Asia-Pacific Journal of Information Technology and Multimedia 14(2) 2025: 525 – 547
https://doi.org/10.17576/apjitm-2025-1402-12

A Context-Aware Constructive Heuristic for Highly-Constrained Room Allocation Problem in Examination Timetabling

Heuristik Konstrukif Peka-Konteks untuk Masalah Peruntukan Bilik yang Sangat Terhad dalam Penjadualan Peperiksaan

Ahmad Abba Datti, Abdulwahab Lawan, Ibrahim Said Ahmad[*]

Bayero University Kano, Nigeria

[*]*Corresponding author: isahmad.it@buk.edu.ng*

ABSTRACT

Examination timetabling is a complex optimization problem with significant implications for student well-being and institutional efficiency. Traditional room allocation methods prioritize capacity but often ignore spatial constraints, leading to long-distance student movement particularly problematic in multi-campus institutions. This paper introduces a context-aware constructive heuristic, termed Closest-Room-First (CRF), which assigns students to exam rooms based on proximity to their departmental buildings. The heuristic is used to generate an initial population of solutions that are subsequently evolved using a tailored implementation of the Non-Dominated Sorting Genetic Algorithm II (NSGA-II). Although this study focuses on minimizing student travel distance as a single objective, NSGA-II was chosen for its extensibility and proven utility in constraint-driven single-objective optimization. Experimental results on nine Purdue benchmarks and a real-world dataset obtained from Bayero University Kano show CRF achieves >80% travel cost reductions compared to Random, Largest Room First, and Best Fit heuristics. All results were statistically validated using the Mann-Whitney U test ($p < 0.001$). Despite higher runtimes, CRF's trade-off is justified in constrained settings. These findings underscore the educational and logistical value of embedding spatial sensitivity in exam timetabling.

Keywords**:** Exam Timetabling, Room Allocation, Constructive Heuristics, Genetic Algorithms, Context-Aware Algorithms.

ABSTRAK

Penjadualan peperiksaan merupakan masalah pengoptimuman yang kompleks dengan implikasi besar terhadap kesejahteraan pelajar dan kecekapan institusi. Kaedah peruntukan bilik tradisional lazimnya mengutamakan kapasiti tetapi sering mengabaikan kekangan ruang, sekali gus menyebabkan pergerakan pelajar jarak jauh—isu yang lebih ketara dalam institusi berbilang kampus. Makalah ini memperkenalkan satu heuristik konstrukif peka-konteks, dinamakan *Closest-Room-First* (CRF), yang memperuntukkan pelajar ke bilik peperiksaan berdasarkan jarak paling hampir dengan bangunan jabatan masing-masing. Heuristik ini digunakan untuk menjana populasi awal penyelesaian yang kemudiannya berkembang melalui

pelaksanaan tersuai *Non-Dominated Sorting Genetic Algorithm II* (NSGA-II). Walaupun kajian ini memfokuskan kepada peminimuman jarak perjalanan pelajar sebagai satu objektif, NSGA-II dipilih kerana sifatnya yang mudah diperluas serta keberkesanan yang terbukti dalam pengoptimuman objektif tunggal berasaskan kekangan. Keputusan eksperimen ke atas sembilan penanda aras Purdue dan satu set data dunia sebenar daripada Universiti Bayero Kano menunjukkan bahawa CRF mencapai pengurangan kos perjalanan melebihi 80% berbanding heuristik Rawak, *Largest Room First*, dan *Best Fit*. Semua keputusan disahkan secara statistik menggunakan ujian Mann-Whitney U ($p < 0.001$). Walaupun masa pemprosesan lebih tinggi, pertukaran yang ditawarkan CRF adalah wajar dalam persekitaran yang sangat terhad. Penemuan ini menekankan nilai pendidikan dan logistik dalam menerapkan kepekaan ruang dalam penjadualan peperiksaan.

Kata kunci: Penjadualan Peperiksaan, Peruntukan Bilik, Heuristik Konstrukif, Algoritma Genetik, Algoritma Peka-Konteks.

INTRODUCTION

Examination timetabling is a critical logistical task faced by educational institutions worldwide. It involves assigning a large number of exams to specific timeslots and rooms while satisfying a complex mix of hard constraints (e.g., no student can take two exams at the same time) and soft preferences (e.g., spreading difficult exams) (E. K. Burke et al., 2006; McCollum & Mcmullan, 2007). Despite extensive research on improving solution quality and computational speed (Adnan et al., 2018; Bashab et al., 2023; Gashgari et al., 2018; Petrovic & Bykov, 2011; Vrielink, 2017; Yousef et al., 2016), many real-world concerns—especially those affecting students directly—remain under-addressed (Abdelhalim & El Khayat, 2016; Ceschia et al., 2022; Goulas & Megalokonomou, 2018).

A notable omission in much of the literature is the consideration of student movement across campus locations. Most room allocation methods are designed to optimize room utilization or minimize the number of unused seats (Aizam et al., 2016; E. Burke et al., 1996; Demeester et al., 2012; Kendall & Hussin, 2005; Mandal et al., 2020; Matias, 2018), without accounting for the physical distance students must travel. This issue is particularly critical in multi-campus institutions, where long walking distances between venues can lead to fatigue, logistical stress, and diminished exam readiness (Bashab et al., 2023; Carlsson et al., 2023; Gu et al., 2025). Existing room allocation techniques—including exact methods such as integer linear programming (Al-Hawari et al., 2020), greedy algorithms, bin-packing heuristics (Soghier & Qu, 2013), and graph colouring approaches (Kubale, 2004) —tend to either ignore spatial constraints or treat room location as incidental. Some models that consider room usage, such as the flex-deluge approach (E. K. Burke & Bykov, 2016) or the N-subset variant (Saharan & Kadian, 2014), still focus primarily on minimizing room underutilization rather than student burden. Furthermore, constructive heuristics for room assignment are often random, without incorporating any context-awareness about student location (De La Rosa-Rivera et al., 2021; Ghaffar et al., 2025; Liu et al., 2023).

This paper addresses these limitations by introducing a context-aware constructive heuristic, termed Closest-Room-First (CRF), that explicitly minimizes student travel distance. The CRF algorithm groups students by department and allocates rooms in increasing proximity to departmental buildings, which serve as logical spatial anchors. Unlike conventional heuristics, CRF assigns students to the closest available room first, thereby reducing physical and psychological strain. Additionally, a lightweight Room Shuffle Mutation operator that perturbs

room assignments during evolution was introduced, encouraging the search algorithm to explore alternate configurations. These components are integrated into an evolutionary optimization framework using NSGA-II—not for its multi-objective ranking, but for its robustness and flexibility in accommodating constraint-rich solution spaces. By embedding location sensitivity directly into the constructive and evolutionary stages of exam timetabling, this research provides a student-centric improvement to scheduling systems—particularly in constrained, multi-campus environments where spatial logistics play a pivotal role.

RELATED WORK

Room allocation in examination timetabling has traditionally been treated as a subproblem focused primarily on optimizing seat usage, minimizing the number of unused rooms or unutilized capacity (Aizam et al., 2016; Ayob et al., 2007; E. K. Burke et al., 1996; Demeester et al., 2012; Leite et al., 2018). These approaches, while effective in addressing room occupancy, often ignore the logistical challenges posed by the physical layout of campuses, especially in multi-building or multi-campus institutions.

Heuristic and metaheuristic strategies have been widely employed for this task, including bin-packing algorithms (Soghier & Qu, 2013), graph colouring (Kubale, 2004), and local search techniques such as Variable Neighbourhood Search (Elloumi et al., 2014). A more recent comprehensive review of educational timetabling highlights the persistent dominance of capacity-driven formulations, despite increasing demand for more context-sensitive models (Ceschia et al., 2022; Tan et al., 2021). Even in works addressing fairness or soft constraint balancing, spatial considerations remain largely unformalized (Eldharif et al., 2024; Jáhn-Erdős & Kővári, 2024). Some efforts have modelled room assignment using subset-sum or knapsack abstractions (Saharan & Kadian, 2014; Tuniki et al., 2020), which enforce room capacity limits but do not capture geographical concerns. A more recent stream of research has explored multi-objective optimization frameworks for timetabling using NSGA-II and other evolutionary algorithms (Abraham et al., 2025; Alabbadi & Abulkhair, 2021; Ma et al., 2023; Ngo et al., 2021; Shen & Xie, 2025), but these typically target trade-offs between fairness, period compactness, or exam spread—not spatial proximity or travel effort. Even when room assignments are optimized in parallel with timeslots (Muklason et al., 2017), the room component is typically capacity-driven. Notably, prior work like Burke and Bykov's (E. K. Burke & Bykov, 2016) flex-deluge strategy and others employing greedy heuristics still treat location as incidental.

