International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys. 14, No.1 (Jun-24)

https://dx.doi.org/10.12785/ijcds/1501124

Efficient Heuristics for Timetabling Scheduling in Blended Learning

Mohamed Jarraya[1]

[1]*Information Technology Department, College of Computation and Informatics, Saudi Electronic University (SEU), Saudi Arabia, Innov'COM, Sup'com , Tunisia*

Received 29 Nov. 2023, Revised 20 Mar. 2024, Accepted 6 Apr. 2024, Published 1 Jun. 2024

Abstract: Blended learning, an innovative educational approach adopted by the Saudi Electronic University, combines traditional face-to-face teaching with online methods to improve educational outcomes. However, implementing it effectively faces challenges, especially in shifting physical classes to virtual formats for faculty and students spread across distant campuses, potentially compromising the core principles of blended learning and reducing its effectiveness. Ideally, each student would attend one face-to-face and one virtual session per course weekly. Yet, operational challenges emerge due to geographical disparities between faculty and student locations, necessitating the conversion of face-to-face sessions into virtual ones. To address these challenges, this study proposes two novel heuristic strategies for scheduling timetables in blended learning contexts. The first heuristic, called Minimum Load Accumulation Heuristic (MLAH), aims to evenly distribute teaching loads among faculty members and time slots while maximizing the number of groups assigned to faculty members from the same campus. The second heuristic, called Average Load Accumulation Heuristic (ALAH), calculates the average load of all faculty members and time slots and reduces the number of iterations searching for the minimum load, as performed by MLAH. These strategies aim to minimize the conversion of face-to-face sessions to virtual, ensure fair distribution of teaching responsibilities, and maintain a balanced allocation of face-to-face and virtual classes throughout the academic year. The paper demonstrates the effectiveness of these algorithms in producing high-quality solutions comparable to those generated by CPLEX, with significantly reduced computational complexity.

Keywords: Timetabling, Scheduling, Optimization , Blended learning.

1. Introduction

University course timetabling is a complex problem that has been the focus of research for several decades. As the size of universities and the number of courses and students continue to grow, the manual process of creating course timetables becomes more difficult and time-consuming. To address this issue, software solutions have been developed to automate the process of creating university timetables [1], [2]. The solutions presented are tailored for learning processes that do not account for the geographical proximity of faculty members to their assigned teaching campuses. The primary contribution of this paper lies in addressing a significant gap in the existing literature concerning timetabling scheduling. While numerous studies have explored various aspects of scheduling in educational settings, none have specifically tackled the challenge of optimizing schedules while considering the distribution of faculty members and student groups among multiple campuses. By introducing novel heuristics and optimization techniques tailored to this specific scenario, my paper offers a pioneering solution that aims to maximize the effectiveness of blended learning programs.

Blended learning, also known as hybrid learning, is a pedagogical approach that combines face-to-face classes with virtual classes to provide an effective learning experience [3], [4], [5], [6]. This innovative approach has gained popularity in recent years, with universities around the world adopting blended learning models that utilize technology, support self-learning, and promote collaboration among learners [3], [4], [7]. By integrating both face-to-face and virtual classes, blended learning has demonstrated its ability to alleviate the isolation commonly encountered by learners in distance learning settings and enhance overall learning outcomes. [8].

The Saudi Electronic University (SEU) is one such university that has embraced blended education and has adopted a flexible model that incorporates both face-to-face and e-learning activities. SEU has a geographically dispersed campus system spread across different cities in the Kingdom.

Balancing face-to-face and virtual classes poses a significant challenge in the implementation of blended learning

E-mail address:m.jarraya@seu.edu.sa

methodologies. At Saudi Electronic University, adherence to this approach entails each student's attendance of one face-to-face session and one virtual session weekly for every enrolled course. However, logistical constraints arise when instructors and students are geographically separated, necessitating the conversion of face-to-face sessions into virtual ones. This adjustment represents a deviation from established blended learning principles, inhibiting direct instructor-student interaction and potentially undermining the pedagogical efficacy of the blended learning paradigm. Converting a face-to-face class into a virtual session due to the instructor and the student group being from different campuses is considered a violation of blended learning principles.

This paper proposes two new heuristics for scheduling timetables in blended learning education, which will be compared in terms of performance and quality with CPLEX. The proposed solution is designed to accomplish three key objectives: optimizing the allocation of instructors to student groups within the same campus to mitigate the need for converting face-to-face sessions to virtual ones; ensuring an equitable distribution of teaching responsibilities among faculty members; and evenly distributing the number of classes across teaching time slots, while adhering to various constraints. By achieving an equitable distribution of classes across time slots, the likelihood of face-to-face sessions being transitioned to online formats due to classroom shortages is also minimized.

The paper is structured as follows: Section 2 discusses related works, while Section 3 provides the problem formulation. Section 4 presents the proposed heuristics, and Section 5 analyzes the computational results. Finally, Section 6 concludes the paper with discussions on future work.

2. Related works

University timetabling software has undergone significant evolution over the years by incorporating advanced optimization algorithms to schedule courses and exams. These algorithms consider various constraints, such as course conflicts, faculty and room availability, and student preferences, resulting in more efficient and effective scheduling solutions [1], [2]. Scheduling problems are common in various fields such as cloud computing [9], [10], manufacturing [11], [12], [13], [14], [15], computer science [16], and crew scheduling [17], [18]. The University Course Timetabling Problem (UCTTP) is a specific scheduling problem that involves assigning courses, faculty members, and students to designated time slots and rooms while satisfying both hard and soft constraints [19], [20], [21], [22].

Approaches to solve the timetabling problem fall into three different classes introduced in the literature. The first class is based on operational research methods, including techniques like graph coloring theory, Integer programming/Linear programming (IP/LP) method, and constraint-based technique (CSPs) [20], [23]. The graph coloring method represents the timetabling problem as a graph coloring problem. The goal is to color the graph vertices with the least number of colors, such that no neighboring vertices have the same color. The resulting timetable must not have any conflicts. [24], [25] presented an approach of coloring the edges of a two-part graph to solve the UCTTP problem, while [26] used graph coloring to schedule classes by assigning courses to a given number of time slots (colors). On the other hand, [27], [28], [29] proposed a genetic coloring approach to solve the UCTTP problem using a combination of graph coloring and genetic algorithms.

The IP/LP method is a mathematical approach used to solve UCTTP by allocating limited resources to maximize interest and minimize costs. IP allows for some or all variables to be defined as non-negative integer values. The method is usually used distinctly, but it can also incorporate constructive heuristics to facilitate the analysis of constraints. The general structure of the IP/LP method for the UCTTP problem involves formulating different groupings into two classes: courses grouping and time intervals grouping. The IP/LP method is influenced by the size of the institute and obtaining optimal solutions can be difficult. [30] proposed a heuristic approach to solve this problem by separating the timetabling problem into two sub-problems: timetabling sub-problem and grouping sub-problem. [31], [32] presented the formulation of the UCTTP problem using integer programming, along with a case study. [33] applied an integer programming approach called IP-0.1 to solve the UCTTP problem, with the main goal being to organize courses and lecturers based on available time slots and classrooms. [34] presented a mixed-integer programming model for solving the university timetabling problem which reduces the number of variables and constraints of the mixed-integer program to manageable levels.

The CSP-based method is a computing system that defines constraints as limitations over a space of facilities. It aims to find a set of consistent values that satisfy the predefined constraints by assigning values to each variable. The problem is represented by a triple of CSP = (X, D, C) where X is a finite set of variables, D is a finite set of domain values, and C is a finite set of constraints that depend on a subset of variables [20].

Several researchers have used the CSP-based method to solve the Course Timetabling Problem. For instance, [35] combined a genetic algorithm with constraint-based reasoning to find a possible and near-optimum solution. [36] presented a case study of university timetabling using an approach based on constraint satisfaction programming. They used ILOC software to implement the CSP approach and allocate faculty members, students, classes, and courses to time slots in the weekly timetabling, while satisfying the constraints [20].

Meta heuristic methods are a set of algorithms used to find approximate solutions to difficult optimization problems. These methods are particularly useful when traditional

optimization techniques become impractical due to the large number of variables or constraints in the problem [37].

