A Co-evolving Timeslot/Room Assignment Genetic Algorithm Technique for University Timetabling

Hiroaki Ueda, Daisuke Ouchi, Kenichi Takahashi, and Tetsuhiro Miyahara

Faculty of Information Sciences, Hiroshima City University
Hiroshima 731-3194 Japan
{ueda, ouchi, takahasi, miyahara}@rea.its.hiroshima-cu.ac.jp

Abstract. We present a two-phase genetic algorithm (TGA) to solve timetabling problems for universities.[1] Here, we use two kinds of populations. The first population is related to class scheduling, and the second one is related to room allocation. These populations are evolved independently, and a cost value of each individual is calculated. Then several individuals with the lowest costs are selected from each population, and these individuals are combined in order to calculate the fitness values. To evaluate the performance of TGA, we apply TGA to several timetabling problems and compare results obtained by TGA with those obtained by the simple GA (SGA). For the timetabling problem based on the curriculum of the Faculty of Information Sciences at Hiroshima City University, TGA finds a solution that satisfies all constraints, but SGA cannot find a feasible solution. From the results for problems generated by an automatic timetabling problem generator, we show that TGA obtains a better solution than the simple GA when the room utilization ratio is high.

1 Introduction

It is well known that the timetabling problem is difficult to solve, since the problem has a huge search space and is highly constrained. Thus, various optimization methods such as simulated annealing techniques, tabu search techniques, and genetic algorithm techniques have been applied to find feasible timetables [1], [2], [3], [4], [5], [6], [7]. In this paper, we present genetic algorithms to obtain a solution to the timetabling problem for universities.

Here, we define a timetabling problem as finding both a class schedule and room allocation which do not conflict with constraints given by users. According to the definition, the main tasks to solve a timetabling problem are class scheduling and room allocation. Since these tasks are closely related, it is hard to complete these tasks separately. However, it seems inefficient to complete these tasks simultaneously. As one method to solve this difficulty, we present a

[1] This research is partly supported by Research Grant from Hiroshima City University.

E. Burke and W. Erben (Eds.): PATAT 2000, LNCS 2079, pp. 48–63, 2001.

technique that uses GA in two phases. We call this method TGA. TGA consists of two evolution phases and uses two types of genotypes. In the evolution phases, the GA-based method performs evolutionary operations for two types of populations. Next, two types of populations are combined to calculate the fitness values, and the values are assigned to the populations. Through computer simulations using the curriculum of Hiroshima City University, we show that TGA finds a feasible solution. Moreover, we apply TGA and the simple GA to timetabling problems which are generated by an automatic timetabling problem generator, and we compare the performance of TGA with that of the simple GA.

The rest of this paper is organized as follows. In the next section, we define the timetabling problem which we focus on in our study. In Section 3, we present TGA and some experimental results are shown in Section 4.

2 Timetabling Problem

Here, we focus on the timetabling problem for Japanese universities and define the timetabling problem as finding both a class schedule and room allocation. Thus, not only assumptions for class scheduling but also assumptions for room allocation exist. The following is a list of assumptions based on a typical timetabling problem for Japanese universities.

(A1) A class consists of a number of students in the same year grade in a department.
(A2) A subject is taught once a week.
(A3) Multiple-class subjects are allowed.
(A4) Multiple-period subjects are allowed.
(A5) Subjects taught by multiple-teacher are allowed.
(A6) There are several kinds and sizes of rooms.

Table 1 shows an example of a timetable. Multiple-class subjects such as Subject 1 must be allocated to the same timeslot. Subjects 2, 4 and 6 are examples of multiple-period subjects, and they use consecutive periods. Subjects 4 and 6 are examples of subjects taught by multiple-teacher. In order to obtain a solution that does not conflict with these assumptions, some constraints must be considered. We call these constraints basic constraints. A list of basic constraints that we use is the following.

C1) No teachers appear more than once in a period.
C2) Multiple-class subjects must be allocated to the same periods.
C3) Multiple-period subjects must use consecutive periods.
C4) A room whose size and kind are suitable for a subject should be allocated to the subject.
C5) No rooms appear more than once in a period.

Since we consider not only class scheduling but also room allocation, we deal with three kinds of basic constraints: constraints for class scheduling, room

Table 1. An example of a university timetable

Class	Period 1	Period 2	Period 3	Period 4	Period 5
A1		**Subject-1** **Staff-1** **Medium-1**	Subject-4 All associate professor of Dept. A Computer_room-1		
B-1	Subject2 Staff-2 Computer_room-1		Subject-5 Staff-4 Small-4		
C-1		**Subject-1** **Staff-1** **Medium-1**	Subject-6 Staff-5 and Staff-6 Small-2		
D-1	Subject-3 Staff-3 Small-1	Subject-7 Staff-7 Small-4	Subject-8 Staff-3 Small-1		

allocation and room clashing. In the above list, C1, C2 and C3 are constraints for class scheduling. C4 and C5 are constraints for room allocation and room clashing, respectively.