To the best of our knowledge, no previous heuristic explicitly prioritizes student location awareness by using proximity to departmental buildings as a formal basis for room allocation. This work contributes by embedding such context-awareness directly into the room assignment process via a domain-specific constructive heuristic and corresponding evolutionary mutation strategy.

METHODOLOGY

This study focuses on the room allocation subproblem within the broader examination timetabling process. It was assumed that a feasible timeslot schedule has already been generated using standard institutional practices, and that the objective now is to assign students to appropriate exam rooms while minimizing logistical burden—specifically, the distance between a student's department and their assigned exam room.

While spatial and temporal constraints may conflict in joint optimization models, they were deliberately decoupled in this work. This staged separation reflects standard operational practice in university examination timetabling, where the timeslot schedule is finalized before room allocation due to administrative workflows and the need to first resolve student scheduling conflicts (E. K. Burke et al., 1994; Ceschia et al., 2022). In such systems, the only case where time and spatial optimization could conflict is when a student has two back-to-back exams in distant locations—the farther apart, the worse the schedule. However, this situation is typically addressed at the time allocation stage through established soft constraints that penalize consecutive exams or enforce buffer periods (Gashgari et al., 2018). These proximity penalties, widely used in both academic practice and timetabling benchmarks, aim to spread each student's exams apart in time, reducing the likelihood of consecutive, long-distance transitions (Ceschia et al., 2022). Because such temporal–spatial conflicts are already minimized during timeslot scheduling, it is both logical and efficient to treat room allocation as an independent second phase, focusing purely on venue assignment and capacity without altering the fixed exam times. This two-phase approach is well-documented in educational timetabling literature and case studies (E. K. Burke et al., 1994; Ceschia et al., 2022; Tan et al., 2021) and aligns with how real institutions manage large-scale exam schedules.

The goal of the proposed model—called the Location-Sensitive Model—is to minimize average student travel distance across all exams and rooms, while satisfying capacity constraints. Each student is assigned a fixed departmental location as a reference point, and rooms are allocated in a way that minimizes the total distance from these origins to assigned exam venues. This formulation aligns with real-world logistical needs in multi-campus settings, where reducing physical movement improves student comfort and performance.

The following subsections present the mathematical model, the derived fitness function, and the underlying assumptions supporting this spatial optimization framework.

OPTIMIZATION FRAMEWORK

The computational framework for the room allocation problem is defined as follows:

$$f = \frac{1}{S}\sum_{s\in S}\sum_{i\in E}\sum_{o\in O}\sum_{r\in R} a_{si}a_{so}a_{sr}d_{or} \tag{1}$$

Subject to

$$\sum_{s\in S} a_{sr} \leq s_r \quad \forall\, r \in R \tag{2}$$

$$\sum_{r\in R} a_{sr} = s_r \sum_{i\in E} a_{si} * a_{sr} \quad \forall\, s \in S \tag{3}$$

Parameters

S: Total number of students.

E: Total number of exams.

R: Total number of rooms.

P: Total number of timeslots (periods).

s: Individual Student.

i: Individual Exam.
r: Individual Room.
p: Individual Period.
s_i: Number of students enrolled in exams i.
s_r: Number of seats in room r.
d_{or}: Distance between origin (department) o and room r obtained from an $o \times r$ distance matrix d.

Decision Variables

a_{ir}: 1 if exam i has been allocated to room r, 0 otherwise.
a_{ip}: 1 if exam i has been allocated to period p, 0 otherwise.
a_{si}: 1 if student s has enrolled in exam i, 0 otherwise.
a_{sr}: 1 if student s has been scheduled in room r, 0 otherwise.
a_{so}: 1 if student s is from department o, 0 otherwise.

The objective function (Equation 1) serves as a cost metric that computes the average distance traveled by students, taking into account their department affiliations and assigned exam rooms. Equation 2 enforces the constraint that the number of students assigned to a room must not exceed its capacity. Equation 3 ensures that all students are allocated to exactly one room per exam they are enrolled in.

FITNESS FUNCTION

Rather than the model itself, the primary contribution in this section lies in the fitness functions derived from this model. This fitness function drives the optimization process, guiding the algorithms—such as NSGAII—to generate solutions that respect location constraints. By evaluating the quality of solutions based on the difficulty and location parameters, the fitness function ensure that the resulting timetables are optimized for logistical feasibility and student comfort. The fitness function (Figure 1) aims to minimize the overall student travel distance while adhering to strict capacity and assignment constraints.

Algorithm 1 Movement Fitness Function

Require: $solution, distanceMatrix, studentList$

```
function MOVEMENT FITNESS(solution, distanceMatrix)
totalDistance ⇐ 0
for student in studentList do
    for exam in student.examList do
        room ⇐ getAssignedRoom(exam, solution)
        department ⇐ getDepartment(student)
        distance ⇐ distanceMatrix[department][room]
        totalDistance ⇐ totalDistance + distance
    end for
end for
return totalDistance
```

FIGURE 1. Movement Fitness Function

The provided function calculates the total distance travelled by students in an exam timetable. It iterates through each student, considering every exam they are enrolled in. For each exam, the function determines the assigned room in the given solution. Subsequently, the department of the student and the room are used as indices to retrieve the corresponding distance from a pre-computed distance matrix. This distance is accumulated for each student and exam, resulting in a total distance value that is returned as the fitness function output.

Time Complexity of Fitness Function

Line 2 initializes *totalDistance* to 0. This is a constant time operation, $O(1)$. Line 3 starts a loop over each student in students. Let's assume there are S students, so this loop runs S times. Line 4 is a nested loop that iterates over each exam in the student's exams. Let E be the number of exams a student takes. Thus, the inner loop runs E times for each student. Line 5 retrieves the room assigned to an exam in constant time, $O(1)$. Line 6 retrieves the department of the student, also in $O(1)$. Line 7 looks up the distance from the department to the room in the *distanceMatrix*. This is a lookup operation in a matrix, which takes constant time, $O(1)$. Line 8 updates *totalDistance*, a constant time operation, $O(1)$. Line 11 returns the total distance, which is $O(1)$. Overall, the outer loop runs S times (once for each student). The inner loop runs E times for each student. Since each step inside the inner loop (Lines 5–8) takes constant time $O(1)$, the total time for the inner loop is $O(E)$. Therefore, the time complexity for each student is $O(E)$, and for S students, the overall complexity is $O(S \bullet E)$.

THE CLOSEST-ROOM-FIRST HEURISTIC

The Closest-Room-First Heuristic is designed to place students in rooms closest to their departments. It achieves this by prioritizing the allocation of exam rooms based on their proximity to a fixed location common to all students taking that exam, thus reducing travel time and associated fatigue.

The strategy employed is the utilization of a domain-specific constructive heuristic to create an initial population of solutions. These solutions will then be used as a starting point by the selected search algorithm with the aim of reducing the time needed to find an optimal solution.

For each exam, the algorithm (Figure 2 and Figure 3), starts by grouping the students that have registered into departments (lines 5 to 12). It then iterates through the departments (lines 15 to 28) and for each department, it sorts the available rooms by increasing proximity to that department (line 19) and then starts placing the students into the closest rooms (lines 20 to 27). The assignment moves to the next closest room when a room has been filled to capacity. The whole process is complete when all students have been allocated.

