Journal of Engineering and Applied Sciences 12 (20): 5166-5181, 2017
ISSN: 1816-949X

A Review of Genetic Algorithm Application in Examination Timetabling Problem

[1, 5]Mazin Abed Mohammed, [1]Mohd Khanapi Abd Ghani, [2]Omar Ibrahim Obaid, [3]Salama A. Mostafa,
[4]Mohd Sharifuddin Ahmad, [6]Dheyaa Ahmed Ibrahim and [1]M.A. Burhanuddin
[1]Biomedical Computing and Engineering Technologies (BIOCORE) Applied Research Group,
Faculty of Information and Communication Technology,
Universiti Teknikal Malaysia, Melaka, Malaysia
[2]College of Education, Al-Iraqia University, Adhamyia, Baghdad, Iraq
[3]Faculty of Computer Science and Information Technology,
Universiti Tun Hussein Onn, Johor, Malaysia
[4]College of Computer Science and Information Technology,
Universiti Tenaga Nasional, Putrajaya, Malaysia
[5]Department of Planning and Follow Up, University Headquarter,
[6]Department of Computer Science, College of Computer Science and Information Technology,
University of Anbar, Anbar, Iraq

Abstract: Many studies have used different types of genetic algorithm in solving examination timetabling problem. The solutions of the genetic algorithm are found to be efficient and reliable. This study provides a comprehensive review of genetic algorithm application in examination timetabling. It presents many examples of using genetic algorithm in finding optimal solutions to the problem of examination timetabling at universities or institutions. Subsequently, it presents the most used techniques to solve the examination timetabling problem such as tabu search and simulated annealing techniques. The objective of the study is to provide an understanding on what have been achieved in solving examination timetabling problem.

Key words: Genetic Algorithm (GA), Evolutionary Algorithm (EA), optimization, examination timetabling, simulated annealing, achieved

INTRODUCTION

This study introduces the background study of evolutionary algorithms which include the idea of genetic algorithm. It presents the theoretical foundations related to genetic algorithm, the beginning of artificial forms, chromosomes and encoding, way of choosing for dying out or for reproduction, crossover, mutation and reversal.

The study reviews issues pertaining to timetabling and scheduling problems, problem solution approaches traditional solutions and methods. It defines the examination timetabling soft and hard constraints and the previous works related to university timetabling problem. Subsequently, it presents the application of genetic algorithm in solving examination timetabling problem and the reasons behind using genetic algorithm in solving examination timetabling problem. Finally, it presents the most used techniques to solve the examination timetabling problem such as Tabu Search and Simulated Annealing techniques.

Evolutionary algorithm: Evolutionary Algorithm (EA) is a subject of evolutionary computing. EA is a generic artificial intelligence computational population-based algorithm employing the metaheuristics optimization technique. EA mechanism uses the concepts of biological evolution, reproduction, mutation, recombination and selection processes (Ashlock, 2006; Beyer *et al*., 2002; Eiben and Smith, 2003). Candidate solutions to the optimization problem play an important role in a population and the fitness function determines the environment within which the solutions 'live' (Holland, 1975). Population evolution occurs when solutions to the optimization problems are repeated. The process that involves individual's evolutionary algorithms is described as Artificial Evolution (AE).

EA is a subset component of AE and it works well in reaching approximate solutions to all types of problems because EA does not make any presumptions to the underlying fitness landscape that has been proven in the field of engineering, art, biology, economics, marketing,

Corresponding Author: Mazin Abed Mohammed, Biomedical Computing and Engineering Technologies (BIOCORE) Applied Research Group, Faculty of Information and Communication Technology,
Universiti Teknikal Malaysia Melaka, Malaysia

genetics, operations research, robotics, social science, physics, politics and chemistry (Giri *et al*., 2013).

EA has also been used in many experimental frameworks to validate theories on biological evolution and natural selection as shown in the works of (Obaid *et al*., 2015; Mohammed *et al*., 2015; Holland, 1975). The book of John Holland "adaptation in natural and artificial system" and De Jong's "adaptation behavior of a class of genetic adaptive system" have set the foundation for the study of Genetic Algorithm (GA). The complexity of computation is a limiting factor in real application of EA (Clune *et al*., 2008), due to fitness function evaluation. One of the solutions to solve this limitation is using the fitness approximation method (Clune *et al*., 2008). The lack of a clear genotype-phenotype distinction is another problem related to EA (Mohammed *et al*., 2012).

In real life, living organism fertilized egg cell undergoes embryogenesis to become phenotype. Through indirect encoding, the genetic search is more full-bodied. This process is intended to reduce lethal mutations and improve the evolution ability of the organism (Ashlock, 2006). The indirect encoding also enables evolution to take advantage of the regularity in the environment (Beyer *et al*., 2002; Mohammed *et al*., 2016; Price *et al*., 2006) that acknowledges five evolutionary algorithm techniques. These techniques are similar but different in the implementation methods, details and the nature of the application. The techniques are genetic algorithm: the most popular type of EA often seeks solutions in the form of string of numbers (binary or reflecting the problem). Genetic algorithm uses mutation and/or recombination methods for optimization problems.

Genetic programming: It provides solutions in the form of computer tools. The fitness of a solution is dictated by the capability to resolve difficult problems.

Evolutionary programming: It is often comparable with genetic programming. The difference is that the design of the program is fixed and its numerical variables are flexible to alter or evolve.

Evolution strategy: It uses vectors and normal numbers as representations to the results and uses same-adaptive mutation range.

Neuroevolution: It resembles a GA programming however the genomes speak to ANN by portraying structure and association weights. The genome encoding is either direct or indirect.

There are many more algorithmic techniques as cited by Price *et al*. (2006), Yang and Press (2010). This includes ant colony optimization, bee's algorithm, differential evolution, firefly algorithm, particle swarm optimization. Harmony search, invasive weed optimization algorithm and Gaussian adaption are added as additional algorithm techniques used (Giri *et al*., 2013).

Liteature review: Optimization of multi-objective facility layout problems in the area of operational research is used (Jannat *et al*., 2011). The study developed a GA for multi-objective facility layout problems taking considerations the quantitative and qualitative approaches. The method generates populations by crossover and mutation process. The approaches succeeded in minimizing total material handling cost and maximizing closeness rating scores and discovered a set of non-dominated solution (Pareto Optimal) for the facility layout. The Time Slot and Subject Group Assignment (TSSGAP) is used by Mushtaq *et al*. (2015) by running a tree search algorithm on the instances. The study concluded that the TSSGAP is NP-hard and that current tree search algorithm may able to solve no solution instances.

GA is used for solving small and large instances timetabling by Zhong *et al*. (2013). The study modified some basic genetic operators that restrain the creation of new conflicts in each character to enhance the algorithm's performance. The study reduces scheduling problems with large number of binary variables to acceptable size by eliminating certain dimensions of the problems and including the dimensions into constraints. The reduction of each gene size is done by grouping several of the binary numbers into one gene value. Such technique has the possibility of solving the full size problem (FER schedule), increases algorithm speed in tens of seconds with the GA approach. By utilizing intelligent operators, the algorithm converges much quicker than the basic algorithm and thus the study represents a good point to solve the FER.

Meta-heuristics is a successful area in solving timetable problems (Mohammed *et al*., 2016). Researchers cautioned that the approach is usually catered for specific problems and cannot be easily applied to other problems. Search methods have been improved recently with the development of hyper heuristics. Iterated Local Search (ILS) performs significantly better than Tabu search, steepest descent and variable neighborhood search methods.

The method employed for high level search is not important within the graph-based heuristic approach for examination timetabling problems. Soft constraints method is used for problem of varying size using GA, random search, hill climbing and simulated annealing (Perzina and Ramik, 2013). The method aims at maximizing the schedule fitness. The parameters that maximize the schedule quality are identified after a fixed number of fitness computations. The study concluded that a naive

crossover operator of GA performs better than random search. However, simulated annealing is found to be much more superior for the search space studied.

