Applying genetic algorithm to university classroom arrangement problem

Bei Wang1*, Yuejie Geng2 and Zhigang Zhang1

1Department of math and physics, University of Science & Technology Beijing, Beijing, 100083, China
2Academic Affairs Office, University of Science & Technology Beijing, Beijing, 100083, China
*Corresponding author’s email: 18811120586@163.com

Abstract. University classroom arrangement problem is considered as a sub-question of the university timetabling problem, which has been proved to be an NP-hard problem in combinatorial optimization field. In this paper, we firstly analyzed the relevant factors and set up a model of the problem. Then the genetic algorithm was applied to optimize the arrangement of classroom considering the moving distance of students between courses, which was caused by room changing. The experimental results have shown that the genetic algorithm could generate feasible solutions and optimize the classroom arrangement.

1. Introduction

The classroom arrangement problem is a sub-problem of the timetable schedule problem, which has been proved to be an NP-hard problem[1]. In 1963, CC Gotlieb proposed a mathematical model of the course arrangement problem firstly[2], which marked the beginning of scientific research in this field. The classroom arrangement is based on the determination of the time and class of the course. In order to optimize the classroom arrangement, Matias. J. B proposed a hybrid genetic algorithm for course arrangement and teaching workload management[3]. Shi Hui took different school districts into account and used the genetic algorithm to optimize the schedule arrangement[4]. Cao Ce-jun described the timetabling problem by a triplet as a parallel machine arrangement problem with machine eligibility restrictions[5]. However, in general, there are few studies on the university classroom arrangement problem, especially by using intelligent algorithms.

The genetic algorithm (GA) is an adaptive heuristic search method based on population genetics [6]. It was firstly proposed by John Holland of the University of Michigan in 1969 and was summarized by De Jong and Goldberg[7,8]. The genetic algorithm simulates the natural selection process when finding an exact or approximate solution. It converts the solution of the problem into a chromosome by coding, and all individuals in each generation form a population. The solution process starts with the initial population and the quality of individuals in each generation is evaluated by fitness value. Individuals with high fitness value are more likely to be selected. The selected individuals produce offspring through crossover and mutation operations. After successive iterations, the population eventually converges to the individual with the highest fitness value, which is considered as the best solution.

Compared with the traditional exact algorithm, the genetic algorithm is likely to gain high-quality approximation within a limited time, especially on large-scale problems where the exact algorithm fails to take effect. As a global search heuristic, the genetic algorithm processes the whole solution space...
simultaneously.
In this paper, the genetic algorithm was applied to optimize the classroom arrangement of two consecutive classes. On the basis of considering the room capacity, the moving distance of students between classes could be reduced by arranging the classrooms reasonably.

2. Problem description
The classroom arrangement problem to be treated in this paper is aimed at reducing the moving distance generated by room changes between courses. The time collection can be divided into timeslots, and each timeslot consists of 90 minutes. Timeslot, course, class, and classroom are four entities considered in this problem.

2.1. Problem definition
The classroom arrangement problem is formulated as follows:

\[ \text{Problem} = \{T, S, D, R, C\} \]

Where \( T = \{t_1, t_2, \ldots, t_n\} \) is a set of time slots, \( S = \{s_1, s_2, \ldots, s_m\} \) is a set of courses, \( D = \{d_1, d_2, \ldots, d_c\} \) is a set of classes. Also, \( R = \{r_1, r_2, \ldots, r_k\} \) is a set of classrooms, where \( r_i = (n_i, l_i) \), and \( n_i, l_i \) denote the number and floor of the classroom \( r_i \) respectively, \( C = \{c_1, c_2, \ldots, c\} \) is a set of constraints.

The classroom arrangement problem considered in this paper is carried out under the condition that the timeslot of the course, class number, and classroom capacity has been arranged previously. An example of timetable schedule and classroom capacity are shown in Table 1 and Table 2.

| Course | Timeslot | Class       |
|--------|----------|-------------|
| \( s_1 \) | \( t_1 \) | \( d_1, d_3, d_8 \) |
| \( s_2 \) | \( t_1 \) | \( d_4, d_6, d_7, d_{11} \) |
| \( s_3 \) | \( t_3 \) | \( d_{12} \) |
| ...    | ...      | ...         |

Table 1 Example of initial timetable schedule

| Classroom | \( r_1 \) | \( r_2 \) | \( r_3 \) | \( r_4 \) | \( r_5 \) | ... |
|-----------|----------|----------|----------|----------|----------|-----|
| Capacity  | \( x_1 \) | \( x_2 \) | \( x_3 \) | \( x_4 \) | \( x_5 \) | ... |

2.2. Constraints
Constraints can be divided into hard constraints and soft constraints. Hard constraints cannot be violated physically so that solutions become feasible and soft constraints are those that should be satisfied as much as possible to approximate the optimal solution.