My approach is different from those presented in the literature by adding new constraints related to the nature of the blended learning by considering the distribution of faculty members and student groups among multiple campuses.

The next section presents the problem formulation and outlines the optimization approach applied.

3. Problem formulation

This paper presents a scheduling problem involving the allocation of faculty members to groups of students belonging to specific campuses and specific time slots. Each course, represented by $k \in \{1, 2, \ldots, K\}$, comprises L_k groups of students. The set of courses is denoted by $C = \{C_1, C_2, \ldots, C_K\}$, and $\mathcal{G}_{\|} = \{G_{k1}, G_{k2}, \ldots, G_{kL_k}\}$ is a set consisting of L_k groups that belong to course k.

A set of M faculty members is referred to as $\mathcal{F} = \{1, 2, \ldots, M\}$, and the reduction in teaching load of a faculty member m in case of administrative responsibilities or a higher rank than other faculty members is represented as R_m. A set of N campuses is denoted by $\mathcal{B} = \{1, 2, \ldots, N\}$. For faculty member m, the assignment vector to the N campuses is denoted by $\mathcal{U}m = \{u_{m1}, u_{m2}, \ldots, u_{mN}\}$, where u_{mn} is defined as 1 if faculty member m is assigned to campus n, and 0 otherwise.

The assignment vector of the l-th group in course k to the N campuses $\mathcal{B}$ is denoted by $\mathcal{V}_{kl} = \{v_{kl1}, v_{kl2}, \ldots, v_{klN}\}$, where v_{kln} is defined as 1 if group l of course k is assigned to campus n, and 0 otherwise.

The assignment vector of course k to the M faculty members $\mathcal{F}$ is denoted by $\mathcal{W}_k = \{u_{k1}, u_{k2}, \ldots, u_{kM}\}$, where w_{km} is defined as 1 if course k is assigned to faculty member m, and 0 otherwise. The set of I teaching time slots is denoted by $\mathcal{T} = \{T^1, T^2, \ldots, T^I\}$.

Table I displays the parameters and variables used in the mathematical formulation.

The mathematical formulation presented in this paper has been specifically selected to address the scheduling problem of SEU. It takes into account the allocation of faculty members to different campuses and time slots, as well as the assignment of student groups to courses and campuses. By formulating the problem mathematically, it is possible to develop an optimized solution that considers various constraints and objectives, ultimately leading to a more efficient and effective scheduling of blended learning courses at SEU.

The decision variable x_{klmi} is used to represent this problem, where the four indexes correspond to the course k, the group l, the faculty member identifier m, and the time slot i. x_{klmi} is equal to 1, if the group l of the course k is assigned to the faculty member m and to the time slot i, and 0 otherwise.

Equation (1) defines $FF(x)$ as the number of groups capable of accommodating blended learning without necessitating the conversion of face-to-face classes into virtual sessions.

$$FF(x) = \sum_{k=1}^{K} \sum_{l=1}^{L_k} \sum_{m=1}^{M} \sum_{n=1}^{N} \sum_{i=1}^{I} x_{klmi} v_{kln} u_{mn} \tag{1}$$

The total number of groups taught by a faculty member m is given by equation (2). It is computed by summing over all courses, groups, and time slots for which the faculty member m is assigned, and their respective decision variables x_{klmi} and v_{kln}.

$$TG^m(x) = \sum_{k=1}^{K} \sum_{l=1}^{L_k} \sum_{n=1}^{N} \sum_{i=1}^{I} x_{klmi} v_{kln} \tag{2}$$

The total number of groups taught in a time slot i is given by equation (3). It is computed by summing over all courses, groups, and faculty members for which the time slot i is assigned, and their respective decision variables x_{klmi}.

$$TS^i(x) = \sum_{k=1}^{K} \sum_{l=1}^{L_k} \sum_{m=1}^{M} x_{klmi} \tag{3}$$

In this paper, two types of metrics are proposed to measure the disparity between faculty members and between teaching time slots. The maximum metric is used to find the faculty member or time slot with the highest workload. Equations (4) and (5) respectively define the maximum workload for faculty members and time slots.

$$D^{Max}_{Faculties}(x) = \max_{m \in \mathcal{F}} TG^m(x), \tag{4}$$

$$D^{Max}_{Slots}(x) = \max_{i \in \mathcal{T}} TS^i(x), \tag{5}$$

The range metric measures the distance between the maximum and minimum workload. Equations (6) and (7) respectively define the range for faculty members and time slots.

$$D^{Range}_{Faculties}(x) = \max_{m \in \mathcal{F}} TG^m(x) - \min_{m \in \mathcal{F}} TG^m(x), \tag{6}$$

Notation	Description
K	number of courses
k	index of the course
L_k	number of groups of the course k
l	index of a group
C	set of K courses
C_k	course k
$\mathcal{G}_k$	set of L_k groups of the course k
M	number of faculty members
m	index of a faculty member
$\mathcal{F}$	set of M faculty members
R_m	teaching load reduction of the faculty member m
N	number of campuses
n	index of a campus
$\mathcal{B}$	set of N campuses
$\mathcal{U}_m$	assignment vector of faculty member m to the N campuses
u_{mn}	boolean variable equal to 1 if the faculty member m is assigned to the campus n
$\mathcal{V}_{kl}$	assignment vector of the group l of the course k to the N campuses
v_{kln}	boolean variable equal to 1 if group l of the course k is assigned to the campus n
I	number of teaching time slots
i	index of a time slot
$\mathcal{T}$	set of I teaching time slots
$\mathcal{W}_k$	assignment vector of the course k to the M faculty members
w_{km}	boolean variable equal to 1 if the course k is assigned to the faculty member m
x	boolean decision vector
x_{klmi}	boolean decision variables of the optimization problem
$FF(x)$	number of groups that can satisfy a blended learning without violation.
$TG^m(x)$	total number of groups tought by a faculty member m
$MaxTG^m$	maximum number of groups that can be tought by a faculty member m
$TS^i(x)$	total number of groups tought in a time slot i
$D^{Max}_{Faculties}(x)$	maximum distance of total number of groups tought by faculty members
$D^{Range}_{Faculties}(x)$	distance between the maximum and the minimum of total number of groups tought by faculty members
D^{Max}_{Slots}	maximum distance of total number of groups tought in all time slots
D^{Range}_{Slots}	distance between the maximum and the minimum of total number of groups tought in all time slots
α_1	minimum load of faculty members
α_2	maximum load of faculty members
β_1	minimum load of teaching time slots
β_2	maximum load of teaching time slots
$\mathcal{X}^{kl}$	set of faculty members belonging to the same campus as the group l of the course k
$\mathcal{Y}^{kl}$	set of time slots where a faculty member from $\mathcal{X}^{kl}$ can be assigned
$\overline{Lf}$	average load of all faculty members
$\overline{Li}$	average load of all time slots

TABLE I. Notations used in the mathematical model

$$D_{Slots}^{Range}(v) = \max_{i\in\mathcal{T}} TS^i(x) - \min_{i\in\mathcal{T}} TS^i(x). \quad (7)$$

An optimization technique proposing a three-dimensional multi-objective lexicographic order has been introduced. The first order in the proposed technique prioritizes the number of groups that can participate in blended learning without converting face-to-face classes into virtual session, followed by the second order that ensures a balanced distribution of groups among faculty members, and finally, the third order seeks to balance the number of groups among teaching time slots.The input to the proposed optimization problem is given by $\mathcal{U}$, $\mathcal{V}$, and $\mathcal{W}$.