MATERIALS AND METHODS

Genetic Algorithm (GA) is a programming technique that imitates biological evolution as a problem-solving technique. In aligning with a specific problem, GA fitness function allows each candidate that needs to be solved to be quantitatively evaluated. The candidates may be the solutions and implementing GA makes the solutions better or improves the fitness level (Obaid *et al*., 2012). The GA evaluates each candidate at random according to the fitness function. Most of the time, the evaluation does not match or work, however by chance a few may work or even show weak or imperfect activity in solving the problem. These positive solution candidates are stimulated, reserved and permitted to reproduce (Mahdi *et al*., 2012).

Copies are made in multiple numbers but they are not made identical to the original. Random changes are introduced during the multiplying process. These digital offspring are the second generation that proceeds to the next GA evaluation process and forms a new pool of candidates that provide better solutions when evaluated in accordance with the GA fitness function. Reproduced candidates that carry weak or worse fitness solutions are discarded. The method works by multiplying candidates that carry better solution or increasing the number of good candidates. Even if in the beginning the chances are small by random changes and random matching, the pool of good candidates is large. The process works by increasing the average fitness of the population by each generation. By repeating the process in many iterations, a high solution potential to the problem is achieved. The multiplicity and the evaluation process works in fraction of seconds. Thus, a solution to the problem can be attained quickly.

Genetic algorithms have proven to be a powerful and successful problem-solving technique (Soule and Ball, 2001; Fleming and Purshouse, 2002; Park and Kim, 2017). The strategy has frequently demonstrated the power of evolutionary principles. Genetic algorithms have been used widely in a variety of fields that is difficult for humans to handle. The solutions are more credible, efficient, faster and of higher complexity than human engineers can solve.

In many cases, genetic algorithms produce solutions far better than the programmers who wrote the algorithm. The process of genetic algorithm is similar to the process of natural selection as illustrated in Table 1. Genetic algorithm is a heuristic search technique that replicates the process of natural evolution. The heuristic is used again and again to produce powerful solutions to optimization and search problems. It is a subset of Evolutionary Algorithm (EA) that uses optimization techniques similar to natural evolution. In natural evolution, a newer generation is either stronger or weaker depending on the type and degree of affect the genes are exposed to. In any problem-solving situation, the stronger candidates must be kept and the weaker candidates are eliminated. In genetic algorithm, a population of strings is called chromosomes, genotype or genome that encodes the candidates with potential solutions.

A candidate with potential solution is called an individual, creature or phenotype (Shah-Hosseini, 2009; Zhang *et al*., 2007). Before a genetic algorithm can be put to work to solve any problem, a method of representation is used to encode potential solutions in a form that a computer can process. Traditionally, solutions are provided in strings of 0 and 1 sec. The evolution begins with a randomly selected pool of candidates. The evaluation process eliminates the weak candidates that could not potentially provide solutions but retains the candidates that have some solution potential. The candidates with potential solution are kept and allowed to duplicate. The evaluation molds the new generation to be better fit and as the next generation is duplicated, the number of fitness candidate increases.

Combination and mutation occurs randomly (Fogel, 2006). The new generation is used in the next iteration of the algorithm. The algorithm terminates when a maximum number of generations has been reached and/or satisfactory fitness level has been achieved for the population. Occasionally, a satisfactory solution is not achieved even if the algorithm terminates due to a maximum number of generations (Mohammed, 2015; Fogel, 2006). Other methods of representation include encoding in array of integers or decimal numbers where each position represents an aspect of the solution. This method gives better precision and is able to handle more complex space (Fleming and Purshouse, 2002).

The third technique is to represent each character as strings of letters where each letter stands for specific aspect of the solution as used in Hiroaki Kitano's grammatical encoding approach (Thede, 2004). The advantage of these representational techniques is that it is easy to identify an operator that causes random changes to the candidates in the population. Figure 1 shows this technique.

Program trees are generally used in genetic programming with noted mathematical expressions. The construction of a genetic algorithm requires:

- A genetic pool of characters in the domain that represent the solution
- A fitness function to evaluate the solution in the domain

Table 1: Similarity between genetic algorithm and evolution/natural selection process

Genetic algorithm	Natural selection/evolution
Create pool of candidates/chromosomes/characters in a domain	Create a population of creatures in the universe
Delete or eliminate weak potential solution candidates through evaluation process	Kill all relatively unfit or weak creatures. In real life those who are weak naturally die faster
The candidates are allowed to reproduce and mutate to produce new generation. The candidates are molded and altered by fitness function so character will be slightly different from the original	In case of non-overpopulation Two or more life forms are allowed or stimulated new generations to mate
The new candidates are evaluated to fit in the original pool	The genetics (DNAs) are combined to produce a new creature causing a few random mutations on the new creature
The evaluation checks if the new generation is fit to be in the original population	Evaluate the new creature and place it in the (constrains of budgets time population. Nature allows the new generation that can fit in the existing world to survive
A maximum set number of candidates has been reached and money)	Reproduction and process continues until a life form reaches a set time (age), character (affect from illness) or overpopulation

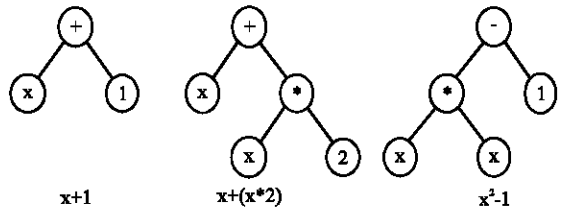

Fig. 1: Program trees (Mitchell, 1998)

A standard representation of the solution in a domain is typically an array of bits. The bits can be easily aligned due to their fixed size that enhances crossover operations (Mohammed, 2015a, b). Variation of shape and sizes are used for different programming. For example, graph form representation is used in evolutionary programming and tree-shaped representation is used in genetic programming. Nevertheless, for the purpose of crossover implementation, it is much simpler to use fixed-size representation. The purpose of a fitness function is to define the character representation and to measure the quality of the character representation. The fitness of the solution ensures that the sum of all characters in the domain is valid. Normally bit value of 0 is valid and 1 is not. In some cases when it is impossible to define the fitness expression, interactive genetic algorithms are used. Once the domain has the genetic representation and the fitness function defined, the algorithm initializes the pool of characters randomly (optimization process), then improves the candidate solution characters through repetitive application of mutation, crossover, inversion and selection operators (Mitchell, 1998).

In the selection process, strong possible solution characters are kept and stimulated. The selected breed then produces newer generation that has greater solution characters that are often referred to as more fit (matching the fitness function). The selection process either rates each selection and lists the best solutions or rates random samples of the population. A stochastic function assures that only small portions of less fit characters are selected for mutation. The function helps keep the diversity of the population big, increases the mutation of strong solution characters and prevents premature convergence that produces poor solutions (Ting, 2005). Roulette wheel selection and tournament selection are popular selection methods. Reproduction generates further generations of populations that are much stronger and fitter (Fig. 2).

The process is done through genetic operators, crossover recombination and mutation. A pair of solution character is selected for reproduction from the domain pool. The new character carries the character of the selected pair (Wang *et al*., 2008) and is then evaluated for its degree of fit. The use of more than one parent is better to reproduce a larger pool of solution characters, better quality characters and at a shorter time. Regrouping, colonized-extinction and migration are also used in GA depending on the problem. The reproduction process continues until conditions for termination are achieved. Termination conditions include:

- The maximum number of possible solutions character has been achieved (this does not offer solution satisfaction)
- The maximum number of set generations has been reached
- A solution is ascertained that satisfies the set minimum criteria. Time and money allocation for the process are reached. The highest possible fitness is obtained and further reproduction no longer produces better results

Chromosome encoding: The initial process of chromosomes encoding is choosing the data types. This is the first major schema variations between John Holland and others (Kraft *et al*., 1997). Holland initially encoded chromosomes in strings of binary digits. There are numerous ways to represents genes creatures, since Holland's work surfaced (Obaid *et al*., 2015; Fogel, 2006). These ways have their own advantages. To associate the problems with the gene forms, the material of the solution must be epitomized as a set of units of