Hard constraints:
\( s_1 \): A classroom can only have one course assigned at the same timeslot.
\( s_2 \): The capacity of the classroom is compatible with the number of students.
\( s_3 \): Students can’t be arranged in different courses and classrooms during the same timeslot.

Soft constraints:
\( s_4 \): Under the condition of satisfying the classroom capacity, select the classroom with smaller capacity firstly.

2.3. Assumption:
The model established is based on the following assumptions:
\( a_1 \): The number of students in each class is the same, and moving distance is calculated by the total
number of floors changing.

\( \alpha_2 \): For classes that do not have courses, the floor is marked as 0.

3. Genetic algorithm

In order to simplify the problem, only two time periods are considered, \( T = \{ t_1, t_2 \} \). The parameter of GA is shown in Table 3.

| Table 3 Symbolic representation in GA |
|---------------------------------------|
| Parameter                | Symbol |
| Population size          | N      |
| Iteration                | \( \omega \) |
| Gap                      | \( \alpha \) |
| Crossover probability    | \( P_c \) |
| Mutation probability     | \( P_m \) |

3.1. Population initialization

A chromosome is composed of two parts which represent the timeslot \( t_2, t_2 \) respectively and the genes represent classrooms for the corresponding course. The structure is shown in Figure 1.

- Figure 1 Chromosome representation

Also, there is an array \( P \) with the same length of the chromosome, which records the course related to genes. The structure is shown in Figure 2.

- Figure 2 Array \( P \) related to the chromosome

3.2. Fitness function

Fitness function is designed to evaluate the quality of the solution. In this problem, it’s defined as:

\[
 f_i = \frac{1}{\sum_{i=1}^{d} |l_{i,j} - l_{i,j}'|}
\]

where \( l_{i,j}, l_{i,j}' \) denotes the floor of class \( i \) in timeslot T1 and T2, respectively.

3.3. Selection

Three selection methods are widely used: Tournament Selection, Proportional Roulette Wheel Selection and Rank-based Roulette Wheel Selection. Proportional Roulette Wheel Selection is applied to this method. The parent is selected with the probability \( P_i \), which is proportional to its fitness value, so the individuals with high fitness value are more likely to be selected. Let \( f_{a_1}, f_{a_2}, \ldots, f_{a_c} \), denote the fitness value of individual \( d_1, d_2, \ldots, d_c \), and the selection probability for individual \( d_i \) is define as,

\[
 P_i = \frac{f_i}{\sum_{i=1}^{d} f_i}
\]

3.4. Genetic Operation

3.4.1. Crossover. There are many types of crossover such as single-point crossover, multi-point
crossover, shuffle crossover, uniform crossover and so on. Two-point crossover is applied with the probability of \( P_c \), and it can be divided into the following three cases:

a) After crossover, all the classrooms are not occupied by more than one course and meet the capacity requirements at the same time. A simple example is shown below.

| Parent1 | \( r_1 \) | \( r_2 \) | \( r_3 \) | \( r_4 \) | \( r_5 \) |
|---------|---------|---------|---------|---------|---------|
| Parent2 | \( r_6 \) | \( r_7 \) | \( r_8 \) | \( r_9 \) | \( r_{11} \) |

| Offspring1 | \( r_6 \) | \( r_7 \) | \( r_8 \) | \( r_4 \) | \( r_5 \) |
| Offspring2 | \( r_1 \) | \( r_2 \) | \( r_3 \) | \( r_9 \) | \( r_{11} \) |

