International Journal of u-and e-Service, Science and Technology
Vol.7, No.4 (2014), pp.277-286
http://dx.doi.org/10.14257/ijunnesst.2014.7.4.25

Solving UTP Containing Combining Classes using GA

Zhang He-nan and Zhang Shao-wen

(School of Economics & Management, Beijing Forestry University, Beijing 100083, China)

Abstract

A university timetabling problem containing combining classes was studied, a corresponding mathematical model was established and an improved genetic algorithm was proposed to solve this problem. To improve the diversity of the initial population and avoid premature convergence, random processing was introduced when generating the initial population; to avoid population degeneration, the strategy of keeping the best individual and a competition mechanism was introduced; suitable crossover and mutation operators were designed. Furthermore, with hybrid programming of Matlab and Access, the efficiency of large-scale data processing was much improved. Example verification indicates that the proposed GA can solve UTP containing combining classes efficiently.

Keywords*: combining classes; genetic algorithm; random processing; keeping the best individual; competition mechanism*

1. Introduction

The university timetabling problem (UTP) is a widespread problem within educational institutions, whether it can be solved efficiently relates to the quality of teachingdirectly. In recent years, as the universities expand their enrollment, the number of students keeps increasing. And with the development of society, professional categories and course diversities keep growing as well. Under such conditions, the resource of teachers, rooms and so on are becoming rare. Therefore, an efficient timetabling has been a shared desire of teachers, students, and school administrators.

The timetabling problem is a combinatorial optimization problem with multiple constraints and objectives, which has been proved to be NP-Hard [1, 2]. Some approaches have been proposed to solve this problem. Early methods include direct heuristics [3], graph coloring [4, 5], integer programming [6-8], network flow techniques [9, 10], all of which have a limitation on the scale of the problem. With the development of computer technology, especially since the 1990s, an increasing number of intelligent algorithms have been applied to solve the timetabling problem, such as genetic algorithm [11-13], simulated annealing algorithm [14], taboo search algorithm [15] and hybrid algorithm [16, 17]. All these intelligent algorithms above have been proved to be efficient.

Among all these intelligent algorithms, genetic algorithm only requires the objective function that affects the search direction and the corresponding fitness function rather than other auxiliary knowledge [18]; genetic algorithm operates a parallel search in the entire space and concentrates on the parts of high performance, which improves the efficiency and does not fall into local optimum easily [19]. With these advantages, genetic algorithm is ideal for solving the timetabling problem. Relevant studies have been conducted, which have some reference value bus also some deficiencies: (1) Constraints are not comprehensive enough, for example, combining classes are rarely taken into consideration [20, 21]; (2) the objective function only depends on violations of the constraints and ignores the fact that different courses are of different importance and different periods have different teaching effectiveness [13, 20]. This paper proposes improved approaches aiming at these deficiencies.

ISSN: 2005-4246 IJUNNEST

2. The University Timetabling Problem

2.1. Problem Description

The timetabling problem is a resource allocation problem that consists of scheduling classes, courses, teachers, rooms and so on, which optimizes the objective function while satisfying a set of constraints of various types. Students, teachers, and administrators all prefer that the same course be arranged in the same room through a semester, therefore, this paper neglects the concept of week and focuses on one week with the most courses. If the courses within this week could be scheduled reasonably, then scheduling of other weeks will be deleting from that of this week.

2.2. Hard Constraints

Hard constraints are constraints that conform to logic and must be satisfied under any circumstances [22]. This paper considers thefollowing four hard constraints:

1) There could be no more than one course at the same time for the same class.
2) There could be no more than one course at the same time for the same teacher.
3) There could be no more than one course at the same time in the same room.
4) The capacity of the room should not be less than the size of the class that takes courses in this room.

Hard constraints decide whether the scheduling is feasible.

2.3. Soft Constraints

Soft constraints are constraints that can be violated but the violations may affect the quality of teaching [22]. This paper considers the following four soft constraints:

1) More important courses should be arranged at periods with better teaching quality.
2) For the same course of the same class, if this course has several arrangements within one week, then they should not be in the same day.
3) For each class, courses should be distributed evenly over each day based on their importance, avoiding too many important courses or too few unimportant courses in one day.
4) The size of the class should be matched with the capacity of the room in order to improve the room utilization.

Soft constraints decide the quality of the scheduling.

