

Stochastic Optimisation Timetabling Tool for university course scheduling

P. Pongcharoen^{a,*}, W. Promtet^b, P. Yenradee^c, C. Hicks^d

^a*Faculty of Engineering, Naresuan University, Pitsanulok 65000, Thailand*

^b*Faculty of Science and Technology, Loei Rajabhat University, Loei 42000, Thailand*

^c*Industrial Engineering Program, Sirindhorn International Institute of Technology, Thammasat University, Pathum Thani 12121, Thailand*

^d*Business School, University of Newcastle upon Tyne, Newcastle NE1 7RU, UK*

Received 14 April 2006; accepted 12 July 2007

Available online 11 September 2007

Abstract

University timetabling is an NP-hard problem, which means that the amount of computation required to find solutions increases exponentially with problem size. Timetabling is subject to hard constraints that must be satisfied in order to produce feasible timetables and soft constraints, which are not absolutely essential. This paper describes the Stochastic Optimisation Timetabling Tool (SOTT) that has been developed for university course timetabling. Genetic Algorithms (GA), Simulated Annealing (SA) and random search are embedded in the SOTT. The algorithms include a repair process, which ensures that all infeasible timetables are rectified. This prevents clashes and ensures that the rooms are sufficiently large to accommodate the classes. The algorithms also evaluate timetables in terms of soft constraints: minimising student movement; avoiding fragmentation in the timetables for students and lecturers; and satisfying lecturers' preferences for the timing of classes. The algorithms were tested using two sets of timetabling data from a collaborating university. Both GA and SA produced very good timetables, but the results obtained from SA were slightly better than those using GA. However, the GA was 54% faster than SA.

© 2007 Elsevier B.V. All rights reserved.

Keywords: Course timetabling; Meta-heuristics; Genetic Algorithms; Simulated Annealing; Repair process

1. Introduction

Timetabling is defined as “the allocation, subject to constraints, of given resources to objects being placed in space–time, in such a way as to satisfy as nearly as possible a set of desirable objectives” (Wren, 1996, p. 53). Timetabling has been proved to be an NP-hard problem (Bardadym, 1996).

This means that the amount of computation required to solve problems increases exponentially with problem size. This makes it time consuming and difficult to manually construct timetables that satisfy the objectives and constraints, particularly for large problems. It is therefore necessary to use efficient search methods to produce optimal or near optimal solutions that satisfy the constraints.

There are different timetabling problems that arise in educational institutions during each academic year. These include course timetabling

*Corresponding author.

E-mail addresses: pupongp@nu.ac.th, pupongp@yahoo.com (P. Pongcharoen).

(lectures, seminars, tutorials, laboratories, etc.) and examinations timetabling. Educational timetabling problems need to be solved in schools, colleges and universities. There are some similarities between these problems, for example, teachers/lecturers can only teach one group of students at a time and rooms have size constraints. However, there are also some key differences. University timetabling problems tend to be larger because there are more programmes, classes and resources. There are also more interactions because students often have more choices, and different programmes may require common teaching. These factors increase complexity. Timetabling in schools is usually more constrained. For example, pupils have specified start, finish and lunch times. A compact timetable, in which there are no free periods, is normally a constraint that does not apply in a university context (Drexel and Salewski, 1997).

The objectives of this paper are to:

1. briefly review previous research on automated timetabling and stochastic optimisation methods;
2. describe the development of the Stochastic Optimisation Timetabling Tool (SOTT) for university course timetabling that includes Genetic Algorithms (GA), Simulated Annealing (SA) and random search;
3. investigate the significance of the repair process for ensuring that the timetables produced are feasible;
4. demonstrate the effectiveness of the proposed approaches using two real cases; and
5. compare the performance of GA, SA and random search through a series of experiments using the SOTT with data obtained from a collaborating university.

This paper first reviews the literature related to the university timetabling problems. Section 3 describes a formulation of the university timetabling problem and the assumptions made. Section 4 describes the heuristics embedded within the SOTT, which includes a repair process for correcting infeasible timetables. Section 5 explains two case studies that used data obtained from a collaborating university in Thailand. Section 6 presents the experimental design and the analysis of the results. The conclusions appear in Section 7.

2. Literature review

There are comprehensive literature reviews on automated procedures for constructing efficient and desirable timetables (Carter and Laporte, 1996; Schaerf, 1999). The approaches adopted have been grouped into four categories (Carter and Laporte, 1996; Burke and Petrovic, 2002): (i) sequential methods, such as graph colouring (Asratian and de Werra, 2002); (ii) clustering methods (Balakrishnan et al., 1992); (iii) constraint-based methods e.g. integer programming (Daskalaki and Birbas, 2005); and (iv) meta-heuristics methods, including GA (Drexel and Salewski, 1997), SA (Thompson and Dowsland, 1998), Taboo Search (TS) (Alvarez-Valdes et al., 2002) and Ant Colony Systems (Azimi, 2005).

The timetabling process is subject to hard constraints that must be satisfied in order to produce feasible timetables and soft constraints, which are not absolutely essential. Table 1 identifies 14 hard constraints and 18 soft constraints that have been considered by previous research. Some hard constraints (e.g. hard constraints numbered 2, 5, 6 and 8) have been widely used, whereas others (e.g. hard constraints numbered 3, 4, 7, 11 and 14) have only been considered by a few researchers. Some constraints have been considered as hard constraints by some researchers and as soft constraints by others. For example, room capacity (Hc_{10}) was considered a hard constraint by Erben and Keppler (1996), Burke and Petrovic (2002) and Socha et al. (2003) but as a soft constraint (Sc_{12}) by Alvarez-valdes et al. (2002), Daskalaki et al. (2004), Gaspero and Schaerf (2001) and Burke et al. (1997).

Burke et al. (1997) and Burke and Petrovic (2002) used datasets that are available on-line from the University of Nottingham (<ftp://ftp.cs.nott.ac.uk/ttp/Data/>), but most other researchers have used unpublished datasets from their own institutions (Erben and Keppler, 1996; Caldeira and Agostinho, 1997; Gaspero and Schaerf, 2001; Alvarez-valdes et al., 2002; Daskalaki et al., 2004).

Both hard and soft constraints used in particular academic institutions may be case specific. For example, Daskalaki et al. (2004) applied integer programming to solve course timetabling problems in Greek universities, where teaching periods are normally 45 min. One of the case studies taken from the Department of Electrical and Computer Engineering at the University of Patras scheduled 72 courses with a total of 211 teaching periods.

Table 1

[illegible]

Alvarez-valdes et al. (2002) applied TS to solve the course timetabling problem using a dataset from the Business School at the University of Valencia, which has 13 teaching periods during each working day. This case study involved the scheduling of 84 courses with a total of 993 weekly teaching periods, delivered by more than 200 academics in 24 rooms.

Stochastic search methods such as GA, SA and TS are particularly suitable for solving complex combinatorial optimisation problems (Nagar et al., 1995). Due to the nature of these problems, it is often impossible to search the whole problem space; although good solutions can usually be found, it is not possible to guarantee that a stochastic search method will find an optimal solution.