Figure 3 Crossover without repeating

b) After crossover, there exists a classroom occupied by two courses. Arrange one of the courses in the original classroom before exchange

| Parent1 | \( r_1 \) | \( r_2 \) | \( r_3 \) | \( r_4 \) | \( r_5 \) |
|---------|---------|---------|---------|---------|---------|
| Parent2 | \( r_5 \) | \( r_6 \) | \( r_7 \) | \( r_8 \) | \( r_9 \) |

| Offspring1 | \( r_6 \) | \( r_7 \) | \( r_4 \) | \( r_5 \) |
| Offspring2 | \( r_1 \) | \( r_2 \) | \( r_3 \) | \( r_9 \) |

| Offspring1 | \( r_6 \) | \( r_7 \) | \( r_4 \) | \( r_5 \) |
| Offspring2 | \( r_1 \) | \( r_2 \) | \( r_3 \) | \( r_8 \) | \( r_9 \) |

Figure 4 Crossover with repeating

c) After crossover, if the classroom capacity does not meet the requirements, choose another room from the empty rooms.

3.4.2 Mutation. The single-point mutation is used, and the mutation point is randomly selected from the whole population with the probability of \( P_m \). The gene of \( r_i \) will transform to another classroom \( r'_i \) which is selected randomly from the empty classrooms.

| Parent | \( r_1 \) | \( r_2 \) | \( r_3 \) | \( r_4 \) | \( r_5 \) |
|--------|---------|---------|---------|---------|---------|
| Offspring | \( r_1 \) | \( r_2 \) | \( r_6 \) | \( r_4 \) | \( r_5 \) |

Figure 5 Single-point mutation

4. Experiment

Matlab language is used to implement the genetic algorithm and determine the best parameter value. The performance of GA with different parameters are compared from two aspects: the results’ quality and running time. The initial course information is presented in Table 4

| Item | Value |
|------|-------|
| No. of time slots(T) | 2 |
| No. of courses(S) | 100 |
| No. of floors (F) | 8 |
| No. of classrooms(R) | 64 |
| No. of classes(D) | 240 |

Table 4 Description of the problem instance

To the determination of the best value of \( N, P_c, P_m, \) taking \( N=50, 100, 150, P_c=0.6, 0.7, 0.8, 0.9 \) and \( P_m = 0.02, 0.05, 0.1, 0.2 \) respectively when \( \alpha=0.9 \) and \( \omega=800 \). Each group of the experiments was performed 20 times to examine the relationship between these three parameters. The results are shown in Table 5-7

| Task | \( P_m=0.02 \) | \( P_m=0.05 \) | \( P_m=0.1 \) | \( P_m=0.2 \) | \( P_m=0.02 \) | \( P_m=0.05 \) | \( P_m=0.1 \) | \( P_m=0.2 \) |
|------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|
| Pc=0.6 | 482 | 460 | 444 | 441 | 6.85 | 7.20 | 7.61 | 7.73 |
| Pc=0.7 | 475 | 452 | 443 | 434 | 8.00 | 8.15 | 8.65 | 8.96 |
| Pc=0.8 | 475 | 455 | 441 | 437 | 8.01 | 8.43 | 8.65 | 9.02 |
| Pc=0.9 | 470 | 449 | 444 | 437 | 9.36 | 9.81 | 9.78 | 10.54 |
Table 6 Average distance and running time (in seconds) when N=100

| Task | Average distance | Average running time |
|------|-----------------|----------------------|
|      | Pm=0.02 | Pm=0.05 | Pm=0.1 | Pm=0.2 | Pm=0.02 | Pm=0.05 | Pm=0.1 | Pm=0.2 |
| Pc=0.6 | 456 | 440 | 431 | 425 | 12.96 | 13.01 | 15.99 | 14.57 |
| Pc=0.7 | 448 | 439 | 432 | 430 | 12.98 | 13.50 | 14.31 | 14.96 |
| Pc=0.8 | 451 | 433 | 426 | 424 | 13.09 | 14.24 | 14.93 | 15.25 |
| Pc=0.9 | 447 | 433 | 423 | 430 | 16.29 | 16.74 | 17.79 | 18.73 |

Table 7 Average distance and running time (in seconds) when N=150

| Task | Average distance | Average running time |
|------|-----------------|----------------------|
|      | Pm=0.02 | Pm=0.05 | Pm=0.1 | Pm=0.2 | Pm=0.02 | Pm=0.05 | Pm=0.1 | Pm=0.2 |
| Pc=0.6 | 444 | 428 | 424 | 423 | 22.94 | 23.29 | 23.52 | 24.33 |
| Pc=0.7 | 446 | 428 | 425 | 423 | 24.61 | 24.28 | 24.05 | 24.88 |
| Pc=0.8 | 439 | 424 | 421 | 415 | 23.04 | 21.49 | 21.45 | 23.30 |
| Pc=0.9 | 438 | 426 | 418 | 422 | 25.53 | 25.66 | 27.27 | 23.86 |

