Düzce Üniversitesi Bilim ve Teknoloji Dergisi

Araştırma Makalesi

Müfredat Tabanlı Ders Çizelgeleme Problemi için Yeni Bir Açgözlü Algoritma

Batuhan COŞAR [a], Bilge SAY [b], Tansel DÖKEROĞLU [c,*]

[a] *Bilgisayar Mühendisliği Bölümü, Mühendislik Fakültesi, Atılım Üniversitesi, Ankara*
[b] *Yazılım Mühendisliği Bölümü, Mühendislik Fakültesi, Atılım Üniversitesi, Ankara*
[c] *Yazılım Mühendisliği Bölümü, Mühendislik Fakültesi, Çankaya Üniversitesi, Ankara*
** Sorumlu yazarın e-posta adresi: tdokeroglu@cankaya.edu.tr*

Öz

Bu çalışma, iyi bilinen Müfredat Tabanlı Ders Çizelgeleme Problemini optimize etmek için yeni bir açgözlü algoritmayı açıklamaktadır. Açgözlü algoritmalar, en iyi çözümü bulmak için yürütülmesi uzun zaman alan kaba kuvvet ve evrimsel algoritmalara iyi bir alternatiftir. Birçok açgözlü algoritmanın yaptığı gibi tek bir buluşsal yöntem kullanmak yerine, aynı problem örneğine 120 yeni buluşsal yöntem tanımlıyor ve uyguluyoruz. Dersleri müsait odalara atamak için, önerilen açgözlü algoritmamız En Büyük-İlk, En Küçük-İlk, En Uygun, Önce Ortalama Ağırlık ve En Yüksek Kullanılamaz ders-ilk buluşsal yöntemlerini kullanır. İkinci Uluslararası Zaman Çizelgesi Yarışması'nın (ITC-2007) kıyaslama setinden 21 problem örneği üzerinde kapsamlı deneyler gerçekleştirilir. Önemli ölçüde azaltılmış yumuşak kısıtlama değerlerine sahip 18 problem için, önerilen açgözlü algoritma sıfır sabit kısıtlama ihlali (uygulanabilir çözümler) rapor edebilir. Önerilen algoritma, performans açısından son teknoloji ürünü açgözlü buluşsal yöntemleri geride bırakıyor.

Anahtar Kelimeler: *Ders zaman çizelgesi oluşturma, Açgözlü algoritmalar, buluşsal yöntemler, eniyileme*

A New Greedy Algorithm for the Curriculum-based Course Timetabling Problem

Abstract

This study describes a novel greedy algorithm for optimizing the well-known Curriculum-Based Course Timetabling (CB-CTT) problem. Greedy algorithms are a good alternative to brute-force and evolutionary algorithms, which take a long time to execute in order to find the best solution. Rather than employing a single heuristic, as many greedy algorithms do, we define and apply 120 new heuristics to the same problem instance. To assign courses to available rooms, our proposed greedy algorithm employs the Largest-First, Smallest-First, Best-Fit, Average-weight first, and Highest Unavailable course-first heuristics. Extensive experiments are carried out on 21 problem instances from the benchmark set of the Second International Timetabling Competition (ITC-2007). For 18 problems with significantly reduced soft-constraint values, the proposed greedy algorithm can report zero hard constraint violations (feasible solutions). The proposed algorithm outperforms state-of-the-art greedy heuristics in terms of performance.

Keywords: *Course-timetabling, Greedy algorithms, heuristics, optimization*

Geliş: …/…/2022, Düzeltme: …/…/2022, Kabul: …/…/2022

I. INTRODUCTION

The course timetabling problem is a well-known optimization problem. Satisfying constraints such as curriculum compactness, matching classroom size with the number of students, minimizing the number of used classrooms, minimizing the distance between classrooms and many other such constraints are some criteria that must be optimized [1]. The Course Timetabling problem has Rooms and Instructors as resources that must be allocated together simultaneously and placed to schedule an hour of a course [2]. Greedy algorithms are efficient polynomial-time approaches that work with heuristics to solve a hard problem. G greedy algorithms often do not provide optimal solutions [3]. However, they are still promising enough to generate approximate solutions in reasonable amounts of time [4].

This study presents a new greedy algorithm (Greedy-CB-CTT) for optimising the Curriculum-Based Course Timetabling (CB-CTT) problem. Periods assigned to all courses in the same curriculum must be distinct as the same student cannot be in two different classrooms on the same day period. The CB-CTT algorithms try to find the best weekly assignment of university lectures to classrooms and day periods [5]. The main goal of this study is to minimize the total number of soft constraint violations while preserving the 28 satisfaction of hard constraints. Since the problem is NP-Hard and large instances of the problem cannot be solved optimally in practical times, the greedy algorithms are good alternatives to brute-force and evolutionary algorithms that spend hours of execution times to discover optimal or best achievable solutions. Instead of using a single heuristic, we simultaneously define and evaluate 120 heuristics on the same problem instance and report the overall best solution [6]. Because according to the No Free Lunch Theorem (NFL) there cannot be a single heuristic that will give the best results for all problems [7].

Our proposed greedy algorithm (Greedy-CB-CTT) sorts the courses using Largest-Course-First, Smallest-Course-First, and Average-Course-First [8], for the number of students enrolled in a course [9]. Other parameters for ordering courses are the number of unavailable hours, the number of courses in a curriculum, the number of lecture hours, and the minimum working days. The second set of ordering heuristics is the availability of the rooms. Their capacity can be ordered according to four criteria: largest, smallest, average-size and best-fit, matching the number of students enrolled in the course being assigned to a classroom. To evaluate the performance of our proposed algorithm, we carried out experiments on 21 problem instances from the Second International Timetabling Competition (ITC-2007) benchmark set [10]. The results verify that the proposed greedy algorithms can report feasible solutions (i.e., zero hard constraint violations) with significantly reduced soft-constraint values.

