Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 841748, 16 pages
http://dx.doi.org/10.1155/2015/841748

Research Article

Modifying Regeneration Mutation and Hybridising Clonal Selection for Evolutionary Algorithms Based Timetabling Tool

Thatchai Thepphakorn,[1,2] Pupong Pongcharoen,[2] and Chris Hicks[3]

[1]*Faculty of Industrial Technology, Pibulsongkram Rajabhat University, Phitsanulok 65000, Thailand*
[2]*Industrial Engineering Department, Centre of Operations Research and Industrial Applications (CORIA), Faculty of Engineering, Naresuan University, Phitsanulok 65000, Thailand*
[3]*Newcastle University Business School, Newcastle University, Newcastle upon Tyne NE1 7RU, UK*

Correspondence should be addressed to Pupong Pongcharoen; pupongp@nu.ac.th

Received 19 September 2014; Accepted 16 December 2014

Academic Editor: Yudong Zhang

This paper outlines the development of a new evolutionary algorithms based timetabling (EAT) tool for solving course scheduling problems that include a genetic algorithm (GA) and a memetic algorithm (MA). Reproduction processes may generate infeasible solutions. Previous research has used repair processes that have been applied after a population of chromosomes has been generated. This research developed a new approach which (i) modified the genetic operators to prevent the creation of infeasible solutions before chromosomes were added to the population; (ii) included the clonal selection algorithm (CSA); and the elitist strategy (ES) to improve the quality of the solutions produced. This approach was adopted by both the GA and MA within the EAT. The MA was further modified to include hill climbing local search. The EAT program was tested using 14 benchmark timetabling problems from the literature using a sequential experimental design, which included a fractional factorial screening experiment. Experiments were conducted to (i) test the performance of the proposed modified algorithms; (ii) identify which factors and interactions were statistically significant; (iii) identify appropriate parameters for the GA and MA; and (iv) compare the performance of the various hybrid algorithms. The genetic algorithm with modified genetic operators produced an average improvement of over 50%.

1. Introduction

Metaheuristics are a class of approximation methods that solve complex optimisation problems that are beyond the scope of classical heuristics and optimisation methods [1]. They have been widely used to solve nondeterministic polynomial (NP) hard problems within acceptable computational time [2]. However, metaheuristic methods are stochastic and cannot guarantee an optimal solution [3]. Evolutionary algorithms (EA) are particularly popular metaheuristics and have been widely applied in the literature. There are three types of EA: evolutionary programming, evolutionary strategies, and genetic algorithms (GA) [4]. Evolutionary programming and evolutionary strategies have been used to solve continuous optimisation problems whilst GA have been mainly used for solving discrete optimisation problems [5].

GA are population based, stochastic search approaches that were inspired by biological evolution. GA include crossover and mutation genetic operations, which are artificial processes for producing new chromosomes. Chromosome selection mimics natural evolution to select a new population for next generation based on individual fitness [6]. GA have been widely applied to solve various optimisation problems [7] including production scheduling [8], course timetabling [9], examination timetabling [10], container packing [11], travelling salesman [12], bankruptcy prediction [13], and machine layout [14]. However, the simple GA may not be effective for solving problems with a very large solution space and many constraints [5].

The term memetic algorithm (MA) is used to describe evolutionary algorithms in which local search is used to a large extent [15]. MAs have received considerable attention

from researchers in many fields [5] including job shop scheduling [16], vehicle routing [17], exam timetabling [18], and nurse scheduling [19]. The MA has also been applied to solve course timetabling problems [20–25].

Genetic operations frequently produce infeasible solutions, which can be (i) discarded; (ii) penalised; or (iii) repaired [6]. However, discarding infeasible solutions or applying a high penalty is only an option when a large proportion of the chromosomes are feasible [9]. Gen and Cheng [6] recommended the repair option. In the algorithms adopted by previous research, the mutation and crossover processes produce a population that includes feasible and infeasible chromosomes. The infeasible chromosomes are then identified and repaired. However, for very large problems that are subject to numerous constraints, the repair process is likely to be highly complex and difficult to design [6]. A complex repair process may be very time consuming [5]. The literature has not considered the development of modified crossover and mutation operators that only produce feasible chromosomes. Such a strategy would likely be more computationally efficient, which would make it possible to conduct more searches within a given execution time.

The performance of evolutionary algorithms is dependent upon the parameters used (such as the population size, number of generations, and the probabilities of crossover and mutation). It is important to identify appropriate values for the parameters in order to obtain the best solutions [26]. There are four experimental strategies: (i) the best-guess approach; (ii) the trial and error approach; (iii) the one factor at a time experimental strategy; and (iv) the factorial experiment [27]. Montgomery [27] suggested that the factorial experiment is the best approach for dealing with several factors. The strategy is to systematically vary the factors together, instead of one at a time. Thus, it is best to use a factorial experiment when investigating appropriate parameter settings for metaheuristic methods. The approach is more reliable, leads to better results, and is more efficient than the alternatives [28].

Artificial immune systems (AIS) are metaheuristics that were inspired by the immune system in biology [29]. There are four main variants of the AIS: danger theory, immune network algorithm (INA), negative selection algorithm (NSA), and clonal selection algorithm (CSA) [30]. AIS have been successfully applied in three application areas: (i) learning; (ii) anomaly detection; and (iii) optimisation [31]. There is only a limited literature on the use of AIS for timetabling. He et al. [32] applied CSA to solve university course timetabling problems in Singapore and benchmark problems. The CSA produced better timetables than GA for all of the problems considered. Malim et al. [33] applied the INA, NSA, and CSA to solve course timetabling problems. The INA produced timetables with the best average fitness, whereas CSA was best in terms of average execution time. Bhaduri [34] was the only researcher to develop a hybrid AIS for timetabling, called GAIN, which included the INA and GA. The GAIN was able to produce optimal feasible timetables faster than GA. However, other researchers have used AIS hybrids in other domains. For example, Zhang et al. [35] combined AIS, the chaos operator, and particle swarm optimisation (PSO), to produce CIPSO, which was used for transportation planning. The approach outperformed GA and PSO in respect of route optimality and convergence time.

The objectives of this paper were to

(i) briefly review the literature on evolutionary algorithms and course timetabling;

(ii) explain the development, process, and features of a novel timetabling tool that incorporates genetic algorithms, local search, the clonal selection algorithm, roulette wheel selection, and the elitist strategy;

(iii) outline a new modified regeneration mutation operator (MRMO) that is based on roulette wheel selection;

(iv) describe the development of a novel local search (LS) algorithm that guarantees the feasibility of new chromosomes generated with the MRMO. This hybrid is called the modified memetic algorithm (MMA);

(v) explain experiments that demonstrated that performance can be improved by the MRMO and MMA using an elitist strategy (ES);

(vi) outline new hybrid algorithms that include the clonal selection algorithm, MRMO+CSA, and MMA+CSA;

(vii) describe the testing of the tool using widely used benchmark problems;

(viii) explain the experimental design and analysis used to investigate the significance of GA parameters and interactions and to identify appropriate parameter settings;

(ix) provide a comparison of the performance of the proposed hybridisations and the other hybridisations of EA which were used to find the best timetables using 14 benchmarking obtained from the literature.

The next section of this paper briefly reviews course timetabling problems, which is followed by a detailed outline of the development of the evolutionary algorithms based timetabling tool and its features, the experimental programme, results, analysis and conclusions.

2. Course Timetabling Problems

"Timetabling is 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" [36, page 266]. There are many types of timetabling problems including employee timetabling, sports timetabling, transportation timetabling, and educational timetabling [28]. Course timetabling arises every academic year in educational institutions (such as high schools, colleges, or universities) and is solved by academic or administrative staff with or without an automated timetabling tool. Course timetabling is known to be a NP-hard problem [37], in which the computational time required to find a solution increases exponentially with problem size [9].

Timetabling problems include hard constraints that must be satisfied in order to produce a feasible timetable and soft constraints, which are desirable but may be violated

[9, page 903]. In the case of course timetabling, it is necessary for a timetable to be feasible for students, lecturers, and classrooms [23]. In a university, a degree programme comprises a set of modules that must be completed by the students registered on the programme. Di Gaspero et al. [38] adopted the following constraints.

(i) Hard constraints.

(a) *Lectures*: all lectures within a module must be assigned to regular periods. All lectures must be scheduled (HC_1).
(b) *Room occupancy*: only one lecture can take place in a room at a given time (HC_2).
(c) *Conflicts*: students and staff can only attend one lecture at a time (HC_3).
(d) *Availabilities*: lecturers must be available for a lecture to be scheduled (HC_4).

(ii) Soft constraints.

(a) *Room capacity*: the room must have sufficient seats for the students on the module (SC_1).
(b) *Minimum working days between lectures*: for a particular module there should be a minimum amount of time between lectures (SC_2).
(c) *Curriculum compactness*: students on a degree programme should have lectures that are consecutive with no gaps (SC_3).
(d) *Room stability*: all lectures of a module should be given in the same room (SC_4).

