Management of Academic Workload Allocation Using Multi-Objective Genetic Algorithm

Manar Salamah Ali

Computer Science Department, King Abdulaziz University, Jeddah, Saudi Arabia

Abstract
Enforcing fairness policies for academic workload distribution and achieving staff satisfaction is of great importance in academic institutions. The amount of effort spent by instructors in teaching individual courses is not only measured by the contact teaching hours with students in classes. Teaching efforts include both in-class and out-of-class activities such as course preparation, teaching, marking exams, marking assignments, and supervising projects. In this paper, the fairness of workload allocation is treated as an optimization problem. We propose a two-dimensional and multi-objective implementation of the Genetic algorithm. The problem is solved using two optimization criteria:1) maximize the fairness workload allocation concerning the actual effort and time spent on the teaching and learning process, and 2) maximize a developed fair eligibility score for instructor and course assignments in workload schedules. The eligibility score is a combined metric which consists of additional factors that may affect the workload allocation decisions such instructor preferences, head of department recommendations, and the level of instructor's expertise in the course. The workload problem is represented using two-dimensional matrices. The experiments are conducted on a real dataset consisting of 32 courses and 10 instructors. The overall performance of the algorithm is measured based on the fitness value and running time of the program. The results on the real dataset show that the proposed algorithm solves the problem efficiently in 395 seconds runtime. The proposed algorithm achieves fair allocation of workload and fair eligibility score with 3.2 and 13 standard deviations respectively. The average eligibility score achieved is 61%.

Key words:
Genetic algorithm; fair academic workload allocation; teaching eligibility scores.

1. Introduction

In the academic domain, faculty members spend considerable effort to meet the educational environment's needs and challenges. Some academic institutions establish a set of well-defined policies and regulations to manage the process of workload distribution and allocation. While other universities initiate general rules and arbitrary constraints and leave the implementation to the department chairs to apply the rules in whichever method they find suitable.

Application of fair workload allocation in academic departments create a positive work-life climate and has a direct impact on the success of faculty [3]. In [15], the authors show how unfair workload is linked to lower job satisfaction. Also, inaccurate estimation of actual teaching effort (ATF) has consequences on research and self-development and productivity.

Academic workload calculation that is based solely on the contact hours with students through lectures or labs is not an accurate measure of the actual workload effort. For example, an instructor might spend up to 5 preparation hours for each one contact hour with students especially if the course is taught by the instructor for the first time, or the topic of the course is not within the field of his expertise. In addition, other factors such as the number of students in the course and the amount of assessed homework and exams in a course contribute to the actual effort spent in fulfilling the teaching and learning activities. Therefore, the Head of Department (HoD) must take into consideration the actual effort required for teaching individual courses and distribute the workload following a transparent, fair, and effective process. Accordingly, workload allocation policies must be designed together with a system to apply it.

Workload allocation is an optimization problem (NP-hard) where exploring all possible solutions with a reasonable amount of courses and instructors has an excessive-high calculation time [4, 6, 8].

Genetic algorithms (GA) are used to solve a broad range of optimization problems such as workload allocation problems [1, 14], sequencing problems [11, 17], machine learning models and image processing [20, 21], and network partition problems[13].

In this paper, we apply the GA to solve the problem of fair academic workload allocation. The main objective of this work is to develop a policy-based algorithm to enforce fair workload distribution. The result of the algorithm is measured according to a combined objective function, which balances both fairness in workload distribution and balances the eligibility score in each schedule. In this

Manuscript received September 5, 2020
Manuscript revised September 20, 2020

DOI: 10.22937/IJCSNS.2020.20.09.8

paper, we develop a combined eligibility measure that provides a metric for scoring (instructor, course) assignments in giving schedule. The eligibility metric includes: 1) the instructor's level of experience in the course, 2) the recommendation of the head of the department, and 3) the course preferences of the instructor. The workload problem is represented as two-dimensional matrices.

The paper is organized as follows: in section 2, the workload allocation problem is discussed. The previous work on academic workload allocation problem is discussed in section 3, and the mathematical model is presented in section 4. The Genetic Algorithm is addressed in section 5. Experimentation and results are shown in section 6, followed by a discussion on results in section 7. Finally, the conclusions are discussed in section 8.

2. Fair Workload Allocation Problem

Many theoretical models have been proposed to enhance transparency and fairness in the academic workload allocation problem [12] [7]. These models discuss specific parameters such as weighing individual teaching and research activities as well as focusing on the overall performance of faculty members. In addition to the primary role of teaching and supervising students, faculty members perform additional tasks such as research-related activities and administrative roles.

Different factors influence the decisions made regarding workload allocation such as: 1) the number of available teaching load, 2) the number of available instructors, 3) the maximum and minimum workloads for instructors, 4) the teaching experience of instructors and how they fit for teaching specific courses, 5) the preference of an instructor to teach particular courses, and 6) the recommendations of the HoD on the instructor-course assignments.

In [2], the author suggests that the teaching workload equation must consider all teaching and learning factors. For example, teaching the same course repeatedly, or teaching the same course for multiple sections are factors that have a direct effect on the time spent on preparation compared to teaching the course for the first time or the lack of experience in the course topic. The preparation time is minimal in the former cases. The provision of teaching assistants or sharing the course with other teachers are also factors that would affect the workload equation [2].

