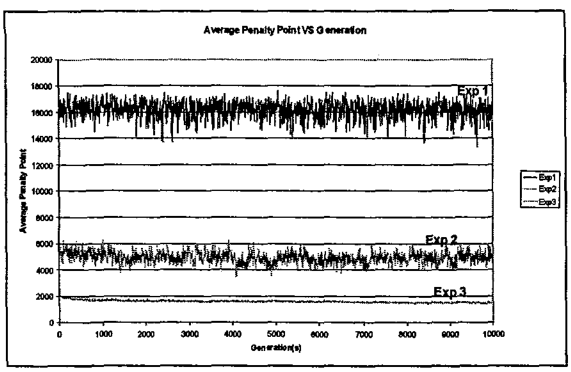

Figure 3 – Performance Graph for G1 (Category 1)

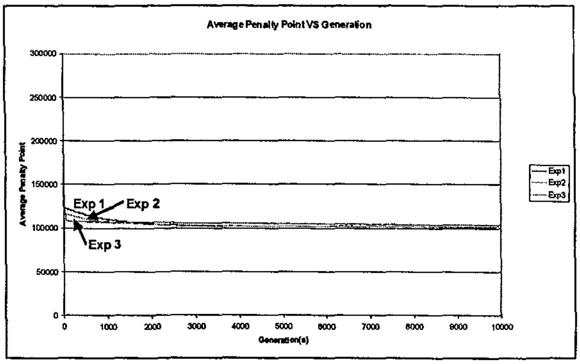

Figure 4 – Performance Graph for G2 (Category 1)

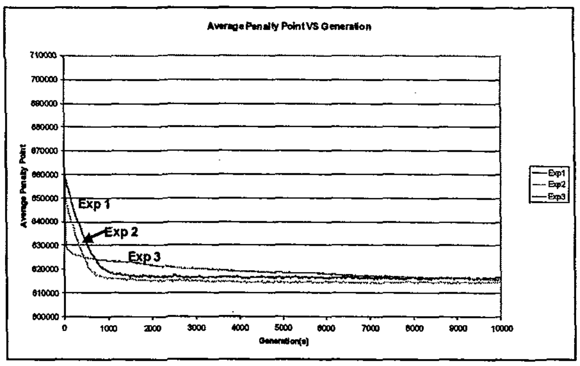

Figure 5 – Performance Graph for G3 (Category 1)

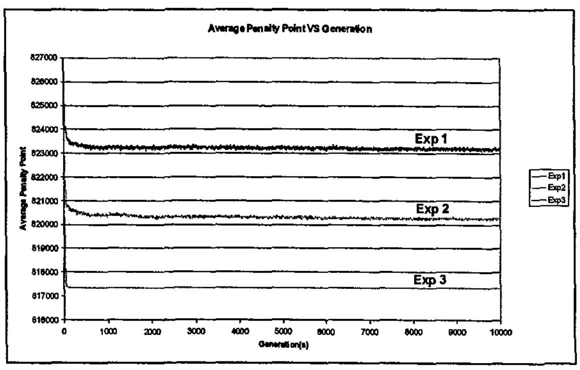

Figure 6 – Performance Graph for G4 (Category 1)

Figure 7, 8, 9 and 10 illustrate the impact of crossover (Category 2) on the system performance for G1, G2, G3 and G4 respectively. Figure 7 and 10 show that Exp 6 has the highest average penalty value followed by Exp 4 and Exp 5. Thus, a setting of high P_C tends not to work well in G1 and G4. Notice also in Figure 7 and 10, the graph for

Exp 5 and Exp 6 fluctuate along the EA process when the P_C is high. On the other hand, Figure 8 and 9 show reduction in Exp 4, Exp 5 and Exp 6. The reduction will then stop after some times. In Figure 8, Exp 6 shows relatively better result than Exp 4 and Exp 5. In Figure 9, graph for Exp 5 overlaps with Exp 6. Both of them show relatively better result than Exp 4. Thus, a high P_C tends to works well in G2 and G3.