As well as basic constraints, we consider additional constraints. These constraints are considered as requests of professors. A list of additional constraints made in this study is the following.

C6) Some teachers should not appear at some periods.
C7) Some rooms cannot be used at particular periods.
C8) For some classes, no subjects appear at particular periods.
C9) Some subjects should be taught immediately after a particular subject is taught.

C6 is a constraint for the professor meeting or the department meeting. In Japanese universities, there are some subjects that are taught by part-time teachers, and periods for these subjects tend to be fixed. In order to reserve rooms and periods for these subjects, we use C7 and C8. Some subject pairs should be treated as a multiple-period subject. C9 is a constraint for these subjects. Similar to basic constraints, C6, C8 and C9 are constraints for class scheduling, and C7 is a constraint for room allocation. To find a timetable (a class schedule and room allocation) that satisfies all of the constraints mentioned above is the subject of our timetabling problem.

3 Genetic Algorithms

3.1 The Simple GA

To compare the performance of TGA with that of the most fundamental GA, we apply the simple GA (SGA) to the timetabling problem. The genotype is

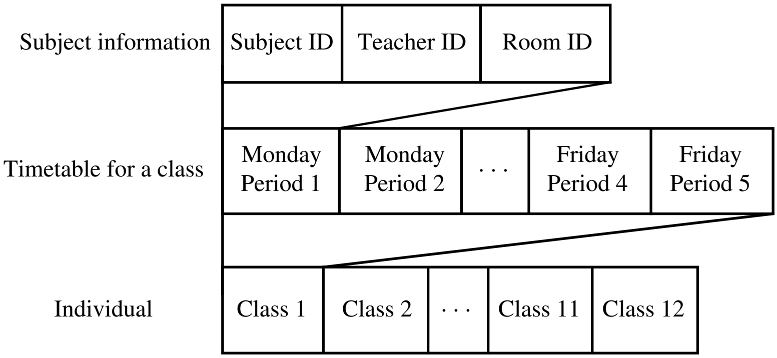


Fig. 1. The genotype for SGA

illustrated in Figure 1. In SGA, genetic operations such as selection, crossover and mutation are applied to individuals repeatedly. These operations are almost the same as those for TGA. Thus, details about these operations are described in the next section.

3.2 The TGA

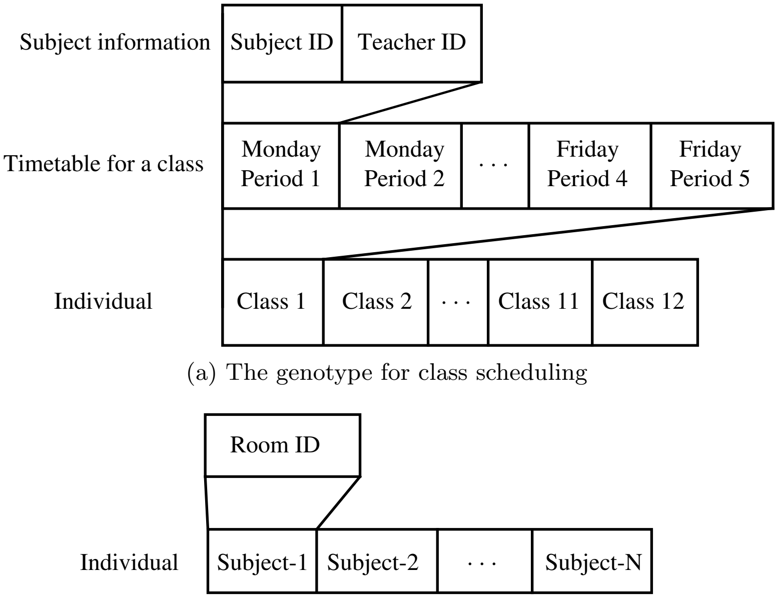
In TGA, we use two types of genotypes. Figure 2 shows the genotypes. One is the genotype for class scheduling, and the other is the genotype for room allocation. The genotype for class scheduling consists of a timetable for each class, and each timeslot in a timetable consists of subject information. Subject information consists of subject ID and teacher ID. The genotype for room allocation consists of room ID.

Figure 3 shows the flow diagram of the TGA. In the first phase, genetic operations are applied to the population for class scheduling, and the cost values related to constraints for class scheduling are calculated. In the second phase, genetic operators are applied to the population for room allocation. Finally, we combine two types of individuals, and the fitness values of these individuals are calculated. These steps are repeated until the stopping criterion is satisfied. Next, we describe details of each genetic operation.