By comparing the results of different parameters, the running time required for the program is roughly proportional to the population size N. As the population size increases, the running time becomes longer, and the result is better. When N=50, the population size is small, resulting in the congenital loss of effective alleles, and the probability of obtaining excellent individuals is smaller; when the population size reaches N=100, the result is significantly improved. If N=150 is adopted, the running time becomes longer while the improvement of the results is limited. Therefore, it is more appropriate to choose the population size of N=100.

In addition, the crossover probability is positively correlated with the convergence rate when the population size is fixed. But the excessive crossover probability easily destroys the existing excellent genes and may cause the loss of the optimal individual. Moreover, the probability of mutation reflects, to some extent, the possibility of producing excellent genes. However, when the probability is too large, it is easy to destroy the existing favorable mode, and the efficiency is low. The running time increases with the growth of the crossover and mutation probability slightly.

Based on the analysis above, the parameter values are determined as follows:

Table 8 Algorithmic parameter

| Parameter               | Value |
|------------------------|-------|
| Population size (N)    | 100   |
| Iteration (ω)          | 800   |
| Gap (α)                | 0.9   |
| Crossover probability (Pc) | 0.8   |
| Mutation probability (Pm) | 0.1   |
5. Results

Figure 6 Experimental results

As shown in Figure 6 and Table 9, moving distance significantly reduces as the number of iterations increases. Through the optimization of the classroom arrangement, the total number of moving floors in the class decreased to 412, 59.6% of the original number. After 736 generations, it can be considered that the population no longer converge, moving distance remains unchanged, and the local optimal solution of the problem is obtained.

Table 9 Experimental results (20 times)

| Item                      | Value |
|---------------------------|-------|
| Shortest distance         | 412   |
| Average distance          | 425.7 |
| Average running time (in seconds) | 14.3 |

However, it can be seen from Figure 6 that the genetic algorithm is good at global search, while poor in local search. It takes 290 iterations to reach 457, 90% of the total decline, but take 139 iterations to complete the remaining 10%.

6. Conclusion and future work

In this paper, a model of the university classroom arrangement problem is established, and then the genetic algorithm is designed based on the characteristics of the problem. The experimental results verify the feasibility and effectiveness of the genetic algorithm proposed. However, only two timeslots and total moving distance are involved in the paper. The moving distance between every two floors hasn’t be considered, which will be a research direction in the future.

Acknowledgments

This work is supported by the Student Research Training Program (SRTP) in University of Science & Technology Beijing, which is gratefully acknowledged.

References

[1] Even, S., Itai, A., Shamir, A. (1975) On the complexity of time table and multi-commodity flow problems. In:16th Annual Symposium on Foundations of Computer Science. The USA. pp. 184-193

[2] Gotlieb, C. (1962) The construction of class-teacher time-tables. In: Communications of the ACM. New York. pp. 312-313

[3] Matias, J. B., Fajardo, A. C., Medina, R. P. (2018) A Hybrid Genetic Algorithm for Course Arrangement and Teaching Workload Management. In:2018 IEEE 10th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM). Baguio. pp. 1-6

[4] Hui, S., Xiao-hong, P., Zhi-hong, W. (2010) Genetic algorithm of genetic-couple cross for curriculum arrangement in multi-campus. Computer Engineering and Applications 46: 240-243.
[5] Ce-jun, C., Qin, Y., Hong-yan, L., Ling-ling, Y. (2012) Genetic Algorithm for the Timetabling Problem in Universities. Industrial Engineering Journal 15: 130-135.

[6] Kumar, M., Husian, M., Uperti, N., Gupta, D. (2010) Genetic algorithm: Review and application. International Journal of Information Technology Knowledge Management 2: 451-454.

[7] Goldberg, D. E., Holland, J. H. J. M. l. (1988) Genetic algorithms and machine learning. 3: 95-99.

[8] Holland, J. H. (1992) Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. MIT press.