Different additional vital factors influence the decision making of workload allocation. First, the suitability of an instructor to teach a specific course. The instructor's experience and seniority have a direct impact on student performance [5]. Second, instructor preference is put into consideration when distributing the teaching load. Some instructors would prefer specific courses because they like teaching these courses or because they have good experience in the course, which reduces the preparation time required by them. Others may prefer courses offered at certain times of the day regardless of the content of the course. Third, the HoD is needed to provide instructors for all offered courses in a course timetable. Sometimes the HoD has to make decisions about who must teach the courses even if the taken decisions do not favor the instructors.

Therefore, thorough and comprehensive workload policies must be developed within institutions that take into consideration all the different factors that influence the fair workload allocation.

3. Previous Work on Academic Workload Allocation

In the literature, researchers have used different optimization methods and algorithms to optimize the academic workload allocation problem (also known as the teacher assignment problem). Solutions have been proposed for variant problem settings (different versions of the problem) and considering different objective functions,

The majority of recent works solved the problem using linear programming models. For example, in [16], a mixed-integer linear programming model was proposed to solve the problem of assigning the most suitable teacher assistants to the tutorials in the department. The objective of the model is to maximize the number of tutorials that are taught by the most suitable assistants. The model assumes that each teaching assistant has a set of defined capabilities for teaching tutorials. In [6], the authors propose a linear regression model for assigning faculty members to courses in Brazilian higher education where the academic performance is linked directly to the appropriate allocation of academic workload. The objective of the model maximizes the contribution of assigning instructors to courses. The authors in [4] proposed an integer linear model for solving the problem of assigning a set of tutors to a set of workshops with an objective function to maximize tutor satisfaction based on their teaching preferences. In [8], a Mixed Integer Linear Programing model (MILP) is developed to solve the teacher assignment problem. The model uses two

optimization criteria. The first criteria is balancing teachers' loads based on the contact hours with students. The second criteria is maximizing teachers' preference. The model solves assignments of up to 40 teachers in reduced calculation time.

Also, many researchers used GA to solve the problem. For example, in [18], the teacher placement problem is represented by one-dimensional array and solved using GA. The study compares different crossover and mutation operators. It concludes that with similar settings, the best results are obtained from ordered crossover and partial shuffle mutation with an average running time 40 minutes. The authors in [14] addressed the timetable problem in assigning teachers to courses. They have employed a hybrid genetic algorithm using a self-adaptive mechanism to guide four tuning operators to increase the optimality of the solutions.

Despite the wealth of research in the area of workload allocation, the problem is still an open optimization research problem for researchers to attempt different problem scenarios, optimization methods, and improved performances.

In the following sections, the problem addressed in this paper, the mathematical formulation of the problem, the developed are discussed in detail.

4. Mathematical Model

In this paper, a solution is proposed for assigning a finite set of instructors to a finite set of courses. Since our approach depends on the ATF, no assumptions are made regarding the maximum workload or the maximum number of courses assigned to an instructor. Also, we assume the set of courses is distinct and does not contain sections of the same course. The objective is to assign instructors to courses such that in given schedule:

1. Each course is assigned to only one instructor.
2. Each instructor must have at least one course in his workload.
3. A balanced workload weight is achieved. The variance between instructor workload weights is minimal in a final accepted workload distribution.
4. A balanced workload eligibility score is achieved. The variance between instructor schedules eligibility scores is minimal in a final accepted workload distribution

When calculating the ATF per semester, the following weights are considered: number of contact hours per semester, the preparation time spent corresponding to 1 contact hour with students (less weight is allocated to courses taught before), number of students in each course, number of assessed activities in each course, and the number of hours required to mark each activity (per student).

The problem is formulated such that a fair workload (ATF) per semester is achieved.

4.1 Mathematical Notation

The following notations are used throughout the paper:

I	Number of instructors available for teaching
C	Number of offered courses for teaching
cr_i	Number of credit hours of course i where $i = 1,2...C$
$prep_{1,k}$	Number of preparation hours corresponding to 1 lecture contact hour if the course is taught for the first time, where $k = 1,2...C$
$prep_{2,k}$	Number of preparation hours corresponding to 1 lecture contact hour if the course is taught before, where $k = 1,2...C$
$weight_{1,k}$	Weight of teaching course k if the course is taught for the first time, where $k = 1,2...C$
$weight_{2,k}$	Weight of teaching course k if the course is taught before, where $k = 1,2...C$
st_i	Number of students in course i where $i = 1,2...C$
nw	Constant denoting number of weeks in a semester
M	Maximum number of assessed activities in courses
$hr_{i,k}$	Number of hours required to mark an activity k in course i, where $i = 1,2...C$ and $k = 1,2,...m_i$
$ep1, ep2, ep$	Given eligibility criteria percentages, where $ep1 + ep2 + ep3 = 1$
$hd_{i,k}$	Recommendation of the HoD for instructor i to teach course k, where $i = 1,2...I$, and $k = 1,2...C$
$pref_{i,k}$	Preference of instructor i to teach course k, $i = 1,2...I$, and $k = 1,2...C$
$spec_{i,k}$	Scale (out of 100) of the experience of instructor i in course k, where $i = 1,2...I$, and $k = 1,2...C$
$e_{i,k}$	A calculated percentage denoting the eligibility of instructor i to teach course k, where $i = 1,2...I$, and $k = 1,2...C$.
es_i	A calculated percentage denoting the eligibility score of instructor i in a given schedule instance, where $i = 1,2...C$
$isTaught_{i,k}$	Binary matrix to indicate if instructor i has taught course k before or not where $i = 1,2...I$, and $k = 1,2...C$.
$assign_{i,k}$	Binary matrix that represents a schedule, where 1 indicates that instructor i has been assigned a course k or 0 otherwise, and $i = 1,2...I$, and $k = 1,2...C$.
wl_i	Calculated weight of actual teaching hours for