Class scheduling phase. Here we describe genetic operations for class scheduling.

1) Selection and Reproduction

We employ roulette selection and the elite keeping strategy. An individual with a low cost has a high probability that the individual is selected in the selection step. The elite is a pair of two types of individuals. Here we denote an individual for class scheduling as C_i, and an individual for room allocation as R_j. An individual pair is denoted as $< C_i, R_j >$. When the sum of the costs for $< C_i, R_j >$ is the minimum in the fitness calculation, the pair is treated as the elite. That is,



(a) The genotype for class scheduling

(b) The genotype for room allocation

Fig. 2. The genotypes for TGA

genetic operations such as crossover and mutation are not applied to C_i and R_j when $< C_i, R_j >$ is the elite.

2) Crossover

We use one-point crossover and crossover points are restricted to borderlines between timetables for any classes. Figure 4 shows an example of the crossover operation. In the figure, a crossover point is the borderline between Class 2 and Class 3.

After crossover, all subjects must appear exactly once due to this restriction, and it seems that a timetable for each class cannot be changed by the crossover operator. However, this problem is solved by employing a repair operation. By use of the crossover operator, many multiple-class subjects tend to be violated. The repair operator modifies timetables in order to repair the violated multiple-class subjects. Thus, the crossover operator also contributes to changing a schedule of each class.

3) Mutation

In mutation, subject information of a particular timeslot is replaced with the subject information of a randomly selected timeslot. If the subject of the timeslot is a multiple-period subject, then subject information in multiple timeslots is exchanged. Figure 5 shows an example of mutation.

4) Repair

After crossover and mutation, several multiple-class subjects may be violated.

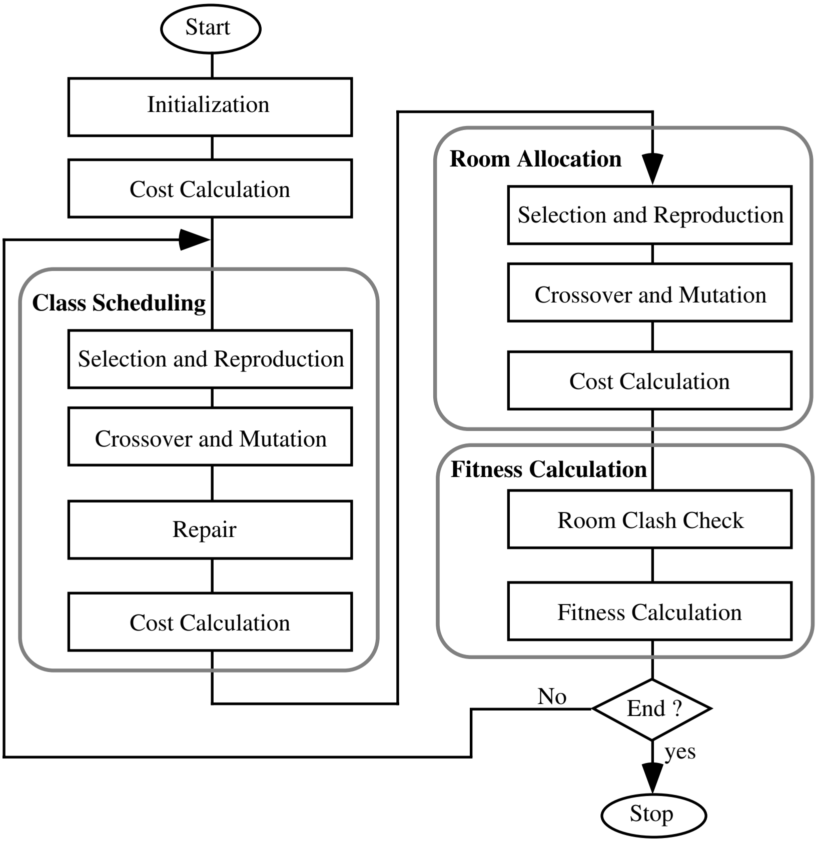


Fig. 3. A flow diagram of TGA

The repair operator tries to repair these subjects by moving the violated subject to a different timeslot if possible. When we cannot repair the violated subject, a penalty is given to the individual to whom the violated subject belongs. We will describe the penalty values when we outline the cost calculation step.

