DOI: 10.6977/IJoSI.202109_6(5).0001
M. C. Chen, S. N. Sze,etc/ Int. J. Systematic Innovation, 6(5), 1-10 (2021)

A Hybrid of Heuristic Orderings and Variable Neighbourhood Descent for a Real Life University Course Timetabling Problem

Mei Ching Chen[1], San Nah Sze[1]*, Say Leng Goh[2] and Sei Ping Lau[1]

[1]Faculty of Computer Science & Information Technology, Universiti Malaysia Sarawak, Jalan Datuk Mohammad Musa, 94300 Kota Samarahan, Sarawak, Malaysia

[2]Optimisation Research Group, Faculty of Computing & Informatics, Universiti Malaysia Sabah Kampus Antarabangsa Labuan, Jln Sungai Pagar, 87000 Labuan, Malaysia

* Corresponding author E-mail: snsze@unimas.my

(Received 8 July 2021; Final version received 22 July 2021; Accepted 7 September 2021.)

Abstract

Academic institutions face timetabling problem every semester. Addressing timetabling problem at academic institutions is a challenging combinatorial optimisation task both in theory and practice. This is due to the size of the problem instances as well as the number of constraints that must be satisfied. Over the years, timetabling problem has attracted many researchers in proposing ways to find an optimal solution. In this paper, we investigate a hybrid of heuristic orderings and variable neighbourhood descent approach in tackling course timetabling problem at the Faculty of Computer Science and Information Technology (FCSIT), Universiti Malaysia Sarawak (UNIMAS). At FCSIT, some events of 4 lecture hours are not evenly spread over minimum working days and some events are conducted until 9 pm. The objectives of the study are to shorten the daily lecture hours and evenly distribute events' lecture. In stage 1, heuristic orderings are utilised to find a feasible solution. In stage 2, a hybrid of heuristic orderings and variable neighbourhood descent approach are utilised to improve the quality of the solution. The proposed algorithm is tested on real-world data instances (semesters 1 and 2 of 2019/2020) of FCSIT, UNIMAS. Results show that certain heuristic ordering (largest degree or the combination of largest degree and largest enrolment) are better than others in generating a feasible solution. In addition, the number of timeslots required by heuristic ordering are less compared to that required by the existing timetabling software. In stage 2, the proposed algorithm manages to achieve soft constraint violations of 0 and 1 for instances for semesters 1 and 2, respectively. However, all HO manage to achieve 0 violation for both instances when the proposed algorithm is executed 30 times. Each neighbourhood structures defined in this study contributes to lowering the soft constraint violations thus ensuring a high-quality timetable. Results show that the order of neighbourhood structures do impact the number of soft constraint (SC1) violations achieved.

Keywords: combinatorial optimisation, course timetabling problem, heuristic orderings, hybrid, variable neighbourhood descent

1. Introduction

Educational timetabling is defined as a task of allocating events such as exams, subjects and courses to rooms and timeslots by fulfilling certain constraints (Tan et al., 2021; Thepphakorn & Pongcharoen, 2020; Tan et al., 2020; Assi et al., 2018). Timetabling is a challenging combinatorial optimisation problem in theory and practice (Schaerf, 1999). Universiti Malaysia Sarawak (UNIMAS) devotes a significant number of resources in developing a feasible and high-quality course scheduleor each faculty. Efficient allocation of courses may result in more effective use of valuable resources (Burke et al., 2005). Therefore, it is crucial to find an optimal configurations for the variables defined to achieve specific objectives (Habashi et al., 2018).

University course timetabling problem (UCTTP) involves allocating a set of courses to limited resources namely lecturers, venues and timeslots by fulfilling certain constraints (Goh et al., 2020; Goh et al., 2019;

Erdeniz & Felfernig, 2018; Goh et al., 2017). UCTTP can be divided into two different categories based on problem settings and requirements, namely curriculum-based course timetabling problem (CBCTTP) and post-enrolment course timetabling problem (PECTTP). UCTTP in UNIMAS is closely related to CBCTTP. Constraints can be classified into two types namely hard and soft. The fulfilment of hard constraints is mandatory in generating a feasible timetable. Meanwhile, the fulfilment of soft constraints is optional but will determine the quality of the timetable generated.