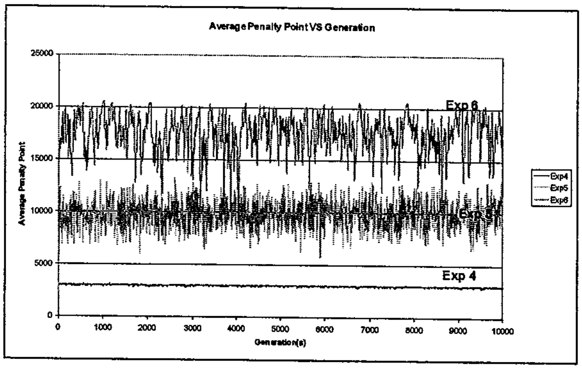

Figure 7 – Performance Graph for G1 (Category 2)

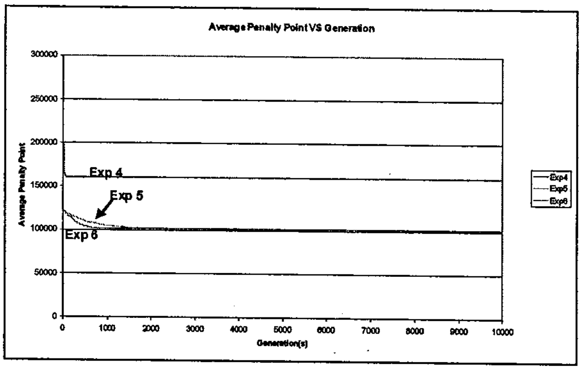

Figure 8 – Performance Graph for G2 (Category 2)

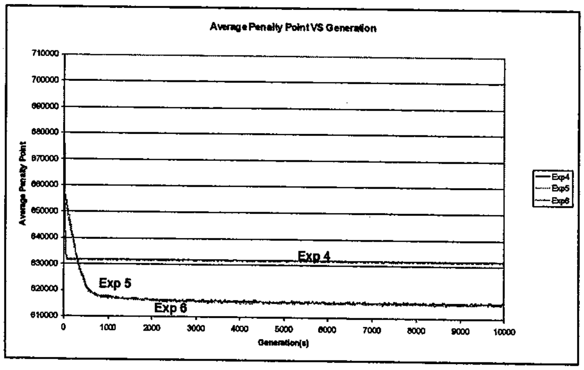

Figure 9 – Performance Graph for G3 (Category 2)

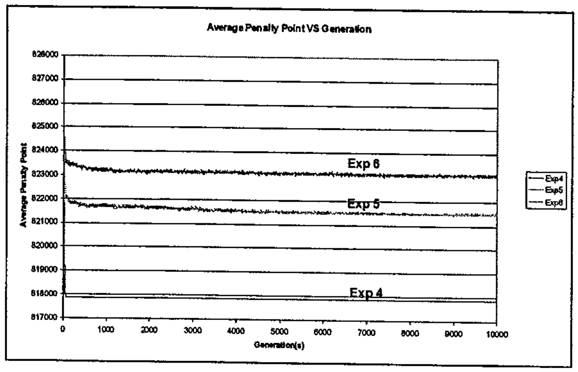

Figure 10 – Performance Graph for G4(Category 2)

Figure 11, 12, 13 and 14 illustrate the impact of mutation (Category 3) on the system performance in G1, G2, G3 and G4 respectively. In Figure 11, Exp 7 fluctuates along the EA process. It also shows that Exp 9 has the lowest average penalty value. In Figure 14, Exp 9 has the highest average penalty value followed by Exp 7 and Exp 8. Thus, a relatively high P_M tends to produce relatively better result in G1 and G4. On the other hands, in Figure 12 and 13, both graphs show Exp 9 has the highest average penalty value followed by Exp 8 ad Exp 7. Note also the Exp 9 in Figure 12 actually show increment in the graph. Thus, a relatively low P_M works well in G2 and G3.

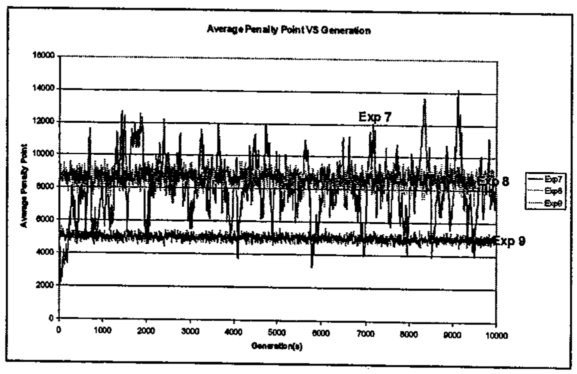

Figure 11 – Performance Graph for G1 (Category 3)

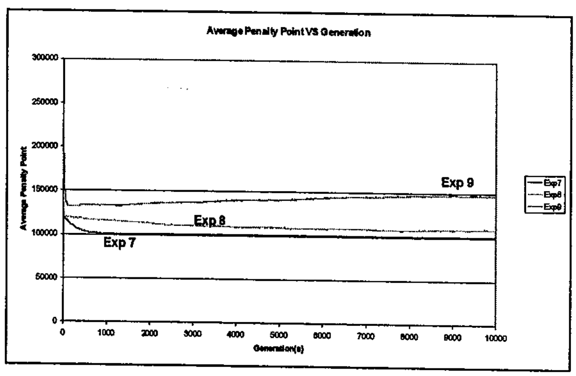

Figure 12 – Performance Graph for G2 (Category 3)