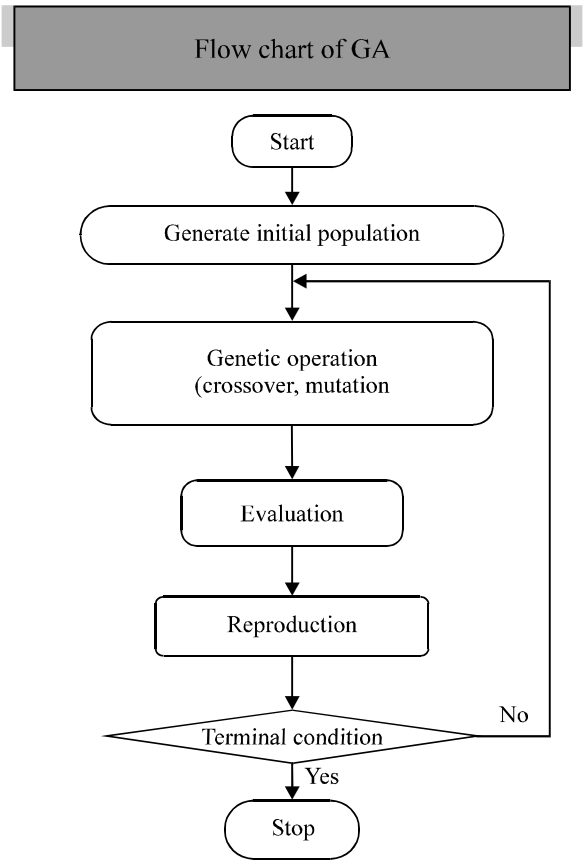

Fig. 2: Summary of the GA process (Wang *et al*., 2008)

information (Obaid *et al*., 2015). This way of association proves accurate. So, when planning for daily activities, the amount of time and the level of priority can be stored in a numerical value in a table.

The relationships between the values can be seen as string of genes. The value in the first row may represents the amount of time spend on meeting a client, the value in the second row may represent the amount of time driving to the airport and etc. Each of the amount of time spent can be converted from base 10-2 to create a fixed width binary number. Therefore, the problem of maximizing the time spent in a day and maximizing activity is construed as a genetic exemplification. A compilation of possible times spent can be encoded, giving a population of time spent creatures. In several cases, the targeted problems might be more handily construed by data types except binary (Osyczka and Kundu, 1995). For example, the times spent in a day of a person appears more identical to the list of real numbers with two decimal places (example: 1.45, 4.30 h) than a list of binary numbers.

The chicken and egg problem: Genotype is an encoded list of genes and phenotype. A phenotype might be encoded to introduce genotype or a genotype might be decoded to introduce a phenotype based on how the idea is put into sentence (Gen and Cheng, 2000). Both sentences refer to the same action. Most of the genes may be affected by external factors such as growth and nutrition (Obaid *et al*., 2015).

Population size: The first thing to do to create a genetic algorithm is to read and encode the entire population of chromosomes. The population's size must be decided and fitness factors must be determined depending on the availability and advancement of computing techniques. If the size of the population is small, the solution may not be satisfied (Gen and Cheng, 2000) because of inadequate exploration (the chromosomes are not allowed to reproduce enough to produce optimal solutions). Nevertheless, smaller population size may generate faster convergence (Obaid *et al*., 2015). On the other hand, if the size of population is too large, then time is wasted in dealing with more data/chromosomes than is required and convergence time is longer (Obaid *et al*., 2015; Park and Kim, 2017). Thus, the right population size depends on the objective of the solutions and the problem at hand. The algorithm determines the right population size based on the experience of the programmer.

Chromosome evaluation: In a pool of random chromosomes, it is highly likely that the chromosomes are extremely unfit for the solution (Obaid *et al*., 2015). In order to distinguish the chromosomes that are much fitter than the others, each chromosome must be evaluated. The distinguishing process requires two important information, one is the information about the chromosome and second, the environment where the target chromosome can survive. To know the environment in which the chromosomes can survive, the programmer must know the problem's description to be partially encoded/decoded. Thus, referring to the time spent in daily activities, one can set rules. For example, rather than spending so much time driving to the airport, it is better to spend time on an express train where the person can perform other activities, save money and safer. Note that the set of rules can be determined based on experience and the aims of the daily task. Set of rules can be applied to each daily activities and the time to be spent on the activities.

However, every rule might be given a relative significance by weighting when there are more than one rules involved (Halim, 2013). Weighting is based on the importance of the activities, hence the objective solution of the chromosomes. In structuring the method to evaluate the chromosomes, the programmer can set the

evaluation to either generate the least costly population or least time consumed based on budget constraints or to generate the fit population. For example, in the time spent example for daily activities, the time most spent might be exemplified with cost. In problems related to optimization, cost is not money or time but in terms of efficiency or life span (Obaid *et al*., 2015; Gen and Cheng, 2000). It is simpler to say that driving to the airport is wasteful and prone to accident (not safe for life). In this case, safety is seen as inversely proportional to cost and life so one might prefer the other (Gen and Cheng, 2000). In referring to optimization techniques, the range of possible solutions is usually referred to as the solution domain and the cost or fitness of each point in the solution domain is referred to as the altitude in the landscape of the problem at hand.

Consequently to look for the domain's minimum cost is also to look at the domain's lowest point of the cost landscape. Usually, in optimization problems, plenty of time is needed to extensively search for the solution in the domain for the minimum cost (Mostafa *et al*., 2012). Such techniques of optimization utilize two techniques to speed up the search. These optimization techniques are referred to as exploitation or exploration. Optimization level increases when the gap between exploration and exploitation is reduced. Optimization is to find a good medium in a shorter time period between exploration and exploitation.

Initializing population: There are two most methods for initializing a population (Barros *et al*., 2012). A population of creature (all of the genetic information about all the creatures in the colony) can be loaded from secondary storage. The storage is like a globe that contains characters and landscape. The data in the storage is a starting point for the directed evolution. Usually the genetic algorithm starts with a random population. The domain for the algorithm is made-up of population of chromosomes or life form with the genetic make-up that is delimited by aimless process (Obaid *et al*., 2015).

Selection methods for extinction or for breeding: The major aim of selection is to produce the most fitness chromosomes in the genes pool (Gen and Cheng, 2000) and to avoid over production of weak genes known as lethal. One of the most important considerations to achieve optimized solutions to a problem is to select the right breed for mutation. Once the population is full, each that matches the fitness function is selected for breeding. If the level of overall fitness is not yet achieved, a desired part of the creatures in the population can be selected for elimination of a species (Obaid *et al*., 2015). This refers to an elitist natural selection operator (Arbaoui *et al*., 2014). To increase solution optimization, early genetic algorithm process uses replacement strategy by replacing two parent's genes with two offspring in each generation. This process maintains the same number of population and avoids overcrowding process which is referred to as crowding strategy (Gen and Cheng, 2000). Now a days, a better crowding strategy is invented that enables a single offspring to replace any of its parents that it most resembles. Nevertheless, this process of gene comparison of an offspring with the parent is expensive and time consuming. Thus, it is not advisable for a study that has budget constraints.

As mentioned in the theory of genetic algorithm section, tournament selection is a technique used to decide which chromosomes to delete or eliminate from the population. In this schema, two chromosomes are compared against each other to find which one is fit according to the fitness function. The one that fits the most is allowed to reproduce and mutate and the one that is least fit is eliminated (Halim, 2013). This is aligned with the survival of the fittest concept in which the weaker is demised.

In the tournament process, each chromosome is matched against each other to fight once with another chromosome similar to a tournament concept. Other method of selection include elitist, fitness-proportionate selection, roulette-wheel selection, scaling selection, rank selection, generalization selection, steady-state selection and hierarchical selection (Ansari and Bakar, 2014). An issue of selection pressure, questioning the number of creatures that are wiped out from each tournament are highlighted in (Gen and Cheng, 2000). Comparing to real life/nature, plagues like flood, storms, tsunami or famines, the number of life forms removed from the earth is large.

However, in genetic algorithm, the number of creatures to be deleted depends on the programmer and if there are constraints to the program. The programmer can set the fitness function to be high and as a result many characters are deleted due to their inability to fit with the function in a set time. The method of selecting chromosomes/creatures in genetic algorithm is different depending on the problems at hand. John Holland's Original Model uses the method of the survival of the fittest in which the healthiest and the strongest chromosomes survive and continue to breed (Gen and Cheng, 2000). Other selection methods pick at random any two creatures for breeding. There are two types of species; high fitness species and low fitness species.