To date, there are many papers on UCTTP either tackling benchmark or real-world UCTTP. For most real-world search problems, automatically generating high-quality solutions is a difficult challenge (Muklason et al., 2019). The objective is to find a feasible timetable with the lowest possible soft constraint violations. Furthermore, the requirements of UCTTP differ across academic institutions as policies and regulations are unique in each institution. This paper is addressing UCTTP at the Faculty of Computer Science and Information Technology (FCSIT), UNIMAS using real-world dataset. We investigate the performance of the hybrid of heuristic ordering (HO) and variable neighbourhood descent (VND). We also compare its performance against the existing timetable which was constructed using commercial timetabling software.

The structure of this paper is as follows. Next section presents the related work on HO and VND. We describe the UCTTP at UNIMAS in Section 3. The proposed algorithm is presented in Section 4. Section 5 presents the numerical results of the research. Finally, conclusions are presented in section 6.

2. Related work

A variety of approaches have been proposed in solving UCTTP. Babaei et al. (2015) had categorized the approaches into five, namely operational research (OR) based techniques, metaheuristic approaches, multi criteria/ objective approaches, intelligent novel approaches and distributed multi agent systems approaches. Each approach has its own advantages. In order to take advantage of each approach, researchers have proposed hybrid approaches in solving UCTTP. Among the hybrid approaches are Hybrid Genetic Algorithm (Akkan & Gülcü, 2018; Matias et al., 2019) and combination of VNS and Tabu Search (Vianna et al., 2020).

VNS is used to solve combinatorial optimisation problem in two phases, namely descent phase and perturbation phase (Hansen et al., 2018). Descent phase helps to achieve local optimum whereas perturbation phase helps to escape from local optimum. VNS is well known for its ability in avoiding traps (local optimum) by considering different neighbourhood structures (Hansen & Mladenovi´c, 2014). Its success has been proven in a wide range of applications with large instances and challenging number of constraints (hard and soft) (Hansen et al., 2018).

Variable Neighbourhood Descent (VND) method was proposed by Borchani et al. (2017) to solve UCTTP for Faculty of Economics and Management Sciences of Sfax in Tunisia. The authors aimed to minimize the total number of holes and the number of isolated lessons. Neighbourhood structures proposed by authors were implemented using simple move. Six real datasets were used as testbeds. Results showed that the proposed algorithm was able to eliminate 52.47% of holes and isolated lessons.

Heuristic ordering (HO) is derived from graph colouring heuristics such as largest degree (LD), saturation degree (SD), largest weighted degree (LWD) and colour degree (CD) (Burke & Petrovic, 2002). In LD, the event with the largest number of conflicts/clashes with other events are assigned first because it is hard to find a valid timeslot for an event that has many conflicts/clashes with other events. LWD associates the number of students with the conflicted events. Therefore, the event with largest number of students is assigned first. In SD, the event with the least number of valid timeslots will be selected for assignment. The valid timeslots for the remaining events are updated in each iteration. Meanwhile, CD takes into consideration the conflict between events to be scheduled with the scheduled events. Priority is given to events with the largest number of conflicts with the scheduled events. These heuristics play an important role in generating initial solutions which quality would then be improved by other methods (Pillay & Özcan, 2019).

Vianna et al. (2020) proposed a hybrid of Variable Neighbourhood Search (VNS) and Tabu Search (TS) in tackling the UCTTP for Federal Fluminense University. Framework for the Implementation of metaheuristics based on Neighbourhood Structure Search (FINESS) framework was used in developing the proposed algorithm which enabled constraints to be added and removed easily. The datasets used in their work were obtained from two undergraduate courses. Results showed that the hybrid produced better solutions than those produced using VNS and TS separately.

Muklason et al. (2019) proposed a Tabu-Variable Neighborhood Search based Hyper-Heuristic algorithm in addressing the UCTTP for the Department of Information Systems, Institut Teknologi Sepuluh Nopember, Indonesia. This approach does not require parameter tuning as required in metaheuristic approaches such as simulated annealing. The algorithm was tested using two real-world datasets from 2017/2018 session. The solution obtained was better in terms of quality compared to the one created manually.

3. Problem description

UNIMAS is one of the public universities in Malaysia established on 24 December 1992. It has 10 faculties offering more than 90 programmes. The timetabling problem in this study is based on the real-world scenario at the Faculty of Computer Science and Information Technology (FCSIT), Universiti Malaysia Sarawak (UNIMAS).