The operator does not repair the violated subjects when some multiple-class subjects or multiple-period subjects are newly violated by repairing the violated subjects. In our method, more complex subjects are repaired earlier than less complex subjects. Here, we define the complexity of a subject as $pC + c$, where p is the number of periods needed by the subject, C is the number of classes in the timetabling problem, and c is the number of classes that the subject is taught. According to the definition, multiple-period subjects that need many periods are assumed to be the most complex subjects, and multiple-class subjects that are taught to many students (classes) are assumed to be more complex subjects. In Table 2, Subject 2 is more complex than Subject 1, since the complexities of Subjects 2 and 1 are 10 and 7, respectively. To explain the behavior of the repair

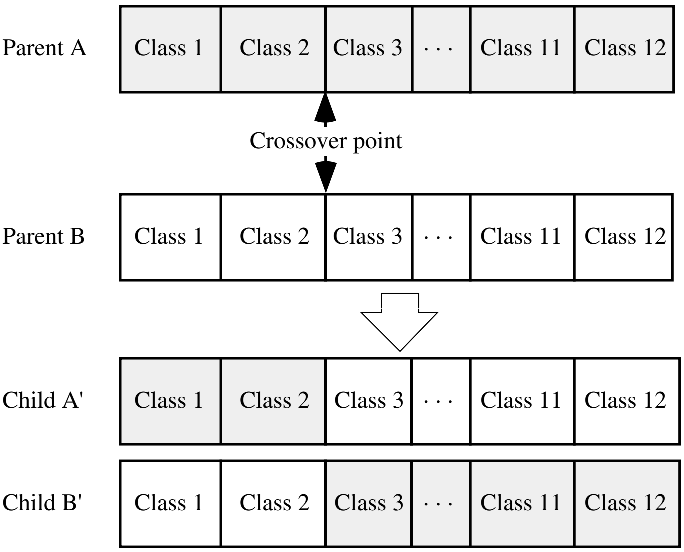


Fig. 4. Crossover for the genotype related to class scheduling

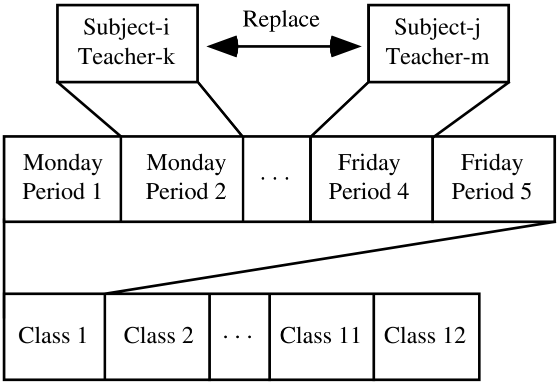


Fig. 5. Mutation for the genotype related to class scheduling

operation, we use this example. In Table 2, multiple-class Subjects 1 and 2 are violated. Since Subject 2 is more complex than Subject 1, repair of Subject 2 is attempted first. The repair operator tries to move Subject 2 for B-1 to periods 3 and 4, or Subject 2 for D-1 is moved to periods 1 and 2. When Subject 2 for B-1 is moved to periods 3 and 4, Subject 7 is newly violated. Thus, Subject 2 for D-1 is moved to periods 1 and 2. In the same manner, Subject 1 for A-1 is moved to period 3. Table 3 shows the timetable after the repair operation, where no multiple-class subjects are violated.

Since the repair operator is a simple operator, its computational costs are not high. However, many violated subjects can be repaired by the operation.

Table 2. An example of a timetable before the repair operation

Class	Period 1	Period 2	Period 3	Period 4	Period 5
A1		**Subject-1** **Staff-1**	Subject-5 Staff-5		Subject-6 Staff-5
B-1	**Subject-2** **Staff-2**		Subject-7 Staff-6 and Staff-7		
C-1		Subject-3 Staff-3	**Subject-1** **Staff-1**		
D-1	**Subject-1** **Staff-1**	Subject-4 Staff-4	**Subject-2** **Staff-2**		Subject-8 Staff-3

Table 3. An example of a timetable after the repair operation

Class	Period 1	Period 2	Period 3	Period 4	Period 5
A1		Subject-5 Staff-5	**Subject-1** **Staff-1**		Subject-6 Staff-5
B-1	**Subject-2** **Staff-2**		Subject-7 Staff-6 and Staff-7		
C-1		Subject-3 Staff-3	**Subject-1** **Staff-1**		
D-1	**Subject-2** **Staff-2**		**Subject-1** **Staff-1**	Subject-4 Staff-4	Subject-8 Staff-3