Another issue is that events or courses may have differing priorities; the generation of infeasible solutions can be avoided by scheduling the highest priority activities first [39].

3. Evolutionary Algorithms Based Timetabling (EAT) Tool

The aim of this research was to generate timetables for lecturers, students, and classrooms that must satisfy all of the hard constraints and minimise the number of violations of the soft constraints proposed by Di Gaspero et al. [38].

The Evolutionary Algorithm based Timetabling (EAT) program was coded using the Tool Command Language and Toolkit (Tcl/Tk) [40]. It was developed in order to construct effective course timetables by using a genetic algorithm (GA) [41] and a memetic algorithm (MA) [42]. Both methods are population based and perform multiple directional search, which achieves a greater diversity than conventional optimisation methods that conduct a single directional search [6]. The MA and GA chromosomes have different components. For MAs, the chromosomes consist of a set of memes, whereas with GAs the chromosomes comprise a set of genes [43]. The key difference is that the memes used by the MA can be self-adapting based upon local search and refinement, whereas genes do not have this capability [6].

The artificial immune system (AIS) was initially proposed in the mid 1980s by Farmer et al. [44]. The clonal selection algorithm is a well-known variants of the AIS that is based upon two immune system principles: clonal selection and affinity maturation [45]. Each antibody (candidate solution) would be cloned proportionally to its antigenic affinity (fitness) value, in which the higher antigenic affinity would have the higher number of cloned antibodies [46]. Affinity maturation is related to hypermutation and receptor editing [46]. The regulation of hypermutation is a rapid accumulation of mutations that depend upon receptor affinity, in which the cell receptor with the higher affinity is mutated by using a mutation rate that is lower than for solutions with lower fitness [46]. Receptor editing provides a mechanism for escaping from the local optima, which increases the diversity of solutions [46]. The elimination percentage $\%B$ specifies how many low affinity antibodies are eliminated from the receptors.

The main procedures within the evolutionary algorithms based timetabling tool are shown in Figure 1. The first step is to the represent events within the timetable as memes/genes. The second step is to combine memes/genes to produce an initial population that represents a set of possible timetables. This part of the algorithm is designed to ensure that all of the candidate solutions are feasible. This is followed sequentially by genetic algorithms, local search, and a clonal selection algorithm, which is repeated for the required number of generations. The GA operators, LS, and CSA are designed to ensure that all of the chromosomes produced are feasible. There is an elitist strategy selection mechanism after the local search processes that selects the chromosomes for the CSA algorithm and also remembers the good solutions in its memory. There is a subsequent roulette wheel selection process after the CSA, which produces a population of chromosomes. A further elitist replacement process substitutes weaker solutions within the population with solutions remembered by the elitist strategy if they are better. The following subsections describe these processes in more detail.

3.1. Meme/Gene Representation. This research used the same data structures for genetic algorithms, memetic algorithms, and the clonal selection algorithm. The terminology used to describe the data structures varies according to the algorithm. With a genetic algorithm a chromosome comprises a set of genes. With a memetic algorithm a chromosome comprises a set of Memes. The clonal selection algorithm described in Section 3.7 uses an identical structure to represent antibodies.

A meme/gene can be encoded using either numeric (binary, integer, or real) or alphanumeric characters [9]. In this work, an integer encoded meme consists of three coded numbers: classrooms ($r = 1, 2, 3, \ldots, R$); days per week ($d = 1, 2, 3, \ldots, D$); and periods or timeslots per day ($t = 1, 2, 3, \ldots, T$). Each meme/gene contains a reference to a classroom, a day, and a timeslot; for example, $\{1, 2, 4\}$ represents an event in the first classroom that takes place on the second days in the fourth timeslot. A chromosome comprises a set of memes/genes that represent a complete timetable.

3.2. Chromosome Initialisation. The chromosome initialisation process takes the following steps.

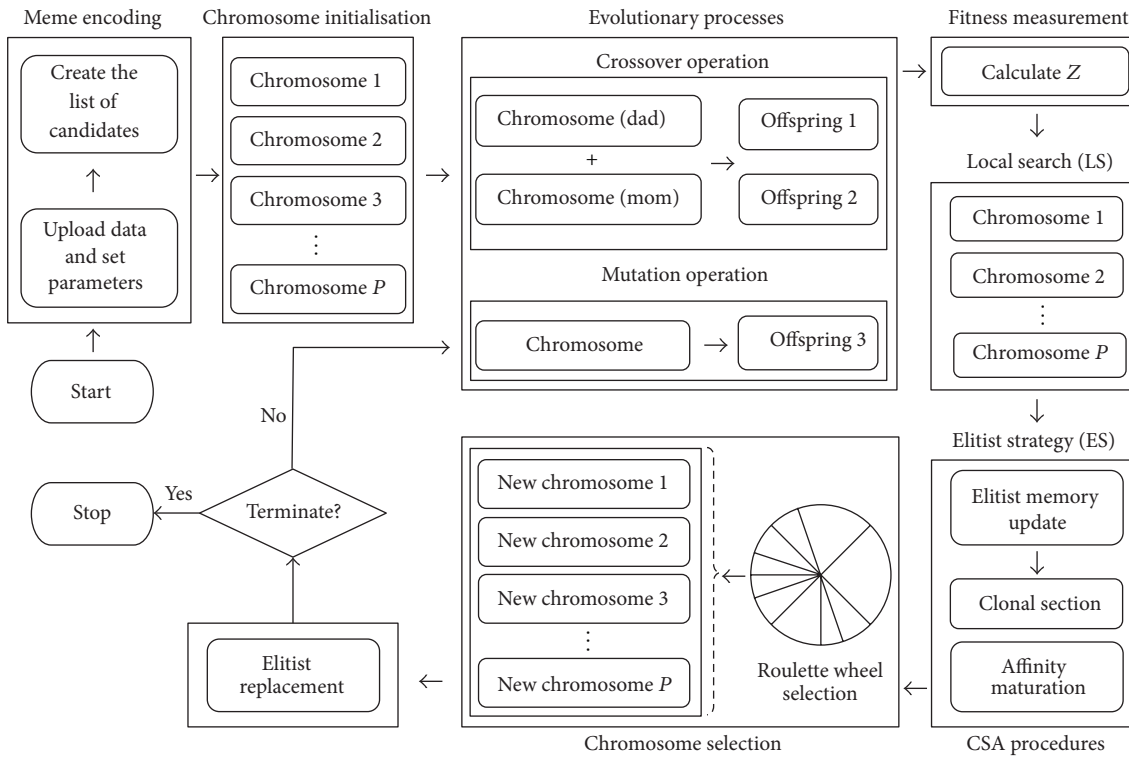

Figure 1: Main procedures of evolutionary algorithms based timetabling tool.

(1) The length of the chromosome required is calculated taking into account the number of degree programmes, modules, and their associated classes.

(2) An empty chromosome is generated with the appropriate length.

(3) The modules are then sorted based upon their relative importance.

(4) The highest priority module is scheduled first: this entails generating memes/genes for all of the classes and randomly assigning them to the chromosome. Before a meme/gene is added a check is made to ensure that the hard constraints are not violated. If there is a violation the algorithm sequentially looks for the next meme/gene that does not contravene the constraints (taking into account the modules in priority order); the process is then repeated in priority order until all the modules have been scheduled.

3.3. Evolutionary Processes. The parent chromosomes are randomly selected for the crossover and mutation genetic operations according to the probabilities of crossover (P_C) and mutation (P_M). The selection of these parameters determines the balance between exploration and exploitation. The crossover operation (COP) produces offspring chromosomes from two parent chromosomes, whereas the mutation operation produces random meme/gene changes in one chromosome. The number of memes/genes within the chromosome that are changed is determined by the mutation rate M_R. Thus, the selection of chromosomes is related to P_M whereas the selection within the chromosome is related to the parameter M_R.

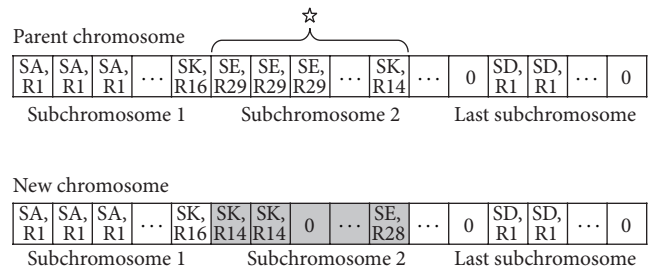

Figure 2: Regeneration mutation operator [9].

The EAT includes three types of crossover operation: one-point crossover (OP), two-point crossover (TP) [47], and position based crossover (PB) [48], which were modified to ensure that only feasible chromosomes can be produced. The modified version of the regeneration mutation [9] was developed to ensure feasible solutions.

Figure 2 illustrates the regeneration mutation operator. It includes three steps. First, a chromosome is randomly selected from the population. Secondly, a section (subchromosome) is selected for regeneration. Finally, a new subchromosome is generated randomly. The remaining genes within the chromosome are inherited from the parent.

