A New Initialisation Method for Examination Timetabling Heuristics

Amjad A. Alsuwaylimi
University of Exeter
Exeter, United Kingdom
Email: aa777@exeter.ac.uk

Jonathan E. Fieldsend
University of Exeter
Exeter, United Kingdom
Email: J.E.Fieldsend@exeter.ac.uk

***Abstract*—Timetabling problems are widespread, but are particularly prevalent in the educational domain. When sufficiently large, these are often only effectively tackled by timetabling meta-heuristics. The effectiveness of these in turn are often largely dependant on their initialisation protocols. There are a number of different initialisation approaches used in the literature for starting examination timetabling heuristics. We present a new iterative initialisation algorithm here — which attempts to generate high-quality and legal solutions, to feed into a heuristic optimiser. The proposed approach is empirically verified on the ITC 2007 and Yeditepe benchmark sets. It is compared to popular initialisation approaches commonly employed in exam timetabling heuristics: the largest degree, largest weighted degree, largest enrollment, and saturation degree graph-colouring approaches, and random schedule allocation. The effectiveness of these approaches are also compared via incorporation in an exemplar evolutionary algorithm. The results show that the proposed method is capable of producing feasible solutions for all instances, with better quality and diversity compared to the alternative methods. It also leads to improved optimiser performance.**

***Keywords*—examination timetabling; scheduling; heuristic initialisation; evolutionary algorithms**

I. Introduction

Exam timetabling is considered an NP-complete optimisation problem [1], [2]. The problem requires the scheduling of the exams corresponding to a set of courses, while satisfying a range of constraints determined by exam timetabling staff. There are commonly two types of constraints: hard constraints which must be satisfied for a solution to be viable (e.g. a student should not sit two exams at the same time), and soft constraints which we would prefer to satisfy, but are not absolutely necessary to meet (e.g. a student should not sit two exams in quick succession), see for example [3]. Concisely: satisfying all the hard constraints produces a feasible timetable; whereas soft constraints can be violated.

Many papers have proposed methods to construct examination timetables. Often graph-colouring forms the basis, e.g. [4], [5]. Computational intelligence approaches such as fuzzy logic [6] and neural networks [7] have been employed, however the bulk of work in this area has employed methods from the broad area of evolutionary computation. For instance [8]–[10] applied tabu search. Simulated annealing was used in [11], [12], and memetic algorithms in [13], [14]. Genetic algorithms were used in e.g. [15], ant colony optimisation in e.g. [16], [17], particle swarm optimisation in e.g. [18]. The great deluge algorithm in [19], and hybridisations of distinct heuristic methods in e.g. [20], [21]. More recently hyper-heuristic approaches have gained popularity [22], [23]. The efficacy of an optimisation heuristic is often dependent on the initial population from which it starts. High quality and diverse initial solutions are generally thought to improve the subsequent performance of a timetabling algorithm [24].

The initial population can be generated by a number of methods. Most commonly these attempt to generate 'high quality' solutions (i.e. ones that have low soft constraint violations, and preferably no hard constraint violations) so that the search is initialised in 'good' areas of design space. This minimises the subsequent run time required of the algorithm to find an acceptable solution. It is this area of algorithm design we are particularly concerned with in this work.

We propose a new initialisation approach here, whose main novelty stems from the way it manages three interacting lists of exams to allocate. These lists are arranged and processed in a step-wise fashion in order to provide a good satisfaction of hard and soft constraints.

The rest of the paper is structured as follows. Section II describes popular initialisation approaches from the literature. Our proposed approach is outlined in Section III. Section IV details the widely used ITC 2007 examination datasets [25], [26], and the more recent Yeditepe datasets [27], which we employ in our empirical work here (Section V). Finally, conclusions and future work are presented in Section VI.

II. Initialisation for exam timetabling problems

Many researchers have studied initialisation methods for evolutionary algorithms (EAs) to help solve timetabling problems. [28] proposed an initialisation method where a random operator was used to ensure diversity to support either a higher quality solution or a more random solution (hard constraints could be violated). They concluded that the strictness of the achievement of the hard constraints could result in an inferior ultimate outcome in regard to the soft constraints. However, it is hard to apply this method to problems that enforce a high plenty weight to the violation of the hard constraints. [29] used this approach to initialise EAs, and presented a comparison between genetic algorithms (GAs), simulated annealing (SA), and multi-start stochastic hill-climbing (MSSH) algorithms in regard to the performance when they started from both seeded solutions and random solutions. Their results show that the GA's performance was the worst among the studied algorithms when it started from random solutions. However, when the algorithms started from seeded solutions, the GA's performance increased to the extent that it performed the best, whilst the SA and MSSH were much less affected. [15] proposed a hybrid approach between heuristic measures and a roulette wheel-style method to enhance the solutions' quality while keeping some diversity between them. Work presented in [30] studied the effect of random sequential initialisation followed by hill-climbing — each solution resulted in a better group of solutions than the former approach.

Graph-colouring heuristic algorithms are also widely used to construct an initial feasible timetable, see for example [1], [4], [31]–[33]. In this initialisation approach, the exams are represented as vertices, while the edges are the conflicts between them. Immediately neighbouring vertices are assigned distinct colours. The colour therefore denotes the time period(s) in the timetable. These heuristic methods give the priority of scheduling based on the level of the difficulty in scheduling (number or weight of edges) in order to first schedule the most difficult exams.

The most common graph-colouring methods are as follows.

Largest degree (LD): in this method the degree represents the number of exams in conflict, and the method gives the priority in scheduling to the exams that have most conflicts with other exams.

Largest weighted degree (LWD): this approach applies a weight to each conflict by counting the number of students participating in that conflict. As such, the priority in scheduling is given to core exams.

Saturation degree (SD): the exam that has the fewest number of free periods for scheduling without violation of any hard constraints will be scheduled first. In order to settle ties between exams, the LD approach is used.