The optimization problem is as follows:

$$\min_{x\in\{0,1\}} \quad (-FF(x), \alpha_2 - \alpha_1, \beta_2 - \beta_1) \quad (8)$$

Subject to:

$$\alpha_1 \leq (\sum_{k=1}^{K}\sum_{l=1}^{L_k}\sum_{n=1}^{N}\sum_{i=1}^{I} x_{klmi}v_{kln} + R_m), m \in \{1, ..., M\}, \quad (9)$$

$$\alpha_2 \geq (\sum_{k=1}^{K}\sum_{l=1}^{L_k}\sum_{n=1}^{N}\sum_{i=1}^{I} x_{klmi}v_{kln} + R_m), m \in \{1, ..., M\}, \quad (10)$$

$$\beta_1 \leq \sum_{k=1}^{K}\sum_{l=1}^{L_k}\sum_{m=1}^{M} x_{klmi}, i \in \{1, ..., I\}, \quad (11)$$

$$\beta_2 \geq \sum_{k=1}^{K}\sum_{l=1}^{L_k}\sum_{m=1}^{M} x_{klmi}, i \in \{1, ..., I\}, \quad (12)$$

$$\sum_{m=1}^{M}\sum_{i=1}^{I} x_{klmi} = 1 \quad \forall k \in \{1, ..., K\}, l \in \{1, ..., L_k\}, \quad (13)$$

$$\sum_{k=1}^{K}\sum_{l=1}^{L_k} x_{klmi} \leq 1 \quad \forall m \in \{1, ..., M\}, i \in \{1, ..., I\}, \quad (14)$$

$$TG^m(x) \leq MaxTG^m \quad \forall m \in \{1, ..., M\}, \quad (15)$$

$$x_{klmi} \leq w_{km} \quad \forall k \in \{1, ..., K\}, \forall l \in \{1, ..., L_k\}, \quad (16)$$
$$\forall m \in \{1, ..., M\}, \forall i \in \{1, ..., I\},$$

$$x_{klmi} \in \{0, 1\}, \forall k \in \{1, ..., K\}, \forall l \in \{1, ..., L_k\}, \quad (17)$$
$$\forall m \in \{1, ..., M\}, \forall i \in \{1, ..., I\},$$

The aim of the optimization problem at hand is to minimize a multi-objective function using Boolean decision variables to schedule courses for faculty members. The objective function comprises three components: the negative value of the function $FF(x)$, the difference between the minimum and maximum number of groups taught by faculty members, and the difference between the minimum and maximum number of groups taught in teaching time slots, as defined in equation (8).

If the range metric is used, the problem involves certain constraints that are defined in equations (9) to (17). Constraints (9) and (10) set the minimum and maximum number of groups that a faculty member can teach. Similarly, constraints (11) and (12) specify the minimum and maximum number of groups that can be taught during a particular time slot. However, if the maximum metric is used, the problem will not include constraints (9) and (11). In this case, α_1 and β_1 will be replaced by zero in equation (8).

Constraints in equations (13) to (17) relate to the scheduling of courses for different faculty members. Constraints (13) ensure that only one faculty member can teach a particular course group, while constraints (14) restrict the maximum number of groups that can be assigned to a faculty member in a particular time slot to one. Constraints (15) limit the number of groups assigned to a faculty member to be less than or equal to their maximum load. Constraints (16) place a restriction on the value of decision variable x_{klmi} to be less than or equal to w_{km}, where w_{km} represents a faculty member's capability to teach a particular course. Finally, constraints (17) outline the range of decision variables, which can only take on Boolean values of 0 or 1.

At SEU, students are grouped based on their level of study and have the freedom to choose courses and groups that align with their curriculum. Therefore, the optimization problem proposed in this context does not take into account any constraints related to student availability.

In the next section, the proposed heuristics will be presented by describing their implementation process.

4. Proposed methods

A. Minimum Load Accumulation Heuristic (MLAH)

Algorithm 1: Initialization scheme

[1] K: Number of courses;
$L = (L_k|k = 1..K)$: Vector of number of groups per course k;
M: Number of faculty members;
N: Number of campuses;
$R = (R_m|m = 1..M)$: Vector of load reduction for faculty members;
$\mathcal{V}_{kl} = (v_{kln}|k = 1..K, l = 1..L_k, n = 1..N)$: Assignment vector of group l of course k to campus n;
$\mathcal{W}_k = (w_{km}|k = 1..K, m = 1..M)$: Assignment vector of course k to faculty member m;
$\mathcal{U}_m = (u_{mn}|m = 1..M, n = 1..N)$: Assignment vector of faculty member m to campus n;
$MaxTG = (TGM^m|m = 1..M)$: Vector of maximum load for faculty members;
$x_{klmj} = (0|k = 1..K, l = 1..L_k, m = 1..M, j = 1..J)$;
Apply MLAH or ALAH

Algorithm 2 is a heuristic method developed to allocate course groups to faculty members and time slots.

Algorithm 2: Minimum Load Accumulation Heuristic (MLAH)

[1] $FF = 0$: Number of groups without blended learning violation;
for $\{k \in C\}$ **do**
  for $\{l \in \mathcal{G}_k\}$ **do**
    $\mathcal{X}^{kl} = \{m \in \mathcal{F}; n \in \mathcal{B} | w_{km} = u_{mn} = v_{kln} = 1\}$;
    $\mathcal{Y}^{kl} = \{i \in \mathcal{T}; m \in \mathcal{X}^{kl} | x_{klmi} = 0\}$;
    if $(\mathcal{X}^{kl} \neq \emptyset$ *and* $\mathcal{Y}^{kl}) \neq \emptyset$ **then**
      $(m^*, i^*) = min_{\{(m,i) \in (\mathcal{X}^{kl}, \mathcal{Y}^{kl})\}} (TG^m, TS^i)$;
      $FF = FF + 1$;
      $x_{klm^*i^*} = 1$;
      : Select the faculty member m^* with the minimum load, from the same campus as the group l of course k, at the time slot i^* with the minimum load as well;
    else
      $\mathcal{Z}^{kl} = \{m \in \mathcal{F}; i \in \mathcal{T} | w_{km} = 1\}$;
      if $(\mathcal{Z}^{kl} \neq \emptyset)$ **then**
        $(m^*, i^*) = min_{\{(m,i) \in (\mathcal{Z}^{kl}, \mathcal{X}^{kl})\} (TG^m, TS^i)}$;
        $x_{klm^*i^*} = 1$;
        : Select the faculty member m^* with the minimum load, from different campus to the group l of course k, at the time slot i^* with the minimum load as well;
      else
        Display(No feasible solution);

The algorithm ensures that the teaching loads are evenly distributed among faculty members and time slots, while also maximizing the number of groups assigned to faculty members from the same campus. The initialization scheme described in 1 specifies key parameters, such as the number of courses, groups per course, faculty members, campuses, load reduction for faculty members, and the maximum load for each faculty member. The initialization scheme also sets up assignment vectors, including $\mathcal{V}_{kl}$, which assigns a group l of course k to campus n, $\mathcal{W}_k$, which assigns course k to faculty member m, and $\mathcal{U}_m$, which assigns faculty member m to campus n. Additionally, the initialization scheme defines a binary variable x_{klmi} that denotes whether a faculty member m is assigned to teach the group l of the course k during time slot i, and a variable FF that tracks the number of instances without blended learning violations.

In the algorithm 2, $\mathcal{X}^{kl}$ is the set of faculty members belonging to the same campus as the group l of the course k, and $\mathcal{Y}^{kl}$ is the set of time slots where a faculty member from $\mathcal{X}^{kl}$ can be assigned. The algorithm starts by looping over all courses and groups. For each group, it identifies the faculty member m^* with the minimum load from $\mathcal{X}^{kl}$ and the time slot i^* with the minimum load from $\mathcal{Y}^{kl}$. If a feasible faculty member is found, the algorithm increments FF, sets the binary variable $x_{klm^*i^*}$ to 1, and proceeds to the next group.

If no feasible faculty member is found from the same campus (i.e., $\mathcal{X}^{kl} = \emptyset$ or $\mathcal{Y}^{kl} = \emptyset$), the algorithm searches for a faculty member m^* with the minimum load from a different campus, again considering the minimum load time slot i^*. If a feasible faculty member is found, the algorithm keeps FF to the same value and sets $x_{klm^*i^*}$ to 1. If no feasible faculty member is found, the algorithm outputs "No feasible solution."

The algorithm prioritizes the teaching time slot with the minimum load to balance the load between faculty members and between teaching time slots. By using this approach, the algorithm aims to assign courses to faculty members while ensuring that their loads are balanced and that blended learning constraints are not violated.

B. Average Load Accumulation Heuristic (ALAH)

Algorithm 3 begins by initializing a counter FF to track the number of groups without blended learning violation. It then calculates the average load of all faculty members, denoted as $\overline{Lf}$, and the average load of all time slots, denoted as $\overline{Li}$.