Time complexity of Closest-Room-First Heuristic

The initialization Steps (Lines 1-4) take constant time operations $O(1)$, so they do not affect the overall complexity significantly. Next is the Department and Enrolment Processing (Lines 5-12) having a complexity of $O(D \times E \times S)$. Setting Distance to Departments (Lines 16-18) has a complexity of $O(R \times D)$, where R represents rooms and D represents departments. Sorting rooms into a priority queue would take $O(RlogR)$ time complexity (Line 19). The Room

assignment (Lines 20-27) has a complexity of $O(E \times S)$, where E is exams, and S is students. Calculating the distance (Algorithm 3 Line 2) is assumed to be $O(1)$ since coordinates are directly available. In summary, the overall time complexity is:

- Enrolment Processing: $O\ (D \times E \times S)$
- Setting Distance to Departments: $O(R \times D)$
- Sorting Rooms: $O(RlogR)$
- Assigning Rooms to Students: $O(E \times S)$
-

Thus, the overall time complexity of the Closest-First Heuristic algorithm becomes:

- $O(D \times E \times S + R \times D + RlogR)$.

In practice, the term $O\ (D \times E \times S)$ will usually dominate if the number of students, exams, and departments are large.

Algorithm 2 Closest-Room-First Heuristic

Require: $Students, Exams, Timeslots, Depts, Rooms$

```
1:  S ⇐ sizeof(Students); E ⇐ sizeof(Exams); T ⇐ sizeof(Timeslots)
2:  D ⇐ sizeof(Depts)R ⇐ sizeof(Rooms)
3:  timetable                                          ▷ E by S matrix
4:  sortedRooms                                        ▷ Priority Queue
5:  for exam in Exams do
6:      for dept in Depts do
7:          for student in Students do
8:              if student.dept == dept then dept.enrollment(exam).add(student)
9:              end if
10:         end for
11:     end for
12: end for
13:
14:
15: for dept in Depts do
16:     for rooms in Rooms do
17:         setDistanceToDepartment(room, dept)
18:     end for
19:     SortedRooms ⇐ Rooms
20:     for student in dept in exam do
21:         currentRoom ⇐ sortedRooms.peek()
22:         if currentRoom.freeSeats ¿ 0 then
23:             student.room ⇐ currentRoom
24:         else
25:             sortedRooms.remove(room)
26:         end if
27:     end for
28: end for
```

Algorithm 3 Procedure setDistanceToDept

```
1: Procedure SETDISTANCTODEPT(dept, room)
2:     return gpsDistance(roomGPSCoords, deptGPScoords)
3: End Procedure
```

FIGURE 1. Closest-Room-First Heuristic

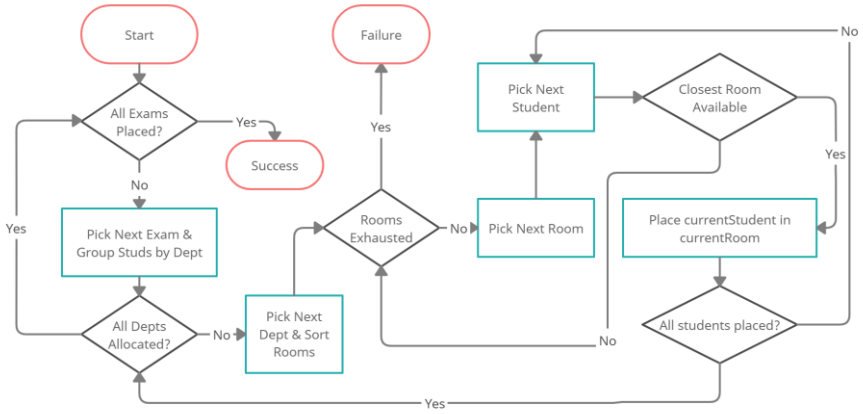

FIGURE 2. Flow Diagram of the Closest-Room-First Heuristic

MUTATION OPERATOR

To enhance solution diversity during the evolutionary search process, a lightweight operator called the Room Shuffle Mutation (Figure 4) was introduced. This mutation operator introduces controlled randomization into room assignments, helping the algorithm escape local optima and explore alternative configurations that may lead to better fitness. The operator works by selecting a solution from the current population and randomly shuffling the room assignments of exams, while maintaining feasibility. Specifically, for each exam, it reassigns its room to a different one randomly selected from the list of available rooms. Since exam-to-timeslot mappings are fixed, and room capacity constraints are enforced, the shuffled assignment remains valid.

This operator is inspired by traditional permutation-based mutations used in scheduling, but it is adapted to maintain structural integrity in the context of spatial allocation. By introducing stochasticity in room assignments, the Room Shuffle Mutation complements the deterministic nature of the Closest-Room-First heuristic and helps the evolutionary process explore a more diverse solution space.

Algorithm 4 RoomShuffleMutationOperator

```
1: Procedure RoomShuffleMutation(solution)
2: tmpSolution ⇐ copy(solution)
3: if solution.length ≥ 2 then
4:     if getRandomValue() ≤ mutationProbability then
5:         tmpRooms ⇐ tmpSolution.getRooms()
6:         shuffle(tmpRooms)
7:         tmpSolution.setRooms(tmpRooms)
8:     end if
9: end if
10: return tmpSolution
11: End Procedure
```

FIGURE 3. RoomShuffle Mutation Operator

Time complexity of the mutation operator

Line 2 creates a copy of the solution. Assuming the copying involves duplicating the exam-to-room assignments, this would take *O(n)* time, where n is the number of exams (or solutionLength). Line 3 ensures that the solution has at least two entries for shuffling. This comparison is *O*(1). Line 4 generates a random number and compares it to the mutation probability. Both operations are *O*(1). Line retrieves the rooms assigned to each exam in the solution. Assuming this returns an array or list, it would take *O(n)* time. Line 6 shuffles the rooms using the Fisher-Yates shuffle, which is typically *O(n)* time, where n is the number of exams. Line 7 sets the shuffled rooms back into the solution. Assuming this is an array or list, it would take *O(n)* time. Thus, the overall time complexity is dominated by the *O(n)* operations for copying, retrieving rooms, shuffling, and setting rooms.

DATASET

Ten datasets were used. Nine, known as the Purdue Dataset, are final exam schedules for semesters from fall 2008 to fall 2012 and then from spring 2009 to spring 2012 at Purdue University. One is from the examination timetable at Bayero University Kano for the 1st semester of 2017/2018 academic session. Tables 1, 2 and 3 summarises the characteristics of the dataset.

The Bayero problem, summarized in Table 4, is characterized by 23,738 students enrolled in 832 exams across 21 departments. There are 32 rooms spread across 2 campuses with a total capacity for 7,121 students. The examination period is 64 timeslots. There are about eight (8) general exams taken by every student of the university. For these courses, at any given timeslot, the enrolment of the students is so high such that the exam requires splitting across almost all rooms spread over the two campuses spanning a wide geographical area. A trivial algorithm can be used to place the students enrolled into as many arbitrary rooms as needed but the problem with this approach is that it may create highly undesirable timetables. This is because students can be placed in rooms that are non-trekkable distance from their usual teaching environment. Apart from additional financial cost on the students, these schedules may also create anxiety and stress due to unfamiliar environment. For the Purdue dataset to be usable in this research, there was a need to augment the original dataset with department location data obtained from google maps.

TABLE 1. Characteristics of Purdue Dataset (2008 to 2009)

Item	Fall 2008	Spring 2009	Fall 2009
Exams	857	794	890
Students	34988	32214	34239
Enrolments	130539	119803	125347
Distribution constraints	13	14	8
Exams fixed in time	166	161	104
Exams fixed in room	74	62	69
Large exams (600+)	24	22	23
Exams needing a room split	2	1	9
Exams with original room	1849	1870	1726
Density	0.02985	0.02927	0.03294
Average periods	26.861 ± 1.005	26.922 ± 0.982	27.494 ± 0.705

Average rooms	261.192 ± 4.846	261.409 ± 3.817	266.016 ± 5.403

Reproduced from (*UniTime | Examination Timetabling Benchmark Benchmark Datasets*, n.d.)