GA have several advantages compared with other optimisation approaches: GA represent problems in terms of coding rather than decision variables (Syarif et al., 2002); domain knowledge is only required for coding the problem and for establishing the objective function used for fitness evaluation (Goldberg, 1989); GA perform multiple directional searches using a set of candidate solutions, whereas most conventional methods conduct single-directional searches (Gen and Cheng, 1997). GA have been widely used for solving production and operations management problems during the last few decades (Aytug et al., 2003; Chaudhry and Luo, 2005).

Several researchers have used GA to solve timetabling problems. Erben and Keppler (1996) developed knowledge-augmented genetic operators to prevent illegal offspring. The authors concluded that the results obtained from GA were promising but more work was required in relation to parameter settings. Ergul (1996) developed an examination timetable scheduler called Generic UNiversity Examination Scheduler (GUNES), which was tested using actual data from a typical university. The schedules produced by the GUNES were far better than those that were prepared manually.

SA has its origins in statistical physics, where annealing involves the slow cooling of a solid until it reaches a low-energy ground state. Kirkpatrick et al. (1983) identified the analogy between minimising the cost function of a combinatorial optimisation problem and annealing, in which the temperature (t) is a control parameter. In general, the SA performs a stochastic search of the neighbourhood space. SA starts with an initial state (s), which includes a point randomly selected from the search (solution) space

and an initial temperature. A next point (s^*) is randomly selected from a set of neighbours to the current point. If the objective function associated with the selected neighbour, $E(s^*)$, is better than the objective value of the initial point $E(s)$, it is accepted as the new point for the next neighbourhood search (NS). Otherwise, it is accepted with a probability $e^{-\Delta E/t}$ where ΔE is the change to the objective value. The next step is then another search for a new set of neighbourhoods for the selection process. The temperature is then reduced to focus on a specific region. A cooling rate (r) is specified that determines the amount of computation required and the quality of solutions. The initial temperature value (t_0), the number of iterations to be performed at each temperature, the cooling rate and the termination criterion are specified by the “SA cooling schedule”.

SA has been used to solve many types of combinatorial optimisation problems including scheduling (Kolonko, 1999) and examination timetabling (Thompson and Dowsland, 1996; Azimi, 2005). Kolonko (1999) modified the algorithm to include a small group of candidate solutions for conducting multidirectional searches. Thompson and Dowsland (1996) investigated the cooling schedules for a wide range of examination timetabling problems from universities and suggested that very slow cooling should be used.

Although there is a comprehensive literature on timetabling, only Azimi (2005) has compared the relative performance of alternative stochastic search methods. He applied four metaheuristic methods (SA, GA, TS and Ant Colony Systems) to solve the examination timetabling problem and found that Ant Colony Systems were the best, followed by TS and GA and the worst was SA. However, Azimi specified parameters in an ad hoc fashion. The performance of stochastic search optimisation methods can be sensitive to the particular parameters used. It is therefore important to select the parameters systematically in order to ensure that the best parameters are used. It is not possible to determine whether Azimi's results were due to the metaheuristics themselves or the particular parameters that were used.

3. Problem formulation and assumptions

Daskalaki et al. (2004, p. 117) defined university timetabling as “the process of assigning university courses to specific time periods throughout the 5 working days of the week and to specific classrooms

suitable for the number of students registered and the needs of each course”. A general course timetabling system commonly requires three main types of information: (i) resource information (including the number of classrooms, their size, lecturers’ availability and their relationships to courses); (ii) programme requirements (e.g. the number of students enrolled and the courses required); and (iii) lecturers’ preferences. In many cases, institutions may have rules and regulations that impose additional soft constraints. For example, the number of lectures per day may be limited to prevent staff and student fatigue. Certain periods may be allocated to non-academic activities, such as sport. Timetabling constraints may be categorised as (i) hard constraints which must be satisfied to generate a feasible timetable; and (ii) soft constraints or preferences can be contravened if necessary. [Burke and Petrovic \(2002\)](#) pointed out that it is usually difficult or impossible to generate timetables that satisfy all individuals’ preferences.

Some regulations or guidelines adopted by universities are generally similar, for example neither a lecturer nor room can be assigned to more than one class in the same period. Others may be different, for example, the avoidance of particular periods. Previous research has considered a wide range of hard and soft constraints (as shown in [Table 1](#)). However, the flexibility of the lecturing format specified by lecturers has not been considered by previous research. For example, if there is a subject that requires three teaching periods per week, the staff member may prefer three single sessions, each of which is scheduled on different days; or three consecutive sessions scheduled on a single day; or one session that is two consecutive periods long together with a further single-period session. This flexibility of the lecturing format is considered to be important by many academics and is accommodated by the SOTT presented in this paper.

The SOTT allows four generic hard constraints to be considered: (i) certain periods can be excluded from timetabling for non-academic purposes; (ii) staff can only teach one session at a time; (iii) students can only attend one session at a time; and (iv) facilities can only be used for one session at a time. In addition, four generic soft constraints may be considered: (i) staff can indicate their preferred lecturing format; (ii) staff can choose to maximise the number of teaching-free days for doing other work; (iii) the maximum number of lectures delivered by a member of staff on a particular day can be specified;

and (iv) students’ movement between rooms can be minimised if required.

The repair process described in the next section rectifies infeasible timetables that violate the hard constraints, whilst GA, SA or random search is used to minimise the violation of the soft constraints. The model can be configured to represent a particular university’s requirements at run time through a user-friendly graphical interface.

4. Stochastic Optimisation Timetabling Tool (SOTT)

The SOTT was designed, developed and tested to solve course timetabling problems using GA, SA or random search. The SOTT program was written in Delphi Pascal and consists of more than 10,000 lines of code (including the graphical user interface and the stochastic optimisation algorithms).

4.1. Genetic algorithms (GA)

GA are based on an analogy of biological evolution, in which the fitness of an individual determines its ability to survive and reproduce ([Gen and Cheng, 1997](#); [Goldberg, 1989](#)). In general, GA include a genetic representation of potential solutions, an initial population, genetic operators, an evaluation function, a selection scheme and stopping criteria ([Aytug et al., 2003](#)).

The GA used within the SOTT is shown in [Fig. 1](#). The algorithm includes a repair process that ensures that all infeasible timetables generated during the evolution process are rectified. This prevents clashes and ensures that the rooms are sufficiently large to accommodate the classes. The algorithm evaluates timetables in terms of how effectively they satisfy the soft constraints. The algorithm includes six main steps: (i) gene encoding; (ii) chromosome initialisation and representation; (iii) genetic operations; (iv) repair; (v) fitness measurement; and (vi) chromosome selection. These elements are explained in the following sections.