5) Cost calculation

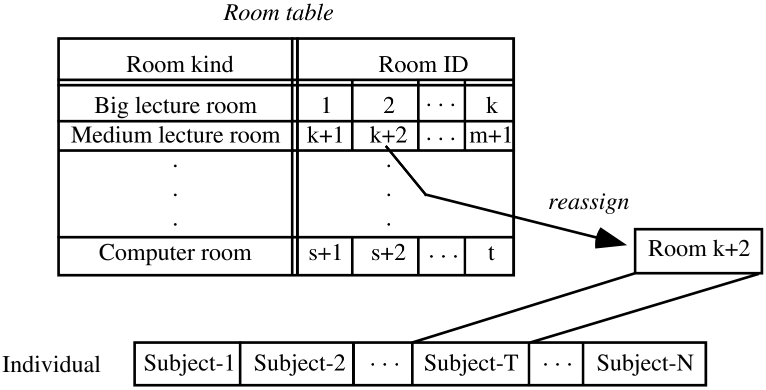
In the class scheduling phase, only constraints for class scheduling are considered for the cost calculation. In Equation (1) $Scheduling\_cost(C_i)$ is defined to be a sum of penalty values:

$$Scheduling\_cost(C_i) = \sum_{\substack{k \in \, all \ constraints \\ for \ class \ scheduling}} Penalty\_value(k). \tag{1}$$

When multiple-period subjects and multiple-class subjects are violated, the penalty values are evaluated according to the extent of its violation. Now, we assume that Subject-T is taught for n classes and the subject needs m consecutive periods. Moreover, we assume the subject uses $p(p \leq n)$ kinds of timeslots in a timetable. The penalty value for violation of the subject depends on m and p, and the value is defined as $(p-1)m$. In Table 2, m and p for Subject 1 are 3 and 1, respectively. Thus, the penalty value for violation of the subject is evaluated as $(3-1)1 = 2$. In the same manner, the penalty value for Subject 2 in the same table is evaluated as $(2-1)2$=2, since Subject 2 uses two kinds of timeslots: i.e. {Period 1, Period 2} and {Period 3, Period 4}.

Room allocation phase. In the room allocation phase, we use similar genetic operations as for the class scheduling phase except for mutation. Thus, we only describe the mutation operator here.

In mutation, we reassign a room to a mutated subject. Figure 6 shows a mutation for Subject-T. When room allocation for a subject is mutated, a room whose size is different to the size of the currently assigned room may be reassigned to the subject, but rooms with different kinds are not assigned to it. For example, we assume that Subject-T in Figure 6 needs a lecture room whose size is medium. When the room allocation for Subject-T is mutated, we may assign a lecture room whose size is not medium: that is, a big or small room may be assigned to the subject, but any computer rooms are not assigned to it.



Room table

Room kind	Room ID			
Big lecture room	1	2	· · ·	k
Medium lecture room	k+1	k+2	. . .	m+1
· · ·		· · ·		
Computer room	s+1	s+2	· · ·	t

Fig. 6. Mutation for the genotype related to room allocation

Fitness calculation phase. In the fitness calculation step, the fitness values are calculated after the room clash check is done. Figure 7 shows the outline of the room clash check. In the room clash check, n individuals with the lowest costs are selected from two kinds of populations, and selected individuals are combined. Any individuals selected from the population for class scheduling are combined with all individuals selected from the population for room allocation. That is, n^2 individual pairs are made. For each pair, we check whether room clashes occur or not, and a room clashing cost is calculated. Here, we denote the room clashing cost for individual pair $< C_i, R_j >$ as $Room\_Clashing\_Cost(C_i, R_j)$, and we assume $Room\_Clashing\_Cost(C_i, R_j)$ is in proportion to the number of room clashes. By using $Room\_Clashing\_Cost(C_i, R_j)$, we update the cost values of all individuals for both populations. Equations (2) and (3) show the functions which update the cost values of individuals related to class scheduling, where $Class\_Cost(C_i)$ represents the updated cost value of C_i. When C_i is selected at the room clash check, we use Equation (2) for calculation of $Class\_Cost(C_i)$. That is, $Class\_Cost(C_i)$ equals the minimum value of the cost of room clash plus $Scheduling\_Cost(C_i)$. When C_i is not selected at the room clash check, $Class\_Cost(C_i)$ is calculated by Equation (3). In (3), the average value of the minimum costs of room clash is a penalty value for not being selected for the room clash check. When we select all individuals from the population for class scheduling, only (2) is used. However, we select some of individuals in our experiments in order to save computational costs.