High fitness species usually evolved from population of chromosomes that mate between two high fitness members. Low fitness species usually referred to as "lethal" are intra-mating of members of a species of a low

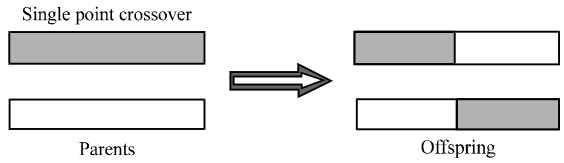

Fig. 3: Single-point crossover

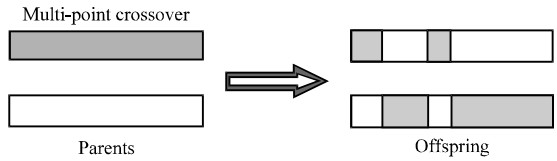

Fig. 4: Two or more-point crossover

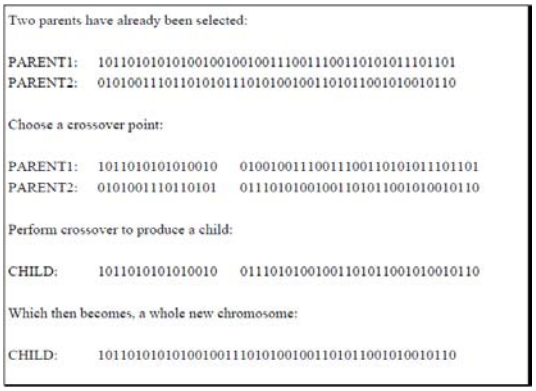

Fig. 5: Crossover with fully encoded genes (Kraft *et al*., 1997; Obaid *et al*., 2015)

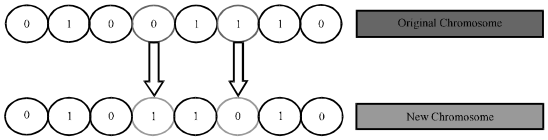

Fig. 6: Mutation process

fitness group with a member of a high fitness group or mating of two low fitness groups. The survival chance of the lethal species is low. The species may survive only in the next few generations.

Crossover, mutation and inversion: After the parent's chromosomes have been selected, the algorithm can allow the breeding process to start. A new chromosome is selected by choosing each gene in the chromosome an allele from either the father or mother. The operation of matching and combining the genes might be achieved in several ways. A single point crossover is the simplest and easiest method (Obaid *et al*., 2015; Kraft *et al*., 1997; Gen and Cheng, 2000). This process is similar to the ones proposed by Holland (1975) in which the genes are encoded in binary strings. The process translates almost any genes to be represented (Obaid *et al*., 2015). The offspring might be yielded using one-point crossover (Fig. 3). A single point crossover is haphazardly selected to happen somewhere in the gene's string. All genetic characteristics before the crossover point in the process reflect the parent (Kraft *et al*., 1997) and all genetic characteristic after the crossover reflects others (Sale and Sherer, 2015).

The difference between single-point and multi-point crossover (Fig. 4) is that in multi-point crossover, the genetic characteristics are extracted from many points in the character and thus producing offspring with multiple characters. The multiple characters optimize the level of fitness solution. Figure 5 illustrates a crossover with fully encoded genes.

The mutation process takes place with a small probability (Limkar *et al*., 2015). A probability test is conducted for each bit in a bit string. If the result is positive, one of these two methods can be performed. In method 1, the bit is flipped (0 changes to 1 and vice versa) and method 2 randomly generates the bit. Randomly generated bit and bit that does not match from the original is flipped to see if the flipped bit matches the original. This process is referred to as inversion. Figure 6 illustrates the mutation process in generating new chromosomes.

Lamarckian operators: In order to optimize the execution of a GA, there are several applications that implement other mechanisms on the chromosomes. Local hill climbing techniques or greedy algorithms can be applied on chromosomes that have been breed. The purpose of applying the technique is to increase the chromosome fitness prior to evaluation. This is referred to as Lamarckian evolution by Gen and Cheng (2000) as an analogy to Lamarck's theory of adaptation (Sherwood, 2015). The lesson learned from a chromosomes life can be passed to its offspring's (Gen and Cheng, 2000). If the algorithm is able to provide good lessons to the parents than the new generation of chromosomes can be optimized to the fittest.

Memetic algorithms: In GA, the integrity of information is a gene and the information is passed on from one generation to the next. This integrity of information is referred to as 'meme' in Mimetic Algorithm (MAs). The unit of information (meme) is read and altered by the chromosomes when it is received. This is similar to the life experience and information received by different individuals interpreted differently by the person and differ

from generation to generation (Gen and Cheng, 2000). As a result, this system reduces the solution space and reduces the number of generations to reach optimized solution (Arbaoui *et al*., 2014). Nevertheless, the system may increase the solution time at each step in reaching the optimal solution. By considering each character's meme, Lamarckian operators are able to compress the search space into smaller ones. This technique works well when an operator works with budget constraints such as time and money.

Repair strategies: In the process of optimizing, there is a possibility of genes to be produced outside the search space (Gen and Cheng, 2000). Examples of a system breeding lecture timetables are given. Two separate timetables that are constructed contain one class booking for a subject. However, if these timetables are not identical and merge properly, the combination (crossover) of these timetables may produce multiple class bookings. Consequently, there is an error in the timetable where students from different classes are assigned to one class. This is more serious in examination timetable scheduling. The reason is that for examination, the administrators and the students do not have ample time to correct the problem. Repair strategy is a method to form the chromosome in the search space to make sure that there is only one booking for each class.

The repair strategy alters the chromosome gene to automatically detect one booking for its class. Rejection scheme rejects any chromosomes which is external to the search space of the problem. A penalty strategy is applied in the search of a much larger search space and thus, penalizes the problem by increasing cost and time. In the case of timetabling problem, the repair strategy is much better than rejection strategy or penalty strategy because it converges the chromosomes in the search space faster and maintains the original search space (Gen and Cheng, 2000). Determining the creature's fitness is the final stage of the evolution of the chromosomes. The process is performed similarly to the initialization process (Obaid *et al*., 2015; Sherwood, 2007). Each complete cycle of iteration is referred to as a generation.

The process can be terminated when the maximum number of chromosomes or possible solutions has been reached, a solution to the problems has been achieved, budget constraints reached and maximum possible fitness has been reached, manual interruption and combination of these factors. The termination process occurs when the algorithm has produced at least one sufficiently fit chromosome or a fit population depending on the optimization target. Moreover, termination occurs when the algorithm has ran out of preset number of generations (Park and Kim, 2017).

Optimizing genetic algorithm performance: In any algorithmic process, altering the variables value improves the quickness and validation of the evolutionary process (Goldberg, 2013). These variables contain mutation's rate, selection method, number of crossovers schemes, constraint weightings and more. In more complex algorithms, the variables are increased thus, increasing the possibilities of alterations to increase speed and effectiveness. There is the most effective range value for each variable in the algorithm.

By effectively combine and accurately interpolate the ranges, optimization is improved. The combination of values at different times increases the possibility of finding an accurate optimization in a shorter time that is similar to randomization (Grefenstette, 2013). There are many other techniques of optimizing algorithm performances including evolution strategy, Neuroevolution, firefly algorithm, particle swarm optimization and harmony search.

RESULTS AND DISCUSSION

The bi-annual construction of a university examination timetable is a common problem for all educational institutions across the globe. Some university examination timetables are manually done with the help of some simple software like Microsoft Excel by a few administrators who are often involved in multitasking activities (scheduling final examination, answering students, lecturer or staff questions, maintaining examination papers submissions and etc). The construction occasionally faces problems like the administrator taking leave, renovations of examination rooms and inadequate facilities, laboratories, studios or equipment. Most examination schedules are taken from the previous semester and modified to work for the new semester. Many institutions are forced to introduce quarterly intake and quarterly examination to cater for the increasing demand of education. For the timetable, this pose higher constraints and utilizing the previous semester schedule is no longer efficient. In every semester a new examination timetable must be constructed to cope with the increasing number of students and classes. The timetable must also be sensitive to the growing number of subjects/specializations every year. Most of the time, the university examination department is not aware of these growth. The university does not provide additional staff, training or infrastructure (fast computers, software programmers, etc.) to cope with these constraints (Mahiba and Durai, 2012).