Largest Enrollment First (LE): this method depends on the arranging exams in decreasing order based on the number of students enrolled.

In addition to the graph-colouring approaches listed above, random schedule allocation (**RD**) is adapted widely to generate initial solutions in exam timetabling [34]. RD orders the examinations list randomly and schedules exams into the first valid period satisfying the hard constraints. Exams are taken sequentially starting from the top of a randomly ordered list. If the scheduling process fails, the process is repeated with a new random ordering.

III. Proposed Initialisation Approach

We now propose a new algorithm to construct solutions for examination timetabling problems, that can be used as seed solutions for meta-heuristic algorithms. The proposed algorithm attempts to schedule the largest number of conflicting exams as possible in the first periods of a timetable, which we name the **Front Section (FS)**, and in the last periods of the timetable named **Back Section (BS)** while also attempting to satisfy all hard and soft constraints. The number of periods in the FS and BS are equal to a given number of spread periods determined by the **Period Spread (PS)** constraint, which aims to spread each student's exams over a given number of periods. The overarching idea behind the proposed algorithm is to satisfy the PS constraint for each student of any exam scheduled into periods of the **Middle Section (MS)** (i.e. the periods after FS periods and before BS periods).

In order to avoid violation of the PS constraint for an exam timetabled in any period, T, of MS, with period spread range X, then the X periods before and after period T must not contain any conflicting exam (i.e. an exam with students in common). On the other hand, if an exam is timetabled in the first period of the FS, then the number of periods N that must not contain any conflict exam is just the X periods which fall after this period, so $N = X$. Also the same period range $N = X$ is used if an exam is timetabled in the last period of the BS, where just the N periods that fall before the last period must be free of conflicting exams. In case an exam is timetabled in the second period of FS, then the number of periods N that must have no conflicting exams in order to satisfy the PS constraint is $N = 1 + X$ (one period falling before and X periods falling after that period T). Also, scheduling an exam in the period before the last period of BS requires the same period range, $N = X + 1$ (i.e. X periods falling before and one period falling after period T). Timetabling an exam in the third period of FS or in the third period before the last period of BS requires $N = X + 2$ and so on until the period number becomes equal to PS (i.e. the last period in the FS or the first period in the BS). Then $N = (X - 1) + X$, which is still lower than timetabling an exam in any period of MS, which will require $2X$ periods $N = X + X$ having no conflict exam(s) to satisfy the PS constraint.

According to the proposed method, for each period the available rooms of a timetable are ordered increasingly by their capacities, also the exams are ordered in three lists which are: **Front List (FL)**, **Back List (BL)**, and **Middle List (ML)**. Each list initially contains all exams to be scheduled.

In the Front List, as detailed in Algorithm 1, the exams are sorted by decreasing number of conflicts with other exams (line 4), subject to 'after' constraints being observed. This list is then processed from the front.

Once any exam in this list is scheduled in any period in the FS, it is removed from the other lists (ML and BL — lines 19–20). On the other hand, if any exam cannot be scheduled in any period in the FS, then it is removed only from the FL and remains in the other lists to be scheduled via one of them later (line 29).

In the Back List, the exams are sorted by decreasing number of conflicts with other exams, subject to 'before' constraints being observed (line 5 of Algorithm 1). This constraint is derived from the 'after' constraint that stipulates which exams must occur after others. The scheduling process starts by attempting to schedule BL exams as much as possible in the BS, starting from the last period of BS until to the first period of BS while satisfying all hard and soft constraints. Any exam scheduled in these last periods will be removed from the BL

Algorithm 1 Front List (FL) heuristic.

```
Require: $PS$                                  ▷ The period spread number
Require: $E$                                   ▷ Exams to be scheduled
Require: $R$                 ▷ Set of room lists, available each period
• Initialisation step
1: $\alpha \leftarrow \emptyset$
2: $FS \leftarrow$ init_FS($PS$)
3: $BS \leftarrow$ init_PS($PS$)
4: $FL \leftarrow$ sort_by_conflict($E$)
5: $BL \leftarrow$ sort_by_conflict($E$)
6: $ML \leftarrow$ random_sorting($E$)
7: for all periods $p$ do                             ▷ For each period
8:     $R_p \leftarrow$ ascend_sort($R_p$)
9: end for
• Schedule $FL$ exams into Front Section periods ($FS$)
10: while $FL$ is not empty do
11:     isScheduled ← false
12:     $p \leftarrow 1$      ▷ Attempt to schedule from the first period
        ($p$) of $FS$ until reaching the last period of $FS$
13:     while $p \leq |FS|$ do
14:         $R_p^i \leftarrow$ first_suitable_room($R_p, \alpha, FL_1$)
15:         if $R_p^i \neq \emptyset$ then                ▷ If it can be scheduled
16:             if satisfies($FL_1$) = true then
17:                 $\alpha \leftarrow$ schedule($\alpha, FL_1, R_p^i$)
18:                 $FL \leftarrow FL \setminus \{FL_1\}$
19:                 $BL \leftarrow BL \setminus \{FL_1\}$
20:                 $ML \leftarrow ML \setminus \{FL_1\}$
21:                 isScheduled ← true
22:                 $p \leftarrow 1$
23:                 Break      ▷ The current exam is scheduled
24:             end if
25:         end if
26:         $p \leftarrow p + 1$
27:     end while
28:     if isScheduled ← false then
29:         $FL \leftarrow FL \setminus \{FL_1\}$   ▷ Del. exam $FL_1$ from $FL$
30:     end if
31: end while
32: return $\alpha$
```

and the ML (Algorithm 2 lines 12–13), otherwise, in case an exam cannot be scheduled at this stage, it is removed only from BL (line 21).