Next, the algorithm iterates over all courses k in set C. For each course k, it iterates over all groups l in set $\mathcal{G}_k$.

If the set $\mathcal{X}^{kl}$ of faculty members belonging to the same campus as the group l of the course k, and the set $\mathcal{Y}^{kl}$ of time slots where a faculty member from $\mathcal{X}^{kl}$ can be assigned are not empty, then the algorithm proceeds to select a feasible faculty member m^* from $\mathcal{X}^{kl}$ whose teaching load is less than $\overline{Lf}$. If no such faculty member is found, the algorithm selects one with a teaching load greater than $\overline{Lf}$ but with the minimum total load TG^m. The algorithm then selects a time slot i^* from $\mathcal{Y}^{kl}$ with the minimum load among those with load less than $\overline{Li}$ or the minimum total load TS^i for those with load greater than or equal to $\overline{Li}$. Once a feasible faculty member and time slot are identified, the algorithm increments the FF counter and schedules group l of course k to faculty member m^* in time slot i^* by setting $x_{klm^*i^*}$ to 1.

If a feasible faculty member cannot be found from the same campus (i.e., $\mathcal{X}^{kl} = \emptyset$ or $\mathcal{Y}^{kl} = \emptyset$), the algorithm searches for one from a different campus with a teaching load less than $\overline{Lf}$. If no such faculty member is found, it selects one with a teaching load greater than $\overline{Lf}$ but with the minimum total load TG^m. The algorithm then selects a time slot i^* as before. If a feasible faculty member is found, the algorithm sets $x_{klm^*i^*}$ to 1 and keeps FF to the same value. If no feasible faculty member is found, the algorithm outputs "No feasible solution."

The Average Load Accumulation Heuristic (ALAH) presented in 3 differs from the Minimum Load Accumulation Heuristic (MLAH) by reducing the number of iterations that search for the minimum load and replacing it with a simple

Algorithm 3: Average Load Accumulation Heuristic (ALAH)

[1] $FF = 0$: Number of groups without blended learning violation;
$\overline{Lf} = K\frac{\sum_k^{L_k} 1}{M}$: Average load of all faculty members ;
$\overline{Li} = K\frac{\sum_k^{L_k} 1}{I}$: Average load of all time slots ;
for $\{k \in C\}$ **do**
  for $\{l \in \mathcal{G}_k\}$ **do**
    $\mathcal{X}^{kl} = \{m \in \mathcal{F}; n \in \mathcal{B} | w_{km} = u_{mn} = v_{kln} = 1\}$;
    $\mathcal{Y}^{kl} = \{i \in \mathcal{T}; m \in \mathcal{X}^{kl} | x_{klmi} = 0\}$;
    if $(\mathcal{X}^{kl} \neq \emptyset \text{ and } \mathcal{Y}^{kl} \neq \emptyset)$ **then**
      Select $\{m \in \mathcal{X}^{kl}\}$;
      if $(m < \overline{Lf})$ **then**
        $m^* = m$;
      else
        $m^* = min_{\{m \in \mathcal{X}^{kl}\}} TG^m$;
      Select $\{i \in \mathcal{Y}^{kl} | x_{klm^*i} = 0\}$;
      if $(i < \overline{Li})$ **then**
        $i^* = i$;
      else
        $i^* = min_{\{i \in \mathcal{Y}^{kl}\}} TS^i$;
      $FF = FF + 1$;
      $x_{klm^*i^*} = 1$;
      : Select the Faculty member $m^* < \overline{Lf}$ or $\geq \overline{Lf}$ with minimum load, from a same campus to group l of course k, during time slot $i^* < \overline{Li}$ or $\geq \overline{Li}$ with minimum load as well;
    else
      $\mathcal{Z}^{kl} = \{m \in \mathcal{F}; i \in \mathcal{T} | w_{km} = 1\}$;
      if $(\mathcal{Z}^{kl} \neq \emptyset)$ **then**
        Select $\{m \in \mathcal{X}^{kl}\}$;
        if $m < \overline{Lf}$ **then**
          $m^* = m$;
        else
          $m^* = min_{\{m \in \mathcal{X}^{kl}\}} TG^m$;
        Select $\{i \in \mathcal{Y}^{kl} | x_{klm^*i} = 0\}$;
        if $(i < \overline{Li})$ **then**
          $i^* = i$;
        else
          $i^* = min_{\{i \in \mathcal{Y}^{kl}\}} TS^i$;
        $x_{klm^*i^*} = 1$;
        : Select the Faculty member $m^* < \overline{Lf}$ or $\geq \overline{Lf}$ with minimum load, from a different campus to group l of course k, during time slot $i^* < \overline{Li}$ or $\geq \overline{Li}$ with minimum load as well;
      else
        Display(No feasible solution);

comparison with $\overline{Lf}$ for faculty load or $\overline{Li}$ for time slot load.

The next section presents the computational results for the proposed heuristics and discusses their effectiveness across various problem sizes.

5. Computational results and discussion

In this section, the computational results and discussion of the proposed heuristics for scheduling timetables in blended learning education are presented. Significant benefits can be provided to Saudi Electronic University, which has a geographically dispersed campus system and requires efficient scheduling to conduct face-to-face classes with minimum building space. To evaluate the effectiveness of the heuristics, their performance is compared with the widely used CPLEX solver on instances of different sizes - small, medium, large and extra-large. The results obtained by each method are analyzed and their strengths and weaknesses are discussed.

It should be noted that the experiments have been performed on a personal computer with a 12th Gen Intel(R) Core(TM) i7-1265U processor, 1.80 GHz clock-pulse, and 16GBytes RAM. The optimization problem is written using the Optimization Programming Language (OPL) under CPLEX Optimization Studio. Additionally, the heuristics have been compiled with Microsoft Visual Studio Community 2022 under Windows 11 Pro.

The efficiency of the proposed heuristics is demonstrated in order to showcase how scheduling challenges faced by Saudi Electronic University, especially in blended learning, can be effectively addressed. The potential benefits of these heuristics make them a valuable tool for delivering high-quality education to students at Saudi Electronic University. The heuristics are capable of minimizing violations of blended learning principles, balancing teaching loads, and evenly distributing face-to-face classes across teaching time slots, which leads to a more effective and engaging learning experience for students.

A. Case study with small size dimension

In this section, a case study is presented that examines the scheduling of five courses within a department, with each course consisting of multiple groups distributed across three different campuses. Six faculty members are employed by the department and are distributed across these three campuses. The details of each course, its respective groups, and the corresponding campus assignment are listed in Table II. Information on the faculty member assigned to each course and their respective campus is provided in Table III and Table IV.

Table V and Table VI present a detailed numerical analysis of the distribution of faculty members to courses/groups and time slots for a specific case involving five courses, six faculty members, six time slots, and three campuses. Table V highlights instances of blended learning violations, which

TABLE II. Courses/Groups/campuses assignment - Case study with small size dimension

Courses	C_1	C_2		C_3			C_4				C_5
Groups	G_1	G_1	G_2	G_1	G_2	G_3	G_1	G_2	G_3	G_4	G_1
campuses	B_1	B_2	B_2	B_3	B_1	B_3	B_1	B_3	B_3	B_1	B_2

TABLE III. Courses/Faculties teaching capabilities - Case study with small size dimension

Courses	C_1		C_2			C_3	C_4					C_5			
Faculties	F_2	F_4	F_2	F_3	F_5	F_4	F_1	F_2	F_3	F_5	F_6	F_2	F_4	F_5	F_6

TABLE IV. Faculties/campuses assignment - Case study with small size dimension

Faculties	F_1	F_2	F_3	F_4	F_5	F_6
campuses	B_1	B_2	B_2	B_3	B_1	B_3

occur when the course/group and the assigned faculty member are from different campuses. The notation $C_kG_lF_mB_n$ represents the assignment of group l from course k in campus n to faculty member m. If $C_kG_lF_mB_n$ is highlighted, it indicates that the assigned faculty member m is not from the same campus n, and all lectures are conducted virtually. Additionally, Table VI presents information on faculty load standard deviation (FSD), minimum faculty load (MinFL), maximum faculty load (MaxFL), time slot load standard deviation (TSD), CPU time in milliseconds (CPU), number of instances without blended learning violation (FF), and the violation rate (VR).