The main contributions of this article are as follows: 120 new heuristics are proposed to solve the CB-CTT problem. Popular task and page ordering heuristics are combined in the Greedy-CB-CTT algorithms to select the results of the best heuristic according to the behaviour of the problem instances. Average course size and average room-size first heuristics are applied to the CB-CTT problem for the first time, and the results are reported. It was possible to obtain better results than classical largest/smallest first greedy heuristic algorithms, yielding solutions giving significantly lower hard and soft constraint violations. In 18 of the 21 problem instances of the ITC-2007 benchmark set, hard constraint violations are reduced to zero in a few milliseconds during the experiments.

In section 2, related studies for state-of-the-art algorithms are presented. The formal problem definition is given in section 3. The details of heuristics and the proposed algorithm are introduced in section 4. The experimental setup obtained results and comparison with state-of-the-art algorithms are reported in section 5. Concluding remarks and future work are provided in section 6 of the study.

II. RELATED WORK

P.I.Tillet presented the results of the feasibility of determining an optimal assignment of lecturers to courses using an operations research model in 1975 [11]. It was one of the first applications in this area. Recent surveys about the university course timetabling problems give detailed information about the operations research techniques, metaheuristics and other intelligent novel methods for the CB-CTT [12–15]. MirHassani et al. analyze the primary considerations of recent papers on the university course timetabling problems and introduce the hard and soft constraints and most currently used objective functions [16]. Bettinelli et al. give a good classification of problem types in this domain: Examination Timetabling, Post-Enrollment-based Course Timetabling, and Curriculum-Based Course Timetabling, the latter being the focus of our study [5]. Initial approaches focused on linear and integer programming models [17–19], recent variants of which still provide promising results as methods, reducing the number of integer variables [20] or using minimal perturbation models to minimize the effects of dealing with the resolution of infeasible parts [21].

Geiger presents local search algorithms for the International Timetabling Competition 2007 (ITC 2007) [22]. His heuristics are based on threshold acceptance, dealing with local optima by a deterministic acceptance of inferior solutions. A stochastic search 1 of neighbors is developed by removing some lecture assignments and reassigning them to new day-period slots randomly. Zhipeng & Jin-Kao develop an Adaptive Tabu Search algorithm for the CB-CTT problem [23]. Their Tabu Search algorithm integrates features such as original double Kempe chains (a method used in the study of graph coloring problems [24]) neighbourhood structure, a penalty-guided perturbation operator and an adaptive search mechanism. Gulcu & Akkan develop two multi-objective Simulated Annealing (SA) algorithms to identify a good Pareto front (the best solution set concerning the objective function) defined by the solution quality (penalty associated with soft-constraint violations) and a robustness measure [25]. Akkan et al. work on a bi-criteria optimisation problem model and solve the CB-CTT problem using a hybrid Multi-objective Genetic Algorithm (GA), which uses Hill Climbing and SA algorithms [26]. Thepphakorn et al. develop a Particle Swarm Optimization (PSO) timetabling application to solve real-world datasets [27]. The results verify that the algorithm has a faster convergence speed than classical GA. Goh et al. combine different local search algorithms into an iterative two-stage procedure, Tabu Search with Sampling and Perturbation and an improved variant of SA; their reported results are competitive to best-known solutions reported in the literature [28]. Bagger et al. apply Benders' decomposition to the solution of the problem to generate cuts that connect the schedule and room allocation [29]. Combining Great Deluge and Tabu Search, iterative local search algorithms combined with SA are among the different methods that have been evaluated [30].

Coster et al. prepared an analysis of the CB-CTT instances [31]. They investigated machine learning methods to automate algorithm selection. They showed how problem space analysis and algorithm selection cooperate. Akkan et al. formulated the problem as a bi-criteria optimization problem [32]. A multi-objective simulated annealing algorithm and a surrogate measure are used to calculate the fitness values. Experiments showed that when the proposed algorithm uses a multi-start approach, it provides the best Pareto optimal solutions. Colajanni & Daniele formulated a completely new model that satisfies the planning constraints and the compactness of the curricula [33]. The preferences of the teachers, the minimum number of working days, maximum capacity, and stability of the classrooms are considered during the study.

III. PROBLEM DEFINITION

The CB-CTT problem is in the class of NP-hard problems. It includes allocating courses, lectures and students to a fixed set of time slots and rooms [12]. This problem must obey hard and soft constraints while performing the allocation of resources. Soft constraints are better satisfied to increase students' and instructors' quality and satisfaction levels using these timetables [34–36]. Due to its exponential growth behaviour, the scheduling cannot be solved in polynomial time. The main goal of algorithms is to minimize the penalties of constraints [12, 35].

- HC1 Each lecture of a course must be scheduled in a distinct time period and a room.
- HC2: Any two lectures cannot be assigned in the same time period and in the same room.
- HC3: an instructor can teach only one course at a given time period, and also, the lectures of courses in the same curriculum are assumed to be attended by the same set of students. Therefore, two courses in the same curriculum cannot be scheduled for the same period.
- HC4: If the lecturer of a course is listed as not available for a given period, then no lectures of the same teacher can be assigned to that period.
- SC1: The number of students registered to a course cannot be greater than the capacity of the classroom where the lecture is assigned. The penalty calculated for a room smaller than the number of registered students is given by the formula (RoomSize - NumberOfStudents) × 2.
- SC2: To make it easier for students to remember in which classroom meets, it is preferred to assign all lectures of a course to the same room.
- SC3: To allow students to attend at least part of a course' s weekly meetings, it is preferred not to schedule all of the course hours to the same day and distribute its lectures to a minimum number of days.
- SC4: If possible, a student' s courses are preferred to be on the same day and closely distributed within the same day. For example, if in a given curriculum, there is one lecture not adjacent to any other lecture in the same curriculum on the same day, a violation is counted.