TABLE 2. Characteristics of Purdue Dataset 2010 to 2011

Item	Spring 2010	Fall 2010	Spring 2011
Exams	835	875	801
Students	30353	34418	31688
Enrolments	113864	129090	113211
Distribution constraints	10	6	1
Exams fixed in time	105	121	99
Exams fixed in room	109	89	170
Large exams (600+)	17	21	17
Exams needing a room split	4	7	13
Exams with original room	1799	1905	1485
Density	0.03025	0.02872	0.03519
Average periods	27.601 ± 0.666	27.460 ± 0.728	27.498 ± 0.710
Average rooms	260.208 ± 7.252	250.693 ± 5.574	234.674 ± 11.772

Reproduced from (*UniTime | Examination Timetabling Benchmark Benchmark Datasets*, n.d.).

TABLE 3. Characteristics of Purdue Dataset 2011 to 2012

Item	Fall 2011	Spring 2012	Fall 2012
Exams	827	780	819
Students	33856	31593	33279
Enrolments	122384	111345	117265
Distribution constraints	6	13	20
Exams fixed in time	58	63	57
Exams fixed in room	70	6	24
Large exams (600+)	18	20	22
Exams needing a room split	20	9	10
Exams with original room	1524	1485	1533
Density	0.03456	0.03586	0.03283
Average periods	28.152 ± 0.406	28.022 ± 0.466	28.145 ± 0.409
Average rooms	256.247 ± 9.800	265.771 ± 4.696	262.892 ± 6.116

Reproduced from (*UniTime | Examination Timetabling Benchmark Benchmark Datasets*, n.d.).

TABLE 4. Characteristics of Bayero Dataset

Item	Value
Exams	832
Students	23738
Enrolments	119383
Distribution constraints	4

Exams fixed in time	0
Exams fixed in room	0
Large exams (600+)	26
Exams needing a room split	13
Exams with original room	0
Density	0.0491
Available periods	64
Available rooms	32
Total Room capacity	7121

EXPERIMENTAL SETUP

All experiments were implemented in Java using the jMetal optimization framework. The experiments were executed on a machine with an Intel Core i7 processor running at 2.9 GHz, with 4 cores and 8 GB of RAM. The room allocation algorithms were embedded within an evolutionary framework to evaluate their performance under consistent and reproducible conditions.

The following experimental parameters were used throughout:

i. Population size: 100
ii. Number of generations: 100
iii. Mutation rate: 0.9
iv. Selection strategy: Binary Tournament Selection
v. Ranking: Fast non-dominated ranking (inherited from jMetal's NSGA-II base class)
vi. Mutation operator: Room Shuffle Mutation

Although the problem is single-objective, the NSGA-II base class in jMetal was retained for its stable and modular evolutionary framework. As a result, fast non-dominated sorting and crowding distance calculations are applied by default. In the absence of multiple objectives, the system effectively regresses to a single-objective behaviour in terms of selection accuracy. While this introduces a slight computational overhead, it is a reasonable trade-off for consistency, stability, and structural compatibility. Since all algorithms are evaluated under identical experimental conditions, comparative fairness is preserved.

The mutation rate was set to 0.9, aligning with prior work on permutation-based evolutionary algorithms (Banzhaf et al., 1996; Shimodaira, 1996), and compensates for the deliberate omission of a crossover operator. In exam timetabling, crossover commonly disrupts feasibility—introducing conflicts that require expensive repair routines—whereas a high mutation rate ensures sufficient exploration while maintaining solution validity. Preliminary tuning with lower rates (e.g., 0.3, 0.5) yielded lower improvements in fitness, reinforcing the choice of a high mutation rate as an efficient alternative to crossover in our domain. To account for algorithmic randomness and ensure statistical reliability, each experiment was executed 30 times per dataset. For each run, the best, average, and worst fitness scores were recorded. Where applicable, statistical significance was assessed using the non-parametric Mann–Whitney U test, which does not require a normal distribution assumption. A confidence level of 95% ($p < 0.05$) was used to determine significance.

The Purdue and Bayero datasets were used for evaluation, as detailed in Section 5. Fitness values were computed using the location-sensitive objective described in Section 2. Solutions that violated room capacity constraints were discarded and reinitialized, ensuring that all evaluated individuals were feasible.

EVALUATION TECHNIQUE

To evaluate the performance of the Closest-Room-First, Largest-Room-First, Best Fit, and Random heuristics, nine Purdue and one Bayero dataset were analysed. The primary metric was the average travel distance for students from a designated reference point to their assigned rooms. Both the heuristics were independently used to create an initial population. Each is then evolved using a single objective Non-dominated Sorting Genetic Algorithm II, and the distance fitness value returned. A test of significance was also conducted on all datasets using the non-parametric Mann-Whitney U test to rule out the possibility of obtaining the results due to chance.

To ensure a comprehensive evaluation, the proposed Closest-Room-First (CRF) heuristic was tested against three location-agnostic baseline strategies commonly found in the exam timetabling literature:

1. *Random assignment*
2. *Largest Room First (LRF):* a greedy heuristic that prioritizes rooms with maximum capacity, and
3. *Best Fit (BF):* which selects the room with the tightest capacity fit for each exam.

These heuristics represent standard constructive approaches that focus solely on capacity optimization without considering student proximity or spatial distribution.

RESULTS AND DISCUSSION

This section presents a comparative evaluation of four room allocation heuristics: Random, Closest-Room-First (CRF), Greedy Largest-Room-First (LRF), and Best-Fit. The CRF heuristic, which explicitly optimizes for student proximity, is benchmarked against capacity-aware but location-agnostic heuristics. Evaluation spans datasets of varying spatial density, allowing us to understand how each heuristic performs in compact versus dispersed institutional layouts.

Figure 5 and Table 5 shows the average travel distance fitness for all four heuristics on two representative dataset groups: a spatially sparse, multi-campus dataset (BUK), and an average of nine spatially dense single-campus Purdue datasets. Lower values indicate better performance.

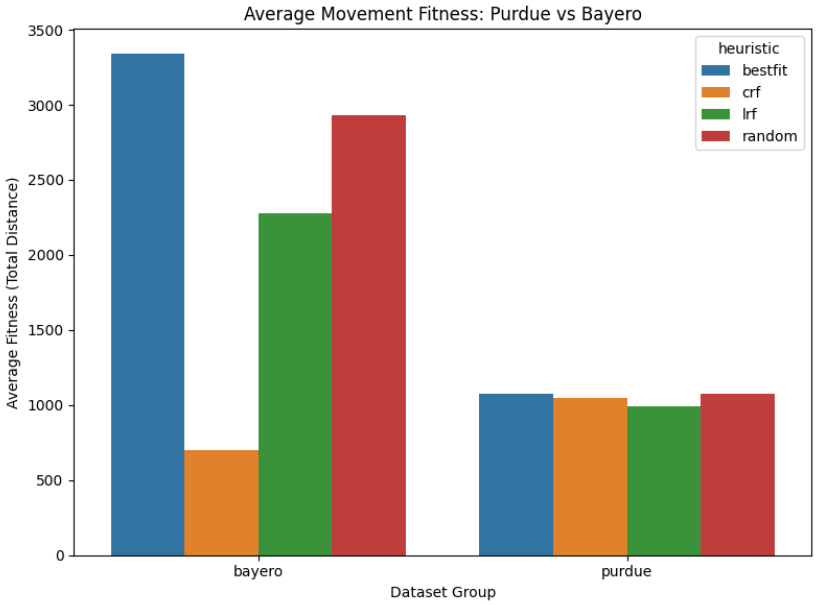

FIGURE 4. Average Fitness of Bayero vs Purdue

TABLE 5. Average fitness (lower is better) across two dataset categories.

Dataset	Random	LRF	Best-Fit	CRF
Purdue (avg)	1,075.14	**994.80**	1,073.61	1,045.38
Bayero	2,850.34	2,274.49	3,338.22	**700.45**

The results show a clear interaction between spatial density and heuristic performance. In the BUK dataset—characterized by a high spatial density score of 2768.18 meters—the CRF heuristic dramatically outperformed all others, achieving a travel cost of 700.45 versus 2274.49 (LRF) and 2850.34 (Random). This reflects a relative improvement margin of over 69% compared to the next best heuristic, LRF. In contrast, in the Purdue datasets, which are far more compact (average spatial density ~1064 meters), LRF achieved the best average performance (994.80), only slightly ahead of CRF (1045.38) and much better than Random (1075.14). This reinforces the idea that CRF's value increases with spatial dispersion, while traditional heuristics suffice in dense layouts.