After analyzing the applied methods for the given case, it was observed that all methods resulted in the same value of $FF = 9$. This implies that 82% of the 11 groups were successfully assigned to faculty members from the same campus without any violation. However, for the remaining 2 groups, the Violation Rate (VR) was calculated to be 0.18, as they were assigned to faculty members from different campuses.

In terms of balancing the teaching load among faculty members (FSD), the CPLEX with DMax method has the highest standard deviation of faculty loads, which is 1.17. Conversely, CPLEX with DRange, MLAH, and ALAH have the same standard deviation for faculty loads, which is 0.98. Therefore, MLAH, ALAH, and CPLEX with DRange provide a better balance of teaching load among faculty members for this specific case.

Examining the standard deviation of teaching load among time slots (TSD), it is observed that all methods have the same standard deviation of 0.41.

The value of MinFL, which represents the minimum load of faculty members, is zero when using CPLEX with DMax and one when using the other methods. This means that in the case of CPLEX with DMax, there is at least one faculty member who does not have any assigned workload, while CPLEX with DRange, MLAH, and ALAH ensure that each faculty member is assigned at least one group.

The value of MaxFL, which represents the maximum load of faculty members, is the same for all methods.

After analyzing the CPU time, MLAH and ALAH were found to have significantly faster processing times than CPLEX with DMax and CPLEX with DRange. CPLEX with DMax was seven times faster than CPLEX with DRange, while MLAH and ALAH had CPU times less than 1 millisecond, which cannot be accurately presented due to the small size of the case study.

This analysis was conducted on a specific case study. Therefore, a range of randomly generated instances will be evaluated in the next subsection to further validate the efficiency and effectiveness of the implemented heuristics.

B. Randomly generated small problems

This subsection presents a comparison of CPLEX with DMax, CPLEX with DRange MLAH, and ALAH, in a small institution setting, equivalent to a level of study. The case study involves five courses with a random number of groups ranging from 1 to 4 groups, six faculty members, and three campuses. The assignment vectors, namely $\mathcal{V}_{kl}$, $\mathcal{W}_k$, and $\mathcal{U}_m$, are generated randomly using a rectangular distribution with a constant probability. The results of this study are discussed in the subsequent paragraphs.

The results presented in Table VII illustrates that CPLEX with DMax, CPLEX with DRange, MLAH, and ALAH all achieve the same violation rate for every instance, indicating that the proposed heuristics are effective in minimizing the violation rate, similar to CPLEX. However, it is worth noting that the violation rate for instance 4 is relatively high at 62%. Upon analyzing the distribution of faculty members and student groups across campuses, it was discovered that there is a campus that has three groups assigned to it, yet no faculty member assigned to teach those groups. As a result, all face-to-face lectures for these three groups will be converted to fully virtual classes. Additionally, there is a course consisting of three groups that are assigned to the same campus, but only one faculty member is available to teach them, and that faculty member belongs to a different campus. Therefore, the remaining two groups will also be converted to fully online lectures. These two groups belong to two different courses that can be taught by faculty members from different campuses.

TABLE V. Results of Computational Analysis for Course/Group/Faculty/campus Assignments to Teaching Time Slots using CPLEX with DMax, CPLEX with DRange, MLAH, and ALAH: A Case Study with Five Courses, Six Faculty Members, Six Time Slots, and Three campuses.

Methods \ Time slots	T_1	T_2	T_3	T_4	T_5	T_6
CPLEX with DMax	$C_2G_2F_3B_2$ $C_2G_1F_2B_2$	$C_4G_4F_1B_1$ $C_4G_3F_6B_3$	$C_4G_2F_6B_3$	$C_4G_1F_1B_1$ $C_3G_3F_4B_3$	$C_5G_1F_2B_2$ $C_3G_1F_4B_3$	$C_3G_2F_4B_1$ $C_1G_1F_2B_1$
CPLEX with DRange	$C_2G_2F_3B_2$ $C_2G_1F_2B_2$	$C_4G_2F_6B_3$	$C_4G_3F_6B_3$ $C_3G_1F_4B_3$	$C_4G_4F_5B_1$ $C_4G_1F_1B_1$	$C_5G_1F_2B_2$ $C_3G_3F_4B_3$	$C_3G_2F_4B_3$ $C_1G_1F_2B_2$
MLAH	$C_1G_1F_2B_1$ $C_4G_1F_1B_1$	$C_2G_1F_3B_2$ $C_4G_2F_6B_3$	$C_2G_2F_2B_2$ $C_4G_3F_6B_3$	$C_3G_1F_4B_3$ $C_4G_4F_5B_1$	$C_3G_2F_4B_3$ $C_5G_1F_2B_2$	$C_3G_3F_4B_3$
ALAH	$C_1G_1F_2B_1$ $C_4G_1F_1B_1$	$C_2G_1F_3B_2$ $C_4G_2F_6B_3$	$C_2G_2F_2B_2$ $C_4G_3F_6B_3$	$C_3G_1F_4B_3$ $C_4G_4F_5B_1$	$C_3G_2F_4B_3$ $C_5G_1F_2B_2$	$C_3G_3F_4B_3$

TABLE VI. Comparison of CPLEX with DMax, CPLEX with DRange, MLAH, and ALAH - Case Study with Five Courses, Six Faculty Members, Six Time Slots, and Three campuses - FSD, MinFL, MaxFL, TSD, CPU Time, FF, and VR

Methods \ Results	FSD	MinFL	MaxFL	TSD	CPU	FF	VR
CPLEX with DMax	1.17	0	3	0.41	60	9	0.18
CPLEX with DRange	0.98	1	3	0.41	440	9	0.18
MLAH	0.98	1	3	0.41	≤ 1	9	0.18
ALAH	0.98	1	3	0.41	≤ 1	9	0.18

Table VII provides information on the comparison of FSDs for different methods based on faculty load. The analysis indicates that CPLEX with DRange generally has fewer FSDs than CPLEX with DMax, MLAH, and ALAH, except for instances 4 and 7 where the range between MaxFL and MinFL is the same for both CPLEX with DMax and CPLEX with DRange. Specifically, in instance 4, CPLEX with DRange has fewer FSDs than MLAH and ALAH as the range between MaxFL and MinFL is smaller. However, in instance 7 where the range between MaxFL and MinFL is the same for all methods, CPLEX with DRange has a higher FSD than CPLEX with DMax, MLAH, and ALAH. Notably, instance 8 exhibits perfect load balancing in terms of faculty and time slots load, where all faculty members have a teaching load of two groups, and all time slots have a teaching load of two groups. In this case, the values of MinFL and MaxFL are the same.

In terms of MinFL, it was found that CPLEX with DRange yields a higher value than the other methods for most random instances. CPLEX with DMax and ALAH provide the lower value of MinFL, while MLAH gives the same value as CPLEX with DRange in some cases. It is essential to consider this parameter in the study because load balancing should not only ensure that all faculty members have a fair teaching load but also that each faculty member has a teaching load whenever possible.

As for time slot load balancing, Table VII indicates that CPLEX with DMax, CPLEX with DRange, MLAH, and ALAH achieve the same TSD. This result confirms that the proposed heuristics guarantee good load balancing among time slots.

Regarding CPU time, Table VII shows that MLAH and ALAH take less than one millisecond for all random instances. Furthermore, as demonstrated in the previous subsection, CPLEX with Dmax is faster than CPLEX with DRange.

Figure 1 presents the faculty and time slot load distribution for MLAH and ALAH in the case of instance 1. MLAH selects the faculty member or time slot with the minimum load, implementing load balancing gradually. In contrast, ALAH compares faculty load or time slot load with a pre-calculated average load, implementing load balancing globally. When the average load is exceeded, ALAH uses the same approach as MLAH. For small-sized problems, MLAH produces better results than ALAH in terms of standard deviation and minimum faculty load. However, their impact on CPU time cannot be determined due to the small dimension size.

C. *Randomly generated medium problems*

In this subsection, a comparison is made between CPLEX with DMax, CPLEX with DRange, MLAH, and ALAH for a medium-sized problem consisting of 30 courses, each of which is assigned a random number of groups between 1 and 12. There are 40 faculty members who are randomly assigned to 3 campuses and 24 time slots. Similar to the previous subsection, the assignment vectors, denoted as $\mathcal{V}_{kl}$, $\mathcal{W}_k$, and $\mathcal{U}_m$, are generated randomly using a rectangular distribution with a constant probability.