2.4. The Objective Function

The quality of the objective function is determined by how well the soft constraints are satisfied. The more the soft constraints are satisfied, the better the objective function is. The objective function in this paper is a comprehensive evaluation of how well all the soft constraints are satisfied.

3. The Mathematical Model for UTP

3.1. Terminology

Small classes: classes that are determined by majors, grades and class ID numbers. For example, “Accounting 10-1” means Class 1, Grade 2010, majoring in accounting.

Classes: A set of small classes that take the same course at the same time in the same room. A classcan consist of only one small class or several small classes.

Subjects: The course that corresponds to a fixed teacher and a fixed class. A class can correspond to more than one subjects while a subject can only correspond to one class.

Periods: Taking 45 minutes for a lesson for example, two to four lessons are always taken continually, such as the first and the second lesson or the first to the fourth lesson in the

morning. Based on this characteristic, this paper divides the daily teaching time into 5 periods: the first and second lesson in the morning is the first period, the third and the fourth lesson in the morning is the second period, and so on.

3.2. Data Structure and Variables

This paper adopts the naming method of property pluses class with the first letter of the latter words capitalized, making it readable. For example, $idTeacher_i$ is the ID number of teacher *i*, $IdSubjectClass_j$ is the set of the ID numbers of the courses that class *j* takes.

The original data consists of the following four tables:

1) CONFIG:*nbPeriodDay*is the number of all the periods of each day, which is 5. *nbDayWeek* is the number of all the days of a week, which is 5. Then the number of all the periods of a week –*nbPeriodWeek*–can be calculated:*nbPeriodWeek*=*nbPeriodDay*·*nbDayWeek*=25. The set of all the periods of a week is *PeriodWeek*= {1, 2, …,*nbPeriodWeek*}。
2) ROOM: There are a total of *nbRoom*rooms，$idRoom_i$ is the ID number of room *i*, $capacityRoom_i$ is the capacity of room *i*.
3) SUBJECT: There are a total of *nbSubject* subjects, $idSubject_i$ is the ID number of subject *i*, $creditSubject_i$ is the credit of subject *i*, $nbPeriodSubject_i$ is the number of periods of a week of subject *i*.
4) SMALLCLASS: There are a total of *nbSmallclass* small classes, $idSmallclass_i$ is the ID number of small class *i*, $sizeSmallclass_i$ is the size of small class *i*.

Other data needed for modeling are generated through hybrid programming of Matlab and Access, including the following two tables:

1) CLASS：There are a total of *nbClass*classes, $idClass_i$ is the ID number of class *i*, $IdSmallclassClass_i$is the set of ID numbers of all the small classes that make up class *i*, $sizeClass_i$ is the size of class *i*, $IdSubjectClass_i$is the set of the ID numbers of the subjects that class *i*take, $nbPeriodClass_i$ is the number of periods class *i* has.
2) TEACHER：There are a total of *nbTeacher* teachers, $idTeacher_i$ is the ID number of teacher *i*, $IdSubjectTeacher_i$ is the set of ID numbers of the subjects teacher *i* teaches.

3.3. Constraints

Hard constraints are described by mathematical language as follows:

1) There could be no more than one course at the same time for the same class:

$$\forall i \in PeriodWeek, \bigcap_{j=1}^{nbRoom} IdSmallclassClass_{(i,idRoom_j)} = \emptyset$$

2) There could be no more than one course at the same time for the same teacher:

$$\forall i \in PeriodWeek, \bigcap_{j=1}^{nbRoom} idTeacher_{(i,idRoom_j)} = \emptyset$$

3) There could be no more than one course at the same time in the same room:

$$\forall i \in PeriodWeek, \forall j \in IdRoom, \bigcap_{k=1}^{nbClass} idClass_{k_{(i,j)}} = \emptyset$$

4) The capacity of the room is not less than the size of the class that takes courses in this room:

$$\forall i \in PeriodWeek, \forall j \in IdRoom, capacityRoom_j \geq sizeClass_{(i,j)}$$

Soft constraints are reflected in the objective function.