To complement the summary comparison, Table 6 and Figure 6 provides a detailed breakdown of average travel-distance fitness for each heuristic across all ten datasets. This reveals the dataset-specific behaviour of each method, and supports the broader conclusions drawn earlier regarding the influence of spatial density.

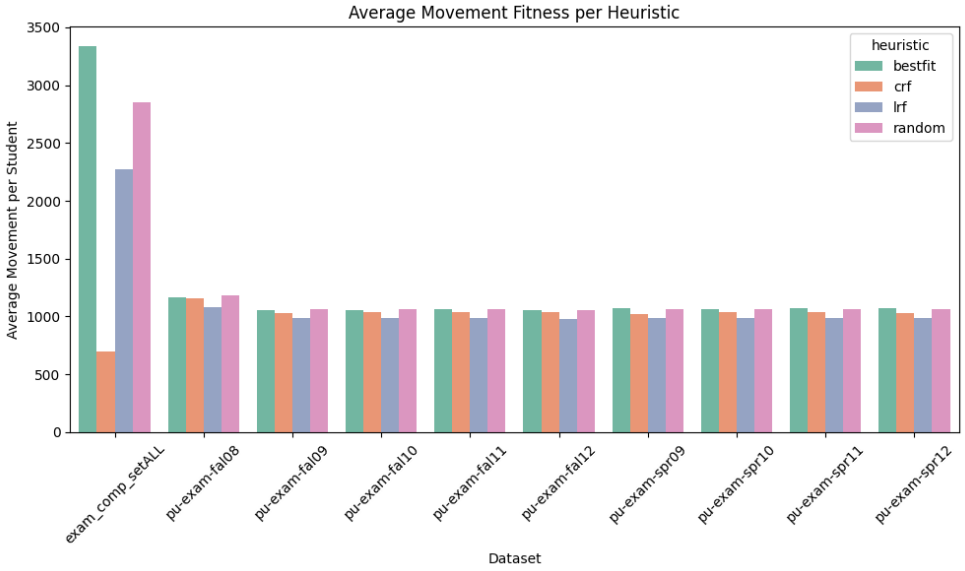

FIGURE 5. Average Movement Fitness per Student Across All Datasets and Heuristics

TABLE 6. Average Fitness per Heuristic Across All Datasets

Dataset	Random	LRF	Best-Fit	CRF
Bayero	2850.34	2274.49	3338.22	700.45
Purdue Fall 08	1186.19	1080.89	1163.14	1155.11
Purdue Fall 09	1065.33	985.90	1056.62	1029.07
Purdue Fall 10	1062.67	984.72	1058.23	1034.90
Purdue Fall 11	1059.40	982.95	1059.38	1034.19
Purdue Fall 12	1057.06	980.86	1053.10	1035.11
Purdue Spring 09	1060.60	985.44	1068.39	1020.40
Purdue Spring 10	1061.20	982.72	1065.34	1037.32
Purdue Spring 11	1061.66	983.02	1068.56	1033.59
Purdue Spring 12	1062.17	986.71	1069.74	1028.74

Table 7 shows the spatial density of each dataset, computed as the average geodesic distance between department buildings and available rooms. The BUK dataset (Bayero) is markedly more geographically dispersed, with a spatial density of over 2700 meters, compared to approximately 1060 meters across all Purdue datasets. This sharp contrast reinforces the importance of location-sensitive room allocation heuristics in multi-campus contexts.

TABLE 7. Spatial Density of Datasets

Dataset	Spatial Density (meters)
Bayero	2768.18
Purdue Fall 08	1190.25
Purdue Fall 09	1077.03
Purdue Fall 10	1064.24
Purdue Fall 11	1064.46
Purdue Fall 12	1063.98
Purdue Spring 09	1062.85
Purdue Spring 10	1065.42
Purdue Spring 11	1064.19
Purdue Spring 12	1063.66

To assess statistical significance, the non-parametric Mann–Whitney U test, which is widely used in examination timetabling research for comparing heuristic performance without assuming normality of the data was applied (E. Burke et al., 2010; Gashgari et al., 2018). In this domain, fitness score distributions are generally not assumed to follow a normal distribution due to the discrete and highly constrained nature of the search space (Carlsson et al., 2023; Qu et al., 2009). The Mann–Whitney U test assumptions of independent samples and ordinal or continuous measurement scales were satisfied. The confidence level was set at 95% ($p < 0.05$) and calculated effect sizes $r = |Z|/\sqrt{N}$ following Cohen's thresholds, where 0.1, 0.3, and 0.5 represent small, medium, and large effects, respectively.

Mann–Whitney U tests were conducted comparing the Closest-Room-First (CRF) heuristic against all three alternatives: Random, Greedy Largest-Room-First (LRF), and Best-Fit. Across all comparisons and datasets, CRF consistently outperformed the baselines with highly significant p-values ($p < 0.001$). In the most spatially sparse dataset (BUK), CRF achieved a large effect size ($r \approx 0.894$) when compared with Random, and moderate effect sizes ($r \approx 0.42$) against LRF and Best-Fit, as shown in Table 8. These values demonstrate not only statistical significance but also substantial practical significance, confirming that CRF's movement-reducing advantage is statistically robust and most pronounced in geographically dispersed environments.

TABLE 8. Mann–Whitney U test results comparing CRF to other heuristics on the BUK dataset (spatially sparse scenario)

Compared With	U Statistic	p-value	Z-score	Effect Size (r)
Random	100.0	< 0.001	3.998	0.894
LRF	0.0	< 0.001	-1.89	0.422
Best-Fit	0.0	< 0.001	-1.89	0.422

Effect of Constructive Heuristics on NSGA-II Optimization

To assess the downstream impact of initial room allocation heuristics, each of the four methods—Closest-Room-First (CRF), Random, Greedy Largest-Room-First (LRF), and Best-Fit—was used to generate the initial population for NSGA-II. The optimization was allowed to proceed with identical parameters across all experiments. The resulting best movement fitness values are presented in Table 9 and visualized in Figure 7.

In the spatially sparse Bayero dataset, the CRF heuristic yielded a dramatic improvement, producing a final fitness of 437.52, substantially outperforming all other heuristics (e.g., Random: 2320.76, LRF: 2792.24, Best-Fit: 2952.81). This highlights CRF's ability to preserve its spatial advantages even after full evolutionary optimization. In contrast, in the compact single-campus Purdue datasets, the final outcomes were more varied. While CRF still performed well in a few cases (e.g., Purdue Fall 09), Random and LRF occasionally produced better evolved solutions, likely due to their capacity to introduce diversity in scenarios where spatial dispersion is less critical. These results confirm that CRF provides a robust foundation in scenarios where minimizing travel distance is a dominant concern. In more compact campuses, however, the flexibility of random or capacity-based heuristics may suffice when combined with a strong evolutionary engine like NSGA-II.