Finally, the Middle List contains all remaining exams that have not been scheduled by Algorithms 1 and 2. The ML exam list is ordered depending on the number of the remaining periods available, where the exams having the fewest number of available periods in the timetable have the highest priority to be scheduled first in any period while satisfying only the hard constraints. The algorithm selects the periods randomly (Algorithm 3, line 2), giving the priority to the previous selected periods in order to maximise the periods' utilisation and keep the diversity. The process stops when all the exams in the middle list are completely scheduled to a feasible period,

Algorithm 2 Back List (BL) heuristic.

```
Require: $BL$                                          ▷ Back list
Require: $ML$                                        ▷ Middle list
Require: $\alpha$                              ▷ Current schedule state
Require: $R$                 ▷ Set of room lists, available each period
1: for all periods $p$ do                             ▷ For each period
2:     $R_p \leftarrow$ ascend_sort($R_p$)
3: end for
• Schedule $BL$ exams into Back Section periods ($BS$)
4: while $BL$ is not empty do
5:     isScheduled ← false
6:     $p \leftarrow |BS|$
7:     while $p \geq 1$ do
8:         $R_p^i \leftarrow$ first_suitable_room($R_p, \alpha, BL_1$)
9:         if $R_p^i \neq \emptyset$ then
10:            if satisfies($BL_1$) = true then
11:                $\alpha \leftarrow$ schedule($\alpha, BL_1, R_p^i$)
12:                $BL \leftarrow BL \setminus \{BL_1\}$
13:                $ML \leftarrow BL \setminus \{BL_1\}$
14:                isScheduled ← true
15:                Break               ▷ Exam is now scheduled
16:            end if
17:        end if
18:        $p \leftarrow p - 1$                          ▷ inverse selecting
19:    end while
20:    if isScheduled ← false then
21:        $BL \leftarrow BL \setminus \{BL_1\}$  ▷ Del. exam $BL_1$ from $BL$
22:    end if
23: end while
24: return $\{\alpha, ML\}$
```

and a feasible solution is returned. However, if there is any exam in this list which has no available period, i.e. it could not be assigned to any feasible period, an infeasible solution is returned instead (Algorithm 3, line 34).

IV. Test problem description

Examination timetabling problems can be categorised as either capacitated or uncapacitated [35]. With regard to capacitated examination problems, room capacity is considered as a hard constraint, which means each exam must be assigned into one of a set of rooms that has sufficient seating capacity. In the uncapacitated examination problems, only timeslot assignment is considered for hard constraints. Both the ITC 2007 and Yeditepe examination datasets are capacitated problems.

A. ITC 2007

The second international timetabling competition (ITC 2007) included three tracks, covering examination and course timetabling [36]. Here we concentrate on its examination datasets, which are derived from real-world timetabling problems. The exam timetabling track involves 12 instances with distinct characteristics, and constraints that are similar to those constraints encountered in practice. Table I describes the ITC 2007 instances' characteristics. The hard constraints are:

Algorithm 3 Middle List (ML) heuristic.

```
Require: $ML$                                   ▷ Middle list
Require: $\alpha$                               ▷ Current schedule state
Require: $R$                ▷ Set of room lists, available each period
Require: $PL$ list of all periods in the timetable.
  • Schedule remaining exams
 1: $SPL \leftarrow \emptyset$                  ▷ Initial empty selected periods list
 2: $rp \leftarrow \texttt{random\_period}(PL)$ ▷ Get a random period
 3: $SPL \leftarrow SPL \cup \{rp\}$           ▷ Add $rp$ to $SPL$
 4: for all periods $p$ do                      ▷ For each period
 5:     $R_p \leftarrow \texttt{ascend\_sort}(R_p)$   ▷ Order rooms by size
 6: end for
 7: $PL \leftarrow PL \setminus \{rp\}$        ▷ Remove $rp$ from $PL$
 8: $sp \leftarrow 1$                           ▷ Index into SPL
 9: while $ML \neq \emptyset$ do
10:     $\texttt{isScheduled} \leftarrow \texttt{false}$
11:     while $sp \leq |SPL|$ do
12:         if $ML_1$ can be scheduled into $SPL_{sp}$ satisfying all the hard constraints into the $i$th smallest room then $\alpha \leftarrow \texttt{schedule}(\alpha, ML_1, R^i_{SPL_{sp}})$
13:             $ML \leftarrow ML \setminus ML_1$   ▷ Remove exam from list
14:             $\texttt{isScheduled} \leftarrow \texttt{true}$
15:             goto line 19                    ▷ Exit loop as exam scheduled
16:         end if
17:         $sp \leftarrow sp + 1$              ▷ select the next period of $SPL$
18:     end while                               ▷ End scheduling $ML_1$ in the $SPL$
19:     if $\texttt{isScheduled} = \texttt{false}$ then
20:         $temp \leftarrow PL$                ▷ Temporary list
21:         while $temp \neq \emptyset$ do
22:             $sp \leftarrow \texttt{random\_element}(temp)$
23:             if $ML_1$ can be scheduled into $sp$ satisfying all the hard constraints into the $i$th smallest room then $\alpha \leftarrow \texttt{schedule}(\alpha, ML_1, R^i_{temp_{sp}})$
24:                 $ML \leftarrow ML \setminus ML_1$   ▷ Remove from list
25:                 $\texttt{isScheduled} \leftarrow \texttt{true}$
26:                 $SPL \leftarrow SPL \cup \{temp_{sp}\}$
27:                 $PL \leftarrow PL \setminus \{temp_{sp}\}$
28:                 goto line 9                 ▷ Exit from loop
29:             else
30:                 $temp \leftarrow temp \setminus \{temp_{sp}\}$
31:             end if
32:         end while
33:     end if
34:     return infeasible solution
35: end while
36: return $\alpha$
```

No conflicts: the exams which have at least one common student must be in different periods.

Room occupancy: it is not possible to exceed the number of seats available for each period and room.

Period duration: exam length must not exceed period length.