3.4. The Objective Function

1) Effect of period arrangements

For a small class *i*, find out the credits of all the subjects and period arrangements relevant to the class. The evaluation function is as follows:

$$f1Smallclass_i = \sum_{m=1}^{nbPeriodWeek} \sum_{n=1}^{nbRoom} weightPeriod_m \cdot creditSubject_{(m,idRoom_n)}$$

$weightPeriod_m$ is the weight of the m[th]period. The weight of the first and the third period of a day is 1, the weight of the second and the fourth period is 0.6, the weight of the fifth period is 0.8. The importance of a course is directly measured by its credit.

2) The same courses should avoid being arranged in the same day

For a small class *i*, the number of the kinds of all the subjects relevant to the class is $nbSubjectSmallclass_i$. $nbSameSubjectSmallclass_i$ is the number of the kinds of subjects that are arranged in the same day. The evaluation function is as follows:

$$f2Smallclass_i = \frac{nbSubjectSmallclass_i}{nbSameSubjectSmallclass_i}$$

3) The courses of the same small class should be distributed evenly over each day

For a small class *i*,calculate the number of periods arranged in each day. The smaller the variance is, the more effective the arrangement is.The evaluation function is as follows:

$$f3Smallclass_i = \frac{1}{var(nbPeriod_{i1}, nbPeriod_{i2}, \dots, nbPeriod_{i5})}$$

4) Room utilization

Calculate the average utilization of all the rooms that have been occupied. The evaluation function is as follows:

$$f4Smallclass_i = \frac{\sum_{m=1}^{nbPeriodWeek} \sum_{n=1}^{nbRoom} \frac{sizeClass_{(m,idRoom_n)}}{capacityRoom_{idRoom_n}}}{}$$

Then the general evaluation function is:

$$f = \frac{1}{10} \cdot f4Smallclass_i \cdot \sum_{i=1}^{nbSmallclass} f1Smallclass_i \cdot f2Smallclass_i \cdot f3Smallclass_i$$

Using this evaluation function as the fitness function in this paper. The higher the value of the fitness function is, the better the solution is.

4. Genetic Algorithm for UTP

4.1. Coding method

To solve the timetabling problem is to arrange all the classes, subjects and teachers reasonably at the right time and place, namely period and room. This paper establishes a two- dimensional matrix with *nbPeriodWeek* rows and *nbRoom* columns. Every nonempty element includes three variables: the ID number of the class, the ID number of the subject, and the ID number of the teacher. The empty elements indicate that there are no arrangements at that time in that room. This coding method can directly satisfy the third hard constraint. Figure 1 shows the coding method.

	Room1	Room2
Period1	$(idClass_{i1}, idSubject_{j1}, idTeacher_{k1})$	$(idClass_{i3}, idSubject_{j3}, idTeacher_{k3})$
Period2		$(idClass_{i2}, idSubject_{j2}, idTeacher_{k2})$
Period3	$(idClass_{i4}, idSubject_{j4}, idTeacher_{k4})$	

Figure 1. Coding Method

4.2. Initial Population

Every initial feasible solution is an individual of the initial population. Figure 2 shows how

a feasible solution is generated.

Based on this method, arranging classes with more periods first may result in the latter classes not being able to be arranged because the solution space is compressed too early. As a result, all the classes are sorted in the order from smallest to largest in number of their weekly periods. In this way, early compression of solution space is avoided and a feasible solution can be generated very quickly.

If the orders of the classes are unchanged every time the feasible solution is generated, it is inevitable that different solutions may have some similarities, which may lead to the local optimal solution. Therefore, it is essential to introduce the random processing to increase the diversities between different individuals, so that GA can search through a larger solution space to find the global optimal solution.

There are two steps of random processing. First, after all the classes are sorted in the order from smallest to largest in number of their weekly periods, classes with the same amount of weekly periods are grouped. Then random sorting is conducted within each group, as shown in Figure 3 and Figure 4. Secondly, randomly sort all the subjects of each class, as shown in Figure 5.