instructor *i*, where $i = 1,2..I$.

The examples throughout the rest of the paper are produced from an arbitrary test data of 9 courses and 4 instructors. The courses are labeled C1, C2 ..,C9, and the instructors are labled I1, ..,I4.

To use the GA for fair workload allocation, several fairness metrics are developed. In the following sections, the metrics are discussed in detail.

4.2 Course Profile

First, a course profile is constructed for each course available for teaching. The profile consists of the weight of the course in terms of the actual effort needed to teach the course. The weight is calculated for two different scenarios: 1) the course is taught by an instructor for the first time and requires significant preparation, and 2) the course has been taught before by an instructor and requires moderate preparation time. Additionally, the time required (per student) to mark any course-related activities (exams, projects, and assignments) is used in calculating the total weight of the course as follows:

For first time teaching,

$$weight_{1,k} = nw * (cr_k + cr_k * prep_{1,k}) + \sum_{t=1}^{M} hr_{k,t} * st_k \quad (1)$$

For courses taught before:

$$weight_{2,k} = nw * (cr_k + cr_k * prep_{2,k}) + \sum_{t=1}^{M} hr_{k,t} * st_k \quad (2)$$

Where *k=1,2..C*, and *M* is the maximum number of activities in courses.

The weight is calculated considering the actual contact hours cr_k, the preparation time per 1 contact hour $(cr_k * prep_{1,k})$ or $(cr_k * prep_{2,k})$, and the number of hours required to assess course activities ($\sum_{t=1}^{M} hr_{k,t} * st_k$). The weight is calculated for nw weeks, which is set to the default of 15 weeks per semester.

In Figure 1, we illustrate how the matrix representing the weights of the 9 courses is calculated. First, the following arbitrary data is provided as an input: the preparation time of courses (considering the two scenarios as in Figure 1.(a)), the number of students in each course (Figure 1.(b)), credit hours of each course (Figure 1.(c)), and assessment hours per each course activity (Figure 1.(d)). Then the weight matrix is produced according to Eq. 1, and 2 as illustrated in Figure 1.(e).

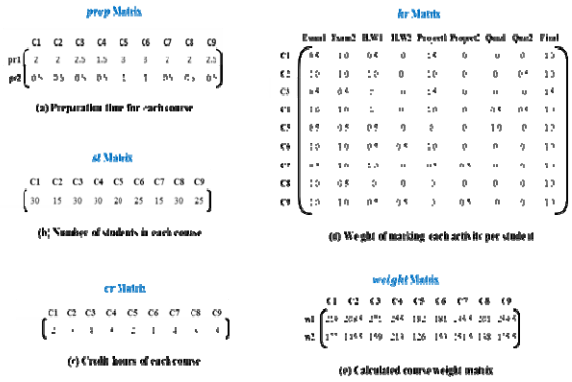

Fig. 1 Calculated course weight matrix

4.3 Eligibility Score

To determine the eligibility of each instructor for teaching a specific course, an eligibility metric is developed which depends on the following eligibility criteria:

a) The experience of the instructor in the course (teaching fitness)
b) The head of department's recommendation for an instructor to teach a course
c) The instructor's preference to teach the course

Each criteria is assigned a scale (out of 100). Accordingly, an eligibility matrix of (instructor, course) pair is produced for each instructor *i*, and each offered course *k*. The eligibility scores are calculated as follows:

$$e_{i,k} = (ep1 * spec_{i,k}) + (ep2 * hd_{i,k} * 100) + (ep3 * pref_{i,k} * 100) \quad (3)$$

Where $ep1 + ep2 + ep3 = 1$

Continuing the steps of the test example, we assume arbitrary values for *ep1*,*ep2*, and *ep3* as 0.35,0.30, and 0.35, respectively. One may argue about the criteria types or the given scales, so it is important to note that these criteria and scales can change according to the standard policies of departments. The values in $spec$, hd, and $pref$ matrices are arbitrarily assumed, as shown in Figure 2(a)-(c). Each $spec_{i,k}$ holds a percentage value (proximity scale) representing the fitness of instructor *i* to teach course *k*. Each $hd_{i,k}$ holds a value of 1 if the HoD recommends that instructor *i* teaches course *k*; otherwise,

the value is 0. And each $pref_{i,k}$ holds the value 1 if instructor *i* prefers to teach course *k* or otherwise the value is 0. The course preferences in our model have no specific order of priority. Accordingly, the eligibility matrix is calculated using Eq. 3, as shown in Figure 2.(d).