Their modified regeneration mutation operator made four modifications to the operator: (i) some memes from the parent chromosome are randomly regenerated so that

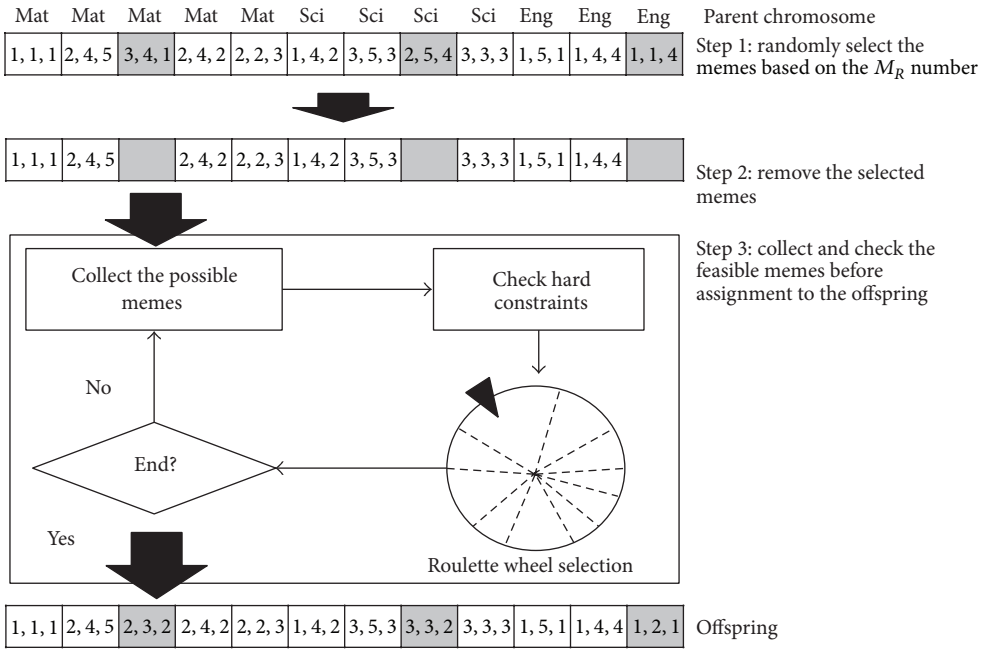

Figure 3: The modified regeneration mutation operation.

beneficial memes are inherited by the offspring; (ii) new feasible memes are assigned into the empty chromosome positions by using roulette wheel selection; (iii) all of the offspring are guaranteed to be feasible chromosomes because of the hard constraint checking before memes which are inserted into the empty positions of a new chromosome; and (iv) the parameter M_R specifies the percentage of memes to be regenerated. A higher setting of M_R increases the amount of exploration, but this may result in beneficial memes from the parent being lost. The modified regeneration mutation procedure is illustrated in Figure 3.

3.4. Fitness Measurement. The total violation index (Z) for a timetable may be calculated using (1) [49].

$$\text{Minimise} \quad Z = \sum_{i=1}^{S} W_i \text{SC}_i \tag{1}$$

$$\text{Subject to:} \quad \text{HC}_j = 0, \quad \forall j, \tag{2}$$

where i is an index relating to the i_{th} soft constraint ($i = 1, 2, 3, \ldots, S$), where S is the number of soft constraints; j is the index for the j_{th} hard constraint ($j = 1, 2, 3, \ldots, H$), where H is the number of hard constraints. SC_i is a variable used to count the number of violations of the i_{th} soft constraint. HC_j is the variable used to count the number of violations of the j_{th} hard constraint. For a timetable to be feasible HC_j must be zero for all the hard constraints. The user can specify the relative importance of the soft constraints by adjusting the weightings W_i for each soft constraint. Higher weightings indicate higher priority of the associated soft constraints. In this work, the weights (W_1–W_4) were set at 1, 5, 2, and 1, respectively, as recommend by Di Gaspero et al. [38].

The GA and MA measure the quality of each chromosome using the objective function from (1) to calculate the total violation index (Z). As the objective is to minimise the number of violations the fitness value, which is determined by [23]

$$\text{Fitness value} = \frac{1}{1 + Z}. \tag{3}$$

3.5. Local Search (LS). The objective of the local search (LS) within the MA is to (i) improve the quality of chromosome or solution, through increased exploitation and (ii) increase the opportunity to quickly discover the global best solution. In this work, two hill-climbing LS heuristics, LS1 and LS2, were adopted from previous work by Thepphakorn et al. [28], as it had been demonstrated that they improved chromosome quality and prevented the generation of infeasible chromosomes. The aim of LS1 is to reduce the number of violations of the first and the forth soft constraints (SC_1 and SC_4), whilst the LS2 aims to reduce the number of violations of the second and the third soft constraints (SC_2 and SC_3). After the LS1 and LS2 procedures, the total violation index (Z) and the fitness values for the new chromosomes are measured again before performing chromosome selection.

3.6. Elitist Strategy (ES). The ES aims to maintain high quality chromosomes from one generation to the next. The ES helps GAs to reach convergence more quickly [13]. This ES is divided into two subprocesses: elitist memory updating, which records the best solutions (with no duplicates) and elitist replacement, which substitutes the worst chromosomes with those remembered if they are better. The elitist replacement process takes place after chromosome selection. The proportion of chromosomes remembered by the ES is

Table 1: Characteristics of course timetabling problems [38].

Problems	Characteristics of course timetabling problems						
	Number of modules	Number of events	Number of classrooms	Number of periods/week	Number of Lecturers	Number degree programmes	Unavailability constraints
1	30	160	6	30	24	14	53
2	82	283	16	25	71	70	513
3	72	251	16	25	61	68	382
4	79	286	18	25	70	57	396
5	54	152	9	36	47	139	771
6	108	361	18	25	87	70	632
7	131	434	20	25	99	77	667
8	86	324	18	25	76	61	478
9	76	279	18	25	68	75	405
10	115	370	18	25	88	67	694
11	30	162	5	46	24	13	94
12	88	218	11	36	74	150	1368
13	82	308	19	25	77	66	468
14	85	275	17	25	68	60	486

determined by a user specified parameter %ES. Previous research had indicated that the most appropriate value for this parameter is 75% [50].

3.7. Clonal Selection Algorithms (CSA). In this research, the memory of the ES is used to produce the hybridisations for (i) the genetic algorithm combined with the CSA and (ii) the memetic algorithm (which had been modified to include hill climbing local search) which was combined with the CSA. When chromosomes are assigned to the elitist memory, the procedure attempts to further improve them through the application of the CSA. The elitist memory is updated if the resultant chromosome is better after the application of the CSA.

All of the chromosomes (antibodies) in the elitist memory are sorted in accordance to their affinities (finesses); the chromosome with the highest affinity is assigned the highest rank $k = 1$, $(k = 1, 2, 3 \ldots, n)$, whilst the chromosome with the lowest affinity is ranked $k = n$. The total number of antibodies (n) for cloning is equal to the number of chromosomes in the ES memory. In the following step, each k rank of antibodies contained in the elitist memory is cloned according to [46]

$$N_c = \sum_{k=1}^{n} \text{round}\left(\frac{\beta \cdot P}{k}\right), \tag{4}$$

where N_c is the total number of cloned antibodies, round $(\cdot)$ is an operator for changing real values into integers, β is the multiplying factor, and P is the population size. In the last step, all of the cloned antibodies are generated using affinity maturation. The regeneration mutation operator is used in this process together with a variable mutation rate M_{Rv} that is an adaptive setting based upon an antibody's ranking. The initial setting for M_{Rv} is determined by the parameter M_R. Antibodies with a lower affinity (ranking) require a value of M_{Rv} that is greater than those with higher ranking [46].

3.8. Chromosome Selection. The classical roulette wheel approach [41] was used in this research. The general concept of the roulette wheel selection is to randomly select which chromosomes in the current population survive into the next population in such a way that their probability of survival depends upon their fitness. This process is terminated when the desired population size has been generated.

4. Experimental Results and Analysis

The objective of the EAT is to construct course timetables with the lowest number of soft constraint violations (Z). The aims of the computational experiments were to (i) identify which main factors and their interactions were statistically significant for the GA; (ii) identify and verify the best parameter settings; (iii) explore the performance of the GA with modified regeneration mutation (called MRMO); and (iv) explore the performance of the proposed hybridisations including MRMO+CSA and MMA+CSA.

The research considered fourteen course timetabling problems that were provided by the third track of ITC2007 [38]. These are summarized in Table 1. The experiments were performed on a personal computer with Intel 2.67 GHz Core 2 Duo CPU and 4 GB of RAM.

4.1. Screening Experiment. The screening experiment had two objectives to identify which factors and first level interactions were statistically significant and to identify the best settings for these factors. The experimental design, shown in Table 2, was used together with data from timetabling problem 1 (a small problem). The factors included (i) the combination of population size and the number of generations (PG), which

Table 2: Experimental factors and levels for the GA.