We can define the CB-CTT problem in terms of N courses $C = c_1, c_2, ..., c_N$ all of which must be scheduled in a set of P periods $T = t_1, t_2, ..., t_P$, and a set of M rooms, $R = r_1, r_2, ..., r_M$. Each course c_i contains l_i lectures that must be scheduled to time periods. A time period is a pair of weekday (D days, typically Monday through Friday) and a timeslot (P periods *1* consisting of D days and H daily timeslots (means $P = D \times H$). Also, there are S curricula, $CR = C_{r1}, C_{r2}, ..., C_{rS}$ where each curriculum C_{rk} is a group of courses that are assumed to share the same students. Typically these curricula will contain must courses of a university department (I : day of week, J : time period of the day, K: Group of students, L: Lecturers, M : Courses, N : Classrooms).

All lecture hours of a given course must be on a different time period. For any given (day, time, course). Any two lectures cannot be assigned in the same period and to meet in the same room. For the day I, and time period J, classroom N, the sum of all $X_{i,j,k,l,m,n}$ (day i , time j , curriculum k , course m, classroom n) for a certain (day, time, classroom) must be zero or one. An instructor can teach only one course at a given time period, and also, the lectures of courses in the same curriculum are assumed to be attended by the same set of students. Therefore, any two courses in the same curriculum cannot be scheduled to the same time (day, period). Lecturer, L, for a given day I, and time period, J, can be assigned to teach at most once. The variable $X_{i,j,k,l,m,n}$ (day i, time j, curriculum k, lecturer l, course m, classroom n) can have only two values 0 or 1, meaning a lecturer is assigned to teach that course on that day period or not. The summation below for any given Lecturer must be zero or one. Two courses in the same curriculum cannot be scheduled to the same time period. For given curriculum K, the day I, and time J, the below summation must be zero or one. If the lecturer of a course (Lecturer L) is not available at a given period (day I, time J), then no lectures of that lecturer can be assigned to that period.

IV. PROPOSED HEURISTICS

Our study develops several heuristics to satisfy the hard and soft constraints. It will be very practical to have several heuristics that can provide solutions satisfying the constraints. Special algorithms reducing soft constraint violations can also be developed separately and can perform small modifications on the resulting timetables where all hard constraints have been eliminated. The proposed Greedy Heuristics are defined in terms of classroom sizes and course sizes [37] as in Tables 1 and 2.

A. COURSE ORDERING HEURISTICS

Highest unavailable hours first heuristic: A high number of unavailable hours listed for a course makes it very difficult to find a suitable time period for that course. For this reason, one of the course ordering heuristics sorts courses in non-increasing order of their unavailable hours. Therefore, scheduling such courses won't conflict with any other course in the same curriculum becomes easier. The largest, smallest, and average course first: As the names imply in these course ordering heuristics, the number of students registered to courses is used to decide which courses are assigned to a classroom first. By assigning courses that will have a larger impact on the resulting timetables first, there will be no need to change these first decisions because of conflicts with less important courses (in terms of the number of registered students). The first course assigned to a classroom can be chosen as the largest size, the smallest size, or the closest to the average size courses. This is achieved by sorting courses in terms of course sizes and choosing the first, the last, or the median course in the sorted list. The chosen course is removed from the list of courses, and the process is repeated until all of the courses are assigned to a classroom.

B. ROOM ORDERING HEURISTICS

In this heuristic, the classrooms are assigned to courses by considering their student capacities. The best-fit, the worst-fit (the largest classroom size), and average size, which is the median size classroom among all of the available classrooms, sorted in order of classroom sizes are used. The Classroom-Heuristics are combined with Course-Heuristics that determine the course to be assigned to that classroom. Finally, a timetable is decided by choosing an available day-time period in that classroom. Since there are four possible ordering choices for Classrooms and ten orders for choosing a Course, there are potentially 40 greedy heuristics that must be investigated to select the one(s) that give the best-expected results. Tables 1 and 2 provide the names of the heuristics used in our proposed algorithm and their ids.

Table 1. *Course ordering heuristics.*

Id	Heuristic Description
1	Course with most students first
2	Course with average number of students first
3	Course with least number of students
4	Course with highest number of unavailable day/periods
5	Course with most number of hours first, if equal then largest unavailable hours
6	Course with most number of unavailable hours first, if equal more students
7	Course in curricula with most courses first, next courses with more students
8	Course in curricula with most unavailable hours first, next more students
9	Courses with highest number of lecture hours
10	Course with highest number of minimum working days, if equal more lecture hours

Table 2. *Room ordering heuristics*

Id	Room ordering heuristics
1	Largest Room First
2	Smallest Room First
3	Average Room First
4	Best Fitting (with enough capacity) Room

C. THE PROPOSED GREEDY ALGORITHM (GREEDY-CB-CTT)

The Greedy-CB-CTT algorithm uses the heuristics given above. Nrooms , Ncourses , and Ncurriculum depict the number of available classrooms, number of courses to be scheduled, and number of curricula with a set of courses, respectively. They are the input data of the algorithm. The output of the algorithm is the list of courses assigned to the classrooms. Mainly, the algorithm works with two main nested loops of rooms and the courses. The ordering heuristics of the classrooms and courses are selected with 40 different methods because four classroom and ten-course ordering alternatives exist. Inside these two loops, available classrooms are selected according to the minimum hard and soft violation values of the assignment choices. We define 120 new heuristics. Ten course ordering heuristics combined with four room ordering heuristics makes a total of 40 distinct heuristics. Each of these 40 heuristics is used in three higher-level heuristics, Heur-1, Heur-2, and Heur-3, which use different techniques to group lectures of courses into minimum working days and curricula.The pseudocode of the Greedy-CB-CTT is given in Algorithm 2.