TABLE 9. Best Final Movement Fitness After NSGA-II for Each Heuristic

Dataset	Best-Fit	CRF	LRF	Random
Bayero	2952.81	**437.52**	2792.24	2320.76
Purdue Fall 08	**1159.26**	1174.37	1185.31	1178.49
Purdue Fall 09	1051.79	**1049.29**	1071.65	1051.81
Purdue Fall 10	1052.32	1113.18	1059.61	**1049.59**
Purdue Fall 11	1053.48	1093.05	1057.01	**1044.84**
Purdue Fall 12	1047.57	1076.34	**1043.56**	1045.61
Purdue Spring 09	1063.42	1043.35	1055.51	**1042.47**
Purdue Spring 10	1060.05	1111.45	1063.05	**1045.12**
Purdue Spring 11	1062.02	1075.13	1062.49	**1046.74**
Purdue Spring 12	1064.78	1093.81	1052.76	**1050.63**

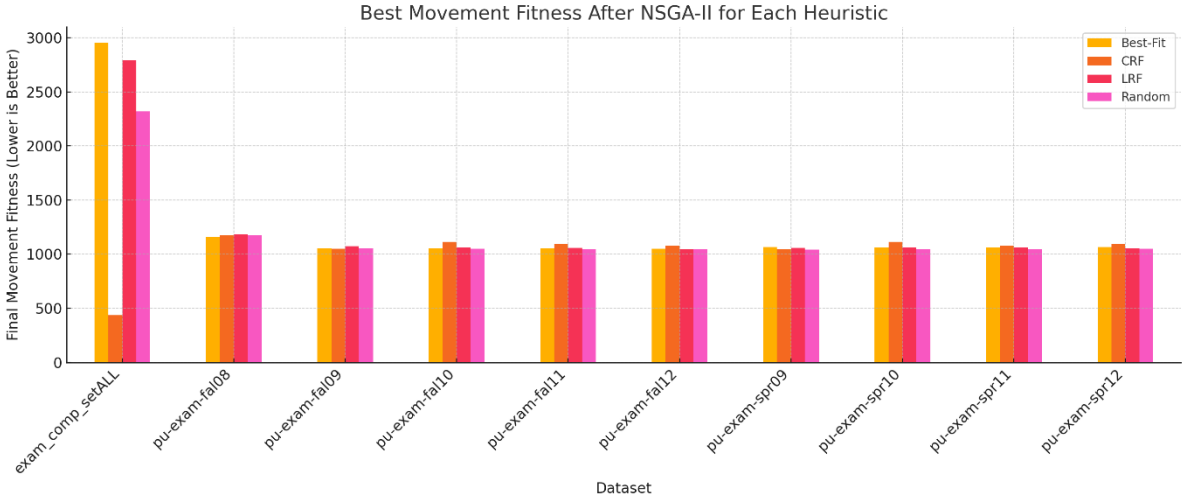

FIGURE 7. Best Movement Fitness after NSGAII for all four heuristics

Runtime and Scalability Analysis

While solution quality is the primary concern in heuristic design, computational efficiency is equally important for practical deployment, particularly in large-scale institutional settings. This section examines the average runtime of CRF and the three baseline heuristics across all ten datasets (Table 10) and analyses how dataset characteristics influence scalability. The goal is to identify structural factors that explain variations in runtime — including the unusually short execution observed for the Bayero dataset — and to assess the trade-offs between computational cost and the movement reduction benefits achieved by CRF.

TABLE 10. Average runtime for each heuristic across the 10 datasets

Dataset	Random	CRF	LRF	Best-Fit
Bayero	0.06	0.03	0.03	0.03
Purdue Fall 08	0.46	0.34	0.28	0.53
Purdue Fall 09	0.45	0.36	0.31	0.49
Purdue Fall 10	0.50	0.34	0.37	0.51
Purdue Fall 11	0.60	0.33	0.30	0.40
Purdue Fall 12	0.43	0.37	0.26	0.38
Purdue Spring 09	0.42	0.32	0.31	0.53

Purdue Spring 10	0.46	0.37	0.35	0.63
Purdue Spring 11	0.38	0.29	0.26	0.39
Purdue Spring 12	0.48	0.43	0.26	0.37

The runtime analysis revealed that, while CRF generally incurred higher computational costs than Random and in some cases LRF, these costs were accompanied by substantial gains in movement reduction, particularly in spatially dispersed datasets. In datasets with many rooms and students (e.g., Purdue), the sorting and assignment overheads magnify, leading to very high slowdown factors compared to Random. Conversely, in the Bayero dataset, CRF's runtime was very short (0.03 mins) because its departmental aggregation logic mapped large, homogeneous student groups almost directly to their nearest rooms, reducing the number of sorting and allocation iterations required. This alignment between dataset characteristics and the heuristic's logic eliminate much of the overhead observed in less spatially constrained datasets. From a scalability perspective, CRF's runtime grows linearly with the number of departments and students per exam but can grow super-linearly if the number of rooms increases substantially, due to the $R \log R$ sorting step. This suggests that CRF remains computationally feasible for medium-to-large institutions, but very large-scale instances may require parallelization or heuristic pruning to maintain lower runtimes.

To further explain the observed runtime patterns, particularly the Bayero anomaly, the CRF's complexity term was related to two key dataset structure indicators — rooms per department (ρ_{RD}) and the Complexity Balance Index (CBI) — which reveal whether runtime is dominated by allocation or room-sorting operations.

CRF's per-dataset runtime is driven by two main complexity contributors in the heuristic:

i. allocation work, dominated by department–exam–student operations, and
ii. room ordering, dominated by sorting available rooms.

From the analysis, the asymptotic cost is:

$$O(D \times E \times S + R \times D + R \log R)$$

In practice, for large instances the allocation term $D \times E \times S$ and the sorting term $R \log R$ are the two drivers. To compare which side dominates on a given dataset, a dimensionless ratio that balances these contributors after normalizing the student count by exams was formed. Let

$$\bar{s} = \frac{S}{E} \text{ (average students per exam)}$$

The Complexity Balance Index was the defined as:

$$CBI = \frac{R \log R}{D \times E \times \bar{s}}$$

- Numerator $R \log R$: the room-sorting workload (per dataset).
- Denominator $D \times E \times \bar{s}$: the allocation workload (department-wise placement work per exam, scaled by average enrollment).

Interpretation of CBI

- Small CBI (↓): allocation dominates; CRF's departmental aggregation tends to run fast (e.g., Bayero).

- Large CBI (↑): room sorting contributes more; runtime tends to be longer (e.g., Purdue).

This index does not predict absolute time; it flags which part of the complexity likely dominates on a dataset and explains the observed runtime patterns in Table 11.

TABLE 11. Dataset Structural Indicators and Complexity Balance Index (CBI)

Dataset	R	D	ρ_{RD}	CBI
BUK	32	21	1.52	0.00032
Purdue Fall 2008	261	21	12.43	0.00285
Purdue Fall 2009	248	21	11.81	0.00269
Purdue Fall 2010	252	21	12.00	0.00278
Purdue Fall 2011	245	21	11.67	0.00270
Purdue Fall 2012	241	21	11.48	0.00266
Purdue Spring 2009	257	21	12.24	0.00282
Purdue Spring 2010	245	21	11.67	0.00269
Purdue Spring 2011	241	21	11.48	0.00267
Purdue Spring 2012	240	21	11.43	0.00266

The results show that BUK has a very low room-to-department ratio ($\rho_{RD} = 1.52$) and the smallest CBI (0.00032), meaning runtime is dominated by allocation rather than sorting. This explains its rapid execution — large, homogeneous student groups are mapped directly to nearest rooms with minimal sorting. In contrast, Purdue datasets have much higher ρ_{RD} values (~11–12) and CBIs (~0.0027–0.00285), indicating room sorting contributes more significantly to runtime. This accounts for their longer execution times despite similar department and student counts. Institutions with very large room counts may require parallelization or pruning strategies to maintain runtime efficiency.

DISCUSSION

This study addresses the room allocation component of the examination timetabling problem, with a particular focus on high-pressure environments where room capacity is limited and physical campus layout introduces logistical challenges. In such settings, minimizing student travel becomes not only desirable but necessary.

Among the datasets evaluated, the BUK dataset exemplifies a severely constrained allocation scenario: only 32 rooms are available for over 20,000 students, and a spatial density score exceeding 2700 meters reflects substantial inter-building distances. In addition, its low room-to-department ratio ($\rho_{RD} = 1.52$) and extremely small Complexity Balance Index (CBI = 0.00032) indicate that runtime is dominated by department–exam–student allocation rather than room sorting, allowing CRF to execute very efficiently. These characteristics justify classifying the problem as a Highly-Constrained Room Allocation Problem and motivate the use of heuristics sensitive to both capacity and spatial layout.