Factors	Levels	Factor values		
		Low (−1)	Medium (0)	High (+1)
PG	3	25 ∗ 100	50 ∗ 50	100 ∗ 25
P_C	3	0.6	0.75	0.9
P_M	3	0.1	0.2	0.3
COP	3	One-point (OP)	Two-point (TP)	Position-based (PB)
M_R	3	0.1	0.5	0.9

Table 3: One-third fraction 3^{5-1} experimental design.

Runs	PG	P_C	P_M	COP	M_R
1	−1	−1	−1	−1	−1
2	−1	−1	−1	0	0
3	−1	−1	−1	1	1
4	−1	−1	0	−1	0
5	−1	−1	0	0	1
⋮	⋮	⋮	⋮	⋮	⋮
79	1	1	1	−1	−1
80	1	1	1	0	0
81	1	1	1	1	1

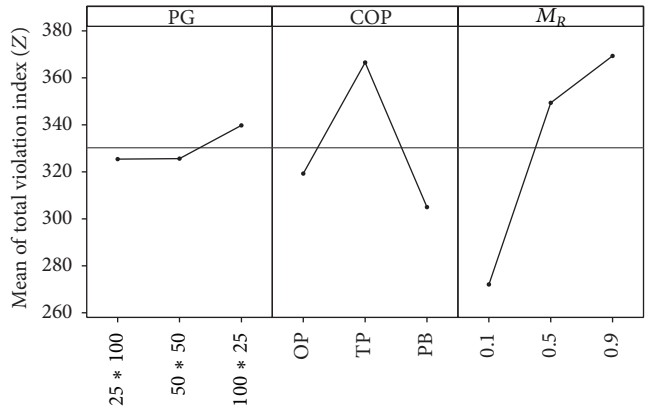

Figure 4: Main effect plots of PG, COP, and M_R factors.

determines the total number of chromosomes generated, which determines the amount of search and influences the execution time. In the computational experiments the value was fixed at 2,500 to limit the time taken for computational search; (ii) the probability of crossover (P_C); (iii) the probability of mutation (P_M); (iv) the crossover operation (COP); and (v) the mutation rate (M_R).

The total number of runs required for a full factorial experiment based on the design in Table 2 would consider all the combinations of the factors in each replication. The total number of runs would therefore be the number of factors times the number of levels times the number of replications, which would be $3^5 = 243$ runs per replication. When resources are limited it is common for researchers to use fractional factorial designs, which use a carefully chosen subset (fraction) of the experimental runs required for a full factorial design. This approach is based upon the sparsity of effects principle that states that a system is usually dominated by main effects and low order interactions [27].

In this experiment, the one-third fraction 3^{5-1} experimental design shown in Table 3 was adopted for the screening experiment, which decreased the number of computational runs by 66.67% per replication compared to the full factorial approach. The first instant problem (see Table 1) was selected and replicated five times using different random seeds. The computational results obtained from the 405 ($3^{5-1} * 5$) runs were analysed using a general linear model form of analysis of variance (ANOVA). Table 4 shows the ANOVA table, which shows the source of variation (*Source*), degrees of freedom (DF), sum of square (SS), mean square (MS), F value, and P value. ANOVA was used to test the null hypothesis that there was no effect (H_0) and the alternative hypothesis (H_1) that there is an effect for each factor and interaction [27]. If a 95% confidence interval is used H_1 is accepted if $P \leq 0.05$, but H_0 cannot be rejected if $P > 0.05$.

Table 4 shows the GA parameters in terms of the main effect and first level interactions. PG, COP, M_R, PG $*$ P_C, PG $*$ COP, PG $*$ M_R, and $P_M * M_R$ were statistically significant with a 95% confidence interval. The random seed number (*seeds*) did not statistically affect the GA performance. However, it is best not to discard parameters having a P value more than 0.05 but less than 0.2 in a screening experiment [26]. Moreover, the most influential factor in this experiment was M_R because it had the highest F value followed by the COP factor.

4.2. Multiple Comparison Analysis. The experimental design considered three different levels for each factor. The alternative hypothesis (H_1) obtained from the ANOVA only identifies that at least one level of a factor has a statistically different mean, but it is not known whether the other levels are significant [27]. Thus, in some cases it is not possible to select the appropriate parameter settings from the ANOVA because it is not known which pairs of results are significantly different. After the screening experiment, appropriate parameter settings for the GA were determined by using the lowest mean obtained from main effect and interaction plots. These are shown in Figures 4 and 5 for the statistically significant GA factors. Figure 4 indicates that the best settings are PG = 25 ∗ 100, COP = PB, and $M_R = 0.1$. Figure 5 shows the best combinations for the interactions, which are PG = 25 ∗ 100 and $M_R = 0.1$; PG = 50 ∗ 50

Table 4: ANOVA analysis of GA parameters.

Source	DF	SS	MS	F value	P value
PG	2	18,274	9,137	7.65	0.001
P_C	2	4,362	2,181	1.83	0.163
P_M	2	4,589	2,295	1.92	0.148
COP	2	280,473	140,236	117.36	0.000
M_R	2	712,893	356,446	298.29	0.000
Seeds	4	7,413	1,853	1.55	0.187
PG $*$ P_C	4	12,304	3,076	2.57	0.038
PG $*$ P_M	4	4,341	1,085	0.91	0.459
PG $*$ COP	4	15,613	3,903	3.27	0.012
PG $*$ M_R	4	44,420	11,105	9.29	0.000
P_C $*$ P_M	4	3,419	855	0.72	0.582
P_C $*$ COP	4	4,544	1,136	0.95	0.435
P_C $*$ M_R	4	3,259	815	0.68	0.605
P_M $*$ COP	4	3,971	993	0.83	0.506
P_M $*$ M_R	4	14,475	3,619	3.03	0.018
COP $*$ M_R	4	8,257	2,064	1.73	0.143
Error	350	418,238	1,195		
Total	404	1,560,845			

Table 5: Pairwise comparisons using Tukey's method for the significant main effects.

Factors	(j)	(i)	Mean difference ($i - j$) Z	T value	P value
PG	25 $*$ 100	50 $*$ 50	0.21	0.05	0.999
	25 $*$ 100	100 $*$ 25	14.36	3.41	0.002
	50 $*$ 50	100 $*$ 25	14.14	3.36	0.002
COP	OP	TP	47.30	11.24	0.000
	OP	PB	−14.28	−3.39	0.002
	TP	PB	−61.58	−14.64	0.000
M_R	0.1	0.5	77.33	18.38	0.000
	0.1	0.9	97.28	23.12	0.000
	0.5	0.9	19.95	4.74	0.000

with COP = PB; PG = 50 $*$ 50 with $P_C = 0.75$; and $M_R = 0.1$ with $P_M = 0.2$.

Tukey's method [27] is a statistical analysis tool that may be used for multiple or pairwise comparisons. The hypothesis testing used by Tukey's method can be defined as follows: H_0 cannot be rejected if the means of pair $i - j$ are equal; that is, the mean difference between the pair $i - j$ is zero (P value > 0.05); otherwise, H_1 will be accepted for pair $i - j$ with P value ≤ 0.05 [27]. Many statisticians prefer to use this approach because the overall error rate is controlled [27]. Tukey's method was therefore applied to detect significant differences in pairs of means in terms of the main and interaction effects. The results obtained with Tukey's comparison are shown in Tables 5 and 6, each of which consists of the significant factors, the pairs of factor levels between i and j considered, the mean difference between i and j (Mean dif. $i - j$), and the T value and the P value.

The comparative results obtained from using Tukey's method for the significant main effects shown in Table 5 indicated that the mean (mean difference $i - j$) difference in penalty Z for the factor PG between 25 $*$ 100 and 50 $*$ 50 was not statistically different (H_0 cannot be rejected) whilst the means of other pairs were different (H_1 will be accepted). Moreover, the means obtained from the pairs of the COP and M_R factors were statistically different from each other pair. As there were many pairs of interactions, only the pairs that had the lowest mean in the interaction plots shown in Figure 5 were selected for pairwise comparisons. The analysis of the results obtained with Tukey's method for the selected significant interactions is shown in Table 6. The mean obtained with the M_R factor at 0.1 was not statistically different when the PG factor was set at either 25 $*$ 100 or 50 $*$ 50. This indicates that all three levels of the PG factor are appropriate for use with the PB crossover. The P_C parameter settings were practicable at all levels when the PG factor was set at either 25 $*$ 100 or 50 $*$ 50. The means obtained for all of the levels of the P_M factor were usable when the M_R factor was set at 0.1. Therefore, the appropriate parameter setting for the GA factors PG, COP, M_R, P_C, and P_M were established as 25 $*$ 100 or 50 $*$ 50, PB, 0.1, 0.6–0.9, and 0.1–0.3, respectively.