Algorithm 1: *AssignDayPeriod(RID,CID,LH) Pseudocode for finding a (Day, Period) for given Course and Room*

```
Input:
  RID : see if room has available (day,period)
  CID : see if course can be assigned (day,period)
  LH : number of lecture hours to be assigned Output: (day,period) on SUCCESS, or FAILURE

for each day in Range(NumberOfDays) do
    for each period in Range(NumberOfPeriods − LH) do
        if (day,period)∈ DRoom[rid].Assigned then
            continue # try next period
        if (day,period)∈ DCourse[cid].UnavailableSet then
            continue # try next period
        if (day,period)∈ DCourse[cid].Assigned then
            continue # try next period
        if (day,period) ∈ DTeacher[Tid].Assigned then
            continue # try next period
        if (day,period)∈ ) DCurriculum[Curid]].Assigned then
            continue # try next period
        for hour in Range(LH) do
            add (day,period+hour) to DCourse[cid].Assigned
            add (day,period+hour) to DCurriculum[Curid]].Assigned
            add (day,period+hour) to DTeacher[Tid].Assigned
            add (day,period+hour) to DRoom[rid].Assigned
        Return SUCCESS_Day_Period
Return FAIL
```

The AssignDayPeriod 1 Algorithm 1 function uses the given Room and Course identifiers to determine a suitable (Day,Period) to schedule a lecture that doesn't violate any of the hard constraints. It is used in all of the heuristic Algorithms 2-4 as it is a shared functionality. MAXPERIOD is the number of lecture slots available on all days, which is six or seven for COMP benchmark but can be defined by problem instance as a dynamic parameter. Algorithm 2 is a naive method that assigns lectures of a course to the first available (Day,Period) slot without attempting to divide lectures of a course to minimum working days. Since the restriction for minimum working days is not enforced, this heuristic must find feasible solutions more easily and acts as a lower bound. Algorithm 3 is intelligent that first decides how to divide the total lecture hours of a course into the designated minimum working days for that course. Then, it calls the AssignDayPeriod to assign lectures of a course to a particular day. Since minimum working days is a soft constraint, the solutions returned by this algorithm will have lower penalty scores compared to Algorithm 2.

Algorithm 2: *Pseudocode of the proposed HEUR-1 Greedy Algorithm*

```
1 Input: RoomList ordered list of classrooms
  CourseList : ordered list of courses to be scheduled
  DCourse['course − id'] : Assigned: set-of-assigned-day-periods ; Curricula: set-of-curricula
  containing that course; Unav: set-of-unavailable-day-periods
  Output: The classroom and (day,period) assigned to a course. The resulting Hard and Soft
  constraint violation reasons and calculated penalties.
2 DCourse['course − id'].Assigned = NULLSET
  DRoom['room − id'].Assigned = NULLSET
  DCurriculum['curr − id'].Assigned = NULLSET
  for (cid in CourseList) do
3   for (rid in RoomList) do
4     LH= DCourse[cid].LectureHours
      for hour in Range(LH) do
5       AssignDayPeriod(rid,cid,1) # assign 1 hour at a time
6 Return DCourse[].Assigned # (day,period) sets for all courses
```

Algorithm 3: *Pseudocode of the proposed MinWorkingDays HEUR-2 Greedy Algorithm*

```
1 Input:
  RoomList ordered list of classrooms
  CourseList : ordered list of courses to be scheduled
  DCourse['course-id']:
  Assigned: set-of-assigned-day-periods
  Curricula: set-of-curricula with that course
  Unavailable: set-of-unavailable-day-periods
  Output: The classroom and (day,period) assigned to a course. The resulting Hard and Soft
  constraint violation reasons and calculated penalties.
2 DCourse['course − id'].Assigned = NULLSET
  DRoom['room − id'].Assigned = NULLSET
  DCurriculum['curr − id'].Assigned = NULLSET
  for (cid in CourseList) do
3   PARTS= DivideIntoMinWorkingDays(DCourse[cid].LectureHours)
    for (rid in RoomList) do
4     for (partHours in PARTS) do
5       AssignDayPeriod(rid,cid,partHours)
6 Return DCourse[].Assigned # (day,period) sets for all courses
```

Algorithm 4: *Pseudocode of the proposed Curriculum-first based HEUR-3 Greedy Algorithm*

```
1 Input:
  RoomList ordered list of classrooms
  CurriculumList : ordered list of Curriculums
  DCourse['course-id']:
  Assigned, Curricula, Unavailable: as above
  Output: The classroom and (day,period) assigned to a course. The resulting Hard and Soft
  constraint violation reasons and calculated penalties.
2 DCourse['course − id'].Assigned = NULLSET
  DRoom['room − id'].Assigned = NULLSET
  DCurriculum['curr − id'].Assigned = NULLSET
  FINISHED = NULLSET
  for (curid in OrderCurriculums(CurriculumList)) do
3   for (cid in OrderCourses(DCurriculum[curid].CourseList)) do
4     if cid ∈ FINISHED then
5       continue
6     PARTS= DivideIntoMinWorkingDays( DCourse[cid].LectureHours for (rid in RoomList) do
7       for (partHours in PARTS) do
8         AssignDayPeriod(rid,cid,partHours)
9 Return DCourse[].Assigned # (day,period) sets for all courses
```

Algorithm 4 exploits the Curriculum-based structure of the CB-CTT model. Another important soft constraint that must be minimized in CB-CTT benchmark instances is to ensure that the lectures of courses belonging to the same Curriculum must be scheduled on the same day, and if possible, the lectures of all courses in the same curriculum must be scheduled consecutively. Since this heuristic schedules courses in the same curriculum in a bundle, it attempts to find a day with enough free hours and can accommodate all of the lectures of the same Curriculum courses assigned to that day. Of course, the total lecture hours of Curriculum courses assigned to a single day must be at most MAXPERIOD, which is the maximum available lecture slots on a single day.