To further evaluate the downstream impact of the constructive heuristics, each was used to initialize the population for NSGA-II. The resulting best movement fitness per dataset is shown in Table 9. In the highly dispersed BUK dataset, the CRF-initialized population yielded the best final solution (437.52), far outperforming all alternatives. This confirms that in spatially sparse environments, CRF not only improves initial fitness but also gives the evolutionary process a strong head start that persists post-optimization. However, in the spatially compact Purdue datasets, the advantage of CRF diminishes. Here, higher room-to-department ratios (~11–12) and larger CBI values (~0.0027–0.00285) mean room sorting plays a more significant role in runtime, and the reduced spatial variation lowers the benefit of proximity-based allocation. In these settings, the Random heuristic often led to the best final fitness, with

occasional wins by LRF and Best-Fit — suggesting that when spatial layout is less critical, diversity introduced by random initialization allows NSGA-II to more effectively explore the solution space and converge to near-optimal room assignments.

The exceptional performance of CRF in spatially sparse settings demonstrates its value not merely as a heuristic, but as a strategic initializer for guiding evolutionary algorithms in real-world, high-stakes deployment scenarios. In institutions with multi-campus configurations or irregular spatial layouts, CRF ensures both immediate and lasting optimization benefits, and the runtime analysis confirms that these gains can be achieved with acceptable computational cost when room-to-department ratios remain low.

CONCLUSION

This study addressed the problem of student-centric, spatially-aware exam timetabling by proposing a context-aware constructive heuristic—Closest-Room-First (CRF)—designed to minimize student travel distances during room allocation. Integrated within the NSGA-II evolutionary framework, CRF consistently improved the initial quality of solutions and, in highly constrained or geographically dispersed environments, led to substantial end-to-end optimization gains.

Our experiments across ten real-world datasets demonstrate that CRF offers significant practical benefits, particularly in multi-campus institutions where physical layout imposes logistical burdens on students. In such settings, CRF achieved over 80% reduction in student movement cost compared to random and capacity-only heuristics, with its advantages persisting even after full NSGA-II evolution. Runtime analysis further shows that these gains can be achieved with acceptable computational cost, especially when room-to-department ratios are low; in such cases, allocation dominates runtime, whereas high ratios shift cost toward room sorting.

The broader contribution of this work lies in showing how domain-specific heuristics—when guided by spatial context—can amplify the performance of general-purpose metaheuristics like NSGA-II, without sacrificing feasibility or scalability. Beyond optimization, the educational impact is equally compelling: timetables that minimize travel stress support better student performance, accessibility, and overall exam-day experience.

LIMITATIONS AND FUTURE WORK

This study has been intentionally scoped to evaluate the Closest-Room-First heuristic within a fixed-timeslot setting and against baseline heuristics that represent common practice in the literature. While this focused approach highlights CRF's contribution in spatially-aware room allocation, it also opens opportunities for broader comparative studies with advanced allocation strategies such as clustering-based or constraint-satisfaction methods. The experiments were designed around movement cost as the primary metric, ensuring a clear measure of CRF's spatial benefits, but leaving room to incorporate additional dimensions such as fairness and accessibility in future work. Likewise, runtime patterns observed under varying room-to-department ratios provide a valuable basis for exploring scalability enhancements and hybridization strategies in larger or more complex institutional settings.

For future work, several avenues are worth exploring. First, the CRF heuristic can be extended and tested in diverse institutional configurations, including those with dynamic room

availability or hybrid exam models. Second, comparisons with clustering-based and constraint-satisfaction-based allocation strategies would deepen understanding of where CRF stands in the broader landscape. Finally, incorporating fairness and accessibility metrics—such as equal walking burdens across departments or students with disabilities—would enhance the equity of the scheduling process.

REFERENCES

Abdelhalim, E. A., & El Khayat, G. A. 2016. A Utilization-based Genetic Algorithm for Solving the University Timetabling Problem (UGA). *Alexandria Engineering Journal*, *55*(2), 1395–1409. https://doi.org/10.1016/j.aej.2016.02.017.

Abraham, O. L., Ngadi, M. A. B., Sharif, J. B. M., & Sidik, M. K. M. 2025. Multi-Objective Optimization Techniques in Cloud Task Scheduling: A Systematic Literature Review. *IEEE Access*, *13*, 12255–12291. https://doi.org/10.1109/access.2025.3529839.

Adnan, F. A., Ab Saad, S., Yahya, Z. R., & Wan Muhamad, W. Z. A. 2018. Genetic Algorithm Method in Examination Timetabling Problem: A Survey. *Regional Conference on Science, Technology and Social Sciences (RCSTSS 2016)*, *Rcstss*, 901–907. https://doi.org/10.1007/978-981-13-0074-5_88.

Aizam, N. A. H., Jamaluddin, N. F., & Ahmad, S. 2016. A survey on the timetabling communities' demands for an effective examination timetabling in universiti malaysia terengganu. *Malaysian Journal of Mathematical Sciences*, *10*, 105–116.

Alabbadi, A. A., & Abulkhair, M. F. 2021. Multi-Objective Task Scheduling Optimization in Spatial Crowdsourcing. *Algorithms*, *14*(3), 77. https://doi.org/10.3390/a14030077.

Al-Hawari, F., Al-Ashi, M., Abawi, F., & Alouneh, S. 2020. A practical three-phase ILP approach for solving the examination timetabling problem. *International Transactions in Operational Research*, *27*(2), 924–944. https://doi.org/10.1111/itor.12471.

Ayob, M., Abdullah, S., & Malik, A. M. A. 2007. A Practical Examination Timetabling Problem at the Universiti Kebangsaan Malaysia. *International Journal of Computer*, *7*(9), 198–204.

Banzhaf, W., Francone, F. D., & Nordin, P. 1996. The effect of extensive use of the mutation operator on generalization in genetic programming using sparse data sets. In H.-M. Voigt, W. Ebeling, I. Rechenberg, & H.-P. Schwefel (Eds.), *Parallel Problem Solving from Nature—PPSN IV* (Vol. 1141, pp. 300–309). Springer Berlin Heidelberg. https://doi.org/10.1007/3-540-61723-X_994.

Bashab, A., Osman Ibrahim, A., Abakar Tarigo Hashem, I., Aggarwal, K., Mukhlif, F., A. Ghaleb, F., & Abdelmaboud, A. 2023. Optimization Techniques in University Timetabling Problem: Constraints, Methodologies, Benchmarks, and Open Issues. *Computers, Materials & Continua*, *74*(3), 6461–6484. https://doi.org/10.32604/cmc.2023.034051.

Burke, E., Curtois, T., Hyde, M., Kendall, G., Ochoa, G., Petrovic, S., Vazquez-Rodriguez, J. A., & Gendreau, M. 2010. Iterated local search vs. hyper-heuristics: Towards general-purpose search algorithms. *IEEE Congress on Evolutionary Computation*, 1–8. https://doi.org/10.1109/CEC.2010.5586064.

Burke, E., Elliman, D., Ford, P., & Weare, R. 1996. Examination timetabling in British universities: A survey. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *1153*, 76–90. https://doi.org/10.1007/3-540-61794-9_52.

Burke, E. K., & Bykov, Y. 2016. An adaptive flex-deluge approach to university exam timetabling. *INFORMS Journal on Computing*, *28*(4), 781–794. https://doi.org/10.1287/ijoc.2015.0680.

Burke, E. K., Elliman, D. G., & Weare, R. 1994. A University Timetabling System Based on Graph Colouring and Constraint Manipulation. *Journal of Research on Computing in Education*, *27*(1), 1–18. https://doi.org/10.1080/08886504.1994.10782112.

Burke, E. K., Mccollum, B., Meisels, A., Petrovic, S., & Qu, R. (2006). *A Graph-Based Hyper-Heuristic for Educational Timetabling Problems*. 1–34.

Burke, E. K., Newall, J. R., & Weare, R. E. 1996. A memetic algorithm for university exam timetabling. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *1153*, 241–250. https://doi.org/10.1007/3-540-61794-9_63.