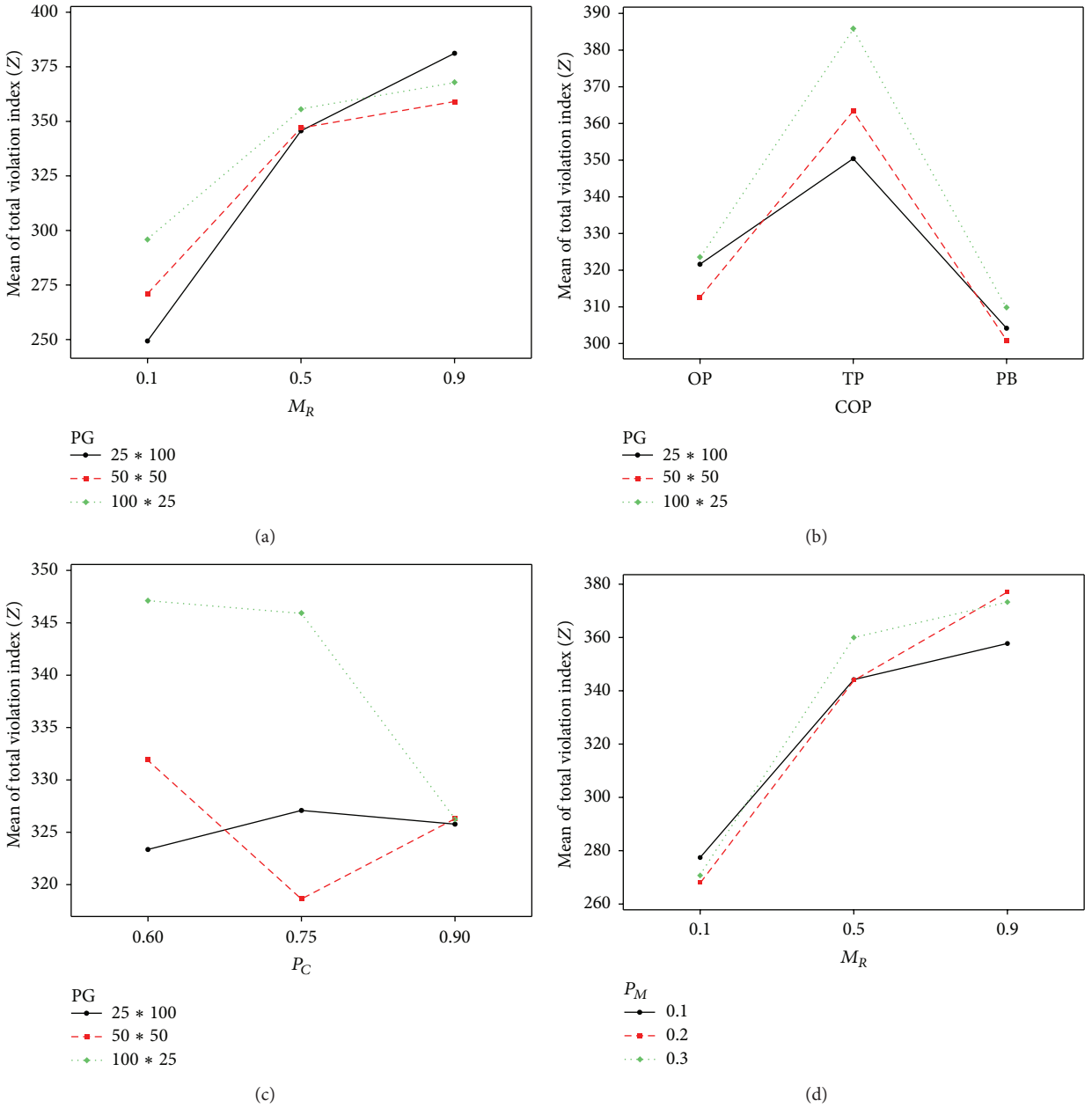

Figure 5: Interaction plots of PG * P_C, PG * COP, PG * M_R, and P_M * M_R.

4.3. Verifying Appropriate Parameter Settings. The significant factors, interactions, and important differences in the means were investigated by the screening experiment and pairwise comparison. Montgomery [27] suggested that the region for significant factors leading to the best possible response should be explored by conducting a second optimisation experiment after the screening experiment. The COP factor was a discrete GA parameter. The previous experiment identified PB as the best setting. PG settings between 25 * 100 and 50 * 50 showed little difference after using the pairwise analysis (see in Table 5), except for the M_R factor. Therefore, the region of M_R around 0.1 should be verified by using an experimental design before carrying out a comparative study.

Five levels of M_R were therefore considered 0.02, 0.06, 0.1, 0.14, and 0.18. The appropriate settings identified by the previous experiments were used for the other parameters. The first instant problem from the ITC2007 was selected and repeated ten times using different random seed numbers. The results obtained from the computational runs for the best so far solution were statistically analysed in terms of minimum (Min), maximum (Max), and average (Avg) penalty value Z as well as the standard deviation (SD) and the execution time (Time) (hour unit).

The experimental results are shown in Table 7. The M_R setting of 0.1 produced the best performance with the lowest average, minimum, and maximum values. The higher values

Table 6: Pairwise comparisons using Tukey's method for the selected significant interaction.

Factors	(j)		(i)		Mean difference $(i-j)$ Z	T value	P value
PG $*$ P_C	25 $*$ 100	0.75	50 $*$ 50	0.75	−8.42	−1.16	0.965
	25 $*$ 100	0.6	25 $*$ 100	0.75	3.73	0.51	0.999
	25 $*$ 100	0.75	25 $*$ 100	0.9	−1.31	−0.18	1.000
	50 $*$ 50	0.6	50 $*$ 50	0.75	−13.24	−1.82	0.671
	50 $*$ 50	0.75	50 $*$ 50	0.9	7.62	1.05	0.981
PG $*$ COP	25 $*$ 100	PB	50 $*$ 50	PB	−3.27	−0.45	1.000
	25 $*$ 100	PB	100 $*$ 25	PB	5.67	0.78	0.998
	50 $*$ 50	PB	100 $*$ 25	PB	8.93	1.23	0.951
PG $*$ M_R	25 $*$ 100	0.1	50 $*$ 50	0.1	21.67	2.97	0.073
	25 $*$ 100	0.1	100 $*$ 25	0.1	46.51	6.38	0.000
	50 $*$ 50	0.1	100 $*$ 25	0.1	24.84	3.41	0.019
P_M $*$ M_R	0.1	0.1	0.2	0.1	−9.42	−1.29	0.933
	0.1	0.1	0.3	0.1	−6.73	−0.92	0.992
	0.2	0.1	0.3	0.1	2.69	0.37	1.000

Table 7: Verifying the optimal setting for M_R.

Factor levels	Significant factor	Best so far solution				
	M_R	Min Z	Max Z	Avg Z	SD	Time (hrs)
Lowest (−2)	0.02	228	312	260.60	24.29	0.11
Low (−1)	0.06	216	286	255.20	21.18	0.12
Medium (0)	0.10	163	254	222.30	25.77	0.12
High (1)	0.14	193	328	260.40	37.33	0.13
Highest (2)	0.18	224	314	256.70	25.32	0.14

of M_R required more computational time than the lower parameter settings. This analysis verified that the optimal setting of M_R for the GA is 0.1.

4.4. Performances of GA with/without the Modified Regeneration Mutation. The objective of this experiment was to explore and compare the performance of GA with/without the modified regeneration mutation operator (MRMO) in terms of the speed of convergence and the quality of the solutions. The appropriate parameter settings for the GA with the MRMO for PG, COP, M_R, P_C, and P_M were found to be 25 $*$ 100, PB, 0.1, 0.75, and 0.2, respectively. The benchmark problems adopted from the third track of the ITC2007 (14 instances) [38] were used to test and compare the performance of the proposed algorithms to find the course timetable with the lowest penalty Z. The computational run for each instance was repeated ten times by using different random seeds. The computational results were analysed in terms of Avg, SD, time (hour unit), and the percentage improvement achieved by the GA with MRMO (%Imp). A t-test was used to compare the means.

Table 8 shows that the performance differences achieved by the GA with/without the MRMO were all statistically significant with a 95% confidence interval using t-test analysis (P value $\leq$ 0.05) for all of the problems. It means that the GA with the MRMO outperformed the GA without the MRMO for all instances, each of which also had the lower Avg and SD values. Moreover, the %Imp value for each problem was distributed between 35.55% and 87.56% but with longer execution time. The average of improvement was up to 51.88%. A comparison of the convergence speed for the proposed methods to investigate the best so far solution is shown in Figures 6 and 7 by using problems number 7 and number 14 from the ITC2007 datasets.

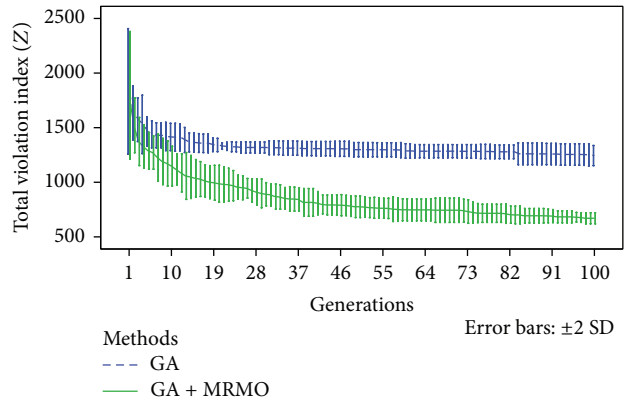

Figure 6: A comparison of convergence achieved by the GA with/without MRMO for problem number 7.