In this part, we explain the theoretical complexity analysis of our proposed algorithms. The results can be achieved in a very short time for all of the 21 benchmark problem instances used in ITC-2007. The proposed algorithm is greedy and its best advantage is that it produces reasonable solutions in polynomial times. The steps of our heuristics are given below. *NC* is the number of Courses and *NR* is the number of Rooms.

Step 1: Sort Courses according to heuristic course order: O(*Nlog N*) where *N* is the number of courses.
Step 2: Sort Rooms according to heuristic room order: O(*Nlog N*), where *N* is the number of rooms.
Step 3: for each Course
Step 4: for each Room
Step 5: find (day, period) such that all hard constraints are satisfied: O(1) × O(1)

Steps 1 and 2 are executed in O(*N logN*) because they require sorting. Steps 3, 4, 5 are O(*NC* × *NR* × *NDays* × *NPeriods*) since each course is considered once, and for each Course each Room is again considered only once, and *NDays* × *NPeriods* is also a constant value. Verifying hard constraints is O(1) because they involve a simple hash array lookup. The overall time complexity becomes: O(*NClogNC*) + O(*NRlogNR*) + O(*NC* × *NR*) × O(1). The running time of our heuristics are equivalent to sorting algorithms where problem size is defined by the larger number of courses (*NC*) and the number of rooms (*NR*). However, if *NC* and *NR* are comparable in size, then the third term O(*NCxNR*) becomes O(N^2).

V. THE EVALUATION OF EXPERIMENTAL RESULTS

We compare our algorithm with three classical greedy algorithms. Although our algorithm is not in the class of evolutionary algorithms, we provide a detailed comparison of our study with state-of-the-art evolutionary algorithms. The results of the evolutionary algorithms are better than our algorithm. However, since most evolutionary algorithms start with a population generated with greedy algorithms, we believe our algorithm can be effectively used in this area and significantly improve the convergence speed of the population-based algorithms. The validator of the competition committee of the ITC-2007 is used to verify our results [10]. Python programming language is used. The PC has eight GB of memory and an I7 2.4 GHz processor.

The total sum of hard and soft constraints is considered when comparing the proposed heuristics. The heuristic that outputs the minimum total sum is selected as the best-performing heuristic, whereas the largest total sum is the worst solution.

A. THE BENCHMARK PROBLEM INSTANCES

The details of the benchmark problem instances used in our experiments are given in Table 3 [10]. The range of the data provided in the columns is 30 to 121 for #courses, 5 to 20 #rooms, 5 to 6 #days, 5 to 6 #periods_per_day, #curricula varying from 13 to 150. It is called #constraints (the number of unavailable periods where a given course cannot be scheduled), ranging from 53 to 1368. comp01 is one of the easiest problem instances with the smallest number of courses, 30, only six classrooms, 14 curricula, and just 53 unavailable periods in the problem set. Two of the hardest problem instances are comp10 and comp12, which have 115 and 88 courses, 67 and 150 curricula, 18 and 11 rooms, 694 and 1368 constraints, respectively.

We give the average results of our proposed heuristics (Heur-1, Heur-2, Heur-3) in Table 4. Heur-1 obtained the maximum number of feasible solutions (6.2). The total sum of Heur-2 is the minimum (84238.1). The worst solutions are generally reported by Heur-3. The execution time of each heuristic is reported to be not more than a few milliseconds during the experiments.

Table 5 reports the best results and the algorithms that have obtained the best result. We obtained a feasible solution (the total sum of the hard constraints is zero) for 18 of 21 benchmark problem instances. The overall hard constraint value is observed to be 0.19 for 21 problem instances. We obtained one or two hard constraint violations for the problem instances that we have not reported any zero hard violation values. The total sum of hard and soft constraints is 867.62 and 867.81, respectively. Heur-1 were the best heuristic, with eight reported best results.

We present the solution of three classical greedy algorithms commonly used in the literature. Our main goal was to outperform these heuristics. The algorithms are the largest course to the largest room first, 1 the smallest course to the smallest room first, and the best-fit greedy algorithms. Table 6 gives the results of the largest course to the largest room first greedy algorithm. In Table 7, we give the results of the smallest course to the smallest room first greedy algorithm. Table 8 presents the result of the best-fit greedy algorithm that assigns the courses to the rooms with minimum empty spaces. No solution has been found among these three algorithms with zero hard violations. It can be observed that the largest course to largest room first algorithm is the one that reports the best solutions in average. This algorithm reports a total point of 713.4 for 21 problem instances. The smallest course to the smallest room first algorithm is the worst among the three algorithms, with the highest 3390.4 points. When a comparison is made with the algorithms we have developed, it can be seen that the performance of our algorithm is much better, with a general average score of 867.1 and the ability to find feasible solutions for 18 problem instances. The execution time of each algorithm was not more than ten milliseconds during the experiments.

Table 3. *The details of the ITC-2007 benchmark problem instances.*

problem	name	#courses	#rooms	#days	#periods/day	#curricula	#constraints
comp01	Fis0506-1	30	6	5	6	14	53
comp02	Ing0203-2	82	16	5	5	70	513
comp03	Ing0304-1	72	16	5	5	68	382
comp04	Ing0405-3	79	18	5	5	57	396
comp05	Let0405-1	54	9	6	6	139	771
comp06	Ing0506-1	108	18	5	5	70	632
comp07	Ing0607-2	131	20	5	5	77	667
comp08	Ing0607-3	86	18	5	5	61	478
comp09	Ing0304-3	76	18	5	5	75	405
comp10	Ing0405-2	115	18	5	5	67	694
comp11	Fis0506-2	30	5	5	9	13	94
comp12	Let0506-2	88	11	6	6	150	1368
comp13	Ing0506-3	82	19	5	5	66	468
comp14	Ing0708-1	85	17	5	5	60	486
comp15	Ing0203-1	72	16	5	5	68	382
comp16	Ing0607-1	108	20	5	5	71	518
comp17	Ing0405-1	99	17	5	5	70	548
comp18	Let0304-1	47	9	6	6	52	594
comp19	Ing0203-3	74	16	5	5	66	475
comp20	Ing0506-2	121	19	5	5	78	691
comp21	Ing0304-2	94	18	5	5	78	463