Educational institutions and timetabling problem: An examination timetabling is a problem of assigning examinations to periods and rooms. The examination

timetable is one of a university's main timetables other than the course timetable. The course timetable is released before a semester starts and the examination timetable is released before the semester ends. There is also another timetable which is the mid-term examination timetable. These timetables are related to each other and yet they can be quite different. Sometimes, it is acceptable for two examinations to be held simultaneously in one examination hall but it is not possible for two subjects to be taught in one classroom in the same time period. Examination halls are often shared by many faculties/departments whereas each department uses its own classroom and generally located in its faculty building (Feng *et al*., 2016; Limkar *et al*., 2015).

Examination timetabling problem description: A university's final examination timetable is a table for coordinating students, teachers, rooms, time slots (periods), class subjects and other examination room types (computer labs, halls, studio, etc.). Every university timetable has its own special problems. Most universities consult experts to find solutions to the problems. The scenario is not as easy as it seems mostly due to the university itself that lacks staff, rooms, facilities, staff work restrictions and student's requirements (Ahmadi *et al*., 2014). University examination timetable involves more human judgment because of the staff's lesser teaching loads and is less constrained than school timetabling. In some institutions where the students are not given choices in their subjects (no electives or co-curriculum subjects), the timetabling is simpler. The task of constructing a university's examination timetable involves several issues including avoiding students to take more than one examination per day or staff to invigilate more than one examination at the same time.

Some subjects require back-to-back room reservation. For example, a subject that involves theoretical examination followed by a practical one. Staffing, additional staff scheduling, assigning special rooms and lessened weekend or evening examination periods are other challenges that the scheduler must consider. The university's examination timetable should not put stress on the students, teachers and administrative staff. For example, by having the students to travel a distance from their homes and not providing ample time for the students to travel from one examination hall to another. The university examination timetable must function to ease the life of teachers, staff and students.

Problem solution approaches: Over the years, many programs and applications have been invented to solve universitie's lecture and examination timetabling problems. UniTime is an open source system designed to construct course and examination timetables. The system permits several universities to arrange labor to construct and quickly change the timetable to serve the educational institution needs and minimize conflicts. The Stochastic Optimization Timetabling Tool (SOTT) has been developed for a university's course timetabling (Rudova, 2015; Hassani and Habibi, 2013). A multi-objective instance of a university's examination timetabling problems is demonstrated in (Chaturvedi, 2013). The instance satisfies universal hard constraints like seating capacity and examinations overlap.

The multi-objective approach requires minimization of the timetable length as well as the number of occurrences of students having to take the examination in consecutive periods or in the same day. The multi-objective algorithm is able to produce feasible solutions without prior information. The effectiveness of this algorithm is better in comparison with other available optimization techniques (Chaturvedi, 2013). Drools Solver algorithm to solve examination scheduling problems considers constraints of physical resource and fulfilling student's satisfaction is used by Gervasi. The related study also investigates the solution for a real world data sets. The study consists of 934 examinations, 36 periods and 48 rooms generated 105,000 possible solutions. The study found that the application is capable of maintaining hard and soft constraints and provides staff and students satisfaction. The algorithm produces a timetable that avoids conflicts and does not add stress to staff and students. The generated timetable avoids large examinations near the end of the schedule, examination of different durations in the same time period and two examinations subsequently held in one examination hall (Ozcan *et al*., 2012).

A case study on examination timetable problem using GA's is discussed by Gonsalves and Oishi (2015). The study concludes that GA can be applied in domains where there is insufficient knowledge or the size and complexity is too high or labor intensive. Crossover swaps is a pre-defined bits amount, usually 1 in GAs. However, the study found that the swapping can lead to convergence of the offspring instead of calculating towards a near fit solution. In this case, mutation is used where 1 bit is set either from 0-1 or vice versa (Gonsalves and Oishi, 2015). The study suggests further improvements of the parameters, the crossover amount, crossover occurrences and mutation

percentage. Further, study also needs to look into defining the optimal values and draw closer parallelism with biological evolution and implement further computational models similar to assigning fitness levels to string parts or introducing new set of input strings during computation.

Traditional solutions and methods: Traditional GA algorithms to solve timetable problems use one crossover and one mutation operator to produce the offspring (new generation). The chosen crossover and mutation methods are critical to the success of the GA. Different crossover and mutation methods work best for inconsistent problems and for several stages of the problems (Sastry *et al*., 2014). Determining the crossover and mutation operators is a difficult task and time consuming. The process is done on one to one trial basis.

A Dynamic Genetic Algorithm (DGA) is proposed (Sastry *et al*., 2014) which simultaneously uses more than one crossover and mutation operators to generate new generations. The crossover and mutation rate changes with the evaluation results. The DGA increases the speed of crossover and mutation and performs better than the traditional algorithm with a single crossover and mutation operator. Parallel Genetic Algorithm (PGA) is used in (Kumar *et al*., 2012), a method that reduces processing time. The result shows that substantial speed of GA is increased with a small communication overhead. The overhead amount is about 13% of the total computational time when using ten processors. The study found improved performance for large number of processors. Increasing processor quantity increases the speed of GA.

Ant colony approaches are used by Sabar *et al*. (2011) to optimize examination timetabling solutions. Ant colony optimization algorithm is a special approach of swarm intelligence. The algorithm is first used in Traveling Salesperson Problem (TSP) and is referred to as Ant System (AS). In an AS, an ant is constructed to visit all nodes in the domain. An ant can be trained to learn from the past to make its decision to choose the next node. The ant applies either a constructive heuristic strategy or a pheromone trail strategy.

The constructive strategy applies one priority rule randomly on the nearest neighbor that the decision to mutate depends on the distances between the nodes that host the ant and the node to be mutated. The pheromone trail strategy depends heavily on the ant itself in finding the shortest path to a node that needs to be mutated. The ant releases a chemical substance called pheromone to find the shortest path and communicate between two locals. The ant chooses the paths with higher pheromone level. The study found that ant colony approaches may solve various joint optimization problems.

Final examination timetabling: The GA approach to examination timetabling requires initial definitions of constraints. The conditions are grouped into two categories. The first category is internal and coordinates the division of periods of time and organization of resources. The second is external and limits the universe of scheduling.

Soft and hard constraints: A timetable must serve and overcome several constraints. Constraints are used by people who deal with timetabling problems without exception. In GA there are two types of constraints; soft and hard constraints. When constructing an examination timetable, hard constraints, require that there are no violations such as two classes cannot be at different locations at the same time (Limkar *et al*., 2015). An extensive list of hard constraints related to final examination scheduling.

Extensive list of final examination hard constraints:

- Final examination class must not be multiple reserved
- Each examination for every subject should be booked just once
- The examination for every subject must not be reserved together
- The examination room must to be enormous to hold the subjects booked for it
- Staff or teacher to invigilate should not be multiple reserved
- The invigilator (staff or lecturer) must be available during the examination and not be booked somewhere else
- Some examinations require special rooms like laboratories, studios, computer labs, etc.
- A few examinations require unique rooms like research centers, studios, PC labs, etc.
- Some examination rooms need to be held consecutively

Soft constraints are limitations that can be abused, however infringement must be minimized. For instance, a last examination must be held nearest to the branch where classes are held. There are numerous soft constraints proposed by various scientists. The soft constraints request of significance is additionally extraordinary. A commonplace list of soft constraints for definite examination timetabling is appeared.

A list of final exam soft constraints:

- The distance between the student's home to the examination room must be minimized
- The examination rooms should be centrally located Staff and lecturers do not wish to invigilate examinations continuously (one examination after the other)
- Most students do not wish their final examinations be held back to back
- All final examinations must be held in one week (examination period)
- Students, staff and lecturers do not wish final examinations to be held on weekends on holidays

In constructing an automated final examination timetabling system, beside the list of hard and soft constraints that must be considered, there is also some exceptions as listed.