The GA with MRMO converged more quickly for problems number 7 and number 14 than the GA without the MRMO (see Figures 6 and 7). Therefore, it can be concluded that the new regeneration mutation based upon roulette

Table 8: Comparative study between GA with/without MRMO.

Problems	GA			GA with MRMO			t-test analysis		%Imp
	Avg Z	SD	Time (hrs)	Avg Z	SD	Time (hrs)	T value	P value	
1	222.30	25.77	0.12	90.60	16.59	0.12	13.59	0.000	59.24
2	710.30	37.97	1.51	416.90	26.74	1.52	19.98	0.000	41.31
3	683.50	28.20	0.67	370.20	26.77	0.68	25.48	0.000	45.84
4	724.70	24.98	0.62	318.70	19.99	0.62	40.13	0.000	56.02
5	1549.60	154.74	0.57	998.70	113.07	0.58	9.09	0.000	35.55
6	1087.70	26.90	0.82	555.70	25.85	0.82	45.09	0.000	48.91
7	1244.50	46.14	1.23	670.50	25.43	1.23	34.45	0.000	46.12
8	689.90	19.17	0.73	300.70	14.48	0.71	51.22	0.000	56.41
9	717.90	24.19	0.87	373.10	21.48	0.85	33.70	0.000	48.03
10	839.20	38.41	0.92	408.70	19.92	0.90	31.47	0.000	51.30
11	403.40	78.19	0.19	50.20	16.43	0.19	13.98	0.000	87.56
12	1397.00	52.02	0.63	854.40	49.25	0.63	23.95	0.000	38.84
13	806.60	37.29	0.71	362.00	23.32	0.71	31.97	0.000	55.12
14	705.90	27.59	0.58	310.30	17.48	0.58	38.30	0.000	56.04
								Avg %Imp	**51.88**

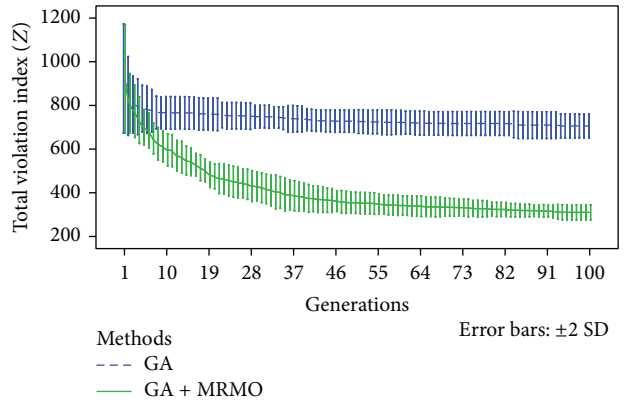

Figure 7: A comparison of convergence achieved by the GA with/without MRMO for problem number 14.

wheel selection was able to improve the GA's performance in terms of solution quality and speed.

4.5. Analysing the Performance Evolutionary Algorithm Hybridisations. The objective of this experiment was to explore the performance of (i) the MRMO with/without local search (the modified memetic algorithm (MMA)); (ii) the addition of the elitist strategy (MRMO+ES and MMA+ES); and (iii) the use of the clonal selection algorithm (MRMO+CSA and MMA+CSA) both in terms of convergence speed and solution quality. The appropriate parameter settings for the MRMO and the MMA were adopted from the previous experiments. The benchmark problems adopted from the third track of the ITC2007 [38] were again used to test and compare the performance of the proposed algorithms to find the course timetable with the lowest penalty Z. The computational run for each instance was repeated ten times by using different random seed numbers. The computational results obtained were analysed statistically in terms of Avg, SD, and time (hour unit), as shown in Table 9. The percentage improvement (%Imp) achieved by the MRMO with/without hybrid heuristics was calculated, whilst the T value obtained by using the t-test method and the P value are also shown in Table 10.

Table 10 shows that almost all of the comparisons between the results obtained from the MRMO and the other hybridisation approaches were statistically significant with a 95% confidence interval (P value ≤ 0.05). For all of the problems the results obtained from the MMA+ES, MRMO+CSA, and MMA+CSA were statistically significant with a 95% confidence interval. Moreover, the MMA+CSA achieved the highest T value, %Imp, and Avg %Imp, which indicates that it was the best configuration. However, the negative or positive T value in Table 10 indicated that the results obtained from some hybrid approaches did not outperform the MRMO for some problems.

According to Tables 9 and 10, the MMA+CSA outperformed the other methods for all instances because it achieved the maximum Avg %Imp of 49.85% and minimum Avg values. However, it also had the longest execution time. Although the Avg %Imp between the MRMO+CSA and the MMA+ES was nearly equal at 42%, the MRMO+CSA required less computational time than both the MMA+CSA and the MMA+ES; it was up to 6.3 times quicker for some instances. Moreover, the Avg %Imp obtained by the MRMO using CSA was better than that using LS (MMA) and ES by approximately 23–26%. The MRMO's execution times using CSA were also up to 5.7 times faster than those using LS but slower than those using ES by up to 3.2 times for some instances. Although the Avg %Imp obtained from the MRMO using ES and LS was less than those using the proposed hybrid methods, the performances of the MRMO+ES and the MMA were better than the MRMO without hybridisations (see in Table 10). The average improvement for almost all problems

Table 9: Performance explorations for EA with/without ES, LS, and CSA hybridisations.

Methods	Statistical analysis	Problems													
		1	2	3	4	5	6	7	8	9	10	11	12	13	14
MRMO + ES	Avg Z	63.6	349.2	325.7	275.4	856.1	442.6	531.6	269.3	322.7	357.9	24.2	805.7	343.7	299.6
	SD	11.1	29.9	14.5	20.7	85.9	31.4	21.7	20.5	13.1	18.0	6.7	57.3	14.9	13.8
	Time (hrs)	0.12	1.53	0.68	0.61	0.58	0.83	1.24	0.72	0.89	0.93	0.20	0.63	0.71	0.59
MMA (MRMO + LS)	Avg Z	46.4	353.5	362.2	249.3	938.9	378.2	423.6	255.7	377.1	340.8	14.9	1032.9	300.1	312.0
	SD	7.5	21.8	25.1	13.8	67.4	19.3	17.4	15.1	22.1	17.1	4.0	59.8	17.3	12.9
	Time (hrs)	0.45	5.09	2.80	2.68	3.64	4.68	7.00	2.94	3.08	4.95	0.60	6.14	3.12	2.93
MMA + ES	Avg Z	33.5	274.7	263.4	176.4	715.9	273.4	297.6	193.7	267.6	240.2	8.0	729.9	232.3	205.8
	SD	5.3	13.1	14.5	13.6	70.1	24.5	17.2	10.4	16.0	11.7	1.8	53.7	9.9	8.3
	Time (hrs)	0.47	5.17	2.74	2.60	4.82	4.60	7.00	3.04	2.91	4.91	0.60	6.81	3.45	2.94
MRMO + CSA	Avg Z	25.1	271.9	248.1	187.1	678.7	293.5	336.7	190.5	251.6	266.8	5.9	678.9	234.5	210.9
	SD	3.4	21.6	18.0	6.7	59.4	22.6	16.2	13.2	15.1	11.7	2.5	65.1	12.6	11.1
	Time (hrs)	0.29	1.89	1.08	1.08	1.05	1.17	1.54	1.12	1.42	1.26	0.64	1.08	1.24	1.00
MMA + CSA	Avg Z	21.2	242.3	231.6	146.6	569.3	230.5	260.0	170.0	242.5	212.5	4.4	672.4	200.5	181.1
	SD	2.7	24.6	20.0	7.3	43.5	19.4	13.4	10.8	12.5	14.0	1.6	65.3	9.2	13.4
	Time (hrs)	0.58	5.47	3.06	3.19	6.20	5.05	7.39	3.27	3.90	5.25	1.03	6.79	3.76	3.25

Table 10: Statistical Analysis using a t-test for EA with/without ES, LS, and CSA hybridisations.