Table 4. *The average results obtained with Heur-1, Heur-2, and Heur-3 for 21 problem instances.*

ALG	Total Hard	Total Soft	Total Sum	Feasible Found	Min at	Max at
Heur-1	94.7	86471.4	86566.1	6.2	0.2	0.0
Heur-2	165.9	84072.2	84238.1	4.0	0.4	0.0
Heur-3	114.9	86759.6	86874.5	5.4	0.1	0.5

Table 5. *The best results and the algorithms that report the solutions.*

problem	ALG	Total Hard	Total Soft	Total Penalty
comp01	Heur-1: 1/5	0	234	234
comp02	Heur-2: 4/8	2	800	802
comp03	Heur-3: 4/1	0	600	600
comp04	Heur-2: 4/7	0	574	574
comp05	Heur-1: 4/8	0	1287	1287
comp06	Heur-3: 4/4	0	838	838
comp07	Heur-1: 1/1	0	1005	1005
comp08	Heur-3: 4/5	0	580	580
comp09	Heur-2: 1/5	0	629	629
comp10	Heur-3: 4/1	0	796	796
comp11	Heur-1: 4/3	0	190	190
comp12	Heur-2: 4/8	0	1129	1129
comp13	Heur-3: 4/3	0	612	612
comp14	Heur-2: 4/7	0	668	668
comp15	Heur-3: 4/1	0	600	600
comp16	Heur-1: 4/1	0	899	899
comp17	Heur-1: 1/8	0	2776	2776
comp18	Heur-2: 4/5	0	426	426
comp19	Heur-3: 4/4	1	607	608
comp20	Heur-1: 4/7	0	953	953
comp21	Heur-1: 1/6	1	2017	2018
Avg		0.19	867.62	867.81

Table 6. *The results of the greedy algorithm which matches the largest course, in terms of number of students, to the largest room.*

problem	HC1	HC2	HC3	HC4	SC1	SC2	SC3	SC4	Total
comp01	0	30	13	0	4	285	20	3	**312**
comp02	0	96	64	0	0	635	74	11	720
comp03	0	56	59	0	0	570	76	6	652
comp04	0	58	44	0	0	515	92	10	617
comp05	0	47	65	0	0	405	230	3	638
comp06	0	105	82	0	0	785	114	9	908
comp07	0	144	84	0	0	900	178	15	1093
comp08	0	52	71	0	0	555	74	7	636
comp09	0	86	60	0	0	585	56	6	647
comp10	0	117	67	0	0	795	122	11	928
comp11	0	18	8	0	0	260	12	2	274
comp12	0	55	97	0	0	555	244	4	803
comp13	0	56	61	0	0	540	84	6	630
comp14	0	90	82	0	0	675	94	7	776
comp15	0	56	59	0	0	570	76	6	652
comp16	0	105	70	0	0	775	110	10	895
comp17	0	82	81	0	0	725	108	8	84
comp18	0	31	44	0	0	425	48	1	474
comp19	0	101	82	0	0	595	96	9	700
comp20	0	98	86	0	0	850	146	9	1005
comp21	0	86	38	0	0	665	106	9	780
total	0	1569	1317	0	4	12665	2160	152	14981
Avg	0	74.7	62.7	0	0.2	603.1	102.9	7.2	713.4

Table 7. *The results of the greedy algorithm which matches the smallest course to the smallest room first.*

problem	HC1	HC2	HC3	HC4	SC1	SC2	SC3	SC4	Total
comp01	0	15	14	0	483	270	1	4	**769**
comp02	0	101	66	0	3483	635	124	8	4250
comp03	0	64	57	0	3247	555	122	7	3931
comp04	0	62	56	0	5496	540	64	6	6106
comp05	0	53	62	0	10412	360	310	3	11085
comp06	0	102	85	0	852	780	144	11	1787
comp07	0	114	77	0	344	915	116	14	1389
comp08	0	63	64	0	3013	550	86	9	3658
comp09	0	83	72	0	3477	540	108	7	4132
comp10	0	126	97	0	204	795	98	10	1107
comp11	0	21	7	0	1306	260	12	3	1581
comp12	0	52	87	0	2577	555	224	1	3357
comp13	0	64	67	0	6342	525	94	9	6970
comp14	0	83	73	0	3098	675	92	7	3872
comp15	0	64	57	0	3247	555	122	7	3931
comp16	0	112	61	0	1746	785	116	9	2656
comp17	0	77	91	0	349	720	158	11	1238
comp18	0	17	51	0	2122	410	56	1	2589
comp19	0	93	84	0	1257	570	108	8	1943
comp20	0	125	76	0	2347	840	166	11	3364
comp21	0	90	48	0	673	665	138	8	1484
total	0	1581	1352	0	56075	12500	2470	154	71199
Avg	0	75.3	64.4	0	2670.2	595.2	117.6	7.3	3390.4