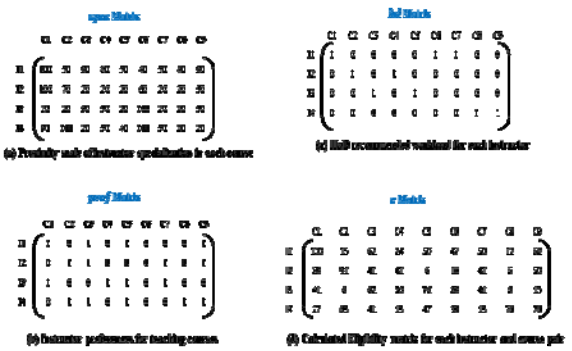

Fig. 2 Calculating eligibility matrix

An instance of schedule is represented by a two-dimensional binary matrix ($assign$). If instructor *i* is assigned course *k*, then $assign_{i,k} = 1$, otherwise the value is 0.

The eligibility score for each instructor in a given workload allocation is calculated according to the following equation:

$$es_i = \frac{\sum_{k=1}^{C} assign_{i,k} * e_{i,k} * cr_k}{\sum_{k=1}^{C} assign_{i,k} * cr_k} \quad (4)$$

Applying Eq. 4 to an arbitrary schedule $assign$, and e and cr from the test example will result in the estimation scores illustrated in Figure 3. It reads as I_1 has achieved a 64% eligibility score in his workload within the schedule.

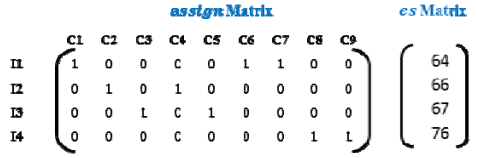

Fig. 3 Instructor eligibility estimation scores in a given schedule.

4.4 Workload Calculation

The weight of the workload of each instructor is calculated based on the courses assigned to the instructor in a given schedule instance. To calculate the actual time required to teach a course, teaching history data is required for each instructor. The teaching history is represented by the binary matrix *isTaught*. If the instructor *i* has taught the course *k* before, then $isTaught_{i,k}$=1 , otherwise the value is 0. Accordingly, the actual workload wl_i is calculated for each instructor *i* in a given schedule instance according to the following process:

Process 1: Calculate the workload for each instructor n terms of the ATF

Input: a two-dimensional schedule *assign*, a two-dimensional matrix *isTaught*.

Output: calculated workload for each instructor in the schedule

Step1: Set $i = 0,$ (for every instructor in the schedule)

Step 2: Set $i = i + 1, wl_i = 0$

Step 2: Set $k = 0,$ (for every course in the schedule)

Step 3:

$$if\ assign_{i,k} = 1\ then\ if\ isTaught = 1\ then\ wl_i = wl_i + weight_{i,k}$$

$$else\ wl_i = wl_i + weight_{i,k}$$

step 4: k=k+1

Step 4: if $k = C$ go to step 5

Step 5: if $i = I$ then stop, otherwise go to Step2.

In Figure 4, we calculate the actual workload for each instructor in the schedule instance in (Figure 3.(b)) given an arbitrary $isTaught$ matrix (Figure 3.(a)).

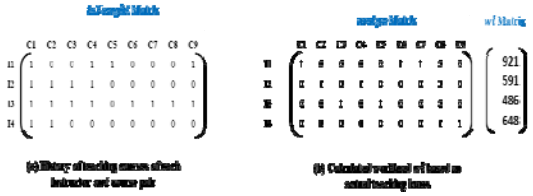

Fig. 4 Calculation of actual teaching hours for each instructor

As seen in Figure 4, the calculated workload according to process 1 results in the following workloads of 921, 591, 486, and 648 hours per semester for instructors 1 to 4, respectively, with a standard deviation of 161 hours. The results show an unfair distribution of actual teaching hours in workloads. To compare with workload calculations that are based only on teaching contact hours using the formula $\sum_{k=1}^{C} assign_{i,k} * cr_k$ and the $assign$ matrix in Figure 4.(b), we get the following workloads of 98, 84, 84, and 84 contact hours per semester for instructors 1 to 4, respectively with a standard deviation of 6 hours which is a misleading fair workload distribution.

4.5 Constraints

The following constraints are applied to every new generation of a schedule instance:

Const.1: In a given schedule, each course must be allocated to only one instructor, or

$$\forall k, \sum_{i=1}^{I} assig\, n_{i,k} = 1, \quad where\ k = 1,2 \ldots C. \tag{5}$$

Const.2: Each instructor must have at least one course assigned to him, or

$$\forall i, \sum_{k=1}^{C} assig\, n_{i,k} > 0, where\ i = 1,2..I. \tag{6}$$

Const. 3: The number of courses assigned to an instructor must are bounded by upper and lower bounds UB and LB, such that:

$$\forall i, LB \leq \sum_{k=1}^{C} assig\, n_{i,k} \leq UB. \tag{7}$$

The above constraints are considered hard constraints because they determine if a schedule instance is feasible and can be considered as a candidate solution or excluded otherwise.