Period related: pairs of exams have time-ordering requirements that must be achieved. Where for any pair of exams ($Exam_1$, $Exam_2$) the following constraints may apply:

- **After constraint:** $Exam_1$ must be scheduled after $Exam_2$.
- **Exam coincidence:** $Exam_1$ and $Exam_2$ must be scheduled simultaneously.
- **Period exclusion:** $Exam_1$ and $Exam_2$ must not be scheduled simultaneously.

Room exclusive: an exam must be roomed alone.

All exams have to be scheduled (a complete solution), and exams cannot be divided between rooms and periods. A feasible solution must satisfy all the hard constraints.

Generally, soft constraints include the following: "**Two Exams in a Row**" constraint that counts the number of occurrences of students having two exams straight after one another, "**Two Exams in a Day**" constraint counts the number of occurrences where two examinations are taken by students in a day but which are not directly adjacent in time, and "**Period Spread (PS)**" constraint that allows organisers to spread a students' exams over a specified number of periods. The PS constraint implicitly contains "two in a row" and "two in a day" constraints, therefore, satisfying the PS constraint means satisfying both the "two in a row" and "two in a day" constraints.

TABLE I
SPECIFICATIONS OF THE 12 INSTANCES OF THE ITC 2007 EXAMINATION TIMETABLING PROBLEM DATASET.

Problem Instance	No. of Exams	No. of Students	No. of Rooms	Days	No. of T.Slots	Conflict Density
Exam 1	607	7891	7	29	54	0.05
Exam 2	870	12,743	49	13	40	0.01
Exam 3	934	16,439	48	12	36	0.03
Exam 4	273	5045	1	7	21	0.15
Exam 5	1018	9253	3	14	42	0.009
Exam 6	242	7909	8	8	16	0.06
Exam 7	1096	14,676	15	40	80	0.02
Exam 8	598	7718	8	40	80	0.05
Exam 9	169	655	3	13	25	0.08
Exam 10	214	1577	48	12	32	0.05
Exam 11	934	16,439	40	9	26	0.03
Exam 12	78	1653	50	7	12	0.18

B. Yeditepe datasets

The Yeditepe dataset is comprised of a number of datasets from the Faculty of Engineering and Architecture of Yeditepe University [27]. In general, they are smaller than ITC 2007 problems with regards to a number of exams, students, and timeslots. Also, this dataset is less constrained compared with ITC 2007 as it contains two hard constraints and only one soft constraint (in ITC 2007, there are seven hard constraints and seven soft constraints). The characteristics of the Yeditepe dataset are given in Table II. The Yeditepe examination timetabling problems' hard constraints are:

Examination conflict (C1): any student must not have more than one exam at any given time period.

Capacity (C2): it is not possible to exceed the number of seats in any given room at any given time period.

TABLE II
THE CHARACTERISTICS OF YEDITEPE DATASET.

Problem Instance	No. of Exams	No. of Students	No. of Rooms	Days	No. of T.Slots	Conflict Density
YUE20011	126	559	2	6	18	0.18
YUE20012	141	591	2	6	18	0.18
YUE20013	26	234	2	2	6	0.25
YUE20021	162	826	2	7	21	0.18
YUE20022	182	869	2	7	21	0.17
YUE20023	38	420	1	2	6	0.2
YUE20031	174	1125	2	6	18	0.15
YUE20032	210	1185	2	6	18	0.14

In this dataset there is only one soft constraint:

Examination spread (C3): the exams for any student on the same day should not be assigned consecutively.

C. Evaluation Function

The number of hard constraint violations must equal zero, whereas the quality of an examination timetable is determined by the soft constraint violations subject to this. The best quality solution is that with the lowest weighted sum of soft constraint violations. The objective function is to minimise the total penalty as result of soft constraint violations defined by (1) for ITC 2007 and (2) for Yeditepe, where the total penalty for the soft constraint violations is calculated for each student s.

1) ITC 2007: Minimise

$$\sum_{s\in S}(w^{2R}C_s^{2R} + w^{2D}C_s^{2D} + w^{PS}C_s^{PS}) + w^{NMD}C^{NMD} + w^{FL}C^{FL} + C^P + C^R. \quad (1)$$

Two exams in a row (C_s^{2R}): penalty given whenever a student s has to attend two distinct exams scheduled in two consecutive periods on the same day.

Two exams in a day (C_s^{2D}): penalty given whenever a student s has to attend two distinct exams scheduled in two non-consecutive periods on the same day.

Period spread (C_s^{PS}): penalty given whenever a student s has to attend more than one exam within a specified period range.

No Mixed durations (C^{NMD}): penalty given whenever any room in any period is assigned to exams with different durations.

Front-load (C^{FL}): penalty incurred by scheduling exams having largest number of students at the end of the examination session.

C^P**:** penalty associated to a **period** whenever it is assigned to an exam.

C^R**:** penalty associated to a **room** whenever it is assigned to an exam.

See [26] for a full description of the ITC2007 exam timetabling problem, weights for each of the above mentioned penalties and the objective function description.

2) Yeditepe: Minimise

$$\sum_{s\in S} w^{2R}C_s^{2R}. \quad (2)$$

Two exams in a row (C_s^{2R}) is a penalty given whenever a student s has to attend two distinct exams scheduled in two consecutive periods on the same day. The weight for this penalty is 1 for all instances of Yeditepe [27].

V. EXPERIMENTAL RESULTS AND COMPARISON

Our experiment has two phases. In the first phase, a set of initial solutions is constructed by the proposed algorithm (OBSI: ordering-based scheduling initialisation) and comparison methods (LD - Largest Degree, LE - Largest Enrollment, LWD - Largest Weighted Degree, SD - Saturation Degree, RD - Random Allocation). In the second phase, a basic evolutionary algorithm (EA) uses the initial population to start its (total) time-limited search.[1]

A. Phase 1: Initialisation

The median results (out of 30 runs for each instance) on both datasets (ITC 2007 examination track and Yeditepe) are shown in Tables III and IV, respectively, including statistical significance. The ITC 2007 results indicate that using our proposed OBSI strategy generally outperforms graph heuristics and random initialisation for initial solution quality. Additionally, the graph heuristics and random initialisation fail to obtain feasible solutions for some of the more difficult instances. Furthermore, OBSI solutions tend to be more diverse in performance, with total soft constraint costs often spread in a wider range than the comparison methods (as shown in Figure 1). Generating an initial population with graph heuristics, on the other hand, tends to occur in a shorter time-frame compared with the OBSI, albeit delivering lower quality solutions.