All this while, each faculty's administrator/timetable planner in UNIMAS constructs course timetable based on curriculum (as information on course pre-registration is not available) manually. They started utilising commercial timetabling software in 2014. In this study, we focus in timetable at FCSIT. Courses offered by FCSIT can be divided into a few categories, namely lecture, lecture with tutorial and lecture with lab. These courses are ranged from 2 to 4 credit hours. The credit hour indicates the number of lecture hours per week for a course. It is recommended to split long lecture hours (4 credit hours course) in 2 days. For example, 2 hours on Monday and another 2 hours on Wednesday, which can be represented as "2+2".

In term of venue, FCSIT conducts lectures at either its own venue (available all the times) or shared venue (only available at certain times). Sharing of venues is a common feature showcased by most academic institutions especially if the venue can accommodate many students. Table 1 shows the capacity of the shared and fixed venues.

Table 1 Teaching venues and its capacity

Feature	Usage	Venue	Quantity	Capacity
Shared	Limited	DK	Vary from semester to semester	500
		BS	Vary from semester to semester	150
Fixed (Faculty)	All the time	TMM	1	120
		MM2	1	100
		ARTLNT ISLAB MM1 TL1 TL2	1 1 1 1 1	80 80 80 80 80
		CSLAB NETLAB1	1 1	60 60
		NETLAB2 TR	1 8	40 40

Table 2 shows the timeslots used in this study. Gray area indicates that the timeslots are blocked. No assignment of faculty courses on these timeslots are allowed. Therefore, only 31 timeslots are allocated for the courses to the latest 5pm for Monday, Tuesday and Thursday, and 12pm for Friday. Table 3 shows the data instances used as testbeds for the algorithm proposed. In this study, all individual courses are referred as events.

Table 2 Timeslots

Day\Time	0800-0900	0900-1000	1000-1100	1100-1200	1200-1300	1300-1400	1400-1500	1500-1600	1600-1700
Monday	01	02	03	04	05	06	07	08	09
Tuesday	10	11	12	13	14	15	16	17	18
Wednesday									
Thursday	19	20	21	22	23	24	25	26	27
Friday	28	29	30	31					

Table 3 FCSIT data instances for academic years 2019/2020

Instance	Events	Rooms	Students	Timeslot requirement	Event enrolment
Semester 1 2019/2020	102	19 (fixed) 4 (shared)	1397	31	5073
Semester 2 2019/2020	77	18 (fixed) 2 (shared)	1040	31	3394

The constraints considered are listed below:

Hard constraints

HC1: Lectures taught by the same lecturer cannot be conducted in the same timeslot.

HC2: Only one lecture can be assigned to a venue at a specific timeslot.

HC3: A room assigned to a lecture must be big enough to accommodate the number of students.
HC4: Lectures for all events must be scheduled.
HC5: Blocked timeslots for lectures must be taken into considerations.
HC6: A student can only attend one lecture at a specific timeslot.

Soft constraints:
SC1: Events with 4 lecture hours are evenly spread over minimum working days.

4. Proposed algorithm

In this study, a two-stage heuristic algorithm is proposed. In stage 1, HO (LD) in descending order is utilised to generate a feasible solution by ensuring all hard constraints are satisfied. In stage 2, a hybrid of HO Largest Enrolment (LE) in descending order and VND is proposed to improve the quality of the solution by satisfying soft constraint as much as possible. This proposed algorithm consolidates the features of both HO and VND, which is not attempted in the existing literature reviews. Fig. 1 shows the general framework for solving UCTTP at the FCSIT, UNIMAS.