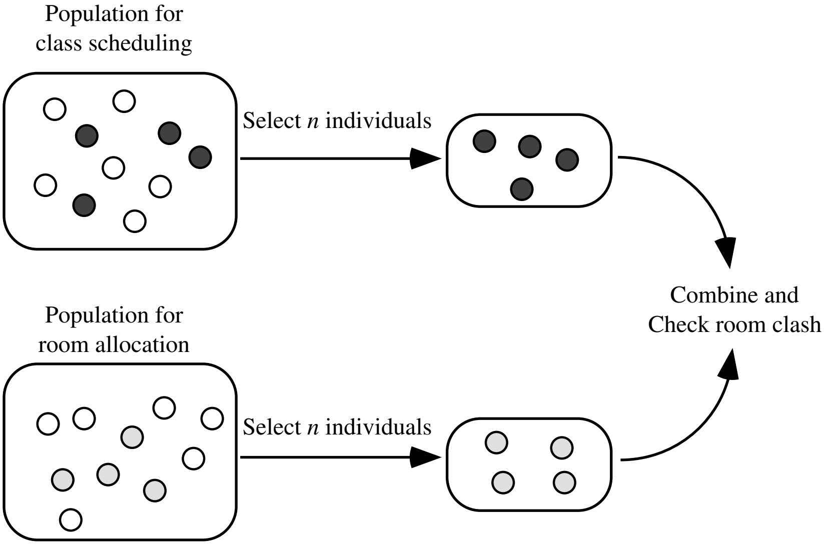


Fig. 7. Overview of the room clash check

As an example of updating the cost values, we consider that there are five individuals C_1–C_5, and C_1–C_3 are selected for the room clash check. We also assume that there are five individuals R_1–R_5, and that R_1–R_3 are selected for the room clash check. In this case, nine individual pairs $< C_i, R_j >$ $(i, j \leq 3)$ are made, and we calculate the room clashing cost for each pair. Then, $Class\_Cost(C_i)$ for $i \leq 3$ is calculated by Equation (2), and $Class\_Cost(C_i)$ for $i \geq 4$ is calculated by Equation (3).

Using the updated cost $Class\_Cost(C_i)$, the fitness value $Class\_fitness(C_i)$ is calculated by Equation (4). The fitness values of individuals for room allocation are calculated in the same manner. The selection and the reproduction steps are performed by using these fitness values.

Although TGA may not be a co-evolution method in the strict sense, it is a co-evolution method in the sense that the fitness values for one population are closely related to those of the other population. For example, $Class\_Cost(C_i)$ may be less than $Class\_Cost(C_k)$ when $Scheduling\_Cost(C_i)$ is greater than $Scheduling\_Cost(C_k)$:

$$\begin{aligned} Class\_Cost(C_i) = \min_j(Room\_Clashing\_Cost(C_i, R_j)) \\ +Scheduling\_Cost(C_i). \end{aligned} \tag{2}$$

$$\begin{aligned} Class\_Cost(C_i) = \operatorname*{average}_k(\min_j(Room\_Clashing\_Cost(C_k, R_j))) \\ +Scheduling\_Cost(C_i). \end{aligned} \tag{3}$$

$$Class\_fitness(C_i) = \max_k(Class\_Cost(C_k)) - Class\_Cost(C_i). \tag{4}$$

4 Experimental Results

The methods mentioned above were implemented in the C language on a Sun Ultra10 (333MHz). Table 4 shows the timetabling problem that we used. This problem derives from the curriculum of the Faculty of Information Sciences at Hiroshima City University in 1997(HCU_97). There are many multiple-class subjects, some multiple-period subjects, and many room types in the problem. Table 5 lists the additional constraints. Due to these constraints, the problem becomes more complex.

Table 4. Parameters of the timetabling problem

Subjects and teachers	no. of subjects	93
	no. of teachers	97
	no. of classes	12
	no. of school days on a week	5
	no. of periods in a date	5
	no. of multiple-class subjects	21
	no. of multiple-period subjects	4
Rooms	no. of small classrooms	19
	no. of medium classrooms	12
	no. of large classrooms	18
	no. of very large classrooms	1
	no. of language laboratory rooms	3
	no. of laboratory rooms	7
	no. of computer rooms	3

Table 5. The number of additional constraints

Constraint type	No. of constraints
C6	20
C7	3
C8	76
C9	28

Table 6. Penalty values for room allocation

		The size of the allocated room			
		Small	Medium	Large	Very large
The adequate size for the subject	Small	0.0	0.2	0.5	0.5
	Medium	1.0	0.0	0.2	0.5
	Large	1.0	1.0	0.0	0.2
	Very large	1.0	1.0	1.0	0.0

Next, we detail the parameter values in our experiments. Crossover and mutation probabilities are 100% and 1%, respectively. The number of selected individuals at the room clash check is 10. The process is terminated when 500 generations are evolved or all constraints are satisfied. These parameter values are decided by performing preliminary experiments, and we obtain good results when we use these values. Apart from the constraints for room allocation, the penalty value is 1.0 per unsatisfied constraint. The penalty values for room allocation are shown in Table 6. When the size of the allocated room is smaller than the adequate size for a subject, the penalty value is 1.0. When the size of the allocated room is slightly larger than the adequate size for a subject, the penalty value is 0.2. We use 0.5 as the penalty value when the size of the allocated room is much larger than the adequate size for a subject.