The results on the Yeditepe problems are provided in Table IV. This shows that the OBSI strategy is able to generate feasible solutions for all instances in contrast to graph heuristics and RD, which again cannot generate a feasible solution for some of the difficult instances. Its performance is less striking than on the ITC 2007 datasets, outperforming some graph heuristics in term of quality on some problems, but on others it performs less well (see Figure 2).

Figures 3 and 4 show the distribution of execution time in milliseconds for the different initialisation approaches on both datasets. Generally speaking the OBSI tends to be relatively faster on the Yeditepe problems (due to the fewer constraints), but often slower on the ITC 2007 problems. Apart from the SD method, the other graph heuristics fail for some instances of ITC 2007 and Yeditepe. Some particular problems instances, such as Exam4 and Exam11 of ITC 2007 and YUE20012, YUE20021, and YUE20032 of Yeditepe, have a large number of exams with high conflict density and few available periods. This can lead to saturation cases where there are no valid periods to schedule the next exam. The main reason for saturation cases occurring is where subroutines schedule exams into periods while giving scheduling priority to exams

[1]These are implemented in Java, with experiments run on an Intel Core (TN) i5-6200U (CPU @ 2.30 GHz with 16 GB RAM) PC running Windows 7 Enterprise with 64-bit operating system. Source code is available at https://github.com/alsuwaylimi/Initialisation-Method-Project.

TABLE III

MEDIAN SOFT CONSTRAINT COST RESULTS, AND TIMINGS ON THE ITC 2007 DATASET. THE HYPHEN SYMBOL "-" MEANS THE METHOD CANNOT PRODUCE FEASIBLE SOLUTIONS FOR THE CORRESPONDING INSTANCE. BOLD AND UNDERLINED RESULTS ARE SIGNIFICANTLY BETTER THAN ALL OTHERS ACCORDING TO THE MANN-WHITNEY U TEST WITH THE HOLM-BONFERRONI CORRECTION.

	OBSI		LD		LWD		LE		SD		RD	
Instances	Median cost	Median t (ms)	Median cost	Median t (ms)	Median cost	Median t (ms)	Median cost	Median t (ms)	Median cost	Median t (ms)	Median cost	Median t (ms)
Exam1	**27987.5**	6130	31805.5	3408	30666.5	3463	30224.5	**2269.5**	31872.5	7714	39630	130463
Exam2	**26661**	10022.5	38974	6973	40838.5	7121	40791	7020	40848	11418.5	150514.5	7817
Exam3	**72713**	23095	102158	12620	101874.5	18704	104357.5	19344	100657	32736.5	249617.5	**9393.5**
Exam4	50139	**1263**	-	-	-	-	-	-	51346.5	2542.5	-	-
Exam5	**74394.5**	11829.5	133080.5	7163	130333	7106	133910	7103.5	135884.5	18150.5	319573.5	45975.5
Exam6	50190	624	53115	686	52190	32845.5	51285	16411	51000	904	63957.5	41347.5
Exam7	**49253**	25232.5	80718.5	11011	77993	10834.5	77386	10829	83277	31225.5	64212.5	31107.5
Exam8	**114559**	7230	136475.5	3104	145734.5	3096	138280.5	3198	134043.5	8197.5	-	-
Exam9	**7705.5**	148	8454.5	62	8550	62	8727.5	62	8998.5	156	11646	874.5
Exam10	66741	452	67936.5	165.5	71808	206	69649	158	67881	872.5	127559.5	8344
Exam11	218227.5	**21067.5**	219267.5	70019.5	-	-	-	-	223552.5	36347	334398.5	86444
Exam12	**10995.5**	452	12327	5187	12748.5	3338	12323	11840	13000.5	**218**	13277.5	7873.5

TABLE IV

YEDITEPE DATASET RESULTS. NOTATION AS IN TABLE III.

	OBSI		LD		LWD		LE		SD		RD	
Instances	Median cost	Median t (ms)	Median cost	Median t (ms)	Median cost	Median t (ms)	Median cost	Median t (ms)	Median cost	Median t (ms)	Median cost	Median t (ms)
YUE20011	831	62	774.5	148	764	163.5	706.5	537.5	713	62	912	62.5
YUE20012	885	**62**	779	2372.5	-	-	-	-	799.5	475.5	886	1057.5
YUE20013	64	15	65	15	63.5	15	64	15	66	**0**	69	11
YUE20021	865	116.5	833.5	2381.5	-	-	-	-	790.5	473	1499.5	116
YUE20022	1093.5	156	1140.5	2098	1112.5	19975.5	1057	51768.5	1110.5	171	1235.5	5938.5
YUE20023	115	**15**	110.5	50.5	108	56	102.5	46.5	117.5	39	116.5	142.5
YUE20031	11285.5	**124**	3306	797.5	4337.5	554	3474	534.5	2582	137	7658	688.5
YUE20032	11378	**156**	-	-	-	-	-	-	**2360.5**	197	-	-