List of exceptions in constructing final exam timetable:

- The final examinations are held at each campus simultaneously
- An examination schedule is not needed for some subjects (e.g., co-curriculum subjects, long distance learning, etc.)
- There are many staff and lecturers to invigilate
- There are many examination rooms to play with
- Examination rooms should be larger to hold the whole class

Constraint satisfaction: Constraint fulfillment in AI field is the conceivable qualities an arrangement of factors can take. Informally, a finite space is a finite group of random components. A constraint fulfillment issue on space contains a group of factors whose qualities must be taken from the area and a group of preconditions, involving any constraint of the permitted values for a gathering of factors. An answer for this issue is an assessment of the considerable number of factors limitations.

In testing, limitations are expressed in consolidated shape, instead of identifying every one of the estimations of the factors that would fulfill the limitation. A standout amongst the most well-known limitations is the finding that the estimations of the factors influenced are all extraordinary. Issues that can be communicated as imperative fulfillment issues for instance are the eight rulers confuse, the Sudoku taking care of issue the Boolean satisfiability issue, booking issues and different issues on diagrams, for example, chart shading issue.

GA application in examination timetabling: Automated timetabling incorporates class and examination timetabling and has been the primary advancement issues for GA applications. The idea utilizes mixture approaches in which a developmental method plays out a search in the area to change. The underlying timetable is bolstered by particular mutation step to produce a final timetable. The technique often produces a timetable that fulfills all hard constraints. In general, GA produces examination timetables that meet hard and soft constraints and the regulation set. When there is no penalty in an algorithm with zero solution it shows that the solution does not exist.

Nature of representation is showed by Pillay (2014) utilizing UniLang. The info language method gives a reasonable and straightforward regular information representation. The method likewise contemplates imperatives and gives quality measures and answers for various and related examination timetabling issues. The utilization of non-specific calculations to take care of timetabling issues is examined by Ahandani *et al*. (2012) and proposed a system that gives more noteworthy adaptability on the presentation of determined limitations and assessment capacity of the arrangements. The review additionally gives tips on the utilization of GA for determination. A bland dialect that joins timetabling issues and its limitations is additionally proposed (Grobner and Wilke, 2002). In the review, the scientist infers that the bland dialect can be utilized to sum up all timetabling issues. The review endeavors to tackle class-teacher timetabling issues for little schools.

A more efficient search algorithm is used by Kajisha and Saito (2000) for one-dimensional Cellular Automata (CAs) with self-replicating structure. In their algorithms, the CA structure is represented by simple fitness function and a genetic algorithm is used effectively in which a gene is used as a rule table. The review found that ideal mutations rate for fitness development are available and some normal examples can be drawn by the genes if the genes are to advance effectively. There are numerous more analysts who utilize diverse algorithmic methodologies and give approaches to advancing arrangements connected for the entire procedure or for part of the procedure.

Reasons for using GA in timetabling: Genetic Algorithm is one of the optimization algorithms that are based on evolution of life (Seto and Kanasugi, 2012). GA is able to solve various problems such as optimization problems and machine learning. GA is used to solve examination schedule for a university by Abbaszadeh and Saeedvand (2014). The study introduces the term NP-complete which means that no method is known that guarantees optimal solution in a period of time. GA has to comply with the hard and soft constraints of examination scheduling. The rooms, staff and students are regarded as characters that have to be arranged to fit specified

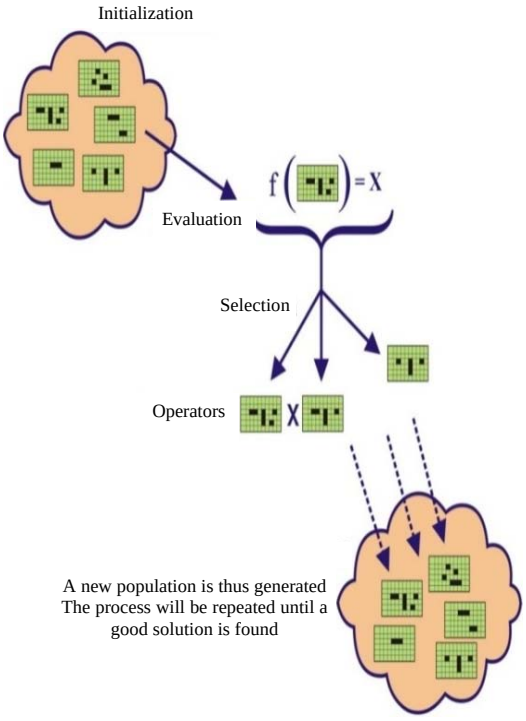

Fig. 7: The parallel connection between ga process and the process of creating an examination timetable

fitness function. The researchers used a hybrid approach for solving problem of a mimetic algorithm. The study used evolutionary algorithm with rank-based fitness proportionate selection combined with a local hill-climber to optimize solutions found by EA. The study produced a result of 40% reduction penalty.

Genetic algorithm is applied to generate schedules for job shops by Mohammed *et al.* (2014). The algorithm has to consider many NP-constraints such as cost, tardiness and etc. The study concludes that GA produce efficient schedules and can be applied to real-world situations. Figure 7 shows the parallel connection between GA process and the process of creating an examination timetable.

The process starts with generating a random population of feasible timetables using a variation on a graph coloring algorithm. The timetables are then evaluated according to a set of criteria, e.g., the length of the timetables, how many days is in the examination week, the number of students who sit in two examinations in a row or how many unused seats there are. The timetables are randomly selected to be the foundation for the next generation. The good timetables are automatically chosen than the bad ones. The Mutation operator randomly changes the period and room in which the examination is to be held while maintaining a feasible timetable. The crossover operator takes pairs of timetables, selects the early examinations from one and the late examinations from the other to produce a new timetable. Any examinations that cannot be placed this way are put in the earliest available period.

Other techniques: This study deals mainly with solutions to university examination timetabling problem. In this study, the most used techniques to solve the examination timetabling problem are discussed.

Evolutionary algorithms: Evolutionary Algorithms (EA) are considered as a good general purpose optimization tool due to their high flexibility accompanied by conceptual simplicity. Moreover, they have proven to be a very effective tool for solving timetabling problems. An EA framework is chosen as the basis to build universal timetable problem solver. Term "EA" is used in this study in its most general sense, it represents a population-based metaheuristic optimization algorithm that uses mechanisms inspired by biological evolution such as reproduction and mutation.

Tabu search: A hyper-heuristic works on a higher level of abstraction than a meta-heuristic and requires no domain knowledge (Mushtaq *et al.*, 2015). A hyper-heuristic has access only to non-domain specific information it receives from the heuristics that it works on. Initially, the hyper-heuristics must know the number, n of heuristics used by the low-level heuristics module. It searches for a good quality solutions guided by its own strategy for making and evaluating the performance of each heuristic known by their generic names H_1, H_2, H_n. The hyper-heuristic does not need to change the name, purpose or implementation details of each low-level heuristics (Hassani and Habibi, 2013). It just has to call a specific heuristic, h and the heuristic solution is to modify and return the result via an evaluation function. The following algorithm shows a general framework of hyper-heuristic.

Algorithm:

```
Algorithm: hyper-heuristic
Begin
    Built an initial solution
    Do
        Select not Tabu heuristics
        Apply the chosen heuristics
        Update solution
    Until terminating condition
End
```

The first solution is produced using a constructive heuristic (highest grade or degree of saturation)

(Mohammed, 2015). Subsequently, a randomized (random examinations to move to a valid slot) is performed for different runs starting with different solutions. In Step 2, we explore the area to find a better solution or local optima (and possibly global optima). The framework is similar to a local search, except that in Step 2, we explore the area by selecting the heuristic to make the current solution works (Abbaszadeh and Saeedvand, 2014).

Simulated annealing: Simulated Annealing (SA) is a generic probabilistic metaheuristic for global optimization problem of finding a good adaptation to the global optimum of a given function in a large search space. It is often used when the search space is discrete (e.g., all tours that visit a given set of cities). For certain problems, simulated annealing are more effective than exhaustive provided the goal was only good to an acceptable solution in a fixed amount of time, rather than the best possible solution (Mushtaq *et al*., 2016).