5. Genetic Algorithm

The GA is inspired by the genetic structure of chromosomes and the three revolutionary genetic operators which happen during breeding of plants and animals. The operators are as follows: (1) the selection operator (selection of the best chromosomes for evolution process), (2) the crossover operation (exchange of sections of parents' chromosomes), and (3) the mutation operator (random modification of a chromosome) [17]. Hence the algorithm mimics evolutionary biology techniques. There exist many extended variations of the genetic algorithm. However, in this paper, we use the canonical GA by Holland in [10] with modifications. The steps of the algorithm are represented in figure 5.

The algorithm is initiated by a random population of chromosomes (candidate solutions). The population is evaluated against a fitness function (objective function) then generates a subsequent population (generation) such that chromosomes with a better fitness value are given a chance to reproduce than the poorer chromosomes. The next generation is reproduced using selection, crossover, and mutation operators in order.

For a binary population where variables can be represented by 0s and 1s, selection means retaining the best performing bit (gene) of strings (chromosomes) from one generation to the next by favoring them for reproduction.

In our model, the chromosome (schedule instance) is represented by a two dimensional binary matrix as illustrated in Figure 4.(b), where columns represent courses and rows represent instructors. The value of a gene in the matrix is 1 if the instructor is assigned to the course or 0 otherwise.

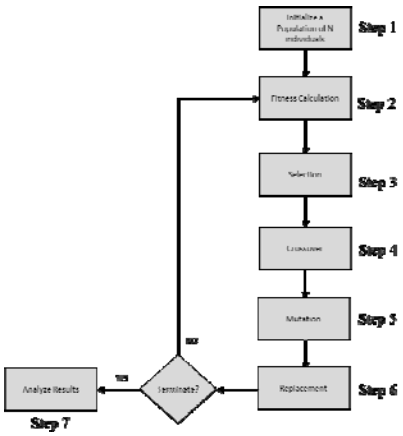

Fig. 5 Genetic Algorithm Flowchart

Step 1: Population Initialization

The initial population is randomly generated. N random and feasible schedule instances are produced. Only schedules satisfying Cosnt. 1-3 are considered potential candidates in the population.

A dataset is initialized by inputting the variables *I,C, nw, ep1,ep2,ep3, M*, *UB*, *LB*, the one-dimensional matrix *cr* of length C, the matrix *prep* of size (2*C), and the matrices of size (I*C): *spec*, *hd*, *pref*, *isTaught*. Accordingly, the workload wl_i is calculated for each instructor in every schedule in the population according to the process 1.

Step 2: Fitness Function

First, the selection for reproduction is performed to create a mating pool that contains the best-fitted schedules from the previous generation. A fitness function f_j is calculated for each schedule in the population where *j* represents the j^{th} schedule in a population of *N* schedules. The fitness function provides a measure of performance with respect to a particular set of parameters

The fitness function for solving the problem of fair workload allocation is a multi-objective function. The objective of the optimization is two-fold: (1) the workload

must be balanced such that the workload is fairly allocated with respect to the ATF required for teaching the allocated courses per individual instructor, and (2) The eligibility score must be balanced in a given schedule.

We denote f_1 and f_2 as the fitness functions for fair workload and eligibility, respectively. The fitness functions are variance functions defined as follows:

$$f_1 = \frac{\sum_{i=1}^{I}(wl_i - \overline{wl})^2}{I-1} \quad (8)$$

Where $\overline{wl}$ is the mean of instructors' workloads wl_i in a given schedule.

$$f_2 = \frac{\sum_{i=1}^{I}(es_i - \overline{es})^2}{I-1} \quad (9)$$

Where $\overline{es}$ is the mean of instructors' eligibility scores es_i in a given schedule.

Then the objective is to minimize the combined fitness function, which is defined as:

$$f = \omega_1 f_1 + \omega_2 f_2 \quad (10)$$

Where $\omega_1 + \omega_2 = 1$, ω_1 and ω_2 represent weighting parameters of workload and eligibility, respectively, which has to be decided by a decision-maker.

Step 3: Selection

The fitness probability is calculated for each schedule in the population according as follows:

Assuming that f_i is the fitness function value for the j^{th} schedule, then the fitness probability of the j^{th} schedule is calculated as:

$$P_j = \frac{f_j}{\sum_{i=1}^{N} f_i} \quad (11)$$

To select the best fitted strings in the sampling pool, selection methods are used. One of the widely used and straight forward selection methods is the Roulette Wheel method [9]. The method maps the fitness probability of each string in the population to contiguous sectors of a wheel where sector sizes are proportional to the fitness probabilities P_i of strings in the population. A random number r is selected such that $0 < r < \sum_{i=1}^{N} P_i$ to select the corresponding sector of a string in the population. Hence, strings with higher fitness function have a greater possibility to be selected for crossover operation. In addition, it is possible to retain copies of the best-fitted strings.