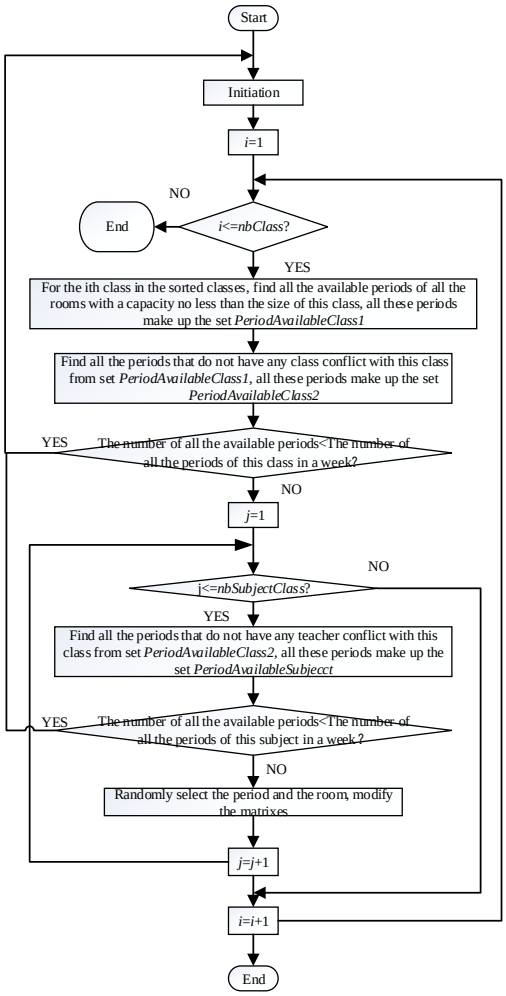

Figure 2. The Process of Generating a Feasible Solution

idClass	68	75	76	6	17	32	71	3	18	21
weekly periods	2			3				4		

Figure 3. ID Numbers of Classes Before Random Processing

idClass	75	76	68	32	17	71	6	21	3	18
weekly periods	2			3				4		

Figure 4. ID Numbers of Classes After Random Processing

ID numbers of subjects before random processing	22	26	86	78	79
ID numbers of subjects after random processing	86	78	26	79	22

Figure 5. Random Processing of Subjects

4.3. Selection

The selection operator adopts the strategy of keeping the best individual, namely the individual with the highest fitness value can directly survive into the next generation. Other individuals will be selected by the roulette method. The higher the fitness value is, the more possible it is to be selected. f_i is the fitness value of individual *i*, then the probability of it being selected is:

$$p_i = \frac{f_i}{\sum_{i=1}^{nbIndividual} f_i}$$

4.4. Crossover

Every solution is generated satisfying a lot of constraints. If the scale of crossover is too large, it is quite possible that too many constraints would be violated resulting in an unfeasible solution, so this paper adopts the two-point crossover. Besides, competition mechanism is introduced to ensure the improvement of the solution by crossover. The operations are as follows.

1) For two parents – Parent 1 (Individual *i*) and Parent 2 –that are about to crossover, find all the positions in the corresponding matrixes with the same coordinates and different ID numbers of subjects. These positions are the candidate positions.
2) Randomly select a position from the candidates. The element at this position in Parent 1 is marked as Element 1, and the element at this position in Parent 2 is marked as Element 2. Exchange Element 1 and Element 2, then there is one more Element 2 and one less Element 1 in Parent 1, one more Element 1 and one less Element 2 in Parent 2. Find other elements the same as Element 2 in Parent 1 and put them in Set 1, and find other elements the same as Element 1 in Parent 2 and put them in Set 2. Both sizes of Set 1 and Set 2 are small, so elements in Set 1 and Set 2 are exchanged by means of traversal to ensure the quality of the crossover operator. 2 parents and 2 offersprings are put into the set $GenerationTemp_i$ if both of the offsprings are feasible, as shown in figure 6.
3) Introduce competition mechanism. Evaluate 2 parents and 2 offsprings in each set $GenerationTemp_i$, select the one with the highest fitness value and put it into the set *GenerationTempBest*. In the end, evaluate all the individuals in set *GenerationTempBest*, select the best one and put it into the next generation.

Parent1	Room1	Room2	Room3
Period1	**Element1**		
Period2			**Element 2**
Period3		Element 1	
Period4	Element 2		

Parent 2	Room1	Room2	Room3
Period1	**Element 2**		**Element 1**
Period2		Element 2	
Period3			
Period4	Element 1		

⇩

Offspring1	Room1	Room2	Room3
Period1	**Element 2**		
Period2			**Element 1**
Period3		Element 1	
Period4	Element 2		

Offspring2	Room1	Room2	Room3
Period1	**Element 1**		**Element 2**
Period2		Element 2	
Period3			
Period4	Element 1		

Figure 6. The Crossover Process

In addition, to control the computing time, a limit of tried attempts in crossover is set. The process of crossover is as shown in Figure 7.