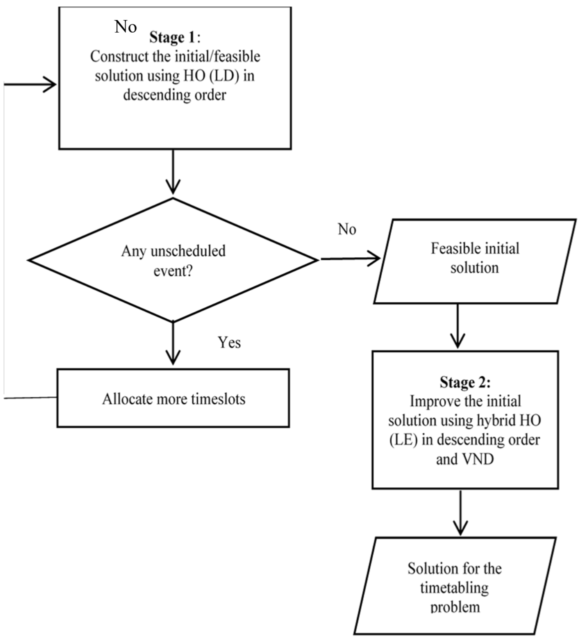


Fig. 1 General framework for solving UCTTP at FCSIT, UNIMAS

In stage 1, HO (LD) in descending order is used for event selection. In LD, the event with the largest number of conflicts/clashes with other events is assigned first. If there is any unscheduled event, more timeslots will be allocated, and stage 1 is repeated to generate a feasible initial solution.

In stage 2, a hybrid of HO (LE) in descending order and variable neighbourhood descent (VND) is used to minimise soft constraint violations. VND is known as best improvement local search. Sequential VND is used where the algorithm will walk through all the neighbourhood structures (NS) in a sequential order. It will start with the first NS and continue with the next one sequentially. Fig. 2 shows the details of this hybrid algorithm. *k* is initialised to 1. The algorithm starts with a feasible initial solution obtained from stage 1. *orderedEvents* is a list of events ordered based on HO (LE) in descending order. For each event in *orderedEvents*, we search the timeslots and venues sequentially until a feasible *candidateSolution* is found. Once it is found, the values of *f(candidateSolution)* and *f(currentSolutiom)* are compared. If the value of *f(candidatesolution)* is less than the value of *f(currentsolution)*, then the *candidatesolution* will be set as the *currentsolution*. Then, the next event in the *orderedEvents* will be considered. Otherwise, if the value of *f(candidatesolution)* is greater than or equal the value of *f(currentsolution)*, then the search to find the next feasible *candidateSolution* will continue. If no feasible *candidateSolution* can be found, the next event in the *orderedEvents* will be considered.

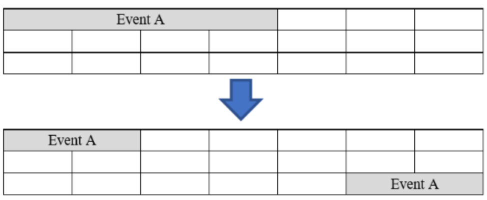
```
PROCEDURE variable neighbourhood descent

Input neighbourhood structures N_k, k=1,2,3,4,5
k ← 1
currentSolution ← initialSolution //initial solution is obtained from stage 1
orderedEvents ← events ordered based on HO

REPEAT
  FOR each e in orderedEvents
    FOR each timeslot
      FOR each venue
        IF feasible (e, timeslot, venue, N_k)
          candidateSolution ← move (e, timeslot, venue, N_k)
          IF f(candidateSolution) < f(currentSolution) THEN
            currentSolution ← candidateSolution
            moved←true;
          END IF
        END IF
        IF moved=true
          Break;
        END IF
      END FOR

      IF moved=true
        Break;
      END IF
    END FOR
  END FOR

    k=k+1
  UNTIL k=5

END PROCEDURE
```

Fig. 2 Hybrid of HO and VND algorithm

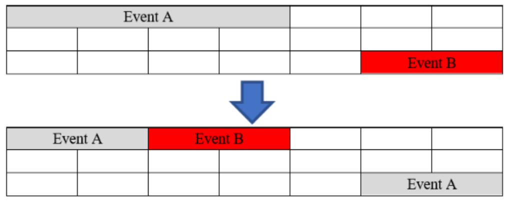
Fig. 3 illustrates the neighbourhood structures (NS) adopted in the proposed algorithm:

- Neighbourhood structure 1 (NS1): attempts to split a course with 4 continuous lecture hours by moving two of its lecture hours to other timeslots.
- Neighbourhood structure 2 (NS2): attempts to split a course with 4 continuous lecture hours by swapping two of its lecture hours with another course with 2 lecture hours.

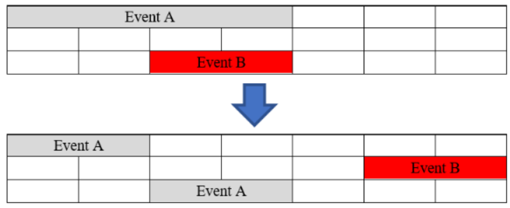
NS1: Split a course with 4 continuous lecture hours by moving two of its lecture hours to other timeslots.