4.1.1. Gene encoding

Genes may be represented as either numeric (binary or real) or alphanumeric characters. [Blazewicz et al. \(1996\)](#) suggested that the binary chromosome representation is often unsuitable for combinatorial optimisation problems because it is difficult to represent potential solutions. In this work, a gene represents a time slot for a teaching

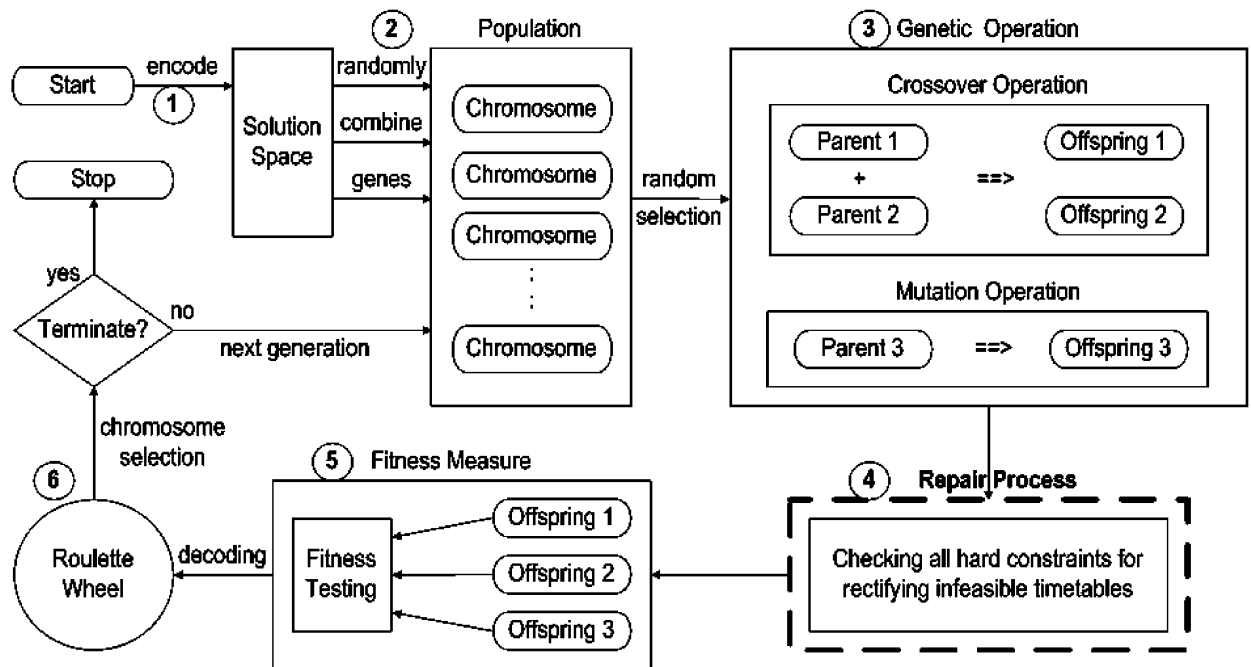


Fig. 1. Structure of the genetic algorithms within the stochastic optimisation timetabling tool.

session. Specified periods can be excluded from the timetabling process. Each gene is an alphanumeric string, which has two parts: a subject code and a room number. For example, if subject A1 is taught in classroom number 1, then it is encoded as SA1, R1. A set of genes representing the whole timetabling problem is then combined to produce an initial chromosome.

4.1.2. Chromosome initialisation and representation

The genes (teaching sessions) associated with each degree programme are randomly assigned to the available periods during the week. The allocation of teaching sessions (genes) is performed sequentially, starting from the beginning of the first period. This allocation process is repeated until all the degree programmes are included in the chromosome. This approach produces a chromosome that is divided into n sub-chromosomes (see Fig. 2), each of which represents a timetable for a degree programme. This prevents timetable clashes by ensuring that each programme only has one class at a time. The whole process is then repeated to produce enough chromosomes to produce a population of the desired size. It should be noted that the number of genes in each sub-chromosome depends on the number of sessions per week. In the SOTT program, both the

number of teaching periods and its duration can be changed easily.

4.1.3. Genetic operations

After a population of chromosomes has been initialised, the next stage within the GA is to randomly select chromosomes that will be subjected to crossover and mutation operations. In crossover, the characteristics of two parents are combined to produce offspring, whilst mutation produces random change in one chromosome. The probability of crossover and mutation are important parameters that affect convergence and the quality of the solution.

There are many types of crossover and mutation operators (see for example, Pongcharoen et al., 2001). The GA within SOTT were developed to include three crossover operators that have been recommended in the literature [One-point and Two-point Crossover (Murata et al., 1996); and Position-Based Crossover (PBX) (Syswerda, 1991)] and three mutation operators [Adjacent Two-day Change and Day Shift Change (DSC) (Murata et al., 1996); and Regenerated Chromosome]. A preliminary experiment, described in Section 5, found that the best crossover operator was PBX and that the DSC mutation operator was the best.

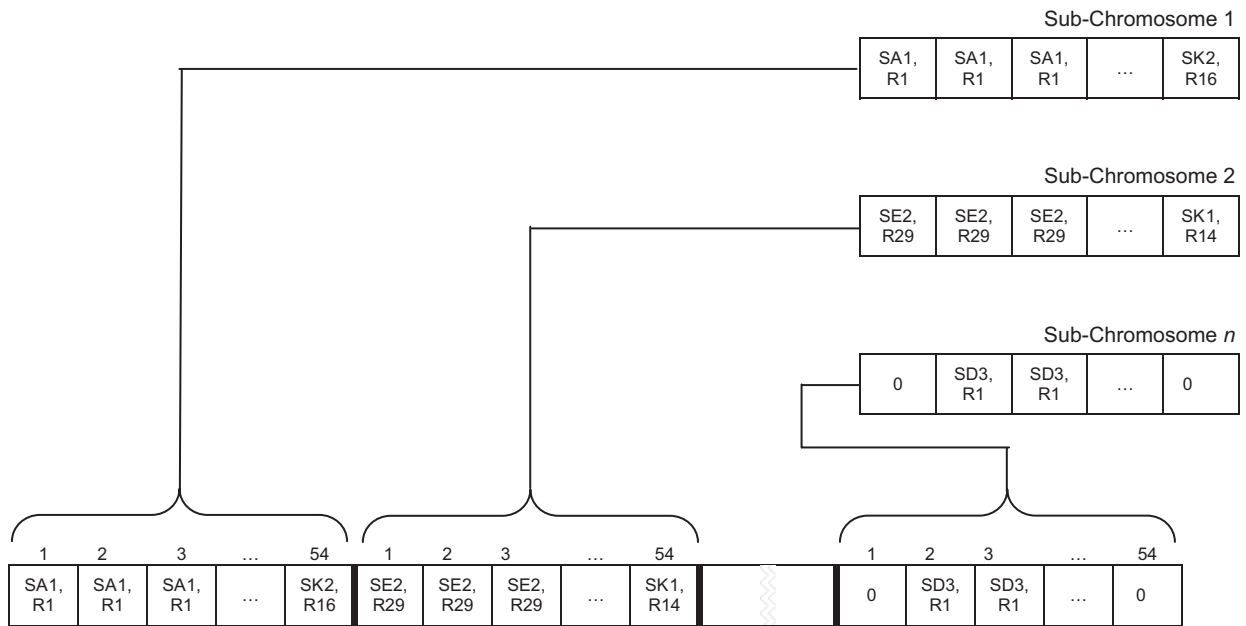


Fig. 2. Chromosome representation.