Table VIII shows that the proposed heuristics are highly effective in reducing the blended learning violation rate, as evidenced by the fact that as the problem size increases, all methods reach a violation rate of zero for all instances.

htbp

TABLE VII. Computational results - Comparison of CPLEX with DMax, CPLEX with Drange, MLAH and ALAH for FSD, TSD, MinFL, MaxFL, CPU time (ms), FF, VR - Random instances with small size dimension

Methods \ instance number		1	2	3	4	5	6	7	8	9	10
CPLEX with DMax	FSD	1.16	0.75	0.84	1.72	0.52	1.1	1.17	0	1.72	1.21
	TSD	0.41	0.41	0.55	0.98	0.52	0	0.41	0	0.41	0.52
	MinFL	0	1	0	0	1	1	0	2	0	0
	MaxFL	3	3	2	4	2	3	3	2	4	3
	CPU	70	70	90	90	80	90	90	70	90	60
	FF	10	9	7	5	8	11	8	10	11	7
	VR	0.23	0.31	0.22	0.62	0.20	0.08	0.38	0.17	0.15	0.30
CPLEX with DRange	FSD	0.41	0.41	0.55	1.83	0.52	0.89	1.33	0	1.47	0.82
	TSD	0.41	0.41	0.55	0.41	0.52	0	0.41	0	0.41	0.52
	MinFL	2	2	1	0	1	1	0	2	0	1
	MaxFL	3	3	2	4	2	3	3	2	4	3
	CPU	330	320	230	250	330	110	1910	70	270	230
	FF	10	9	7	5	8	11	8	10	11	7
	VR	0.23	0.31	0.22	0.62	0.20	0.08	0.38	0.17	0.15	0.30
MLAH	FSD	0.98	0.75	0.84	1.94	1.03	0.89	1.17	0	1.33	0.82
	TSD	0.41	0.41	0.55	0.75	0.52	0.63	0.41	0	0.41	0.52
	MinFL	1	1	0	0	0	1	0	2	0	1
	MaxFL	4	3	2	5	3	3	3	2	4	3
	CPU	≤ 1	≤ 1	≤ 1	≤ 1	≤ 1	≤ 1	≤ 1	≤ 1	≤ 1	≤ 1
	FF	8	9	7	5	8	11	8	10	11	7
	VR	0.23	0.31	0.22	0.62	0.20	0.08	0.38	0.17	0.15	0.30
ALAH	FSD	0.41	1.17	0.84	1.94	1.03	0.89	1.17	0	1.33	0.82
	TSD	0.41	0.41	0.55	0.41	0.52	0	0.41	0	0.41	0.52
	MinFL	2	0	0	0	0	1	0	2	0	1
	MaxFL	3	3	2	5	3	3	3	2	4	3
	CPU	≤ 1	≤ 1	≤ 1	≤ 1	≤ 1	≤ 1	≤ 1	≤ 1	≤ 1	≤ 1
	FF	10	9	7	5	8	11	8	10	11	7
	VR	0.23	0.31	0.22	0.62	0.20	0.08	0.38	0.17	0.15	0.30

MLAH outperforms CPLEX with DMax, CPLEX with DRange, and ALAH in reducing FSD in most random instances, except for instance 5 where CPLEX with DRange has a lower value. For instances 6 and 10, CPLEX with DRange and MLAH have the same FSD value. In terms of FSD, ALAH only outperforms MLAH in instance 1. In all random instances, MLAH outperforms CPLEX with DMax in reducing FSD.

By examining the range between MaxFL and MinFL, CPLEX with DRange has a lower range than CPLEX with DMax, MLAH, and ALAH. MLAH has a lower range than CPLEX with DMax and ALAH. Regarding the minimum load of faculty members, CPLEX with DRange and MLAH ensure that no faculty members are without any teaching load, while in many instances, CPLEX with Dmax and ALAH have at least one faculty member without any teaching load.

The proposed heuristics are always highly efficient compared to CPLEX methods in terms of CPU time. However, for these size instances, ALAH is two to four times faster than MLAH due to its lower complexity when the faculty member's load and time slot's load are smaller than the average load of all faculty members and time slots.

TSD remains the same for all random instances and all proposed methods.

D. Randomly generated large problems

In this subsection, the number of courses has been increased to 60, each of which is randomly assigned a number of groups between 1 and 12. The number of faculty members has also been increased to 70, while the number of time slots remains 24 and the number of campuses is three. The studied size problem is equivalent to the size of a college. the Input vectors such as $\mathcal{V}_{kl}$, $\mathcal{W}_k$, and $\mathcal{U}_m$ are generated randomly, similar to the medium and small size problems.

Table IX demonstrates that the proposed heuristics minimize blended learning violations very effectively, as the VR is consistently zero for all random instances. Moreover, as the problem size increases, the likelihood of converting blended learning into full online learning decreases.

TABLE VIII. Computational results - Comparison of CPLEX with DMax, CPLEX with Drange, MLAH and ALAH for FSD, TSD, MinFL, MaxFL, CPU time (ms), FF, VR - Random instances with medium size dimension

Methods \ instance number		1	2	3	4	5	6	7	8	9	10
CPLEX with DMax	FSD	1.62	1.59	1.5	2.52	1.48	1.7	2.23	2.1	2.36	2.2
	TSD	0.2	0.99	0.34	0.44	0.82	0.41	0.34	0.41	0.88	0.48
	MinFL	1	1	1	1	0	0	2	0	0	0
	MaxFL	6	6	6	9	6	7	8	7	8	8
	CPU	28360	7430	39420	16470	18080	38720	31570	33670	40440	36400
	FF	191	174	189	222	200	235	213	187	207	196
	VR	0	0	0	0	0	0	0	0	0	0
CPLEX with DRange	FSD	1.25	1.41	1.38	2.04	0.93	0.94	1.76	1.53	1.57	1.41
	TSD	0.2	0.44	0.34	0.44	0.48	0.41	0.34	0.41	0.49	0.38
	MinFL	3	3	3	4	4	5	4	3	4	4
	MaxFL	6	6	6	9	6	7	8	7	8	8
	CPU	88030	77350	79960	173840	91610	81100	157750	90130	51250	60080
	FF	191	174	189	222	200	235	213	187	207	196
	VR	0	0	0	0	0	0	0	0	0	0
MLAH	FSD	1	1.23	1.06	1.71	0.96	0.94	1.67	1.14	1.39	1.41
	TSD	0.2	0.44	0.34	0.44	0.48	0.41	0.34	0.41	0.49	0.38
	MinFL	3	2	2	3	3	5	3	3	4	3
	MaxFL	8	7	7	10	7	8	11	8	9	8
	CPU	8	28	26	28	25	36	31	23	64	23
	FF	191	174	189	222	200	235	213	187	207	196
	VR	0	0	0	0	0	0	0	0	0	0
ALAH	FSD	0.95	1.82	1.11	1.87	1.84	0.97	1.97	1.33	1.66	1.45
	TSD	0.2	0.44	0.34	0.44	0.48	0.41	0.34	0.41	0.49	0.38
	MinFL	3	0	1	1	0	3	1	1	1	2
	MaxFL	8	7	7	10	7	8	11	8	9	8
	CPU	4	10	7	8	9	15	19	9	12	7
	FF	191	174	189	222	200	235	213	187	207	196
	VR	0	0	0	0	0	0	0	0	0	0

In terms of faculty member load balancing, MLAH outperforms CPLEX with DRange, CPLEX with DMax, and ALAH in reducing FSD. MLAH provides the same range between MaxFL and MinFL as CPLEX with DRange for all instances, except for instance 10 where CPLEX with DRange is slightly better. On the other hand, ALAH and CPLEX with DMax cannot guarantee a teaching load for some faculty members because MinFL is zero for some random instances.

Regarding TSD, all proposed methods provide the same value for all instances, similar to the results obtained for other problem sizes.

When analyzing CPU time, ALAH is consistently two to three times faster than MLAH. CPLEX with DMax is also faster than CPLEX with DRange, by a factor of two to eighteen.