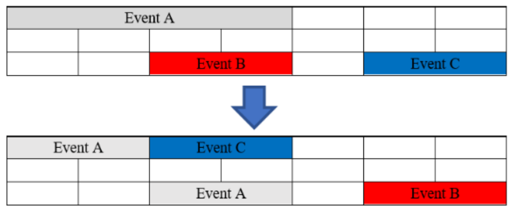
NS2: Split a course with 4 continuous lecture hours by swapping two of its lecture hours with another course with 2 lecture hours.



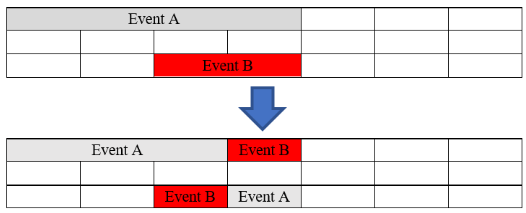
NS3: Split a course with 4 continuous lecture hours by executing 2 moves involving another course.



NS4: Split a course with 4 continuous lecture hours by executing 2 swaps involving 2 other courses.



NS5: Split a course with 4 continuous lecture hours by swapping one of its lecture hours with another course or by moving one lecture hour to other timeslot.



**Note: Column – timeslot, Row - venue*

Fig. 3 Neighbourhood structures: NS1 to NS5

- Neighbourhood structure 3 (NS3): attempts to split a course with 4 continuous lecture hours by executing 2 moves involving another course.
- Neighbourhood structure 4 (NS4): attempts to split a course with 4 continuous lecture hours by executing 2 swaps involving 2 other courses.
- Neighbourhood structure 5 (NS5): attempts to split a course with 4 continuous lecture hours by swapping one of its lecture hours with another course or by moving one lecture hour to other timeslot.

NS3 and NS4 are new neighbourhood structures introduced and included in the proposed algorithm to further improve the quality of the timetable. As shown in Fig. 3, Event A is selected from a list ordered by HO (LE) in descending order. Whereas Event B and Event C are selected when the timeslots and venues are scanned sequentially. The five different NS are used to improve the connectivity of the search space and therefore the quality of the solution. If the resulting solution from applying the NS is feasible (not breaching any hard constraints), it is returned as a *candidateSolution* and evaluated for acceptance.

5. Numerical result

The algorithms are coded using visual basic (VB.Net). We use Microsoft Access as the database management software. Table 4 shows the distance to feasibility (number of unallocated courses) for initial solutions generated using different HO in stage 1. In LD, the event with the largest number of conflicts/clashes with other events is assigned first. Whereas in LE, the event with the largest number of enrolments is assigned first. In (LD+LE), both LD and LE are taken into considerations when allocating events to timetable.

Table 4 Distance to feasibility (number of unallocated courses) for initial solutions generated using different HO in stage 1. $N = 30$ runs.

HO	Instance					
	Semester 1 (31 timeslots)		*Semester 1 (35 timeslots)*		*Semester 2 (31 timeslots)*	
	Best	Average	Best	Average	Best	Average
LD Ascending	11	13.43	6	8.87	5	7.23
LD Descending	1	2.03	0**	0.27	0**	0.43
LE Ascending	12	14.20	7	8.10	6	7.47
LE Descending	5	5.80	1	1.73	1	1.00
(LD + LE) Ascending	12	15.80	8	11.07	6	7.13
(LD + LE) Descending	2	3.93	0	0.60	0	0.27
Random	8	8.00	5	5.00	3	3.00

Note: ** Selected HO ordering in stage 1 (used as feasible initial solution in stage 2)

As shown in Table 4, both LD and (LD + LE) in descending order manage to find feasible solutions for semester 2's instance using 31 timeslots. However, they failed to do so for semester 1's instance using the same number of timeslots. This is because the instance for semester 1 is larger than that of semester 2 in terms of events, students and course enrolment. There are 77 events (282 lecture hours) for semester 2, compared to 102 events (347 lecture hours) for semester 1. Furthermore, FCSIT has limited timeslots, since Wednesday and Friday afternoons are blocked. As the size of the data instance grows larger, this makes allocating lecture hours a challenging task. Nevertheless, the algorithm manages to find feasible solution for semester 1's instance when the number of allocated timeslots is increased to 35 (6 pm). In a comparison, the solution generated by existing timetabling software required 48 timeslots (9 pm) to achieve feasibility.