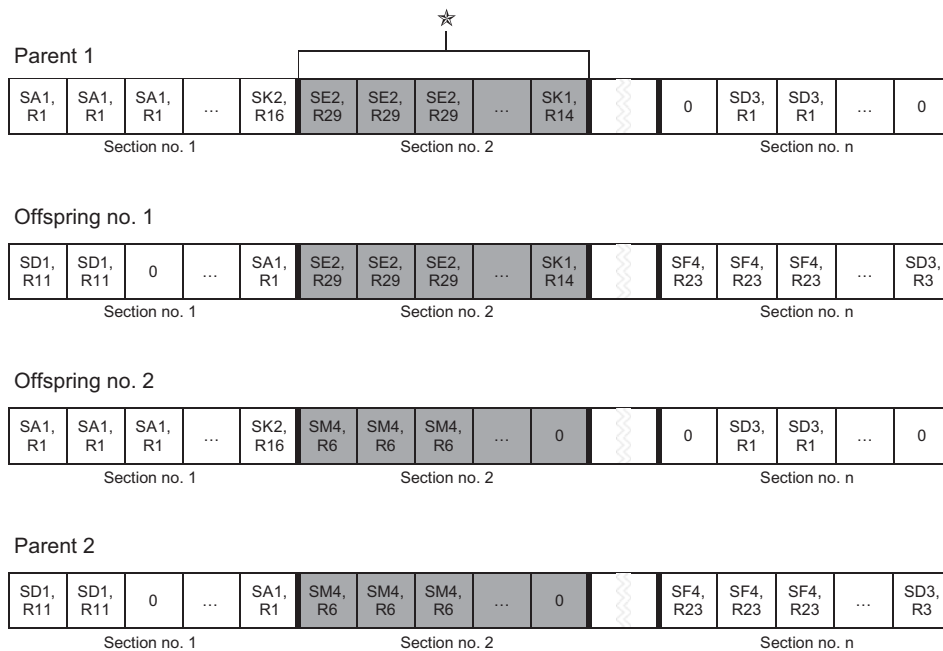


Fig. 3. Position-Based Crossover for timetabling problem.

The PBX operation randomly swaps selected sub-chromosomes between parents to produce offspring. This process is shown in Fig. 3. The second sub-chromosome for the first offspring is inherited from

the first parent whilst the remaining sub-chromosomes are inherited from the second parent. The process is repeated to produce a second offspring. It should be noted that the swapping process in the

PBX crossover is performed on whole sub-chromosomes, each of which represents the students' timetable for a particular programme. This means that the students' timetables produced by the PBX will not violate the first and third hard constraints providing that both parent chromosomes represent feasible timetables.

The DSC mutation operator is illustrated in Fig. 4. The process begins by randomly selecting two separate daily schedules. It then cyclically shifts sessions to the next day. An example is shown in Fig. 4. Tuesday and Friday were randomly selected. The Tuesday schedule was shifted to Wednesday and then the Wednesday schedule was shifted to Thursday and so on. It should be noted that if there were certain periods on any day that are excluded from timetabling for non-academic purposes (for example, if a session on the seventh to twelfth period of Wednesday was excluded), the Wednesday schedule would not be involved in the cyclical rotation. This would mean that the Tuesday schedule would be shifted to Thursday. The Thursday schedule would then be shifted to Friday and the Friday schedule would be shifted to Tuesday. This mechanism prevents the creation of infeasible offspring that violate the hard constraint that excludes specified periods.

4.1.4. Repair process

There are three ways to deal with infeasible solutions: (i) discard them; (ii) apply a high penalty in the fitness function, so that they are unlikely to survive; or (iii) repair them (Blum and Roli, 2003). Discarding infeasible solutions or applying a high penalty is only an option when a large proportion of the population is feasible. Otherwise, the GA process would be terminated if all the chromosomes in the population were infeasible. Several researchers have used a repair process for rectifying infeasible chromosomes that were generated during evolution process (e.g. Caldeira and Agostinho, 1997; Colorni et al., 1998; Pongcharoen et al., 2002, 2004). The repair processes are problem specific. The GA within SOTT includes a repair process that ensures that all infeasible timetables are rectified.

The steps involved in ensuring that the staff are only timetabled to teach one session at a time are shown in Fig. 5. Firstly, the degree programme timetables are mapped into staff timetables. This process starts at the beginning of the first gene and works through to the end of the chromosome. Each gene includes a subject code that is used as a primary key to find the member of staff required. After the transformation, a gene in the lecturers' timetables contains: (i) the subject code (SA1); (ii) the room number (R1); (iii) the degree pro-

Parent												
						Period						
	1	2	3	4	5	6	7	8	9	10	11	12
Mon	SE2,R28	SE2,R28	SE2,R28		SE1,R28	SE1,R28	SE1,R28	SD4,R11	SD4,R11	SD4,R11		
Tue	SD9,R28	SD9,R28	SD9,R28	SD9,R28								
Wed												
Thu	SD7,R9	SD7,R9	SD7,R9		SD6,R28	SD6,R28	SD6,R28					
Fri		SD8,R28	SD8,R28	SD8,R28				SK7,R33	SK7,R33	SK7,R33		
Offspring												
						Period						
	1	2	3	4	5	6	7	8	9	10	11	12
Mon	SE2,R28	SE2,R28	SE2,R28		SE1,R28	SE1,R28	SE1,R28	SD4,R11	SD4,R11	SD4,R11		
Tue		SD8,R28	SD8,R28	SD8,R28				SK7,R33	SK7,R33	SK7,R33		
Wed	SD9,R28	SD9,R28	SD9,R28	SD9,R28								
Thu												
Fri	SD7,R9	SD7,R9	SD7,R9		SD6,R28	SD6,R28	SD6,R28					

Fig. 4. Day Shift Change mutation operator.

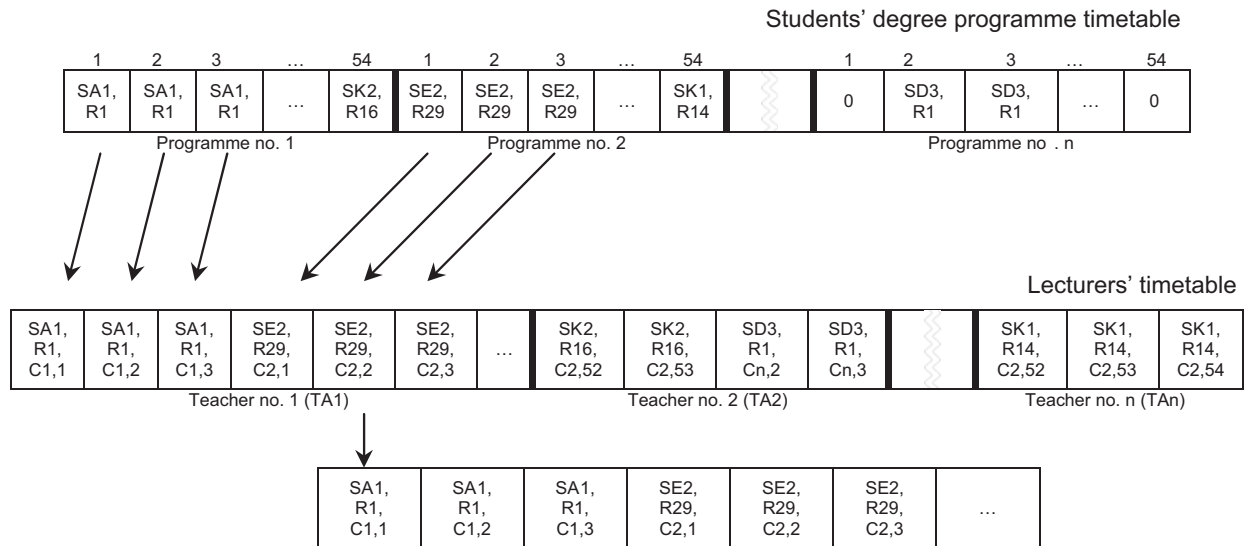


Fig. 5. The repair process for an infeasible timetable that violates Hc_2 .

gramme code (C1); and (iv) the period (slot 1). Each sub-chromosome represents a timetable for a particular member of staff.