The name and inspiration come from annealing in metallurgy, a technique involving heating and controlled cooling of the material to the size of the crystals increase and reduce their defects. The heat causes the atoms to unstuck from their initial position (a local minimum of the internal energy) become random wander through states of higher energy the slow cooling gives them more chances of finding configurations with lower internal energy than the original.

Graph coloring: In the area of graph theory, graph coloring is a special case of graph labeling and is an assignment of labels traditionally called "colors" to elements of a graph subject to certain limitations. In its simplest form, it is a way of coloring the vertices of a graph such that no two adjacent vertices have the same color which is called a vertex color. A border color has a color to each edge so that no two adjacent edges share the same color and a face coloring of a planar graph indicates a color to each face or region so that no two faces that share a border have the same color. Vertex coloring is the starting point of the subject and other coloring problems can be converted into a vertex version.

Graph coloring enjoys many practical applications and theoretical challenges. Besides the traditional types of problems, different restrictions may be imposed on the graph or the way a color is assigned or even the color itself. It has even reached popularity with the general public in the form of the popular Sudoku. Graph coloring is still a very active area of research (Mohammed, 2015). Vertex color models a number of planning problems. In the cleanest form, a certain set of jobs should be assigned to time slots where each job requires such a slot. Jobs can be scheduled in any order but pairs of jobs can conflict in the sense that they cannot be assigned to the same time slot, perhaps because they both rely on a shared resource (Bello *et al*., 2008).

CONCLUSION

Examination scheduling is one of the most important real life tasks and many universities are facing problems with constraints in producing a perfect timetable. Examination scheduling is known to be NP-hard and the parallelism between GA and generating timetable process make GA a popular method in solving timetable problems. Methodologies like General Algorithms (GAs), Evolutionary Algorithms (EAs), etc. have been applied to achieve optimum solutions with mixed success. There are many techniques in solving examination timetable problems, some looks at the general or overall algorithm and some looks at part of the processes. However, all of the studies aim at improving the quality and effectiveness of the GAs. Genetic algorithm effectively demonstrates the ability to solve complex optimization problems. The future of GA is promising in solving examination timetabling problems at university level. If a university is willing to fully support the initiative in applying GA in solving their scheduling problems the prospect of applying GA will be huge.

ACKNOWLEDGEMENT

This research has been funded and supported by fellowship scheme (UTeM Zamalah scheme) by Universiti Teknikal Malaysia, Melaka, Malaysia

REFERENCES

Abbaszadeh, M. and S. Saeedvand, 2014. A fast genetic algorithm for solving university scheduling problem. IAES. Intl. J. Artif. Intell., 3: 7-15.

Ahandani, M.A., M.T.V. Baghmisheh, M.A.B. Zadeh and S. Ghaemi, 2012. Hybrid particle swarm optimization transplanted into a hyper-heuristic structure for solving examination timetabling problem. Swarm Evol. Comput., 7: 21-34.

Ahmadi, F., R. Tati and S. Safavi, 2014. A novel approach for university course scheduling in Islamic Azad University. J. Current Res. Sci., 2: 909-914.

Ansari, A. and A.A. Bakar, 2014. A comparative study of three artificial intelligence techniques: Genetic algorithm, neural network and fuzzy logic, on scheduling problem. Proceedings of the 2014 4th International Conference on Artificial Intelligence with Applications in Engineering and Technology, December 3-5, 2014, IEEE, Selangor, Malaysia, ISBN:978-1-4799-7910-3, pp: 31-36.

Arbaoui, T., J.P. Boufflet, K. Hu and A. Moukrim, 2014. Exam timetabling at university of technology of compiegne: A memetic approach. Proceedings of the 10th International Conference on the Practice and Theory of Automated Timetabling, August 26-29, 2014, University of Technology of Compiegne, Compiegne, France, pp: 438-441.

Ashlock, D., 2006. Evolutionary Computation for Modeling and Optimization. 1st Edn., Springer Science+Business Media, New York, ISBN: 0-387-22196-4.

Barros, R.C., M.P. Basgalupp, A.C.P.L.F. de Carvalho and A.A. Freitas, 2012. A survey of evolutionary algorithms for decision-tree induction. IEEE Trans. Syst. Man Cybern. Part C: Applic. Rev., 42: 291-312.

Bello, G.S., M.C. Rangel and M.C.S. Boeres, 2008. An approach for the class teacher timetabling problem using graph coloring. Pract. Theor. Autom. Timetabling, 1: 1-6.

Beyer, H.G., H.P. Schwefel and I. Wegener, 2002. How to analyse evolutionary algorithms. Theor. Comput. Sci., 287: 101-130.

Chaturvedi, J., 2013. Application of quantum evolutionary algorithm to complex timetabling problem. Open Sci. Repository Comput. Inf. Sci., 1: e70081951-e70081951.

Clune, J., C. Ofria and R.T. Pennock, 2008. How a Generative Encoding Fares as Problem-Regularity Decreases. Parallel Problem Solving from Nature-PPSN X. PPSN 2008 Lecture Notes in Computer Science, Vol. 5199, September 13-17, 2008, Springer, Berlin, Germany, pp: 358-367.

Eiben, A.E. and J.E. Smith, 2003. Introduction to Evolutionary Computing. Springer, New York, USA., ISBN-13: 9783540401841, Pages: 199.

Feng, X., Y. Lee and I. Moon, 2016. An integer program and a hybrid genetic algorithm for the university timetabling problem. Optim. Methods Software, 32: 1-25.

Fleming, P.J. and R.C. Purshouse, 2002. Evolutionary algorithms in control systems engineering: A survey. Control Eng. Pract., 10: 1223-1241.

Fogel, D.B., 2006. Evolutionary Computation: Toward a New Philosophy of Machine Intelligence. 3rd Edn., John Wiley & Sons, New York, USA., ISBN:13:978-0-471-66951-2, Pages: 273.

Gen, M. and R. Cheng, 2000. Genetic Algorithms and Engineering Optimization. Vol. 7, John Wiley & Sons, Hoboken, New Jersey, USA., Pages: 499.

Giri, B.K., F. Pettersson, H. Saxen and N. Chakraborti, 2013. Genetic programming evolved through bi-objective genetic algorithms applied to a blast furnace. Mater. Manuf. Processes, 28: 776-782.

Goldberg, D.E., 2013. The Design of Innovation: Lessons from and for Competent Genetic Algorithms. Vol. 7, Springer, Berlin, Germany, ISBN:978-1-4757-3645-8, Pages: 247.

Gonsalves, T. and R. Oishi, 2015. Artificial immune algorithm for exams timetable. J. Inf. Sci. Comput. Technol., 4: 287-296.

Grefenstette, J.J., 2013. Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms. Psychology Press, Park Drive, UK., Pages: 259.

Grobner, M. and P. Wilke, 2002. A general view on timetabling problems. Ph.D Thesis, University of Erlangen-Nuremberg, Erlangen, Germany.

Halim, A., 2013. A micro-genetic algorithm approach for soft constraint satisfaction problem in university course scheduling. Ph.D Thesis, Universiti Utara Malaysia, Changlun, Malaysia.

Hassani, M.S.A. and F. Habibi, 2013. Solution approaches to the course timetabling problem. Artif. Intell. Rev., 39: 133-149.

Holland, J.H., 1975. Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence. 1st Edn., University of Michigan Press, Ann Arbor, MI., USA., ISBN-13: 9780472084609, Pages: 183.

Jannat, S., A.A. Khaled and S.K. Paul, 2010. Optimal solution for multi-objective facility layout problem using genetic algorithm. Proceedings of the 2010 International Conference on Industrial Engineering and Operations Management, January 9-10, 2010, Bangladesh University of Engineering and Technology, Dhaka, Bangladesh, pp: 1-6.

Kajisha, H. and T. Saito, 2000. Synthesis of self-replication cellular automata using genetic algorithms. Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks, Vol. 5, July 27-27, 2000, IEEE, Tokyo, Japan, ISBN:0-7695-0619-4, pp: 173-177.

Kraft, D., F. Petry, B. Buckles and T. Sadasivan, 1997. Genetic Algorithms for Query Optimization in Information Retrieval: Relevance Feedback. In: Genetic Algorithms and Fuzzy Logic Systems: Soft Computing Perspectives, Sanchez, E., T. Shibata and L.A. Zadeh (Eds.). World Scientific Publishing Co. Pte. Ltd., London, pp: 155-173.