Methods	Statistical analysis	Problems													
		1	2	3	4	5	6	7	8	9	10	11	12	13	14
MRMO versus MMA	T value	7.68	5.81	0.69	9.04	1.44	17.41	25.33	6.81	−0.41	8.17	6.61	−7.28	6.74	−0.25
	P value	0.000	0.000	0.500	0.000	0.173	0.000	0.000	0.000	0.687	0.000	0.000	0.000	0.000	0.808
	%Imp	48.79	15.21	2.16	21.78	5.99	31.94	36.82	14.97	−1.07	16.61	70.32	−20.89	17.1	−0.55
	Avg %Imp =		**18.51**												
MRMO versus MRMO + ES	T value	4.27	5.34	4.62	4.76	3.18	8.8	13.14	3.95	6.34	5.98	4.63	2.04	2.09	1.52
	P value	0.001	0.000	0.000	0.000	0.006	0.000	0.000	0.001	0.000	0.000	0.001	0.057	0.054	0.148
	%Imp	29.8	16.24	12.02	13.59	14.28	20.35	20.72	10.44	13.51	12.43	51.79	5.7	5.06	3.45
	Avg %Imp =		**16.38**												
MRMO versus MMA + ES	T value	10.36	15.1	11.09	18.6	6.72	25.06	38.42	19.01	12.47	23.05	8.07	5.4	16.18	17.09
	P value	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	%Imp	63.02	34.11	28.85	44.65	28.32	50.8	55.62	35.58	28.28	41.23	84.06	14.57	35.83	33.68
	Avg %Imp =		**41.33**												
MRMO versus MRMO + CSA	T value	12.23	13.34	11.98	19.75	7.92	24.14	34.99	17.79	14.63	19.43	8.43	6.8	15.2	15.16
	P value	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	%Imp	72.3	34.78	32.98	41.29	32.04	47.18	49.78	36.65	32.56	34.72	88.25	20.54	35.22	32.03
	Avg %Imp =		**42.17**												
MRMO versus MMA + CSA	T value	13.05	15.19	13.12	25.57	11.21	31.81	45.18	22.89	16.63	25.49	8.77	7.04	20.39	18.55
	P value	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	%Imp	76.6	41.88	37.44	54	43	58.52	61.22	43.47	35	48.01	91.24	21.3	44.61	41.64
	Avg %Imp =		**49.85**												

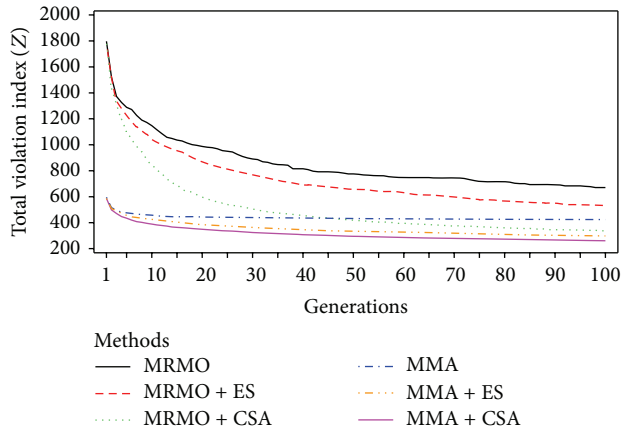

FIGURE 8: A comparison of convergence amongst EA hybridisations for problem number 7.

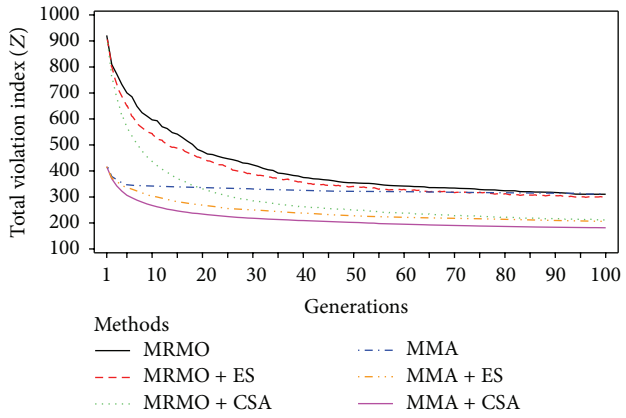

FIGURE 9: A comparison of convergence for EA hybridisations for problem number 14.

was around 16–19%. A comparison of the results in terms of average convergence speeds of the proposed hybrid methods to find the best so far solution is shown in Figures 8 and 9. These were based upon problem numbers 7 and 14 from the ITC2007 which represent medium and large problem sizes.

The MMA+CSA's converged more quickly than the other algorithms for problems 7 and 14. The next best convergence was achieved by MMA+ES (Figures 8 and 9). The MRMO+CSA had low performance in early generations. However, the average of best so far solutions found in the last generation was close to the average of best so far solutions obtained by the MMA+CSA. Moreover, the LS strategy hybridisation in the MRMO including the MMA+CSA, the MMA+ES, and the MMA was able to find a better average of best so far solutions in early generations than the other methods without LS. Therefore, it can be concluded that the LS, ES, and CSA strategies were able to improve the MRMO's performance in terms of the solution quality and speed.

5. Conclusions

The evolutionary algorithms based timetabling (EAT) tool was developed to use genetic algorithms (GA) and memetic algorithms (MA) to solve university course timetabling problems. The work made a number of significant research contributions. A common problem with genetic algorithms is that many chromosomes within a population may represent infeasible solutions. This work developed new one-point, two-point, and position-based crossover operators and a modified regeneration mutation operators that guaranteed that all of the chromosomes generated represented feasible solutions. Likewise the chromosome initialisation process was designed to produce feasible chromosomes. The research also developed novel hybrids that included genetic algorithms, local search, and a clonal selection algorithm together with roulette wheel and elitist selection. The tool was tested using 14 datasets obtained from the third track of ITC2007 [38], which have been widely used by previous researchers.

The experimental work adopted a sequential experimental design. The screening experiment used a one-third fraction of the 3^{k-1} experimental design [27] with five factors, each of which had three levels. The factors PG, COP, M_R, PG $*$ P_C, PG $*$ COP, PG $*$ M_R, and P_M $*$ M_R were statistically significant with a 95% confidence interval. Main effect plot analysis found the best settings to be PG $=$ 25 $*$ 100, COP = PB, and M_R = 0.1. The best combinations for the interactions were PG $=$ 25 $*$ 100 and M_R $=$ 0.1; PG $=$ 50 $*$ 50 with COP = PB; PG = 50 $*$ 50 with P_C = 0.75; and M_R = 0.1 with P_M = 0.2.

A further analysis using pairwise comparison found the appropriate parameter setting for PG, COP, M_R, P_C, and P_M to be 25 $*$ 100 or 50 $*$ 50, PB, 0.1, 0.6–0.9, and 0.1–0.3, respectively. A further experiment verified that the best setting for M_R was 0.1, as it produced the best performance with the lowest average, minimum and maximum penalty values.

The comparative results indicated that the MRMO outperformed the GA for all problems, with an average improvement of up to 51.88%. The MRMO converged more quickly than GA. In terms of hybrid comparisons, the MMA+CSA outperformed all the other methods; there was an average improvement of 49.85% compared to the MRMO. The second best hybrid was the MRMO+CSA. The MMA+CSA also converged more quickly and the best so far solutions were better than for all the other hybrid methods for all generations. Although the performance of the MRMO+CSA was the second rank in terms of an average of %Imp, it required up to 6.3 times less computational time less than the MMA+CSA. The ES, LS, and CSA embedded within the EAT tool were able to improve the EA's performances in terms of solution quality and its convergence but at the expense of longer execution time.

Thus, the development of novel hybrids has been shown to be an effective approach to solve a wide range of timetabling problems. The proposed approaches have been shown to provide good solutions quickly.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

The authors would like to acknowledge the Naresuan University for financial support on publication.

References

[1] I. H. Osman and G. Laporte, "Metaheuristics: a bibliography," *Annals of Operations Research*, vol. 63, pp. 513–623, 1996.

[2] R. Lewis, "A survey of metaheuristic-based techniques for university timetabling problems," *OR Spectrum*, vol. 30, no. 1, pp. 167–190, 2008.

[3] E.-G. Talbi, *Metaheuristics: From Design to Implementation*, John Wiley & Sons, 2009.

[4] C. Blum and A. Roli, "Metaheuristics in combinatorial optimization: overview and conceptual comparison," *ACM Computing Surveys*, vol. 35, no. 3, pp. 268–308, 2003.

[5] S. Salcedo-Sanz, "A survey of repair methods used as constraint handling techniques in evolutionary algorithms," *Computer Science Review*, vol. 3, no. 3, pp. 175–192, 2009.

[6] M. Gen and R. Cheng, *Genetic Algorithms and Engineering Design*, John Wiley & Sons, New York, NY, USA, 1997.

[7] S. S. Chaudhry and W. Luo, "Application of genetic algorithms in production and operations management: a review," *International Journal of Production Research*, vol. 43, no. 19, pp. 4083–4101, 2005.

[8] P. Pongcharoen, C. Hicks, and P. M. Braiden, "The development of genetic algorithms for the finite capacity scheduling of complex products, with multiple levels of product structure," *European Journal of Operational Research*, vol. 152, no. 1, pp. 215–225, 2004.

[9] P. Pongcharoen, W. Promtet, P. Yenradee, and C. Hicks, "Stochastic optimisation timetabling tool for university course scheduling," *International Journal of Production Economics*, vol. 112, no. 2, pp. 903–918, 2008.

[10] Z. N. Azimi, "Hybrid heuristics for examination timetabling problem," *Applied Mathematics and Computation*, vol. 163, no. 2, pp. 705–733, 2005.