The next step is to check whether the staff timetables are feasible. In the example shown in Fig. 5, the first lecturer has a class for subject SA1 with three consecutive time slots of 1, 2 and 3 for the degree programme C1 in room number 1. However, the next three genes indicate a clash because the lecturer also has a class for subject SE2 for class C2 in room 29 at the same time. This means that the timetable for the first lecturer is infeasible and needs to be corrected by the repair process.

The repair process starts by identifying the classes that clash (in this case SA1 and SE2). One of the classes is then randomly selected for reallocation (say SA1). The algorithm then searches through the member of staff's timetable, the room timetable and the programme timetable to identify a time slot that can accommodate the class to be reallocated. If there is an appropriate time slot the class is moved. Otherwise, it is necessary to fill in gaps within the lecturer's timetable. To do this a subject is chosen randomly and the classes are split into shorter sessions (for example, a three-period class may be split into a 1-2, 2-1 or 1-1-1 format); time slots are identified and the classes are reallocated. Unfortunately, this will contravene the first soft constraint if the lecturer had specified a preference for the 3-0 pattern. If this is the case, the fitness function (described in the next section) applies a penalty.

4.1.5. Fitness measurement

The next stage of the GA is to measure the quality of each timetable (chromosome) in terms of a fitness value. There is a soft constraint violation index (SCVI) associated with each soft constraint (described in Section 2). For example, if $SCVI_1 = 10$, this means that the first soft constraint has been violated 10 times. The overall objective is to minimise the total violation index (TVI), which is the weighted sum of the violation indices. This fitness function is

$$TVI = w_1 SCVI_1 + w_2 SCVI_2 + w_3 SCVI_3 + w_4 SCVI_4$$

where w_1 – w_4 are the weighting factors and $SCVI_1$ – $SCVI_4$ are the violation indices associated with constraints Sc_1 – Sc_4 .

The relative weights given to the four SCVIs specify the relative priorities to be given to the soft constraints in the optimisation process. A soft constraint can be emphasised by giving it a high weighting or it can be ignored by specifying a weighting of zero. This flexibility helps the model to cope with the soft constraints that may be used or unused for course timetabling in particular universities. In this study, equal weights were used for the soft constraint violation indices.

4.1.6. Chromosome selection

The final stage of the GA is to select the chromosomes that are to survive to the next

generation. The Roulette wheel approach (Goldberg, 1989) was used for chromosome selection. It is based on the Delphi Pascal random number generator that produces numbers in the range 0–1. The probability of survival and number of replicates of a chromosome in the next generation is determined by its fitness. The GA process is repeated until the required number of generations has been completed.

4.2. Simulated annealing (SA)

In this work the general SA procedure was modified to include the repair process (as described in Section 4.1.4) for rectifying infeasible solutions produced during the process of NS. The SA parameters included initial (t_0) and final ($t_{\min}$) temperatures, NS and cooling rate (r). In some situations (e.g. when the number of candidate solutions generated or the computational time is limited), the number of iterations (N) for NS may be specified. The cooling rate is determined using the formula $(t_{\min}/t_0)^{1/N-1}$ (Glass and Potts, 1996). The pseudo code of the proposed SA is shown in Fig. 6.

5. Case study

Although the particular course timetabling problems considered in this paper are based on the specific requirements of the Faculty of Science and Technology from a collaborating university in Thailand, they are typical of those faced by other universities. The faculty offers bachelor-level degree programmes, which are of 4 years' duration. For timetabling purposes, each day consists of 12 50 min time slots (see Fig. 7). The periods excluded from the timetables for non-academic purpose are shown in grey colour. The SOTT is generic and there is a

graphic user interface within the SOTT that allows users to easily modify the number and duration of time slots. Table 2 summarises the characteristics of two sets of timetabling data, which were collected and used to test the model. Some subjects may be taught by more than one lecturer.

In the case study, five specific hard constraints (Hc) were considered:

- Hc₁ the University regulations state that there must be no classes on Wednesday afternoons;
- Hc₂ staff can only teach one session at a time;
- Hc₃ the University regulations state that there must be no classes during the fourth period on Mondays;
- Hc₄ students can only attend one session at a time;
- Hc₅ facilities can only be used for one session at a time.

In addition, four soft constraints (Sc) were included:

- Sc₁ the lecturing format preferred by the staff. For example, if there is a subject that requires three teaching periods per week, the staff member may prefer a single session that is three periods long; or three sessions that are one period long; or one session that is two periods long and a single period session;
- Sc₂ staff may choose to maximise the number of teaching-free days for doing other work;
- Sc₃ the daily load for staff should not exceed a specified number of periods;
- Sc₄ students should remain in the same classroom as much as possible.

```

Specify initial ( $t_0$ ) and final ( $t_{\min}$ ) temperatures and current solution ( $s$ )
 $t_n = t_0$ 
 $r = (t_{\min}/t_0)^{1/N-1}$  ( $N$  is no. of iteration by reducing from initial to final temperatures)
Repeat
  Randomly select neighbour solutions ( $s^*$ ) of current solution ( $s$ )
  If  $s^*$  is infeasible then repair  $s^*$ 
  Calculate the modification of the objective value [ $\Delta E = E(s^*) - E(s)$ ]
  If [ $(\Delta E \leq 0)$  or ( $\text{random}[0,1] \leq \exp(-\Delta E/t)$ )] then  $s = s^*$ 
  Reduce temperature ( $t_{n+1} = rt_n$ ) (where  $n = 1, 2, \dots, N$ )
Until a stopping criterion is satisfied ( $t_n \leq t_{\min}$ )

```

Fig. 6. Pseudo code of the Simulated Annealing including repair process.

Days	Period												
	1	2	3	4	Lunch time	5	6	7	8	9	10	11	12
Monday	1	2	3			4	5	6	7	8	9	10	11
Tuesday	12	13	14	15		16	17	18	19	20	21	22	23
Wednesday	24	25	26	27		28	29	Campus activities					
Thursday	30	31	32	33		34	35	36	37	38	39	40	41
Friday	42	43	44	45		46	47	48	49	50	51	52	53

Fig. 7. The University week.

Table 2
Timetabling case studies

Datasets	No. of degree programmes	No. of lecturers	No. of subjects	No. of classes
Small size	10	46	69	251
Large size	30	111	128	758

6. Experimental results and analysis

A three-stage sequential experiment was carried out in this work. The first was a preliminary experiment to find the best settings of the GA and SA parameters that used a full factorial design of experiments approach (Montgomery, 2001). The second stage was a comparative experiment that compared the performance of the different stochastic search methods within SOTT using the best parameters that had been identified during the first stage. The last experiment identified the significance of the repair process. The design of each experiment, the results and analyses are provided in the following sections.

6.1. Preliminary experiment

The performance of the stochastic search methods depends on the parameters and operators used. The preliminary experiment was designed to identify the best configurations for both GA and SA.