Table 8. *The results of of the best-fit greedy algorithm.*

problem	HC1	HC2	HC3	HC4	SC1	SC2	SC3	SC4	Total
comp01	2	26	13	0	88	300	12	3	403
comp02	1	81	76	0	0	655	104	4	763
comp03	1	70	64	0	0	555	78	6	639
comp04	0	53	49	0	0	520	80	8	808
comp05	0	67	45	0	0	430	186	0	616
comp06	0	97	66	0	0	765	116	9	890
comp07	1	167	79	0	0	895	162	9	1066
comp08	1	68	68	0	0	560	78	10	648
comp09	0	62	65	0	0	540	94	4	638
comp10	2	128	76	0	0	805	102	9	916
comp11	0	35	7	0	0	270	0	1	**271**
comp12	0	90	77	0	0	560	234	3	979
comp13	0	78	59	0	0	530	70	7	607
comp14	0	66	76	0	0	675	118	6	799
comp15	1	70	64	0	0	555	78	6	639
comp16	0	112	63	0	0	775	120	8	903
comp17	3	101	72	0	0	725	126	8	859
comp18	0	57	32	0	0	440	20	0	460
comp19	2	92	52	0	0	595	74	6	675
comp20	0	126	92	0	0	835	144	8	987
comp21	0	140	31	0	0	675	136	10	821
total	14	1786	1226	0	88	12660	2132	125	15205
Avg	0.7	85.0	58.4	0	4.2	602.9	101.5	5.9	724.0

Table 9. *Competition results of ITC-2007; the best discovered results on all the 21 competition instances (given in boldface).*

problem	Müller	Lü&Hao	Atsuta	Geiger	Clark	Batuhan
comp01	5	5	5	5	10	**234**
comp02	51	55	50	111	111	802
comp03	84	71	82	128	119	600
comp04	37	43	35	72	72	574
comp05	330	309	312	410	426	1287
comp06	48	53	69	100	130	838
comp07	20	28	42	57	110	1005
comp08	41	49	40	77	83	580
comp09	109	105	110	150	139	629
comp10	16	21	27	71	85	796
comp11	0	0	0	0	3	190
comp12	333	343	351	442	408	1129
comp13	66	73	68	622	113	612
comp14	59	57	59	90	84	668
comp15	84	71	82	128	119	600
comp16	34	39	40	81	84	899
comp17	83	91	102	124	152	2776
comp18	83	69	68	116	110	426
comp19	62	65	75	107	111	608
comp20	27	47	61	88	144	953
comp21	103	106	123	174	169	2018
Avg	79.8	80.9	85.8	180	132.5	916

B. COMPARISON WITH STATE-OF-THE-ART ALGORITHMS

We compare the experimental performance results of our algorithms with state-of-the-art algorithms in the literature. Although our algorithms are not in this class of algorithms, we think these results can give a good understanding of the recent studies about the CB-CTT problem. In Table 9, we can compare the best-known results reported in the literature with those discovered by our algorithms. The algorithms we compare against our solutions are obtained from recent papers [22, 23, 38–40]. The total scores of the results on 21 problems are presented. The best schedules are given in bold digits. The best algorithms in the literature are primarily evolutionary and require much time to converge, whereas our algorithms spend only a few millisecond optimization time.

VI. CONCLUSION AND FUTURE WORK

We outperformed classical greedy algorithms, the largest course to the largest room first, the smallest course to the smallest room first, and the best-fit greedy algorithms. The performance of our algorithm is better, with a general average score of 867.80 and finding feasible solutions for all problem instances. The most important advantage of our algorithm is that it is much faster than evolutionary approaches and works well with even large problem instances. While the evolutionary algorithms spend hours of computation time, our algorithm can get feasible solutions in a few milliseconds. The No Free Lunch Theorem (NFL) [7] tells us that there will always be new ideas and approaches leading to better optimization algorithms to solve a given problem. Instead of being forgotten in a short time, it is far more likely that most of the currently known optimization methods have at least one niche, one area where they are excellent. It has been experimentally shown that greedy heuristics can help eliminate hard constraint violations in CB-CTT. These results verify that it might as well be possible to find new greedy algorithms to eliminate at least a substantial portion of soft constraint violations. Further research is needed to determine which greedy heuristic would perform better on a given CB-CTT problem instance. New benchmark problem instances are also being introduced. It can be interesting to apply our proposed algorithm to these new problem sets and observe the results, justifying our experimental findings on different benchmark datasets.

VII. REFERENCES

[1] P. Michael, and K. Hadavi, "Scheduling: theory, algorithms and systems development," *Operations research proceedings*, Berlin, 1992, pp. 35-42.

[2] W. Ruegg, *A history of the university in Europe: Volume 3, universities in the nineteenth and early twentieth centuries*, vol. 3, Cambridge University Press, 2004, pp.1800-1945.

[3] D. Ronald, and V.N. Temlyakov, "Some remarks on greedy algorithms," *Advances in computational Mathematics*, vol. 5, pp. 173-187, 1996.

[4] A.R. Barron, A. Cohen, W. Dahmen, RA. DeVore, "Approximation and learning by greedy algorithms," *The annals of statistics*, vol. 36, no. 1, pp. 64-94. 2008.

[5] A. Bettinelli, V. Cacchiani, R. Roberti, and P. "Toth An overview of curriculum-based course timetabling," *Top*, vol. 23, no. 2, pp. 313-349, 2015.

[6] E.K. Burke, M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and R. Qu, "Hyper-heuristics: A survey of the state of the art," *Journal of the Operational Research Society*, vol. 64, no. 12, pp. 1695-1724. 2013.

[7] D.H. Wolpert, and W.G. Macready, "No free lunch theorems for optimization," *IEEE transactions on evolutionary computation,* vol. 1, no. 1, pp. 67-82, 1997.

[8] G. Dosa, J. Sgall, "Optimal analysis of best bin packing," *International Colloquium on Automata, Languages, and Programming,* 2014, pp.429-441.

[9] K. Fleszar, and C. Charalambous, "Average-weight-controlled bin-oriented heuristics for the one-dimensional bin-packing problem," *European Journal of Operational Research,* vol. 210, no. 2, pp. 176-184, 2011.

[10] G.L. Di, B. McCollum, and A. Schaerf, "The second international timetabling competition (ITC-2007): Curriculum-based course timetabling (track 3)," Technical Report 1.0, Queen's University, Belfast, United Kingdom, 2007.