[11] P. Thapatsuwan, P. Pongcharoen, C. Hicks, and W. Chainate, "Development of a stochastic optimisation tool for solving the multiple container packing problems," *International Journal of Production Economics*, vol. 140, no. 2, pp. 737–748, 2012.

[12] P. Pongcharoen, W. Chainate, and S. Pongcharoen, "Improving artificial immune system performance: inductive bias and alternative mutations," in *Artificial Immune Systems*, vol. 5132 of *Lecture Notes in Computer Science*, pp. 220–231, Springer, 2008.

[13] Y. Zhang, S. Wang, and G. Ji, "A rule-based model for bankruptcy prediction based on an improved genetic ant colony algorithm," *Mathematical Problems in Engineering*, vol. 2013, Article ID 753251, 10 pages, 2013.

[14] S. Vitayasak and P. Pongcharoen, "Machine selection rules for designing multi-row rotatable machine layout considering rectangular-to-square ratio," *Journal of Applied Operational Research*, vol. 5, no. 2, pp. 48–55, 2013.

[15] E. K. Burke and J. D. Landa Silva, "The design of memetic algorithms for scheduling and timetabling problems," in *Recent Advances in Memetic Algorithms*, W. Hart, J. E. Smith, and N. Krasnogor, Eds., pp. 289–311, Springer, Berlin, Germany, 2005.

[16] J.-H. Yang, L. Sun, H. P. Lee, Y. Qian, and Y.-C. Liang, "Clonal selection based memetic algorithm for job shop scheduling problems," *Journal of Bionic Engineering*, vol. 5, no. 2, pp. 111–119, 2008.

[17] R. Tavakkoli-Moghaddam, A. R. Saremi, and M. S. Ziaee, "A memetic algorithm for a vehicle routing problem with backhauls," *Applied Mathematics and Computation*, vol. 181, no. 2, pp. 1049–1060, 2006.

[18] S. Abdullah, H. Turabieh, B. McCollum, and P. McMullan, "A tabu-based memetic approach for examination timetabling problems," in *Rough Set and Knowledge Technology*, vol. 6401 of *Lecture Notes in Computer Science*, pp. 574–581, Springer, Berlin, Germany, 2010.

[19] E. Özcan, "Memes, self-generation and nurse rostering," in *Practice and Theory of Automated Timetabling VI*, vol. 3867 of *Lecture Notes in computer Science*, pp. 85–104, Springer, 2006.

[20] E. Özcan, A. J. Parkes, and A. Alkan, "The interleaved constructive memetic algorithm and its application to timetabling," *Computers and Operations Research*, vol. 39, no. 10, pp. 2310–2322, 2012.

[21] S. Abdullah and H. Turabieh, "On the use of multi neighbourhood structures within a Tabu-based memetic approach to university timetabling problems," *Information Sciences*, vol. 191, pp. 146–168, 2012.

[22] M. Basikhasteh and M. A. Movafaghpour, "Hybridizing genetic algorithm with biased chance local search," *World Academy of Science, Engineering and Technology*, vol. 80, pp. 354–359, 2011.

[23] R. Lewis and B. Paechter, "Finding feasible timetables using group-based operators," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 3, pp. 397–413, 2007.

[24] D. Qaurooni and M.-R. Akbarzadeh-T, "Course timetabling using evolutionary operators," *Applied Soft Computing Journal*, vol. 13, no. 5, pp. 2504–2514, 2013.

[25] B. T. Tesfaldet, "Automated lecture timetabling using a memetic algorithm," *Asia-Pacific Journal of Operational Research*, vol. 25, no. 4, pp. 451–475, 2008.

[26] P. Pongcharoen, D. J. Stewardson, C. Hicks, and P. M. Braiden, "Applying designed experiments to optimize the performance of genetic algorithms used for scheduling complex products in the capital goods industry," *Journal of Applied Statistics*, vol. 28, no. 3-4, pp. 441–455, 2001.

[27] D. C. Montgomery, *Design and Analysis of Experiments*, John Wiley & Sons, New York, NY, USA, 8th edition, 2012.

[28] T. Thepphakorn, P. Pongcharoen, and C. Hicks, "An ant colony based timetabling tool," *International Journal of Production Economics*, vol. 149, pp. 131–144, 2014.

[29] L. N. de Castro and J. Timmis, "An artificial immune network for multimodal function optimization," in *Proceedings of the Congress on Evolutionary Computation (CEC '02)*, pp. 699–704, IEEE, 2002.

[30] D. Dasgupta, "Advances in artificial immune systems," *IEEE Computational Intelligence Magazine*, vol. 1, no. 4, pp. 40–49, 2006.

[31] E. Hart and J. Timmis, "Application areas of AIS: the past, the present and the future," *Applied Soft Computing Journal*, vol. 8, no. 1, pp. 191–201, 2008.

[32] Y. L. He, S. C. Hui, and E. M. K. Lai, "Automatic timetabling using artificial immune system," in *Algorithmic Applications in Management*, N. Megiddo, Y. F. Xu, and B. H. Zhu, Eds., vol. 3521 of *Lecture Notes in Computer Science*, pp. 55–65, Springer, Berlin, Germany, 2005.

[33] M. R. Malim, A. T. Khader, and A. Mustafa, "Artificial immune algorithms for university timetabling," in *Proceedings of the 6th International Conference on Practice and Theory of Automated Timetabling (PATAT '06)*, pp. 234–245, 2006.

[34] A. Bhaduri, "University time table scheduling using genetic artificial immune network," in *Proceedings of the International Conference on Advances in Recent Technologies in Communication and Computing*, pp. 289–292, IEEE, Kottayam, India, October 2009.

[35] Y. Zhang, Y. Jun, G. Wei, and L. Wu, "Find multi-objective paths in stochastic networks via chaotic immune PSO," *Expert Systems with Applications*, vol. 37, no. 3, pp. 1911–1919, 2010.

[36] E. K. Burke and S. Petrovic, "Recent research directions in automated timetabling," *European Journal of Operational Research*, vol. 140, no. 2, pp. 266–280, 2002.

[37] R. Qu, E. K. Burke, and B. McCollum, "Adaptive automated construction of hybrid heuristics for exam timetabling and graph colouring problems," *European Journal of Operational Research*, vol. 198, no. 2, pp. 392–404, 2009.

[38] L. Di Gaspero, B. McCollum, and A. Schaerf, "The second international timetabling competition (ITC-2007): curriculum-based course timetabling track," in *Proceedings of the 14th RCRA Workshop on Experimental Evaluation of Algorithms for Solving Problems with Combinatorial Explosion*, M. Gavanelli and T. Mancini, Eds., Rome, Italy, 2007.

[39] E. K. Burke and J. P. Newall, "Solving examination timetabling problems through adaption of heuristic orderings," *Annals of Operations Research*, vol. 129, pp. 107–134, 2004.

[40] J. K. Ousterhout and K. Jones, *Tcl and the Tk Toolkit*, Addison-Wesley, Reading, Mass, USA, 2nd edition, 2009.

[41] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Reading, Mass, USA, 1989.

[42] P. Moscato and M. G. Norman, "A memetic approach for the travelling salesman problem—implementation of a computational ecology for combinatorial optimisation on message-passing systems," in *Proceedings of the International Conference on Parallel Computing and Transputer Applications*, pp. 177–186, 1992.

[43] E. Elbeltagi, T. Hegazy, and D. Grierson, "Comparison among five evolutionary-based optimization algorithms," *Advanced Engineering Informatics*, vol. 19, no. 1, pp. 43–53, 2005.

[44] J. D. Farmer, N. H. Packard, and A. S. Perelson, "The immune system, adaptation, and machine learning," *Physica D: Nonlinear Phenomena*, vol. 22, no. 1–3, pp. 187–204, 1986.

[45] D. Dasgupta, S. Yu, and F. Nino, "Recent advances in artificial immune systems: models and applications," *Applied Soft Computing Journal*, vol. 11, no. 2, pp. 1574–1587, 2011.

[46] L. N. de Castro and F. J. von Zuben, "Learning and optimization using the clonal selection principle," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 3, pp. 239–251, 2002.

[47] T. Murata, H. Ishibuchi, and H. Tanaka, "Genetic algorithms for flowshop scheduling problems," *Computers & Industrial Engineering*, vol. 30, no. 4, pp. 1061–1071, 1996.

[48] G. Syswerda, "Scheduling optimisation using genetic algorithm," in *Handbook of Genetic Algorithms*, pp. 332–349, 1991.

[49] T. Thepphakorn and P. Pongcharoen, "Heuristic ordering for ant colony based timetabling tool," *Journal of Applied Operational Research*, vol. 5, no. 3, pp. 113–123, 2013.

[50] P. Thapatsuwan, J. Sepsirisuk, W. Chainate, and P. Pongcharoen, "Modifying particle swarm optimisation and genetic algorithm for solving multiple container packing problems," in *Proceedings of the International Conference on Computer and Automation Engineering (ICCAE '09)*, pp. 137–141, IEEE, Bangkok, Thailand, March 2009.