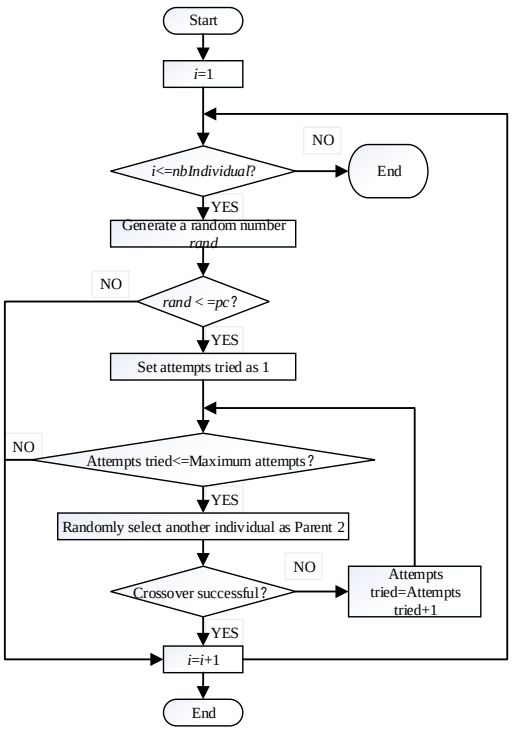

Figure 7. The Crossover Process

4.5. Mutation

For the parent that is about to mutate, randomly select two rows in its matrix. An offspring is generated by exchanging these two rows. This offspring, by the coding methods, is also a feasible solution. The mutation process is shown in Figure 8. After the mutation, competition mechanism is also introduced to select a better one between the parent and the offspring ant put it in the next generation.

Parent	Room1	Room2	Room3
Period1	**Element 1**		**Element 2**
Period2			Element 3
Period3		**Element 4**	

⇩

Offspring	Room1	Room2	Room3
Period1		**Element 4**	
Period2			Element 3
Period3	**Element 1**		**Element 2**

Figure 8. The Mutation Process

4.6. Population Evolution

The process of the population evolution is shown in Figure 9.

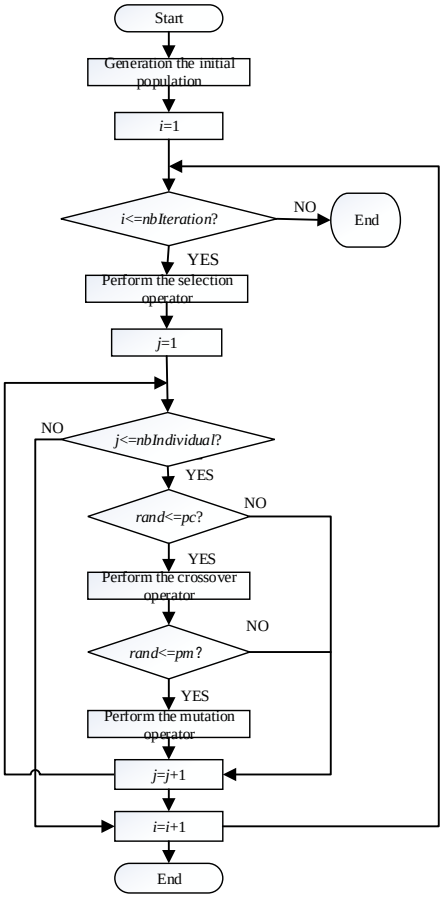

Figure 9. The Process of the Population Evolution

4.7. The Checking Mechanism

To verify the algorithm, this paper introduces 4 checking mechanisms.

1) If there is a class appearing in more than one room at the same period.
2) If there is a teacher appearing in more than one room at the same period.
3) If the capacity of the room is no less than the size of the class taking courses in the room.
4) If the number of all the elements in a solution equals the number of all the periods of all the classes.

The results verify that each individual in the population has satisfied all the hard constraints and all classes, subjects and teachershave been arranged.

5. A Case Study

5.1. Experimental Data

The experimental data are extracted from the timetabling of the 2013~2014 school year of a school in a university, including 79 classes (which contain 93 different small classes), 145 subjects, 85 teachers,and 18 rooms.

5.2. Parameter Settings

The number of all the individuals in the initial population is 60, the iteration times is 90, the crossover probability is 0.8, and the mutation probability is 0.2.