6.2. Experiment A: investigation on the GA parameters and operators

The experimental factors and levels considered in Experiment A are shown in Table 3. The first factor, a combination of the population size and the number of generations (P/G), determined the amount of search and the computational time. To limit the execution time the total number of

Table 3
GA experimental factors and its levels

Factors	Level values		
	Low (−1)	Medium (0)	High (+1)
Combination of population size and number of generations (P/G)	250/200	500/100	1000/50
Probability of crossover (% C)	0.6	0.75	0.9
Probability of mutation (% M)	0.01	0.1	0.2
Crossover Operator (COP)	One point	Two point	Position based
Mutation Operator (MOP)	Re-generation	Day shift change	Adjacent 2 day

chromosomes generated (total candidate solutions) was fixed at 50,000. Test runs had identified that this was sufficient to achieve convergence within less than 1 h on a personal computer. The levels of the probabilities of crossover (% C) and mutation (% M) were chosen on the basis of the results of De Jong (1975), Erben and Keppler (1996) and Todd (1997). The fourth factor was the crossover operator (one point, two point (Murata et al., 1996) and PBX (Syswerda, 1991)). The last factor was the mutation operator used (re-generation, day shift change (Murata et al., 1996) and adjacent 2-day mutation (Murata et al., 1996)). This experiment used a full factorial design that was replicated five times using the small problem. Table 4 shows the analysis of variance (ANOVA) results obtained from 1215 runs within the experiment.

All the main factors, apart from the probability of crossover, were statistically significant with a 95% level of confidence (p value of ≤ 0.05). Although the % C was not statistically significant in this case, a value of % C had to be specified for the GA to work.

Table 4
Analysis of variance (ANOVA) table on GA parameters and operators

Source	DF	Sum of squares	Mean square	F	p
P/G	2	208.933	8.680	4.40	0.012
%C	2	7.323	4.529	2.30	0.101
%M	2	14.079	8.163	4.14	0.016
COP	2	227.264	100.319	50.87	0.000
MOP	2	102.138	24.790	12.57	0.000
Seed	4	771.577	192.894	97.81	0.000
P/G $\times$ %C	2	2.581	1.291	0.65	0.520
P/G $\times$ %M	2	45.619	22.809	11.57	0.000
P/G $\times$ COP	2	14.167	7.083	3.59	0.028
P/G $\times$ MOP	2	0.541	0.270	0.14	0.872
%C $\times$ %M	2	0.719	0.359	0.18	0.833
%C $\times$ COP	2	6.626	3.313	1.68	0.187
%C $\times$ MOP	2	0.363	0.181	0.09	0.912
%M $\times$ COP	2	2.141	1.070	0.54	0.581
%M $\times$ MOP	2	7.681	3.841	1.95	0.143
COP $\times$ MOP	2	1.796	0.898	0.46	0.634
Error	1180	2327.222	1.972		
Total	1214	3740.770			

The probability of crossover was therefore arbitrarily set at 0.9. Main effect plots for each of the main factors suggested that the best solutions were obtained with: (i) a population of 500; (ii) 100 generations; (iii) a probability of mutation of 0.2; (iv) the PBX operation; and (v) the DSC mutation operation. These settings were used in the remaining experiments. It should be noted that the random seed was also found to be statistically significant. It was therefore a potential nuisance factor.

6.3. Experiment B: investigation on the SA parameters and operators

This experiment investigated appropriate settings for the SA parameters. It included the initial and final temperatures and the NS operator. The selection of experimental factors and levels are shown in Table 5.

The first two factors (initial (t_0) and final ($t_{\min}$) temperatures) were considered in the range of 50–5000 and 0.0001–1, respectively. These values were chosen in accordance with previous research (Lu, 1991; Glass and Potts, 1996; Triki et al., 2005). Three mutation operators (described in Section 4.1.3) were applied as NS operations. The experiment was based on a full factorial experiment that

Table 5
SA experimental factors and its levels

Factors	Level values		
	Low (−1)	Medium (0)	High (+1)
Initial temperature (t_0)	50	500	5000
Final temperature ($t_{\min}$)	0.0001	0.01	1
Neighbourhood Search (NS) Operator	Re-generation	Day shift change	Adjacent 2 day

Table 6
Analysis of variance (ANOVA) table on SA parameters

Source	DF	Sum of squares	Mean square	F	p
T_0	2	0.28	0.16	0.04	0.962
$T_{\min}$	2	148.55	62.44	15.68	0.000
NS	2	7023.66	3237.76	813.28	0.000
$T_0 \times T_{\min}$	2	1.96	0.56	0.14	0.869
$T_0 \times$ NS	2	0.82	0.69	0.17	0.842
$T_{\min} \times$ NS	2	180.52	89.37	22.45	0.000
$T_0 \times T_{\min} \times$ NS	2	1.11	0.56	0.14	0.869
Error	120	477.73	3.98		
Total	134	7834.64			

was replicated five times using the small problem. The experimental results obtained from 135 runs were analysed using the general linear model form of ANOVA shown in Table 6.

From Table 6, it can be seen that the final temperature ($t_{\min}$) and NS operation were statistically significant with a 95% confidence level. Main effect plots suggested that SA produced the best results with the regeneration operator and $t_{\min} = 0.0001$. The initial temperature (t_0) was arbitrarily set at 50.

6.4. Comparative experiment

This experiment compared the performance of the stochastic search methods within the SOTT program. This experiment was based on the small problem. The results obtained were compared in terms of the quality of results produced and the execution time required. The stochastic search methods were limited to 50,000 candidate solutions to maintain reasonable execution times. The best parameter settings identified by the

previous experiments were used. Each experiment was replicated 30 times with different random number seeds. The experiments were run on a personal computer with a 930 MHz Intel processor and 512 MB of RAM.

Table 7 compares the methods in terms of the mean and standard deviation of the TVI and the execution times. SA produced a lower mean and standard deviation than the GA and the random search method. However, the GA was quicker than the other methods. The execution time was influenced by the complexity of the algorithm and the time taken to perform the repair process. The random method was the simplest algorithm but it required most execution time because all 50,000 solutions produced needed to be repaired. The GA was more complicated than SA but required less repair time because it produced less infeasible solutions.

Table 7
Results obtained from the SOTT using GA, SA and RM

Methods	Total violation index (TVI)		Execution time (min)	
	Mean	Std. dev.	Mean	Std. dev.
Genetic algorithms (GA)	3.07	1.74	47.40	0.62
Simulated annealing (SA)	0.83	0.38	102.43	1.10
Random method (RM)	10.73	0.74	107.09	3.51

6.5. Experiment to evaluate the repair process

This experiment aimed to identify the significance of the repair process that was used to rectify infeasible timetables generated during the evolution process. Experiments were carried out for both the small and large problems. During the experiments, the number of infeasible chromosomes rectified within each generation (relating to each of five hard constraints, H_c) was recorded. Figs. 8 and 9 show the number of infeasible chromosomes repaired during each generation using the small and large problems, respectively. From both figures, it can be seen that all the chromosomes generated during the evolution process always satisfied the first and the fourth hard constraints. H_{c4} was always satisfied due to the chromosome initialisation process (described in Section 4.1.2). H_{c1} was always satisfied due to the crossover and mutation operators used (described in Section 4.1.3).