Step 4: Crossover

In a crossover, randomly selected parents from the sampling pool are combined to form a pool for mating in the hope of producing offspring with higher fitness values. The crossover is done by swapping string segments of the selected parents at a random crossover point X. If we assume that parents of schedules P1 and P2 are selected at random for mating, then crossover is performed as illustrated in Figure 6.

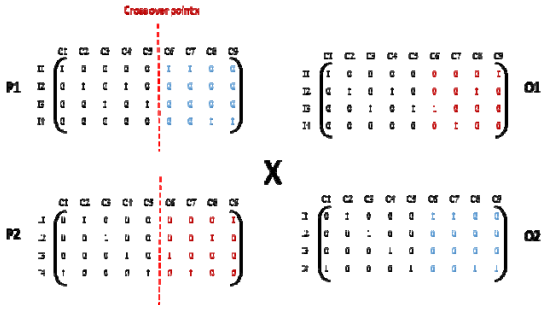

Fig. 6 Crossover operation at random point 6

Crossover is not applied to all couples. A random crossover probability P_c is determined to decide the proportion of couples to go through crossover operation. Studies showed that the best value of P_c is to keep it in the range 0.65 to 0.85, and the default is 0.75 [19]. Each couple reproduces two children. P_c is global GA parameter. This crossover strategy is guaranteed to hold Const. 1. However, Const. 2 and 3 could be violated. Hence, an offspring is added to the pool only if the offspring is a feasible schedule.

Step 5: Mutation

The mutation is applied to offspring individually to prevent the search algorithm from trapping into a local optimum and to explore the search space by introducing new genetic structures. The bits are mutated at randomly chosen positions of randomly selected schedules by flipping the bit from 1 to 0 or from 0 to 1. Mutation frequency is controlled by a global GA parameter P_m which is a small value between 0 and 1, with a default value 0.1. The mutated gene (instructor, course) is selected according to the following process: two random integers R_1 and R_2 are generated where $1 \leq R_1 \leq I$ and $1 \leq R_2 \leq C$. The value gene (R_1,R_2) is flipped.

The mutated schedules are accepted if and only if the result of the mutation is a feasible solution (satisfying constraints 1-3).

Step 6: Replacement Strategy

Replacement is the last breeding step in the algorithm where the chromosomes in the current pool are either replaced or survives to the next generation. There exist various replacement strategies such as generational replacement, steady-state replacement, and elitism. We use the steady-state replacement strategy. The best-fitted N schedules from the pool are kept in the population, and the remaining schedules are excluded.

Step 7: Terminate the algorithm

The GA terminates in case one of the following termination conditions are true: an optimal solution is found, or the maximum number of iterations is reached. Since the fitness function is a balancing function, there is no one optimal solution. Instead, a set of near-optimal solutions are targeted.

6. Experimentation and Results

The experiments were carried out a real dataset consisting of 10 instructors and 32 courses from a Computer Science department. The dataset set is initialized as discussed in section V. Since a multi-objective optimization is applied, there does not exist one single optimized solution to the problem. In addition, a 100% balance in workloads or eligibility scores is not feasible. Hence we follow the Pareto analysis approach, by selecting the best solutions from a set of optimal solutions.

The experiments are planned as follows: run batches with different values of P_c, ω_1, and ω_2. In the first batch, we set N=200 and apply one crossover point. To expand the search space, in the second batch, we set N=300 and use two crossover points. In each run, we select the best schedules in the population based on optimized fitness values. The results showing in Table 1 represent average values for the best 15 schedules in each run. We set the maximum number of iterations to 10^4, *UB*=4, and *LB*=2.

The experiment was run on a PC with intel(R) Core(TM)i7-6500U and 8GB RAM. The algorithm was programmed in java. The results of the experiments are illustrated in Table 1. The data in Table 2 demonstrates a candidate solution to the problem. In the next section, the results are discussed.

7. Discussion

In the model, we include combined metrics for criteria that might affect the decisions in the workload assignment other than the ATF. The values assigned to ω_1, ω_2, ep_1, ep_2, and ep_3 has to be decided by the decision-maker and used for tradeoffs depending on the policies of the department. For example, if only instructor preference is giving priority in the targeted schedules, then ep_3 is set to 1. For population sizes less 150, the algorithm did not produce significant value. We set the population size to 200, and tuned *Pc*, ω_1 and ω_2 to different values while using one crossover operator for generating new schedules. We observed the best results are achieved with *Pc* =0.75 and ω_1 = 0.8. To increase the search space, we increased N to 300, increased P_c to 0.85, and used two crossover points. There was a significant improvement in the results with minimized deviations for ATF and eligibility scores.

We combine the real dataset values and a generated optimized schedule in a single table (Table 3) to visualize how the algorithm generated the solution. The algorithm has achieved a fair workload with average of 455, deviation of 3.2, and 445 seconds of runtime. Also, a fair eligibility score is achieved with average 62 and a deviation of 13. The error rates however were 50% for both instructor and HoD preferences. It is an acceptable result especially that we are not considering error rates in our implementation of the problem. Finally, the results are compared with the recommendation made by the HoD (bold frames in Table 3). The calculations of the schedule recommended by the HoD have a workload deviation of 123, were the workloads ranged between780 and 443. The deviation of the eligibility scores is 11 with average 70%.