5.3. Experimental Results

First, generate the initial population without the random processing. The results of evolution are shown in Figure 10.

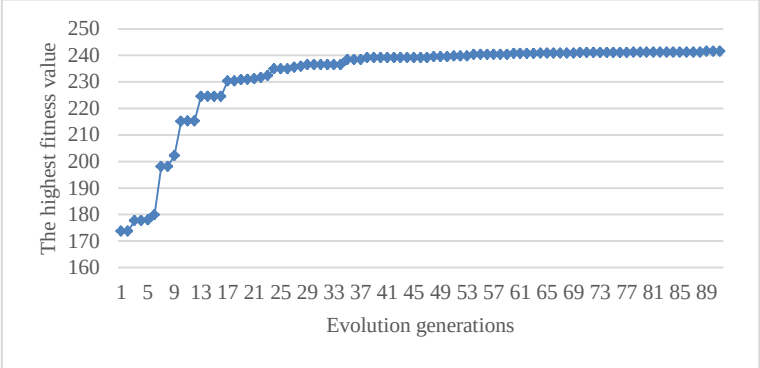

Figure 10. Population Evolution without Random Processing

As shown above, the evolution slows down around the 40th generation. That is due to insufficient diversity in the initial population which leads to the local optimal solution.

Next, random processing is introduced while generating the initial population. The results are compared between the original genetic algorithm and the improved genetic algorithm, as shown in Figure 12 and Table 1. The original genetic algorithm does not introduce the strategy of keeping the best individual; the crossover operator only compares the two offsprings and keeps the better one, and the crossover process is terminated once it is successful; the mutation operator only keeps the offspring.

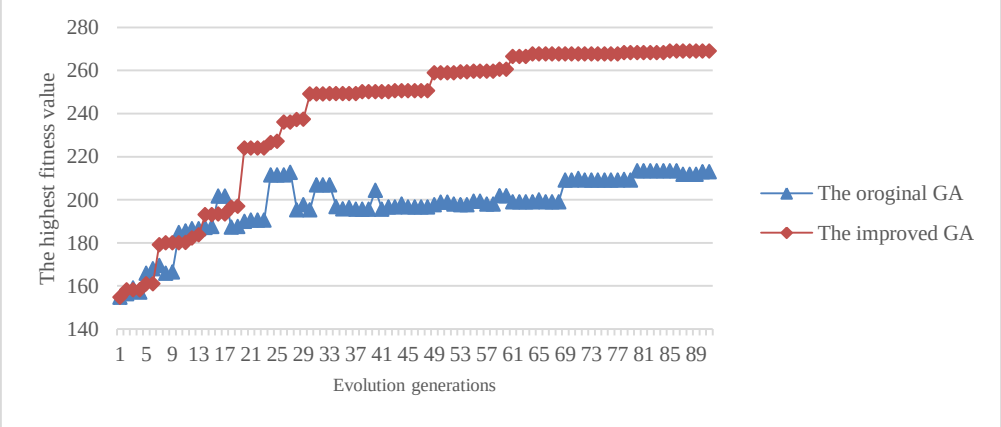

Figure 11. Comparisons of Population Evolutions between Two Kinds of GA

Through comparative study, it could be seen that the improved GA can avoid degeneration and is more efficient with better solutions.

Table 1. Comparisons of Scheduling Effect

	The original GA	The improved GA
The highest fitness value	212.97	268.97
The standard deviation of all the periods of each day	44.55	37.90
The number of subjects arranged several times in a day	23	10
Room utilization	41.27%	42.56%

The sum of all the periods each day of all the small classes is calculated when assessing uniformity because many classes contain more than one small class. The number of all the periods in a week of all the small classes is 2853.

As seen from table 1, the number of subjects arranged several times in a day is not very small, that is because the computing method is based on the small classes. There are 444 subjects in total. If a class consists of 5 small classes and this class has one subject that is arranged several times in a day, then the computing result is 5 subjects that are arranged in a day.

6. Conclusion

Combining classes are quite common in the university timetabling problem. This paper establishes the mathematical models and proposes the improved genetic algorithm aiming at this characteristic. The experimental results verify that the algorithm this paper proposes can solve the university timetabling problem containing combining classes efficiently.