Figure 8 shows the changes of the cost values for the elite. The curves indicate average values over ten trials. Obviously, TGA finds a better solution than SGA. Even in early generations, TGA finds the elite with smaller cost. In TGA, we combine two kinds of individuals with the lowest costs, thus TGA finds a better elite in the early evolution phase. Table 7 shows details of the experimental results. The worst (best) case in the table shows the cost of the elite at the final generation in the worst (best) case. The average cost in the table shows the average cost of the elites for ten trials. CPU time is the average CPU time for ten trials. Both for the worst and best cases, we obtain good results by TGA. In the best case, all constraints are satisfied.

Table 8 shows a part of the timetable obtained by TGA. This table shows the schedule and room allocation on Friday. In the table, Sxx represents Subject ID, and SRxx, MRxx, LRxx, LLxx and CR represent room IDs. Boldface subjects such as **S78** are multiple-class subjects, and subjects in italics such as *S99* are multiple-period multiple-class subjects. There are no violations of constraints nor room clashes. Table 9 shows the schedule on Thursday. In the table, Txx

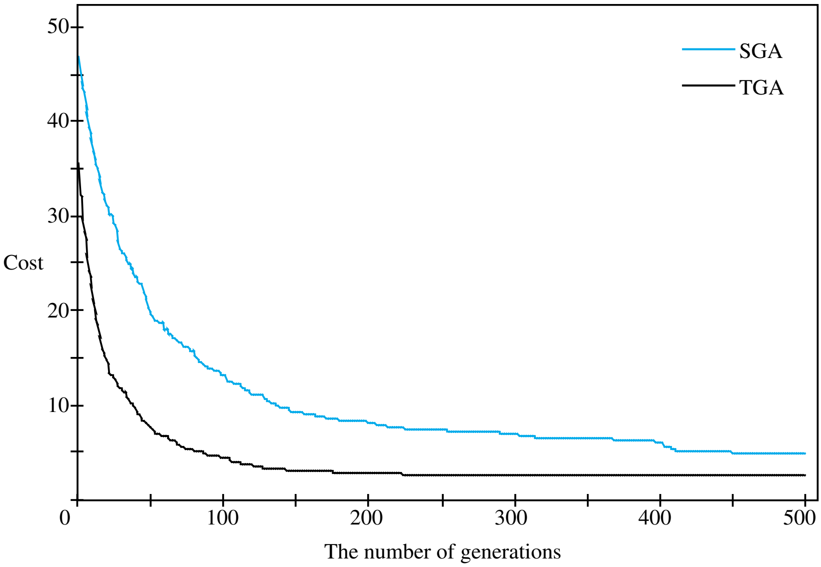


Fig. 8. Changes of the cost values of the elite

Table 7. Costs of the elite for ten trials

	TGA	SGA
The worst case	5.0	7.0
The best case	0.0	1.0
The average cost	2.5	4.8
CPU (s)	23.5	37.5

represents teacher ID, and boldface subjects are taught by professors. In our experiments, we consider the constraints that any professors must not have any classes on Thursday afternoon because of a professor meeting. Thus, there are no subjects taught by professors in the afternoon in Table 9. Subjects in italics such as *S33* and *S34* are treated as multiple-period subjects.

Finally, we applied TGA and SGA to several timetabling problems. These problems were generated by the automatic timetabling problem generator which is implemented in C. This tool generates timetabling problems randomly by using some parameters such as the number of classes, the average number of subjects per class, the room utilization ratio, and so on. Table 10 shows some parameter values that we used. For all problems in this table, the number of classes, the number of school days in a week, and the number of periods in a day are assumed to be 12, 5 and 5, respectively. The room utilization ratio is the probability that a room is used in a period. Thus, the room utilization ratio multiplied by the number of periods in a week gives the average frequency of the use of a room per week. Non-lecture rooms in Table 10 indicate special-purpose rooms such as

Table 8. An example of the timetable obtained by TGA (Friday)

Class	Period 1	Period 2	Period 3	Period 4	Period 5
A1	S4, MR12	S3, LR8			
B1	S18, LR7	S15, SR4		S20, LL2	**S22, MR11**
C1		S27, LL2			
D1					**S22, MR11**
A2			*S99, CR*	*S99, CR*	*S99, CR*
B2		S43, LR6	*S99, CR*	*S99, CR*	*S99, CR*
C2		S51, LR2	*S99, CR*	*S99, CR*	*S99, CR*
D2	S58, LL2		*S99, CR*	*S99, CR*	*S99, CR*
A3	S73, LR2	**S70, LR3**			
B3	**S78, LR1**	**S70, LR3**			
C3	**S78, LR1**				
D3			S89, LR5	S86, LR5	

Table 9. An example of the timetable obtained by TGA (Thursday)