Kumar, K., R.S. Sikander and K. Mehta, 2012. Genetic algorithm approach to automate university timetable. Intl. J. Tech. Res., 1: 47-51.

Limkar, S., A. Khalwadekar, A. Tekale, M. Mantri and Y. Chaudhari, 2015. Genetic Algorithm: Paradigm Shift Over a Traditional Approach of Timetable Scheduling. In: Proceedings of the 3rd International Conference on Frontiers of Intelligent Computing: Theory and Applications, Satapathy, S., B. Biswal, S. Udgata and J. Mandal (Eds.). Springer, Berlin, Germany, pp: 771-780.

Mahdi, O.A., M.A. Mohammed and A.J. Mohamed, 2012. Implementing a novel approach an convert audio compression to text coding via hybrid technique. Intl. J. Comput. Sci. Issues, 9: 53-59.

Mahiba, A.A. and C.A.D. Durai, 2012. Genetic algorithm with search bank strategies for university course timetabling problem. Procedia Eng., 38: 253-263.

Mitchell, M., 1998. An Introduction to Genetic Algorithms Cambridge, Massachusetts. MIT Press, Cambridge, UK.

Mohammed, M.A., 2015. Design and implementing an efficient expert assistance system for car evaluation via fuzzy logic controller. Intl. J. Comput. Sci. Software Eng., 4: 60-68.

Mohammed, M.A., 2015. Investigating role of knowledge auditing in profile of the business UNIT-information technology and computer center university of Anbar. Intl. J. Enhanced Res. Manage. Comput. Appl., 4: 10-18.

Mohammed, M.A., A.B. Khateeb and D.A. Ibrahim, 2016. Case based reasoning shell frameworkas decision support tool. Indian J. Sci. Technol., Vol. 9, 10.17485/ijst/2016/v9i42/101280.

Mohammed, M.A., A.K. Belal and D.A. Ibrahim, 2016. Human interaction with mobile devices on social networks by young and elderly people: Iraq a case study. Indian J. Sci. Technol., Vol. 9, 10.17485/ijst/2016/v9i42/101281.

Mohammed, M.A., A.T.Y. Aljumaili and H.A. Salah, 2014. Investigation the role of cloud computing in the business value for optimal criteria. Intl. J. Enhanced Res. Sci. Technol. Eng., 3: 111-118.

Mohammed, M.A., M.S. Ahmad and S.A. Mostafa, 2012. Using genetic algorithm in implementing capacitated vehicle routing problem. Proceedings of the 2012 International Conference on Computer and Information Science (ICCIS), June 12-14, 2012, IEEE, Ramadi, Malaysia, ISBN:978-1-4673-1937-9, pp: 257-262.

Mohammed, M.A., O.I. Obaid and M.S. Ahmad, 2015. Using Genetic Algorithm in Solving Capacitated Vehicle Routing Problem. OmniScriptum Publishing, Saarbrucken, Germany,.

Mostafa, S.A., M.S. Ahmad and M. Firdaus, 2012. A soft computing modeling to case-based reasoning implementation. Intl. J. Comput. Appl., 47: 14-21.

Mushtaq, A.D., M. Hojabri, D. Hamdan and M.H. Ali, 2015. Maximum power prediction for PV system based on P and O Algorithm. J. Adv. Appl. Sci., 3: 113-118.

Obaid, O.I, M.A. Mohammed and M.S. Ahmad, 2015. Solving Examination Timetabling Problem by using Genetic Algorithm. OmniScriptum Publishing, Saarbrucken, Germany,.

Obaid, O.I., M. Ahmad, S.A. Mostafa and M.A. Mohammed, 2012. Comparing performance of genetic algorithm with varying crossover in solving examination timetabling problem. J. Emerg. Trends Comput. Inf. Sci., 3: 1427-1434.

Osyczka, A. and S. Kundu, 1995. A new method to solve generalized multicriteria optimization problems using the simple genetic algorithm. Struct. Multi. Optim., 10: 94-99.

Ozcan, E., M. Misir, G. Ochoa and E.K. Burke, 2012. A Reinforcement Learning: Great-Deluge Hyper-Heuristic. In: Modeling, Analysis and Applications in Metaheuristic Computing: Advancements and Trends: Advancements and Trends, Yin, P.Y. (Ed.). National Chi Nan University, Taiwan, pp: 37-412.

Park, J. and K.Y. Kim, 2017. Meta-modeling using generalized regression neural network and particle swarm optimization. Appl. Soft Comput., 51: 354-369.

Perzina, R. and J. Ramik, 2013. Timetabling problem with fuzzy constraints: A self-learning genetic algorithm. Constraints, 3: 105-113.

Pillay, N., 2014. A survey of school timetabling research. Ann. Oper. Res., 218: 261-293.

Price, K., R.M. Storn and J.A. Lampinen, 2006. Differential Evolution: A Practical Approach to Global Optimization. Springer, New York, USA., ISBN-13: 9783540313069, Pages: 539.

Rudova, H., 2015. University Course Timetabling: From Theory to Practice. Masaryk University, Brno, Czech Republic.

Sabar, N.R., M. Ayob, G. Kendall and R. Qu, 2011. A honey-bee mating optimization algorithm for educational timetabling problems. Eur. J. Operat. Res., 216: 533-543.

Sale, M. and E.A. Sherer, 2015. A genetic algorithm based global search strategy for population pharmacokinetic pharmacodynamic model selection. Br. J. Clin. Pharmacol., 79: 28-39.

Sastry, K., D.E. Goldberg and G. Kendall, 2014. Genetic Algorithms. In: Search Methodologies, Burke, E.K. and K. Graham (Eds.). Springer, Berlin, Germany, ISBN:978-1-4614-6939-1, pp: 93-117.

Seto, S. and A. Kanasugi, 2012. A novel distributed genetic algorithm with redundant binary number. Proceedings of the 2012 8th International Conference on Information Science and Digital Content Technology, Vol. 2, June 26-28, 2012, IEEE, Adachi, Japan, ISBN:978-8-9886-7870-1, pp: 273-276.

Shah-Hosseini, H., 2009. The intelligent water drops algorithm: A nature-inspired swarm-based optimization algorithm. Int. J. Bio-Inspired Comput., 1: 71-79.

Sherwood, L., 2015. Human Physiology: From Cells to Systems. Cengage learning, Boston, Massachusetts.

Soule, T. and A.E. Ball, 2001. A genetic algorithm with multiple reading frames. Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation, July 07-11, 2001, Morgan Kaufmann Publishers, San Francisco, California, ISBN: 1-55860-774-9, pp: 615-622.

Thede, S.M., 2004. An introduction to genetic algorithms. J. Comput. Sci. Colleges, 20: 115-123.

Ting, C.K., 2005. On the Mean Convergence Time of Multi-Parent Genetic Algorithms Without Selection. In: Advances in Artificial Life ECAL Lecture Notes in Computer Science, Capcarrere, M.S., A.A. Freitas, P.J. Bentley, C.G. Johnson and J. Timmis (Eds.). Springer, Berlin, Germany, pp: 403-412.

Wang, Y.M., N.F. Xiao, H.L. Yin, E.L. Hu and C.G. Zhao *et al*., 2008. A two-stage genetic algorithm for large size job shop scheduling problems. Intl. J. Adv. Manuf. Technol., 39: 813-820.

Yang, X.S. and L. Press, 2010. Nature-Inspired Metaheuristic Algorithms. 2nd Edn., University of Cambridge, Cambridge, UK., ISBN:13:978-1-905986-28-6, Pages: 147.

Zhang, J., H.S.H. Chung and W.L. Lo, 2007. Clustering-based adaptive crossover and mutation probabilities for genetic algorithms. IEEE. Trans. Evol. Comput., 11: 326-335.

Zhong, J.H., M. Shen, J. Zhang, H.S.H. Chung and Y.H. Shi *et al*., 2013. A differential evolution algorithm with dual populations for solving periodic railway timetable scheduling problem. IEEE. Trans. Evol. Comput., 17: 512-527.