References

[1] S. Even, A. Itai and A. Shamir, "On the complexity of time table and multi-commodity flow problems", [J], SIAM Journal of Computation, vol. 5, no. 4, **(1976)**, pp. 691-703.
[2] W. M. Carter and C. Tovey, "When is the classroom assignment problem hard?", [J], Operations Research, vol. 40, no. 1S, **(1989)**,pp. 28-39.
[3] W. Junginger, "Timetabling in Germany – a survey", [J], Interface, vol. 16, no. 4, **(1986)**,pp. 66-74.
[4] G. A. Neufeld and J. Tartar, "Graph coloring conditions for te existence of solutions to the timetable problem", [J], Communications of the ACM, vol. 17, no. 8, **(1974)**, pp. 450-453.
[5] D. de Werra, "An Introduction to timetabling", [J], European Journal of Operational, vol. 19, no. 2, **(1985)**, pp. 151-162.
[6] W. Shih and J. A. Sullivan, "Dynamic course scheduling for college faculty via zero-one programming", [J], Decision Sciences, vol. 8, no. 4, **(1977)**, pp. 711-721.
[7] R. H. McClure and C. E. Wells, "A mathematical programming model for faculty course assignments", [J], Decision Sciences, vol. 15, no. 3, **(1984)**, pp. 409-420.
[8] A. Tripathy, "School timetabling - a case in large binary integer linear programming", [J], Management Science, vol. 30, no. 12, **(1984)**, pp. 1473-1489.
[9] R. Ostermann and D. de Werra, "Some experiments with a timetabling system", [J], OR Spectrum, vol. 3, no. 4, **(1982)**, pp. 199-204.
[10] J. M. Mulvey, "A classroom/time assignment model", [J], European Journal of Operational Research, vol. 9, no. 1, **(1982)**, pp. 64-70.
[11] E. Yu and K. S. Sung, "A genetic algorithm for a university weekly courses timetabling problem", [J], International Transactions in Operational Research, vol. 9, no. 6, **(2002)**,pp. 703-717.
[12] H. Ueda, D. Ouchi and K. Takahashi, *et al*, "Comparisons of genetic algorithms for timetabling problems", [J], Systems and Computers in Japan, vol. 35, no. 7, **(2004)**, pp. 1-12.
[13] M. S. Kohshori and M. S. Liri, "Multi population hybrid genetic algorithms for university course timetabling", [J], International Journal for Advances in Computer Science, vol. 3, no. 1, **(2012)**, pp. 12-22.
[14] Y. Liu, D. Zhang and S. C. H. Leung, "A simulated annealing algorithm with a new neighborhood structure for the timetabling problem", [C], Proceedings of the first ACM/SIGEVO Summit on Genetic and Evolutionary Computation, **(2009)**.
[15] K. N. T. T. Minh, N. D. T. Thanh and K. T. Trang, *et al*, "Using tabu search for solving a high school timetabling problem", [M], //Studies in Computational Intelligence, **(2010)**, pp. 305-313.
[16] S. N. Jat and S. Yang, "A hybrid genetic algorithm and tabu search approach for post enrolment course timetabling", [J], Journal of Scheduling, vol. 14, no. 6 , **(2011)**, pp. 617-637.
[17] M. Chiarandini, M. Birattari and K. Socha, *et al*, "An effective hybrid algorithm for university course timetabling", [J], Journal of Scheduling, vol. 9, no. 5, **(2006)**, pp. 403-432.
[18] X. P. Wang and L. M. Cao, "Genetic algorithm-theory, application and software implementation", [M], Xi'an Jiaotong University Press, **(2002)**.
[19] L. Wang, "Job shop scheduling and genetic algorithm", [M], Tsinghua University Press, **(2003)**.
[20] X. F. Wang and X. N. Li, "Innovative GA code scheme applied in course arrangement", [J], Computer Engineering and Design, vol. 29, no. 17, **(2008)**, pp. 4565-4567.
[21] H. C. Li and Z. H. Dong, "Decimal immunization GA used to solve UTP", [J], Systems Engineering - Theory & Practice, vol. 32, no. 9, **(2012)**, pp. 2031-2036.
[22] R. Lewis, "A survey of metaheuristic-based techniques for university timetabling problems", [J], OR Spectrum, vol. 30, no. 1, **(2008)**, pp. 167-190.