Carlsson, M., Ceschia, S., Di Gaspero, L., Mikkelsen, R. Ø., Schaerf, A., & Stidsen, T. J. R. 2023. Exact and metaheuristic methods for a real-world examination timetabling problem. *Journal of Scheduling*, *26*(4), 353–367. https://doi.org/10.1007/s10951-023-00778-6.

Ceschia, S., Di Gaspero, L., & Schaerf, A. 2022. Educational timetabling: Problems, benchmarks, and state-of-the-art results. *European Journal of Operational Research*, *308*(1), 1–18. https://doi.org/10.1016/j.ejor.2022.07.011.

De La Rosa-Rivera, F., Nunez-Varela, J. I., Puente-Montejano, C. A., & Nava-Muñoz, S. E. 2021. Measuring the complexity of university timetabling instances. *Journal of Scheduling*, *24*(1), 103–121. https://doi.org/10.1007/s10951-020-00641-y.

Demeester, P., Bilgin, B., De Causmaecker, P., & Van Den Berghe, G. 2012. A hyperheuristic approach to examination timetabling problems: Benchmarks and a new problem from practice. *Journal of Scheduling*, *15*(1), 83–103. https://doi.org/10.1007/s10951-011-0258-5.

Eldharif, E. A., Sallabi, O. M., & Mosa Eltharif, A. A. 2024. An Improved Genetic Algorithm Tool for Exam Timetabling Solutions. *2024 IEEE 4th International Maghreb Meeting of the Conference on Sciences and Techniques of Automatic Control and Computer Engineering (MI-STA)*, 672–677. https://doi.org/10.1109/mi-sta61267.2024.10599714.

Elloumi, A., Kamoun, H., Jarboui, B., & Dammak, A. 2014. The classroom assignment problem: Complexity, size reduction and heuristics. *Applied Soft Computing*, *14*(10), 677–686. https://doi.org/10.1016/j.asoc.2013.09.003.

Gashgari, R., Alhashimi, L., Obaid, R., & Palaniswamy, T. 2018. A Survey on Exam Scheduling. *2018 1st International Conference on Computer Applications & Information Security (ICCAIS)*, 1–5..

Ghaffar, A., Din, I. U., Tariq, A., & Zafar, M. H. 2025. Hybridization and artificial intelligence in optimizing university examination timetabling problem: A systematic review. *Review of Education*, *13*(2). https://doi.org/10.1002/rev3.70071.

Goulas, S., & Megalokonomou, R. 2018. Marathon, Hurdling or Sprint? The Effects of Exam Scheduling on Academic Performance. *IZA Discussion Paper*, *11624*. https://www.iza.org/publications/dp/11624/marathon-hurdling-or-sprint-the-effects-of-exam-scheduling-on-academic-performance.

Gu, X., Krish, M., Sohail, S., Thakur, S., Sabrina, F., & Fan, Z. 2025. From Integer Programming to Machine Learning: A Technical Review on Solving University Timetabling Problems. *Computation*, *13*(1), 10. https://doi.org/10.3390/computation13010010.

Jáhn-Erdős, S., & Kővári, B. 2024. Exploring fair scheduling aspects – Through final exam scheduling. *Pollack Periodica*, *19*(1), 151–156. https://doi.org/10.1556/606.2023.00780.

Kendall, G., & Hussin, N. M. 2005. A tabu search hyper-heuristic approach to the examination timetabling problem at the MARA university of technology. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 3616 LNCS*, 270–293. https://doi.org/10.1007/11593577_16

Kubale, M. (Ed.). 2004. *Graph colorings*. American Mathematical Society.

Leite, N., Fernandes, C. M., Melício, F., & Rosa, A. C. 2018. A cellular memetic algorithm for the examination timetabling problem. *Computers and Operations Research*, *94*, 118–138. https://doi.org/10.1016/j.cor.2018.02.009.

Liu, Y., Ming, H., Luo, X., Hu, L., & Sun, Y. 2023. Timetabling optimization of classrooms and self-study rooms in university teaching buildings based on the building controls virtual test bed platform considering energy efficiency. *Building Simulation*, *16*(2), 263–277. https://doi.org/10.1007/s12273-022-0938-4.

Ma, H., Zhang, Y., Sun, S., Liu, T., & Shan, Y. 2023. A comprehensive survey on NSGA-II for multi-objective optimization and applications. *Artificial Intelligence Review*, *56*(12), 15217–15270. https://doi.org/10.1007/s10462-023-10526-z.

Mandal, A. K., Kahar, M. N. M., & Kendall, G. 2020. Addressing examination timetabling problem using a partial exams approach in constructive and improvement. *Computation*, *8*(2). https://doi.org/10.3390/COMPUTATION8020046.

Matias, J. B. 2018. A Fair Course Timetabling Using Genetic Algorithm with Guided Search Technique. *2018 5th International Conference on Business and Industrial Research (ICBIR)*, 77–82.

McCollum, B., & Mcmullan, P. 2007. The Second International Timetabling Competition: Examination Timetabling Track. *Electrical Engineering (2007), Track 2*, 1–21.

Muklason, A., Parkes, A. J., Özcan, E., Mccollum, B., & Mcmullan, P. 2017. Fairness in examination timetabling: Student preferences and extended formulations. *Applied Soft Computing Journal*, *55*, 302–318. https://doi.org/10.1016/j.asoc.2017.01.026.

Ngo, S. T., Jaafar, J. B., Aziz, I. A., Nguyen, G. H., & Bui, A. N. 2021. Genetic Algorithm for Solving Multi-Objective Optimization in Examination Timetabling Problem. *International Journal of Emerging Technologies in Learning (iJET)*, *16*(11), 4. https://doi.org/10.3991/ijet.v16i11.21017.

Petrovic, S., & Bykov, Y. 2011. *A Multiobjective Optimisation Technique for Exam Timetabling Based on Trajectories* (pp. 181–194). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-45157-0_12.

Qu, R., Burke, E. K., & McCollum, B. 2009. Adaptive automated construction of hybrid heuristics for exam timetabling and graph colouring problems. *European Journal of Operational Research*, *198*(2), 392–404. https://doi.org/10.1016/j.ejor.2008.10.001.

Saharan, S., & Kadian, K. 2014. A multi-objective genetic room allocation in examination scheduling using graph coloring. *2014 International Conference on Signal Propagation and Computer Technology, ICSPCT 2014*, 514–518. https://doi.org/10.1109/ICSPCT.2014.6885025.

Shen, Y., & Xie, W. 2025. A dynamic multi-objective optimization approach for integrated timetabling and vehicle scheduling. *Swarm and Evolutionary Computation*, *95*, 101960. https://doi.org/10.1016/j.swevo.2025.101960.

Shimodaira, H. 1996. A new genetic algorithm using large mutation rates and population-elitist selection (GALME). *Proceedings Eighth IEEE International Conference on Tools with Artificial Intelligence*, 25–32. https://doi.org/10.1109/TAI.1996.560396.

Soghier, A., & Qu, R. 2013. Adaptive selection of heuristics for assigning time slots and rooms in exam timetables. *Applied Intelligence*, *39*(2), 438–450. https://doi.org/10.1007/s10489-013-0422-z.

Tan, J. S., Goh, S. L., Kendall, G., & Sabar, N. R. 2021. A survey of the state-of-the-art of optimisation methodologies in school timetabling problems. *Expert Systems with Applications*, *165*. https://doi.org/10.1016/j.eswa.2020.113943.

Tuniki, C., Kunta, V., & Trupthi, M. 2020. A System for Efficient Examination Seat Allocation. *Advances in Intelligent Systems and Computing*, 449–460.

UniTime | Examination Timetabling Benchmark Benchmark Datasets. (n.d.). UniTime. Retrieved April 21, 2025, from https://www.unitime.org/exam_datasets.php.

Vrielink, R. O. 2017. Practices in timetabling in higher education institutions: A systematic review. *Annals of Operations Research*. https://doi.org/10.1007/s10479-017-2688-8.

Yousef, A. H., Salama, C., Jad, M. Y., El-gafy, T., Matar, M., & Habashi, S. S. 2016. *A GPU Based Genetic Algorithm Solution for the Timetabling Problem*. 103–109.