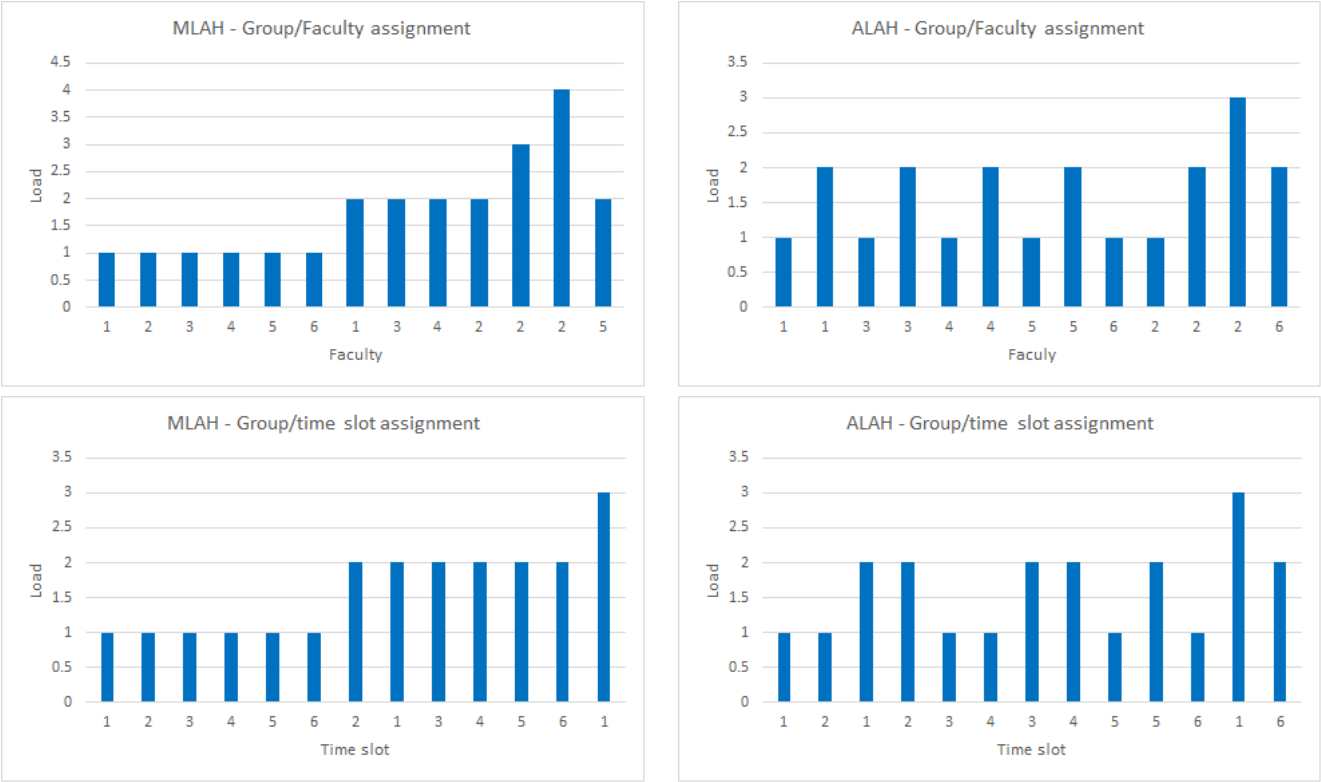

Figure 1. Faculty and Time slots load distributions for MLAH and ALAH - small size dimension

TABLE IX. Computational results - Comparison of CPLEX with DMax, CPLEX with Drange, MLAH and ALAH for FSD, TSD, MinFL, MaxFL, CPU time (ms), FF, VR - Random instances with large size dimension

Methods \ instance number		1	2	3	4	5	6	7	8	9	10
CPLEX with DMax	FSD	1.49	2.14	2.16	2.01	1.81	0.84	1.53	2.22	1.53	1.54
	TSD	0.46	0.44	0.34	0.77	1.14	0.97	0.2	0.48	0.2	0.88
	MinFL	1	1	0	1	1	2	1	1	2	1
	MaxFL	6	8	8	8	8	6	7	8	7	7
	CPU	36820	74760	78580	42930	79160	74290	50390	70160	69890	43730
	FF	319	402	381	399	443	394	407	404	383	396
	VR	0	0	0	0	0	0	0	0	0	0 height
CPLEX with DRange	FSD	1.11	1.77	1.55	1.6	1.02	0.49	1.33	1.79	1.15	0.8
	TSD	0.46	0.44	0.34	0.49	0.51	0.5	0.2	0.38	0.2	0.51
	MinFL	3	4	4	4	5	5	4	4	4	5
	MaxFL	6	8	8	8	8	6	7	8	7	7
	CPU	672070	118590	158790	445160	332720	218350	156240	286510	504640	335810
	FF	319	402	381	399	443	394	407	404	383	396
	VR	0	0	0	0	0	0	0	0	0	0
MLAH	FSD	0.67	1.44	1.28	1.28	0.79	0.49	1.01	1.47	0.79	0.66
	TSD	0.46	0.44	0.34	0.49	0.51	0.5	0.2	0.38	0.2	0.51
	MinFL	3	4	4	4	5	5	4	4	4	4
	MaxFL	6	8	8	8	8	6	7	8	7	7
	CPU	28	112	82	113	138	96	58	124	109	54
	FF	319	402	381	399	443	394	407	404	383	396
	VR	0	0	0	0	0	0	0	0	0	0
ALAH	FSD	0.63	1.69	1.65	1.48	0.85	0.49	1.08	1.69	0.9	0.78
	TSD	0.46	0.44	0.34	0.49	0.51	0.5	0.2	0.38	0.2	0.51
	MinFL	3	0	0	0	4	5	2	0	2	2
	MaxFL	6	8	8	8	8	6	7	8	7	7
	CPU	9	37	40	50	43	33	19	34	33	18
	FF	319	402	381	399	443	394	407	404	383	396
	VR	0	0	0	0	0	0	0	0	0	0

6. Conclusion and future work

In conclusion, this paper introduces two innovative heuristics designed to schedule timetables effectively in blended learning education. The objectives are to reduce violations of blended learning principles, ensure equitable teaching loads, and evenly distribute face-to-face classes throughout teaching time slots. While the study is driven by the specific challenges encountered at Saudi Electronic University, its findings are applicable to institutions with geographically dispersed campuses adopting blended learning methodologies. The optimization problem proposed here addresses offline timetabling scheduling at SEU, where student groups, faculty members, and time slots are assigned before the semester begins and student registration commences.

The heuristics presented in this paper provide effective solutions for minimizing blended learning violations, comparable to those achieved by CPLEX but with significantly reduced computational time. Computational analyses show that MLAH outperforms ALAH and CPLEX methods in balancing faculty member loads, especially when dealing with larger problem sizes approaching university dimensions.

Future research can explore several directions. Firstly, the representation of faculty member expertise could be refined by utilizing a scale of preferences ranging from low to high, rather than a binary can-teach or cannot-teach approach. Secondly, a software solution can be developed using web and mobile frameworks, granting students and faculty members convenient access to their schedules and timely notifications. Lastly, comprehensive reporting and analytic capabilities can be incorporated to generate insights on course scheduling trends, resource utilization, and student/faculty distribution, thereby facilitating informed decision-making and efficient resource allocation.

References

[1] A. Kayanda, L. Busagala, and M. Tedre, "Design and implementation of timetabling software for an improved decision making at the tanzanian higher education context: A design science approach," in *2019 IEEE AFRICON*, 2019, pp. 1–4.

[2] C. Eke, A. Garcia-Dominguez, J. van Mourik, and I. Khan, "A case study of model-driven engineering for automated timetabling," in *2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*, 2019, pp. 203–210.

[3] X. Xu, "A survey on students' satisfaction with college english iv course in blended learning mode," in *2022 2nd International Conference on Big Data Engineering and Education (BDEE)*, 2022, pp. 221–227.

[4] S. Hrastinski, "What do we mean by blended learning?" *TechTrends*, vol. 63, no. 5, pp. 564–569, 2019.