Class	Period 1	Period 2	Period 3	Period 4	Period 5
A1					
B1	**S21, T9**	**S21, T9**			
C1					S24, T104
D1			*S33, T70*	*S34, T70*	S36, T106
A2	S44, T54		S38, T53	S45, T64	
B2	S48, T57		T38, T53	S45, T64	
C2				S45, T64	
D2	S63, T72	S97, T72		S45, T64	S62, T71
A3					S74, T56
B3		**S75, T7**			
C3	S81, T75			S84, T66	
D3	**S91, T27**		S90, T67		S74, T56

computer rooms. Since the difference between TGA and SGA rests in whether or not the genotype is split, we focus on the results when the room utilization ratio is changed. Thus, the only difference between these problems is the parameter values with respect to the room utilization ratios. Other parameter values are chosen in order to make the complexity of the problems equal to that of HCU_97. While the average number of multiple-class subjects is different from the number of multiple-class subjects in Table 4, the total number of timeslots needed by multiple-class subjects in the problems is almost equal to that in HCU_97.

Table 11 shows results for these problems. In the table, CC indicates the class cost of the elite, and RC equals the room cost plus the room clashing cost for the elite. That is, CC plus RC is equal to the total cost of the elite. In P1 and P2, TGA and SGA find the solution that satisfies all constraints. However,

Table 10. Parameters for generated timetabling problems

Parameter type	Problems			
	P1	P2	P3	P4
The average number of subjects per class	17.5	17.5	17.5	17.5
The average number of subjects per teacher	4.0	4.0	4.0	4.0
The average number of multiple-period subjects	3.5	3.5	3.5	3.5
The average number of multiple-class subjects	3.5	3.5	3.5	3.5
The room utilization ratio for lecture rooms	0.1	0.1	0.5	0.5
The room utilization ratio for non-lecture rooms	0.1	0.5	0.1	0.5

Table 11. Results for timetabling problems generated by ATPG

Problems	TGA			SGA		
	CC	RC	CPU (s)	CC	RC	CPU (s)
P1	0.0	0.0	3.6	0.0	0.0	9.1
P2	0.0	0.0	5.1	0.0	0.0	14.8
P3	0.0	0.0	38.4	1.0	3.0	48.8
P4	4.0	4.0	48.0	1.0	10.5	48.0

TGA finds better solutions than SGA when the average room utilization ratio is high.

5 Conclusion

We have described a genetic algorithm with two phases to solve a university timetabling problem. The key points of the TGA are the use of two types of genotypes for the evolution phases and combination of these genotypes for the fitness calculation. In the experiment for the timetabling problem in Hiroshima City University, TGA finds a feasible solution. For problems generated by the automatic timetabling problem generator, TGA does not find a feasible solution when the average room utilization ratio is high. However, we show that TGA find better solutions than the simple GA.

Improvement of the repair operator with the use of more intelligent methods such as a local search technique, and comparisons between TGA and other methods remain as important goals for future work.

References

1. Mitchell, M.: An Introduction to Genetic Algorithms. MIT Press, Cambridge, MA (1998)
2. Carter, M.W., Laporte, G.: Recent Developments in Practical Course Timetabling. In: Burke, E., Carter, M. (eds.): The Practice and Theory of Automated Timetabling: Selected Papers. Lecture Notes in Computer Science, Vol. 1408. Springer-Verlag, Berlin Heidelberg New York (1998) 3–19

3. Fukushima, M., Tanaka, S.: Adaptive Genetic Algorithms for Solving School Timetabling Problems. IEICE Trans. D-I, J82-D-I(6), (1998) 883–885 (in Japanese)
4. Rich, Daivid C.: A Smart Genetic Algorithm for University Timetabling. In: Burke, E., Ross, P. (eds.): The Practice and Theory of Automated Timetabling: Selected Papers. Lecture Notes in Computer Science, Vol. 1153. Springer-Verlag, Berlin Heidelberg New York (1998) 181–197
5. Erben, W., Keppler, J.: A Genetic Algorithm Solving a Weekly Course-Timetabling Problem. In: Burke, E., Ross, P. (eds.): The Practice and Theory of Automated Timetabling: Selected Papers. Lecture Notes in Computer Science, Vol. 1153. Springer-Verlag, Berlin Heidelberg New York (1998) 198–211
6. Burke, E., Newall, J-P.: Multi-Stage Evolutionary Algorithm for Timetable Problem. IEEE Trans. Evol. Comput. (1999)
7. Peachter, B., Rankin, R.C., Cumming, A.: Improving a Lecture Timetabling System for University-Wide Use. In: Burke, E., Carter, M. (eds.): The Practice and Theory of Automated Timetabling: Selected Papers. Lecture Notes in Computer Science, Vol. 1408. Springer-Verlag, Berlin Heidelberg New York (1998) 156–165