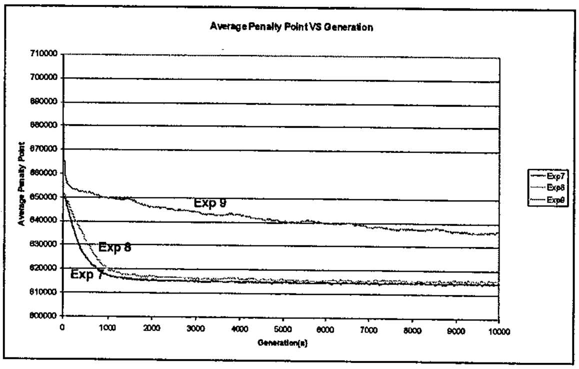

Figure 13 – Performance Graph for G3 (Category 3)

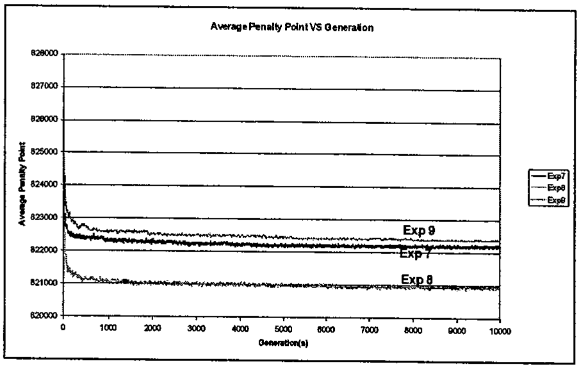

Figure 14 – Performance Graph for G4 (Category 3)

Conclusion

A study of the impact of different data densities on the search operator rates in the Matrix-based EAs Model in examination room assignment problem has been presented in this paper. We have conducted a series of experiments in study the behavior of reproduction, crossover and mutation operator with different data densities. Results are discussed from two different perspectives: quality of the merit individual and performance of the EAs-based system.

For G1 and G4 that show a high and extremely low data density respectively, a setting of relatively high P_R and P_M tends to produce better results. This is because both groups represent an extreme state: G1 has limited empty seats whereas G4 has too many empty seats to perform optimization. The system tends to start with a relatively better solution at the initial stage. Therefore, we need reproduction and mutation to fine-tune the results. The crossover operator tends to disrupt the process in G1 and G4.

For G2 and G3 that show an intermediate and low data density respectively, a setting of high P_C tends to produce better results. This is because G2 and G3 have appropriate empty seats and crossover operator will employ those empty seats to perform optimization in getting a better solution. A relatively high P_R and P_M tend not to work well in G2 and G3.

References

[1] Davis, L. 1991. *HANDBOOK OF GENETIC ALGORITHMS.* Chichester, New York: Van Nostrand Reinhold.

[2] Langdon, W. B. 1995. Genetic Programming and Data Structures. Ph.D. Thesis. Department of Computer Science, University College London.

[3] Luan, F., and Yao, X. 1996. Solving Real World Lecture Room Assignment Problems by Genetic Algorithms. *Complexity International: An Electronic Journal of Complex Systems Research*, vol. 3.

[4] Wallet, B. C., Marchette, D. J., and Solka, J. L. 1996. A Matrix Representation for Genetic Algorithms. In Proceedings of Automatic Object Recogntion IV of SPIE Aerosense, 1996 (Sadjadi,F.A., St.Anthony., ed.). 206 – 214. Orlando, FL, USA.

[5] Wong, Li-Pei., and Khader, A.T. 2002. Solving Examination Room Allocation Problem by Evolutionary Algorithm. In Proceedings of the International Conference on Operations Research for Development. Chennai, India.

[6] Garey, M. R., and Johnson D. S. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* San Francisco: W. H. Freeman Co.

[7] French, S. 1982. *Sequencing and Scheduling: An Introduction to the Mathematics of the Job-Shop.* Chichester, England: Ellis Horwood Ltd.

[8] Burke, E. K., Elliman, D. G., Ford, P., and Weare, R. F. 1995. Examination Timetabling in British Universities - A Survey. In PATAT '95 Proceedings of the 1st International Conference on the Practice and Theory of Automated Timetabling (Burke, E. K., Corne, D., Paechter, B. and Ross, P., ed.). 423 – 434. Napier University, Edinburgh, UK.

[9] Koza, J. R. 1992. *Genetic Programming On the Programming of Computers by Means of Natural Selection.* Cambridge, Massachusettes, London, England: The MIT Press.