[5] M. Vithanapathirana, "Blended learning as an emerging approach to teacher education in higher education in sri lanka: Lessons from a state-of-the-art review," *University of Colombo Review*, vol. 2, p. 61, 05 2021.

[6] N. Megahed and E. Ghoneim, "Blended learning: The new normal for post-covid-19 pedagogy," *International Journal of Mobile and Blended Learning*, vol. 14, pp. 1–15, 01 2022.

[7] R. Shoukat, I. Ismayil, Q. Huang, O. Mohamed, M. Younas, and R. Munir, "A comparative analysis of blended learning and traditional instruction: Effects on academic motivation and learning outcomes," *PloS one*, vol. 19, p. e0298220, 03 2024.

[8] R. A. Rasheed, A. Kamsin, and N. A. Abdullah, "Challenges in the online component of blended learning: A systematic review," *Computers Education*, vol. 144, p. 103701, 2020.

[9] M. Jarraya and S. Elloumi, "Load balancing scheduling algorithms for virtual computing laboratories in a desktop-as-a-service cloud computing services," *Computer Communications*, vol. 192, pp. 343–354, 2022.

[10] K. Mishra and S. Majhi, "A state-of-art on cloud load balancing algorithms," *International Journal of Computing and Digital Systems*, vol. 9, pp. 201–220, 01 2020.

[11] J. Luo, S. Fujimura, D. E. Baz, and B. Plazolles, "Gpu based parallel genetic algorithm for solving an energy efficient dynamic flexible flow shop scheduling problem," *Journal of Parallel and Distributed Computing*, vol. 133, pp. 244–257, 2019.

[12] D. El-Baz, B. Fakih, R. S. Nigenda, and V. Boyer, "Parallel best-first search algorithms for planning problems on multi-core processors," *The Journal of Supercomputing*, vol. 78, pp. 3122–3151, 2022.

[13] A. Rossi and M. Lanzetta, "Integration of hybrid additive/subtractive manufacturing planning and scheduling by metaheuristics," *Computers Industrial Engineering*, vol. 144, p. 106428, 2020.

[14] J. Hu, H. Xiong, and G. Yang, "A hybrid algorithm for job shop scheduling problem with consideration of work timetable," in *2021 40th Chinese Control Conference (CCC)*, 2021, pp. 1663–1668.

[15] X. Wang, K. Xing, Y. Feng, and Y. Wu, "Scheduling of flexible manufacturing systems subject to no-wait constraints via petri nets and heuristic search," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 51, no. 10, pp. 6122–6133, 2021.

[16] J. xiang Luo, D. E. Baz, R. Xue, and J. Hu, "Solving the dynamic energy aware job shop scheduling problem with the heterogeneous parallel genetic algorithm," *Future Gener. Comput. Syst.*, vol. 108, pp. 119–134, 2020.

[17] S. Pang and M.-C. Chen, "Optimize railway crew scheduling by using modified bacterial foraging algorithm," *Computers Industrial Engineering*, vol. 180, p. 109218, 2023.

[18] S. Jütte, *Large-Scale Crew Scheduling; Models, Methods, and Applications in the Railway Industry*. 65189 - Wiesbaden GERMANY: Springer Gabler Wiesbade, 2019.

[19] M. C. Chen, S. N. Sze, S. L. Goh, N. R. Sabar, and G. Kendall, "A survey of university course timetabling problem: Perspectives, trends and opportunities," *IEEE Access*, vol. 9, pp. 106 515–106 529, 2021.

[20] H. Babaei, J. Karimpour, and A. Hadidi, "A survey of approaches for university course timetabling problem," *Computers Industrial Engineering*, vol. 86, pp. 43–59, 2015, applications of Computational Intelligence and Fuzzy Logic to Manufacturing and Service Systems.

[21] A. Houar, T. H. Betaouaf, and Y. Bensmain, "Scheduling pedagogical tasks for university timetables: A field study," in *2022 14th International Colloquium of Logistics and Supply Chain Management (LOGISTIQUA)*, 2022, pp. 1–5.

[22] E. Alexa, D. Zmaranda, and E. V. Moisi, "Modeling and solving the university timetabling problem using the constraints satisfaction model: A case study," in *2021 13th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*, 2021, pp. 1–6.

[23] A. Sovis, C. Patikirige, and Y. Pandigama, "Enhanced timetable scheduling: A high-performance computational approach," in *2023 8th International Conference on Information Technology Research (ICITR)*, 2023, pp. 1–6.

[24] H. A. Razak, Z. Ibrahim, and N. M. Hussin, "Bipartite graph edge coloring approach to course timetabling," in *2010 International Conference on Information Retrieval Knowledge Management (CAMP)*, 2010, pp. 229–234.

[25] P. Nandal, A. Satyawali, D. Sachdeva, and A. S. Tomar, "Graph coloring based scheduling algorithm to automatically generate college course timetable," in *2021 11th International Conference on Cloud Computing, Data Science Engineering (Confluence)*, 2021, pp. 210–214.

[26] S. Biswas, S. Nusrat, N. Sharmin, and M. M. Rahman, "Graph coloring in university timetable scheduling," *International Journal of Intelligent Systems and Applications*, vol. 15, pp. 16–32, 06 2023.

[27] K. Malhotra, K. Vasa, N. Chaudhary, A. Vishnoi, and V. Sapra, "A solution to graph coloring problem using genetic algorithm," *ICST Transactions on Scalable Information Systems*, 03 2024.

[28] M. Asham, M. Soliman, and R. Ramadan, "Trans genetic coloring approach for timetabling problem," *Artificial Intelligence Techniques - Novel Approaches Practical Applications*, vol. 1, pp. 17–25, 05 2011. [Online]. Available: https://www.ijcaonline.org/specialissues/ait/number1/2824-205

[29] E. Psarra and D. Apostolou, "A combination of genetic algorithms and local search to solve a real data university timetable scheduling problem," in *2023 14th International Conference on Information, Intelligence, Systems Applications (IISA)*, 2023, pp. 1–8.

[30] F. Awad, A. Al-Kubaisi, and M. Mahmood, "Large-scale timetabling problems with adaptive tabu search," *Journal of Intelligent Systems*, vol. 31, pp. 168–176, 01 2022.

[31] "An integer programming formulation for a case study in university timetabling," *European Journal of Operational Research*, vol. 153, no. 1, pp. 117–135, 2004, timetabling and Rostering.

[32] F. L. Farias de Freitas and G. Valentim Loch, "An approach for minimizing time to degree based on university course timetabling problem," *IEEE Latin America Transactions*, vol. 21, no. 6, pp. 752–757, 2023.

[33] M. A. Bakır and C. Aksop, "A 0-1 integer programming approach to a university timetabling problem," *Hacettepe Journal of Mathematics and Statistics*, vol. 37, 2008. [Online]. Available: https://www.researchgate.net/publication/265153670_A_0-1_integer_programming_approach_to_a_university_timetabling_problem

[34] E. Rappos, E. Thiémard, S. Robert, and J.-F. Hêche, "A mixed-integer programming approach for solving university course timetabling problems," *Journal of Scheduling*, vol. 25, 08 2022.

[35] S. bin deris, S. Omatu, H. Ohta, and P. Saad, "Incorporating constraint propagation in genetic algorithm for university timetable planning," *Engineering Applications of Artificial Intelligence*, vol. 12, pp. 241–253, 06 1999.

[36] L. Zhang and S. Lau, "Constructing university timetable using constraint satisfaction programming approach," in *International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06)*, vol. 2, 2005, pp. 55–60.

[37] J. Obit, "Developing novel meta-heuristic, hyper-heuristic and cooperative search for course timetabling problems," *PhD Thesis, University of Nottingham*, 2010.

Dr. Mohamed Jarraya holds a Ph.D. and master's degrees from the Laboratory of Analysis and Architecture of Systems (LAAS-CNRS) at Paul Sabatier University of Toulouse, France. He earned his High Engineering Diploma in Computer Science from the Engineering National School of Tunis (E.N.I.T), El Manar University, Tunisia. Dr. Jarraya currently serves as an Assistant Professor in the Department of Information Technology at the College of Computation and Informatics, Saudi Electronic University (SEU). His research interests encompass performance analysis, cloud computing, and optimization.