[11] P.I. Tillett, "An operations research approach to the assignment of teachers to courses," *Socio-Economic Planning Sciences,* vol. 9, no. 3-4, pp. 101-104, 1975.

[12] 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.

[13] S. Abdullah, H. Turabieh, B. McCollum, and P. McMullan, "A hybrid metaheuristic approach to the university course timetabling problem," *Journal of Heuristics,* vol. 18, no. 1, pp. 1-23, 2012.

[14] I. Boussaïd, J. Lepagnot, and P. Siarry, "A survey on optimization metaheuristics," *Information sciences,* vol. 237, pp. 82-117, 2013.

[15] T. Dokeroglu, E. Sevinc, T. Kucukyilmaz, and A. Cosar, "A survey on new generation metaheuristic algorithms," *Computers & Industrial Engineering,* vol. 137, no. 106040, 2019.

[16] S.A. MirHassani, and F. Habibi, "Solution approaches to the course timetabling problem," *Artificial Intelligence Review,* vol. 39, no. 2, pp. 133-149, 2013.

[17] A. Gary, and C. Robert, "Matching Faculty to Courses," *College and University,* vol. 46, pp. 83-89, 1971.

[18] J.S. Dyer, and J.M. Mulvey, "An integrated optimization/information system for academic departmental planning," *Management Science,* vol. 22, no. 12, pp. 1332-1341, 1976.

[19] W. Shih, and J.A. Sullivan, "Dynamic course scheduling for college faculty via zero-one programming," *Decision Sciences,* vol. 8, no. 4, pp. 711-721, 1977.

[20] N.Christian, F. Bagger, S. Kristiansen, M. Sørensen, and T.R. Stidsen, "Flow formulations for curriculum-based course timetabling," *Annals of Operations Research,* vol. 280, no. 1, pp. 121-150, 2019.

[21] A.E. Phillips, C.G. Walker, M. Ehrgott, and D.M. Ryan, "Integer programming for minimal perturbation problems in university course timetabling," *Annals of Operations Research,* vol. 252, no. 2, pp. 283-304, 2017.

[22] M.J. Geiger, "Applying the threshold accepting metaheuristic to curriculum based course timetabling," *Annals of Operations Research,* vol. 194, no.1, pp. 189-202, 2012.

[23] Z. Lu, and J.K. Hao, "Adaptive tabu search for course timetabling," *European journal of operational research,* vol. 200, no. 1, pp. 235-244, 2010.

[24] T. Dokeroglu, and E. Sevinc, "Memetic Teaching–Learning-Based Optimization algorithms for large graph coloring problems," *Engineering Applications of Artificial Intelligence*, vol. 102, no. 104282, 2021.

[25] A. Gulcu, and C. Akkan, "Robust university course timetabling problem subject to single and multiple disruptions," *European Journal of Operational Research*, vol. 283, no. 1, pp. 630-646., 2020.

[26] C. Akkan and A. Gulcu, "A bi-criteria hybrid Genetic Algorithm with robustness objective for the course timetabling problem," *Computers and Operations Research*, vol. 90, pp. 22-32, 2018.

[27] T. Thepphakorn, and P. Pongcharoen, "Variants and parameters investigations of particle swarm optimisation for solving course timetabling problems," *International Conference on Swarm Intelligence*, 2019, pp. 177-187.

[28] S. LengGoh, G. Kendall, and N.R. Sabar, "Improved local search approaches to solve the post enrolment course timetabling problem," *European Journal of Operational Research*, vol. 261, no. 1, pp. 17-29., 2017.

[29] N.C.F. Bagger, M. Sorensen, and TR. Stidsen, "Benders' decomposition for curriculum-based course timetabling," *Computers and Operations Research*, vol. 91, pp. 178-189, 2018.

[30] T. Song, S. Liu, X. Tang, X. Peng, and M. Chen, "An iterated local search algorithm for the University Course Timetabling Problem," *Applied Soft Computing*, vol. 68, pp. 597-608, 2018.

[31] A.De Coster, N.Musliu, A.Schaerf, J.Schoisswohl, and K.Smith-Miles, "Algorithm selection and instance space analysis for curriculum-based course timetabling," *Journal of Scheduling*, vol. *25, no* 1, pp. 35-58, 2022.

[32] C.Akkan, A.Gülcü, and Z.Kuş, "Bi-criteria simulated annealing for the curriculum-based course timetabling problem with robustness approximation," *Journal of Scheduling*, pp. 1-25, 2022.

[33] G.Colajanni, and P.Daniele, "A new model for curriculum-based university course timetabling," Optimization Letters, vol. 15, no 5, pp. 1601-1616., 2021.

[34] H. Asmuni, "Fuzzy methodologies for automated university timetabling solution construction and evaluation," Ph.D. dissertation, University of Nottingham, United Kingdom, 2008.

[35] J.H. Obit, "Developing novel meta-heuristic, hyper-heuristic and cooperative search for course timetabling problems," Ph.D, University of Nottingham, United Kingdom, 2010.

[36] T.A. Redl, "A study of university timetabling that blends graph coloring with the satisfaction of various essential and preferential conditions," Ph.D., Rice University Houston, USA, 2004.

[37] B.M. Cosar, "New greedy algorithms to optimize the curriculum-based course timetabling problem, " M.S thesis, Atilim University, Ankara, Turkey, 2021.

[38] T. Muller, "ITC2007 solver description: a hybrid approach," *Annals of Operations Research,* vol. 172, no. 1, pp. 429-446, 2009.

[39] M. Atsuta, K. Nonobe, and T. Ibaraki, "ITC-2007 Track2: an approach using general CSP solver," Citeseer, 2008.

[40] M. Clark, M. Henz, and B. Love, "Quikfix—a repair-based timetable solver," *The Seventh International Conference on the Practice and Theory of Automated Timetabling*," Citeseer, 2008.