8. Conclusion

It is necessary for academic institutions to apply optimal ways to produce fair and balanced workload allocation for their academic instructors. The teaching effort is not only calculated by the contact hours with students. Teaching effort includes substantial time spent on preparations and the assessment of student work. Hence, it is crucial that teaching efforts are calculated and balanced based on detailed metrics for taught courses. We propose a genetic algorithm solution for fair workload allocation based on comprehensive metrics and actual teaching effort. The algorithm succeeded in achieving a fair allocation reasonable eligibility score. Further enhancement can be applied to by considering several other workload metrics, such as weights of research and

administrative, and study how this could be introduced to solve the problem.

Table 1: Experiment results on a real dataset with I=10 and C=32.

N	Cross Over	Parameters				Pareto optimal solutions				Time in seconds
		P_c	P_m	ω_1	ω_2	Workload deviation	Workload average	Eligibility deviation	Eligibility average	
200	1 crossover point	0.5	0.1	0.5	0.5	56	624	12	28	213.4
		0.5	0.1	0.8	0.2	40	564	8	39	210.4
		0.75	0.1	0.5	0.5	32	567	8	33	225.8
		0.75	0.1	0.8	0.2	23	564	8	46	220.5
300	2 crossover points	0.75	0.1	0.5	0.5	18	568	14	55	330.7
		0.75	0.1	0.8	0.2	17	532	16	48	328.9
		0.85	0.1	0.5	0.5	10	508	8	56	393.1
		0.85	0.1	0.8	0.2	5	460	9	61	394.6

Table 2: Solution instance from the experiments

Course Name	C#	Credit	# of Stud.	$prep_{1,k}$	$prep_{2,k}$	hr_i	I1	I2	I3	I4	I5	I6	I7	I8	I9	I10
programming 1	C1	4	15	187.5	97.5	67.5	80	80	100	80	80	80	80	100	80	100
programming 2	C2	4	30	285	195	165	80	80	100	60	80	80	80	100	80	100
programming 3	C3	4	25	212.5	92.5	62.5	80	60	100	60	80	80	80	100	80	100
Network 1	C4	3	30	330	195	150	20	20	20	20	20	100	100	60	20	20
Network 2	C5	3	15	187.5	97.5	52.5	20	20	20	20	20	100	100	60	20	20
Software Engineering 1	C6	3	30	285	195	150	20	100	80	20	100	80	20	50	50	100
Software Engineering 2	C7	3	15	202.5	112.5	67.5	20	100	50	20	100	20	20	20	20	100
Database 1	C8	3	30	210	97.5	75	100	90	50	80	100	50	50	100	50	100
Database 2	C9	3	15	202.5	90	67.5	20	80	50	20	100	50	20	80	20	100
Operating Systems	C10	3	30	270	180	135	80	70	60	50	50	50	50	50	50	50
Digital Logic	C11	3	30	285	150	105	100	50	50	50	80	100	50	50	50	50
Discrete Structures 1	C12	3	30	255	120	75	100	50	50	50	90	50	80	80	50	50
Discrete Structures 2	C13	2	30	225	165	135	80	50	80	20	60	20	60	100	20	20
Computer Architecture	C14	3	30	345	210	165	100	50	50	50	100	50	50	50	50	50
Data Structure 1	C15	3	30	255	165	75	80	80	100	100	100	50	50	50	50	50
Data Structure 2	C16	3	15	255	120	75	80	50	100	80	50	50	50	50	50	50
Programming Languages	C17	3	30	240	127.5	105	80	50	50	80	90	50	80	50	50	50
Artificial Intelligence 1	C18	3	25	305	170	125	100	20	50	100	20	80	20	20	100	20
Algorithm Design	C19	3	30	330	240	195	50	50	100	50	50	50	80	50	50	50
Artificial Intelligence 2	C20	3	20	250	160	70	100	20	20	100	20	50	20	20	100	20
HCI1	C21	2	15	127.5	97.5	67.5	20	100	20	20	50	20	20	80	20	80
HCI2	C22	3	30	225	157.5	135	20	80	20	20	50	20	20	80	20	50
Graphics 1	C23	3	15	187.5	97.5	52.5	20	20	20	20	20	20	20	20	100	80
Graphics 2	C24	3	30	210	165	75	20	20	20	20	20	20	20	20	100	80
Multimedia	C25	3	30	255	165	120	20	20	20	20	20	20	20	80	80	100
Data Analytics	C26	3	15	217.5	127.5	82.5	20	20	20	100	20	20	20	20	100	20
HPC	C27	3	15	217.5	82.5	37.5	20	80	20	50	20	20	20	80	20	20
Computer Forensics	C28	2	15	165	105	75	20	20	20	60	20	20	20	100	50	20
Web Development	C29	3	15	187.5	97.5	52.5	20	80	20	20	50	20	20	80	20	100
Compiler Construction	C30	3	20	255	165	120	50	50	50	50	50	80	50	80	50	50
Applied Math 1	C31	4	30	390	210	150	50	50	50	50	50	80	100	50	50	50
Applied Math for 2	C32	4	15	277.5	97.5	37.5	50	50	50	50	50	50	100	80	50	50
Total Workload							458	458	458	455	450	458	450	458	453	453
Eligibility Score							51	48	57	86	64	75	62	41	75	62