Fig. 8 shows that most chromosomes generated during the evolution process did not satisfy three hard constraints (H_{c2} , H_{c3} and H_{c5}). Of the 500 chromosomes in the first generation, 491 chromosomes were repaired because they violated H_{c2} whilst 155 and 25 chromosomes were also repaired because they violated H_{c5} and H_{c3} , respectively. The large number of repairs is necessary, indicating that the repair process was a crucial component of the methodology. The number of infeasible

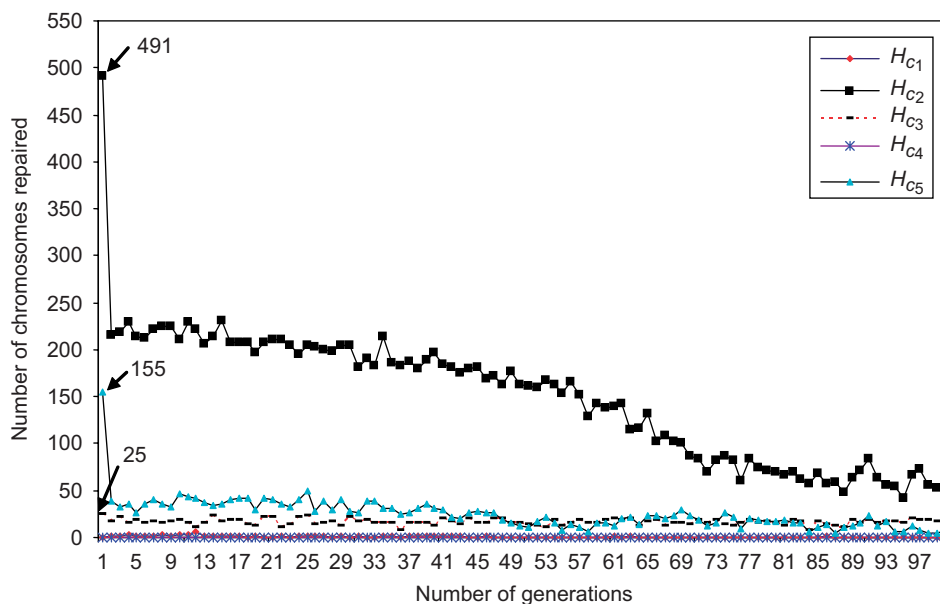


Fig. 8. Number of infeasible chromosomes found using small problem.

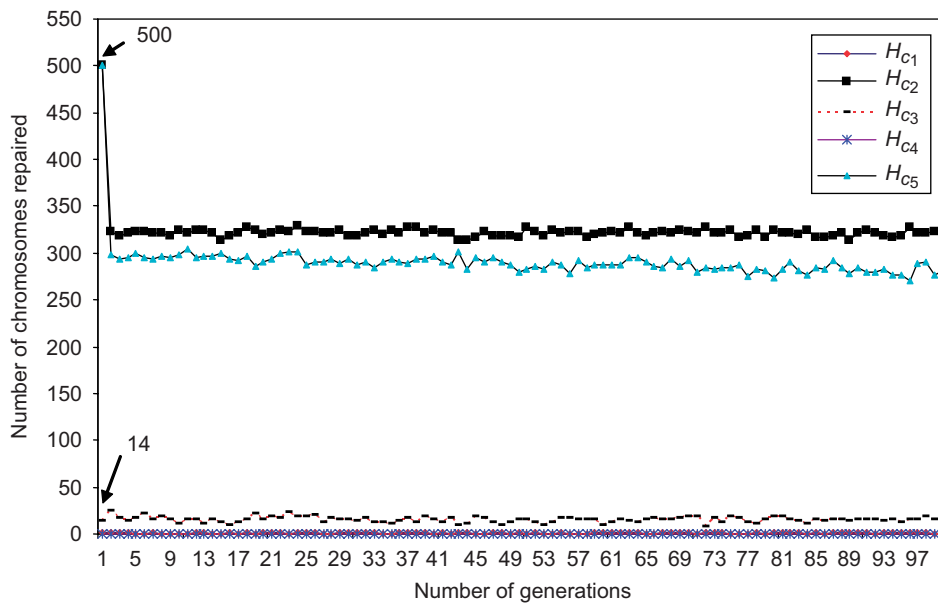


Fig. 9. Number of infeasible chromosomes found using large problem.

timetables declined dramatically in the early generations and then gradually reduced for the remaining generations especially for the small problem. For the large problem, the number of infeasible timetables that violated the hard constraints was relatively constant after the first generation (see Fig. 9). This is because the larger problem was far more complex, was more constrained and had a larger search space.

7. Conclusions

University timetabling is an NP-hard problem, which means that the amount of computation required increases exponentially with problem size. The timetabling process is subject to hard constraints that must be satisfied in order to produce feasible timetables and soft constraints that are desirable, but not absolutely essential. This paper has described a Stochastic Optimisation Timetabling Tool (SOTT) that has been developed for university course timetabling. Genetic Algorithms (GA), Simulated Annealing (SA) and random search are embedded in the SOTT program. The algorithms include a repair process, which ensures that all infeasible timetables generated during the evolution process are rectified. This prevents clashes and ensures that rooms are sufficiently large to accommodate the classes. The SOTT program developed allows lecturers to specify the lecture

format for their courses. The algorithm evaluates timetables in terms of soft constraints: minimising student movement; avoiding fragmentation in the timetable for students and lecturers; and satisfying lecturers' preferences for the timing of classes.

The algorithms were tested using two sets of timetabling data from a collaborating university in Thailand. The repair process was shown to be a crucial aspect of the methodology. The best experimental results obtained from SOTT satisfied all hard constraints for both the small and large problems. This work applied a systematic method for selecting parameters and operators. It has also compared alternative stochastic search methods for solving timetables through a series of statistical experimental designs. These issues have mostly been ignored by previous research. The experimental results produced for the small problem showed that the timetables obtained from the stochastic search methods almost satisfied all the soft constraints. The average execution times using GA were 54% and 56% faster than SA and Random Method (RM), respectively. However, the results obtained from SA yielded a slightly lower total violation index than those produced by GA and RM.

Acknowledgements

This work was supported by the Thailand Research Fund under Grant no. MRG4880126

and the Naresuan University Research Fund (NURF49). The authors would also like to acknowledge Mr. Warattapop Chainate for programming and debugging support.