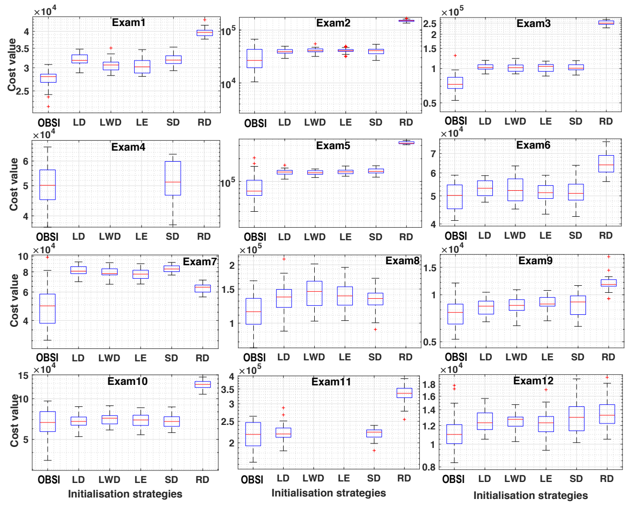

Fig. 1. Boxplots of the soft constraint costs for all initialisation strategies performed on ITC 2007 dataset, where OBSI: ordering-based scheduling initialisation, LD: largest degree, LWD: largest weighted degree, LE: largest enrollment, SD: saturation degree and RD: random. Y-axis on log scale.

having largest degree (LD), largest weighted degree (LWD), or largest enrollment degree (LE). SD prevents saturation cases arising by dynamically ordering exams depending on a number of available periods and giving scheduling priority to exams which have the fewest number of periods available.

We further analyse our results by conducting statistical tests, as shown in Tables III, IV,V,VI, using the Mann-Whitney U test followed by the Holm-Bonferroni method to compensate for multiple hypothesis testing.

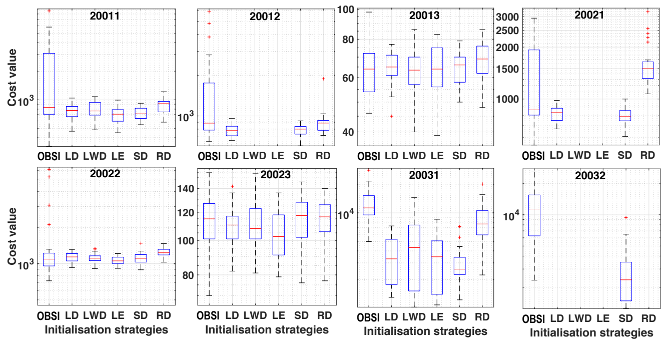

Fig. 2. Boxplots of the soft constraint costs for all initialisation strategies performed on Yeditepe dataset. Notation as in Figure 1. Y-axis on log scale.

B. Phase 2: Optimisation

An EA was implemented following the approach of [37] (which eschews crossover) to quantify the effectiveness of the initialisation on subsequent meta-heuristic performance. We utilise the initial solutions that have been constructed in phase 1 by the OBSI, as well as other initialisation methods, and use these to determine the start populations of the EA. The EA population size is 40. Total run time (initialisation plus optimisation) is capped at 8 minutes. Each child solution in

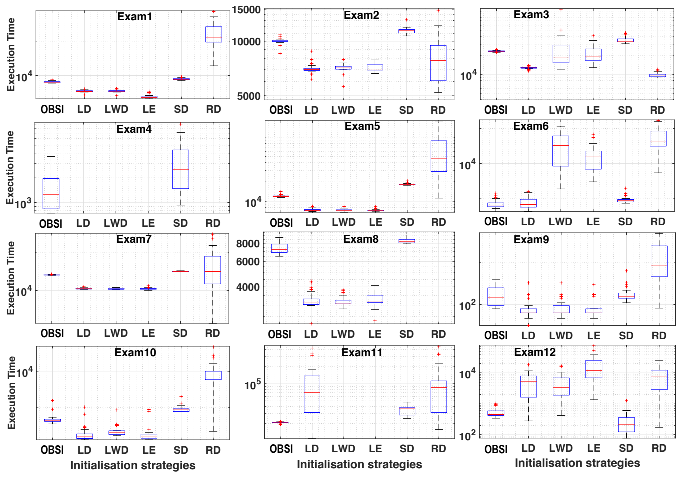
Fig. 3. Boxplots of the log execution time in milliseconds for all initialisation strategies on the ITC 2007 dataset.

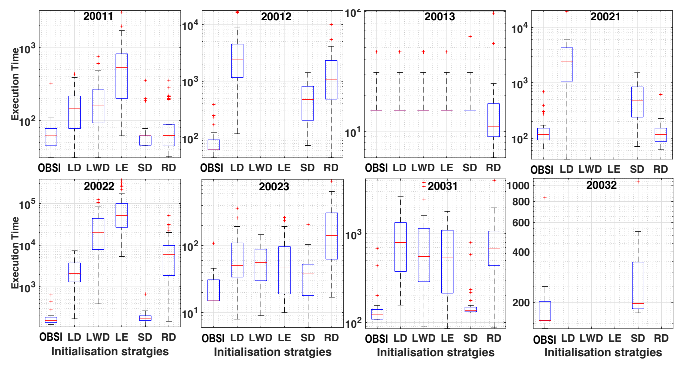
Fig. 4. Boxplots of the log execution time of all initialisation strategies on the Yeditepe dataset. Notation as in Figure 3.

the population for every generation (iteration) is generated by applying a combination of both light and heavy mutation [13]. Light mutation chooses a number of exams randomly from any period in the timetable and attempts to reschedule them at any other period while satisfying all hard constraints. The heavy mutation is performed by disturbing all the exams in one or more periods in the timetable. The periods to be disturbed are determined by the procedure from [13]. After applying the two mutation operators at any time to every parent population member, the subsequent generation's parent population is populated by performing roulette wheel selection on the combined parent and child set.

Results indicate that the OBSI strategy gives not only relatively high quality solutions compared to other initialisation approaches, but that these solutions are effective for seeding an efficient population-based search. The results for most instances using OBSI result in a final solution surpassing the quality of runs using other initialisation strategies (see Tables V and VI for the results of the ITC 2007 and Yeditepe problems respectively).