Workload Deviation	**3.207802986**
Eligibility Score Deviation	**13.0418557**
Workload Average	**455.1**
Eligibility Average	**62.1**
Ints. Preference Error Rate	**0.5**
HoD Preference Error Rate	**0.47**

Grey highlight means that Instructor i has been assigned course k in the schedule

X The numbers in in each (course, instructor) assignment represents the experience of the instructor in the course

Bold frame means the HoD recommends that instructor i teaches course k

X Blue means that the instructor i prefers to teach course k

X Underlined means the instructor has taught the course before

X Red means no instructors preferred to teach the course

References

[1] M. Abbasi, E. M. Pasand, and M. R. Khosravi, "Workload Allocation in IoT-Fog-Cloud Architecture Using a Multi-Objective Genetic Algorithm," *Journal of Grid Computing,* pp. 1-14, 2020.

[2] E. Bitzer, "Attempting a fair and equitable academic workload distribution in a faculty of education," *South African Journal of Higher Education,* vol. 21, no. 1, pp. 23-37, 2007.

[3] L. Boyd, "Exploring the utility of workload models in academe: a pilot study," *Journal of Higher Education Policy and Management,* vol. 36, no. 3, pp. 315-326, 2014.

[4] G. Caselli, M. Delorme, and M. Iori, "Integer Linear Programming for the Tutor Allocation Problem: A Practical Case in a British University," *arXiv preprint arXiv:2005.09442,* 2020.

[5] J. Coenen, I. Cornelisz, W. Groot, H. Maassen van den Brink, and C. Van Klaveren, "Teacher characteristics and their effects on student test scores: A systematic review," *Journal of economic surveys,* vol. 32, no. 3, pp. 848-877, 2018.

[6] J. J. da Cunha Jr and M. C. de Souza, "A linearized model for academic staff assignment in a Brazilian university focusing on performance gain in quality indicators," *International Journal of Production Economics,* vol. 197, pp. 43-51, 2018.

[7] S. Dekeyser, R. Watson, and E. Baré, "Comparing academic workload models: How Australian universities resource teaching activities," in *Tertiary Education Management*

Conference: From Rhetoric to Reality. Retrieved from https://www. atem. org. au/documents/item/576, 2016.
[8] B. Domenech and A. Lusa, "A MILP model for the teacher assignment problem considering teachers' preferences," *European journal of operational research,* vol. 249, no. 3, pp. 1153-1160, 2016.
[9] D. E. Goldenberg, "Genetic algorithms in search, optimization and machine learning," ed: Addison Wesley, Reading: MA, 1989.
10] J. H. Holland, *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.
[11] A. A. R. Hosseinabadi, J. Vahidi, B. Saemi, A. K. Sangaiah, and M. Elhoseny, "Extended genetic algorithm for solving open-shop scheduling problem," *Soft computing,* vol. 23, no. 13, pp. 5099-5116, 2019.
[15]K. O'Meara, C. J. Lennartz, A. Kuvaeva, A. Jaeger, and J. Misra, "Department conditions and practices associated with faculty workload satisfaction and perceptions of equity," *The Journal of Higher Education,* vol. 90, no. 5, pp. 744-772, 2019.
[16] X. Qu, W. Yi, T. Wang, S. Wang, L. Xiao, and Z. Liu, "Mixed-integer linear programming models for teaching assistant assignment and extensions," *Scientific Programming*, 2017.
[17] C. R. Reeves, "A genetic algorithm for flowshop sequencing," *Computers & operations research,* vol. 22, no. 1, pp. 5-13, 1995.
[18] P. Rosa, H. Sriwindono, R. Nugroho, A. Polina, and K. Pinaryanto, "Comparison of Crossover and Mutation Operators to Solve Teachers Placement Problem by Using Genetic Algorithm," in *Journal of Physics: Conference Series*, vol. 1566, p. 1-8, 2020.
[19] E. G. Shopova and N. G. Vaklieva-Bancheva, "BASIC—A genetic algorithm for engineering problems solution," *Computers & chemical engineering,* vol. 30, no. 8, pp. 1293-1309, 2006.
[20] Y. Sun, B. Xue, M. Zhang, G. G. Yen, and J. Lv, "Automatically Designing CNN Architectures Using the Genetic Algorithm for Image Classification," *IEEE Transactions on Cybernetics,* 2020.
[21] H. Zhi and S. Liu, "Face recognition based on genetic algorithm," *Journal of Visual Communication and Image Representation,* vol. 58, pp. 495-502, 2019.

Manar Salamah Ali received the M.Sc. degree in advanced computing from the Imperial College London, and the Ph.D. degree in web service-based transaction management from the University of Leicester. She has assumed a variety of roles during her professional career, including the Chairwoman of the Computer Science Department, various consultancy roles, and the Vice Director of the Knowledge-Based E-Portal Center, King Abdulaziz University, where she is currently an Assistant Professor in the Department of Computer Science. Her current research interests include the use of machine learning and computer vision in image and video processing as well as automated assessment of human and machine translations.