References

- Alvarez-Valdes, R., Crespo, E., Tamarit, J.M., 2002. Design and implementation of a course scheduling system using tabu search. *European Journal of Operational Research* 137 (3), 512–523.
- Asratian, A.S., de Werra, D., 2002. A generalised class–teacher model for some timetabling problems. *European Journal of Operational Research* 143 (3), 531–542.
- Aytug, H., Khouja, M., Vergara, F.E., 2003. Use of genetic algorithms to solve production and operations management problems: A review. *International Journal of Production Research* 41 (17), 3955–4009.
- Azimi, Z.N., 2005. Hybrid heuristics for examination timetabling problem. *Applied Mathematics and Computation* 163 (2), 705–733.
- Balakrishnan, N., Lucena, A., Wong, R.T., 1992. Scheduling examinations to reduce second-order conflicts. *Computers and Operations Research* 19, 353–361.
- Bardadym, V.A., 1996. Computer-aided school and university timetabling: The new wave. In: Burke, E., Ross, P. (Eds.), *Practice and Theory of Automated Timetabling*, Lecture Notes in Computer Science, vol. 1153. Springer, Berlin, pp. 22–45.
- Blazewicz, J., Ecker, K.H., Pesch, E., Schmidt, G., Weglarz, J., 1996. *Scheduling Computer and Manufacturing Processes*. Springer, Berlin.
- Blum, C., Roli, A., 2003. Metaheuristics in combinatorial optimisation: Overview and conceptual comparison. *ACM Computing Surveys* 35 (3), 268–308.
- Burke, E.K., Petrovic, S., 2002. Recent research directions in automated timetabling. *European Journal of Operational Research* 140 (2), 266–280.
- Burke, E.K., Kingston, J., Jackson, K., Weare, R., 1997. Automated university timetabling: The state of the art. *The Computer Journal* 40 (9), 565–571.
- Caldeira, J.P., Agostinho, C.R., 1997. School timetabling using genetic search. In: *Proceedings of the Practice and Theory of Automated Timetabling*, University of Toronto, Toronto, pp. 115–122.
- Carter, M.W., Laporte, G., 1996. Recent developments in practical examination timetabling. In: Burke, E., Ross, P. (Eds.), *Practice and Theory of Automated Timetabling*, Lecture Notes in Computer Science, vol. 1153. Springer, Berlin, pp. 3–21.
- Chaudhry, S.S., Luo, W., 2005. Application of genetic algorithms in production and operation management: A review. *International Journal of Production Research* 43 (19), 4083–4101.
- Colomi, A., Dorigo, M., Maniezzo, V., 1998. Metaheuristics for high school timetabling. *Computational Optimisation and Application Journal* 9, 277–298.
- Costa, D., 1994. A tabu search algorithm for computing an operational timetable. *European Journal of Operational Research* 76 (1), 98–110.
- Daskalaki, S., Birbas, T., 2005. Efficient solutions for a university timetabling problem through integer programming. *European Journal of Operational Research* 160 (1), 106–120.
- Daskalaki, S., Birbas, T., Housos, E., 2004. An integer programming formulation for a case study in university timetabling. *European Journal of Operational Research* 153 (1), 117–135.
- De Jong, K.A., 1975. Analysis of behaviour of a class of genetic adaptive systems. Doctoral Dissertation, University of Michigan, Michigan.
- Drexel, A., Salewski, F., 1997. Distribution requirements and compactness constraints in school timetabling. *European Journal of Operational Research* 102 (1), 193–214.
- Erben, W., Keppler, J., 1996. A genetic algorithm solving a weekly course-timetabling problem. In: Burke, E., Ross, P. (Eds.), *Practice and Theory of Automated Timetabling*, Lecture Notes in Computer Science, vol. 1153. Springer, Berlin, pp. 198–211.
- Ergul, A., 1996. GA based examination scheduling experience at Middle East Technical University. In: Burke, E., Ross, P. (Eds.), *Practice and Theory of Automated Timetabling*, Lecture Notes in Computer Science, vol. 1153. Springer, Berlin, pp. 212–226.
- Gaspero, L.D., Schaerf, A., 2001. A case-study for Easy-Local++: The course timetabling problem. Research Report, Università degli Studi di Udine, Italy.
- Gen, M., Cheng, R., 1997. *Genetic Algorithms and Engineering Design*. Wiley, New York.
- Glass, C.A., Potts, C.N., 1996. A comparison of local search methods for flow shop scheduling. *Annals of Operations Research* 63, 489–509.
- Goldberg, D.E., 1989. *Genetic Algorithms in Search, Optimisation and Machine Learning*. Addison-Wesley, Reading, MA.
- Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P., 1983. Optimisation by simulated annealing. *Science* 220 (4598), 671–679.
- Kolonko, M., 1999. Some new results on simulated annealing applied to the job shop scheduling problem. *European Journal of Operational Research* 113, 123–136.
- Lu, Y., 1991. Solving combinatorial optimisation problems by simulated annealing, genetic algorithms and neural networks. Master's Thesis, The University of Minnesota, USA.
- Montgomery, D.C., 2001. *Design and Analysis of Experiments*, fifth ed. Wiley, NY.
- Murata, T., Ishibuchi, H., Tanaka, H., 1996. Genetic algorithms for flow shop scheduling problems. *Computers and Industrial Engineering* 30 (4), 1061–1071.
- Nagar, A., Haddock, J., Heragu, S., 1995. Multiple and bicriteria scheduling: A literature survey. *European Journal of Operational Research* 81, 88–104.
- Pongcharoen, P., Stewardson, D.J., Hicks, C., Braiden, P.M., 2001. Applying designed experiments to optimise the performance of genetic algorithms used for scheduling complex products in the capital goods industry. *Journal of Applied Statistics* 28 (3&4), 441–455.
- Pongcharoen, P., Hicks, C., Braiden, P.M., Stewardson, D.J., 2002. Determining optimum genetic algorithm parameters for scheduling the manufacturing and assembly of complex products. *International Journal of Production Economics* 78 (3), 311–322.
- Pongcharoen, P., Hicks, C., Braiden, P.M., 2004. The development of genetic algorithms for the finite capacity scheduling of complex products, with multiple levels of products

- structure. *European Journal of Operational Research* 152 (1), 215–225.
- Schaerf, A., 1999. A survey of automated timetabling. *Artificial Intelligence Review* 13 (2), 87–127.
- Socha, K., Sampels, M., Manfrin, M., 2003. Ant algorithms for the university course timetabling problem with regard to the state-of-the-art. In: *Proceedings of the Third European Workshop on Evolutionary Computation in Combinatorial Optimisation*. Springer, UK.
- Souza, M.J.F., Maculan, N., Ochi, L.S., 2001. A GRASP-tabu search algorithm to solve a school timetabling problem. In: *Proceedings of the Fourth Metaheuristics International Conference*, Porto, Portugal, pp. 53–58.
- Syarif, A., Yun, Y., Gen, M., 2002. Study on multi-stage logistic chain network: A spanning tree-based genetic algorithm approach. *Computers & Industrial Engineering* 43 (1–2), 299–314.
- Syswerda, G., 1991. Scheduling optimisation using genetic algorithm. *Handbook of Genetic Algorithms*, pp. 332–349.
- Thompson, J.M., Dowsland, K.A., 1996. General cooling schedules for a simulated annealing based timetabling system. In: Burke, E., Ross, P. (Eds.), *Practice and Theory of Automated Timetabling*, Lecture Notes in Computer Science, vol. 1153. Springer, Berlin, pp. 345–363.
- Thompson, J.M., Dowsland, K.A., 1998. A robust simulated annealing based examination timetabling system. *Computers and Operations Research* 25 (7–8), 637–648.
- Todd, D., 1997. Multiple criteria genetic algorithms in engineering design and operations. Doctoral Dissertation, University of Newcastle upon Tyne, Newcastle upon Tyne, UK.
- Triki, E., Collette, Y., Siarry, P., 2005. A theoretical study on the behaviour of simulated annealing leading to a new cooling schedule. *European Journal of Operational Research* 166, 77–92.
- Wren, A., 1996. Scheduling, timetabling and rostering—A special relationship. In: Burke, E., Ross, P. (Eds.), *Practice and Theory of Automated Timetabling*, Lecture Notes in Computer Science, vol. 1153. Springer, Berlin, pp. 46–75.