VI. Conclusion and Future Work

This paper presents a new strategy for constructing initial examination timetables. Compared against other popular initialisation strategies, our results demonstrate that this strategy outperforms graph heuristic and random approaches on nearly all of the ITC 2007 benchmark instances for both quality and diversity and also on some of Yeditepe benchmark instances for quality. Incorporation within a simple EA has demonstrated the advantage that using OBSI to generate an initial population of high quality and diverse solutions provides in the final timetables returned. This is for the same total time cost (initialisation time plus optimisation time). We look forward to integrating this new approach in some recently proposed state-of-the-art timetabling meta-heuristics in future work.

Acknowledgments

Amjad Alsuwaylimi thanks Northern Border University, Saudi Arabia and the Saudi Arabia Cultural Bureau in the UK for his PhD Scholarship.

References

[1] D. de Werra, "An introduction to timetabling," *European journal of operational research*, vol. 19, no. 2, pp. 151–162, 1985.

[2] A. Schaerf, "A survey of automated timetabling," *Artificial intelligence review*, vol. 13, no. 2, pp. 87–127, 1999.

[3] B. Sigl, M. Golub, and V. Mornar, "Solving timetable scheduling problem using genetic algorithms," in *Information Technology Interfaces, 2003. ITI 2003. Proceedings of the 25th International Conference on*. IEEE, 2003, pp. 519–524.

[4] M. W. Carter and G. Laporte, "Recent developments in practical examination timetabling," in *International Conference on the Practice and Theory of Automated Timetabling*. Springer, 1995, pp. 1–21.

[5] E. K. Burke, B. McCollum, A. Meisels, S. Petrovic, and R. Qu, "A graph-based hyper-heuristic for educational timetabling problems," *European Journal of Operational Research*, vol. 176, no. 1, pp. 177–192, 2007.

[6] H. Asmuni, E. K. Burke, J. M. Garibaldi, and B. McCollum, "Fuzzy multiple heuristic orderings for examination timetabling," in *International Conference on the Practice and Theory of Automated Timetabling*. Springer, 2004, pp. 334–353.

[7] P. H. Corr, B. McCollum, M. McGreevy, and P. McMullan, "A new neural network based construction heuristic for the examination timetabling problem," in *Parallel Problem Solving from Nature-PPSN IX*. Springer, 2006, pp. 392–401.

[8] L. Di Gaspero and A. Schaerf, "Tabu search techniques for examination timetabling," in *International Conference on the Practice and Theory of Automated Timetabling*. Springer, 2000, pp. 104–117.

[9] G. Kendall and N. M. Hussin, "A tabu search hyper-heuristic approach to the examination timetabling problem at the mara university of technology," in *International Conference on the Practice and Theory of Automated Timetabling*. Springer, 2004, pp. 270–293.

[10] G. M. White and B. S. Xie, "Examination timetables and tabu search with longer-term memory," in *International Conference on the Practice and Theory of Automated Timetabling*. Springer, 2000, pp. 85–103.

[11] E. K. Burke and J. P. Newall, "Enhancing timetable solutions with local search methods," in *International Conference on the Practice and Theory of Automated Timetabling*. Springer, 2002, pp. 195–206.

[12] J. M. Thompson and K. A. Dowsland, "A robust simulated annealing based examination timetabling system," *Computers & Operations Research*, vol. 25, no. 7-8, pp. 637–648, 1998.

[13] E. K. Burke, J. P. Newall, and R. F. Weare, "A memetic algorithm for university exam timetabling," in *international conference on the practice and theory of automated timetabling*. Springer, 1995, pp. 241–250.

[14] E. K. Burke and J. L. Silva, "The design of memetic algorithms for scheduling and timetabling problems," in *Recent Advances in Memetic Algorithms*. Springer, 2005, pp. 289–311.

TABLE V
MEDIAN EA RESULTS AND MEDIAN ABSOLUTE DEVIATION (MAD) AFTER 8 MINUTES — ITC2007 PROBLEMS. NOTATION AS IN TABLE III.

	OBSI		LD		LWD		LE		SD		RD	
Instance	Median	MAD	Median	MAD	Median	MAD	Median	MAD	Median	MAD	Median	MAD
Exam1	**12955**	509.5	16802	580.5	15920.5	189	15739	136	15508.5	350.5	15731	178.5
Exam2	**2228.5**	273.5	5450.5	532	3961	204.5	3942.5	63.5	3904	112	3823	132.5
Exam3	**17442.5**	82.5	27457.5	100	27364.5	57.5	28166	410.5	27963	108.5	32174.5	1459.5
Exam4	**24345.5**	510	-	-	-	-	-	-	32664.5	957	-	-
Exam5	**7113.5**	240.5	24574.5	1867.5	18245	326.5	16831	81.5	16794.5	610.5	17743.5	124.5
Exam6	**29310**	12	32142.5	47.5	32520	200	31700	120	31567.5	102.5	31812.5	105
Exam7	**18460.5**	12	39860.5	1190.5	35726	129.5	35893.5	554	34587.5	326	32941	230
Exam8	**15837.5**	190	20077	965.5	18097.5	463.5	17490	91.5	17358.5	259	-	-
Exam9	**1530.5**	34.5	2095.5	57	2039.5	18	2020.5	43.5	1973.5	22.5	2006.5	37.5
Exam10	**15219**	446	17948.5	165	17669	55	17509.5	146	17620	148.5	17502.5	133
Exam11	**54594.5**	593.5	87940	4322	-	-	-	-	79346	472.5	89335.5	3855
Exam12	**6340.5**	79.5	7543	95.5	7587	133	7502	82	7458.5	107	7650	182

TABLE VI
MEDIAN EA RESULTS AND MEDIAN ABSOLUTE DEVIATION (MAD) AFTER 8 MINUTES — YEDITEPE PROBLEMS. NOTATION AS IN TABLE III.

	OBSI		LD		LWD		LE		SD		RD	
Instance	Median	MAD	Median	MAD	Median	MAD	Median	MAD	Median	MAD	Median	MAD
YUE20011	**112**	7.5	257	15	240	13	241	9.5	231.5	14	246.5	5.5
YUE20012	**219**	18	355.5	10	-	-	-	-	375	7	364.5	10.5
YUE20013	**41**	7.5	55.5	5	56	6	54	10	56	8	59	7
YUE20021	**181**	14.5	382.5	11	-	-	-	-	374.5	12	364	13
YUE20022	**341.5**	34.5	568.5	21.5	542.5	5.5	551.5	11.5	561.5	11.5	563	15
YUE20023	**71.5**	6	100.5	8.5	98	12	92.5	13.5	107.5	12	107	10.5
YUE20031	**327.5**	18	587.5	11.5	569	21.5	564	12	541.5	24	570	9.5
YUE20032	**647**	21.5	-	-	-	-	-	-	888	13.5	-	-

[15] R. Weare, E. Burke, and D. Elliman, "A hybrid genetic algorithm for highly constrained timetabling problems," *Department of Computer Science*, 1995.

[16] K. A. Dowsland and J. M. Thompson, "Ant colony optimization for the examination scheduling problem," *Journal of the Operational Research Society*, vol. 56, no. 4, pp. 426–438, 2005.

[17] M. Eley, "Ant algorithms for the exam timetabling problem," in *International Conference on the Practice and Theory of Automated Timetabling*. Springer, 2006, pp. 364–382.

[18] S.-C. Chu, Y.-T. Chen, and J.-H. Ho, "Timetable scheduling using particle swarm optimization," in *Innovative Computing, Information and Control, 2006. ICICIC'06. First International Conference on*, vol. 3. IEEE, 2006, pp. 324–327.

[19] E. Burke, Y. Bykov, J. Newall, and S. Petrovic, "A time-predefined local search approach to exam timetabling problems," *Iie Transactions*, vol. 36, no. 6, pp. 509–528, 2004.

[20] M. Caramia, P. Dell'Olmo, and G. F. Italiano, "New algorithms for examination timetabling," in *International Workshop on Algorithm Engineering*. Springer, 2000, pp. 230–241.

[21] L. T. Merlot, N. Boland, B. D. Hughes, and P. J. Stuckey, "A hybrid algorithm for the examination timetabling problem," in *International Conference on the Practice and Theory of Automated Timetabling*. Springer, 2002, pp. 207–231.

[22] A. Muklason, "Hyper-heuristics and fairness in examination timetabling problems," *Philosophy*, 2017.

[23] N. Pillay and E. Özcan, "Automated generation of constructive ordering heuristics for educational timetabling," *Annals of Operations Research*, vol. 275, no. 1, pp. 181–208, 2019.

[24] E. K. Burke, J. P. Newall, and R. F. Weare, "Initialization strategies and diversity in evolutionary timetabling," *Evolutionary computation*, vol. 6, no. 1, pp. 81–103, 1998.

[25] B. McCollum, P. McMullan, E. K. Burke, A. J. Parkes, and R. Qu, "The second international timetabling competition: Examination timetabling track," Technical Report QUB/IEEE/Tech/ITC2007/-Exam/v4. 0/17, Queen's University, Belfast, Tech. Rep., 2007.

[26] B. McCollum, P. McMullan, A. J. Parkes, E. K. Burke, and R. Qu, "A new model for automated examination timetabling," *Annals of Operations Research*, vol. 194, no. 1, pp. 291–315, 2012.

[27] A. J. Parkes and E. Ozcan, "Properties of yeditepe examination timetabling benchmark instances," in *Proceedings of the 8th International Conference on the Practice and Theory of Automated Timetabling*, 2010, pp. 531–534.

[28] D. Corne and P. Ross, "Peckish initialisation strategies for evolutionary timetabling," in *International Conference on the Practice and Theory of Automated Timetabling*. Springer, 1995, pp. 227–240.

[29] ——, "Some combinatorial landscapes on which a genetic algorithm outperforms other stochastic iterative methods," in *AISB Workshop on Evolutionary Computing*. Springer, 1995, pp. 1–13.

[30] E. Burke and J. Newall, "Investigating the benefits of utilising problem specific heuristics within a memetic timetabling algorithm," *Workin Paper NOTTCS-TR-97-6, dept. of Computer Science, University of Nottingham, UK*, 1997.

[31] R. Qu, E. K. Burke, B. Mccollum, L. T. G. Merlot, and S. Y. Lee, "A survey of search methodologies and automated system development for examination timetabling," *Journal of Scheduling*, pp. 55–89, 2009.

[32] E. K. Burke and S. Petrovic, "Recent research directions in automated timetabling," *European Journal of Operational Research*, vol. 140, no. 2, pp. 266–280, 2002.

[33] M. Ayob, A. M. Ab Malik, S. Abdullah, A. R. Hamdan, G. Kendall, and R. Qu, "Solving a practical examination timetabling problem: a case study," in *International Conference on Computational Science and Its Applications*. Springer, 2007, pp. 611–624.

[34] S. K. Jha, "Exam timetabling problem using genetic algorithm," *International Journal of Research in Engineering and Technology*, vol. 3, no. 5, pp. 649–654, 2014.

[35] M. N. M. Kahar and G. Kendall, "The examination timetabling problem at universiti malaysia pahang: Comparison of a constructive heuristic with an existing software solution," *European journal of operational research*, vol. 207, no. 2, pp. 557–565, 2010.

[36] B. McCollum, A. Schaerf, B. Paechter, P. McMullan, R. Lewis, A. J. Parkes, L. D. Gaspero, R. Qu, and E. K. Burke, "Setting the research agenda in automated timetabling: The second international timetabling competition," *INFORMS Journal on Computing*, vol. 22, no. 1, pp. 120–130, 2010.

[37] G. N. Beligiannis, C. N. Moschopoulos, G. P. Kaperonis, and S. D. Likothanassis, "Applying evolutionary computation to the school timetabling problem: The Greek case," *Computers & Operations Research*, vol. 35, no. 4, pp. 1265–1280, 2008.