Table 5 shows the number of timeslots required by the existing UNIMAS timetabling software in obtaining a feasible solution for semester 1's instance. A total of 48 timeslots required. As shown, some of the lectures are conducted until 9 pm. This will consume extra resources such as electricity cost. One the other hand, Table 6 shows the timeslots required by our approach in generating a feasible solution for the same instance. A total of 35 timeslots required, where the latest lectures end at 6pm. Comparatively, there are only 3 timeslots compared to 12 timeslots from existing timetable (Table 5) are scheduled after 5pm.

Table 7 shows the number of soft constraint (SC1) violations of the proposed VND algorithm with different HO for semester 1's instance. From the table, the lowest number of soft constraint (SC1) violations achieved is 0 using LE Ascending, LD Descending, LE Descending, (LD+LE) Descending and random ordering. The number 0 indicates that all the courses can be spread over minimum working days (2 days). Note that the number of allocated timeslots is 35. Each NS defined in this study contributes to lowering the soft constraint violations thus ensuring a higher-quality timetable.

Table 8 shows the number of soft constraint (SC1) violations of the proposed VND algorithm with different HO for semester 2's instance. From the table, the lowest number of soft constraint (SC1) violations achieved is 1 using LE Ascending, LD Descending, LE Descending, (LD+LE) Descending and random ordering. The number 1 indicates that there is one course which cannot be spread over minimum working days.

Table 5 The number of timeslots required by the existing timetabling software (semester 1's instance).

Day\Time	0800 - 0900	0900-1000	1000-1100	1100-1200	1200-1300	1300-1400	1400-1500	1500-1600	1600-1700	1700-1800	1800-1900	1900-2000	2000-2100
Monday	01	02	03	04	05	06	07	08	09	10	11	12	13
Tuesday	14	15	16	17	18	19	20	21	22	23	24	25	26
Wednesday													
Thursday	27	28	29	30	31	32	33	34	35	36	37	38	39
Friday	40	41	42	43	44	45	46	47	48				

Table 6 The number of timeslots required by our approach (semester 1's instance).

Day\Time	0800-0900	0900-1000	1000-1100	1100-1200	1200-1300	1300-1400	1400-1500	1500-1600	1600-1700	1700-1800
Monday	01	02	03	04	05	06	07	08	09	10
Tuesday	11	12	13	14	15	16	17	18	19	20
Wednesday										
Thursday	21	22	23	24	25	26	27	28	29	30
Friday	31	32	33	34	35					

Table 7 The number of soft constraint (SC1) violations of the proposed VND algorithm with different HO (semester 1's instance) with 35 timeslots.

	LD Ascending	LE Ascending	(LD+LE) Ascending	LD Descending	LE Descending	(LD+LE) Descending	Random
Initial solution	32	32	32	32	32	32	32
NS1	6	5	6	5	5	5	6
NS1 + NS2	4	3	4	3	3	3	5
NS1 + NS2 + NS3	3	1	3	3	3	3	3
NS1 + NS2 + NS3 + NS4	3	1	3	2	3	2	3
NS1 + NS2 + NS3 + NS4 + NS5	1	0	1	0	0	0	0

It is hard to spread a course with many students when; 1) the number of venues (high seating capacity) that can fit the students is limited, 2) the timetable is tight (as not many vacant places are available, and it is difficult to satisfy the conflict requirement).

In order to achieve 0 soft constraint (SC1) violations, the number of allocated timeslots needs to be increased to 32. As shown in Table 9, the lowest number of soft constraint (SC1) violations achieved is 0 which is achieved using LE Ascending, LE Descending and (LD+LE) Descending. By increasing the number of allocated timeslots, more venues (high seating capacity) that can accommodate large number of students, are made available. This increases the chances of a course with many students being spread over minimum working days.

Table 10 shows the number of soft constraint (SC1) violations of the proposed VND algorithm with different HO for semester 2's instance with 31 timeslots when NS is applied in different order. From the table, the lowest number of soft constraint (SC1) violations achieved is 0 by using LD Ascending. This shows the order of NS is one of the parameters which will impact the number of soft constraint (SC1) violations achieved in this study.

In further analysis, the proposed algorithm is executed 30 times for both instances (semesters 1 and 2). The aim is to find the best and average of soft constraint (SC1) violations. Each run uses different initial solution generated from LD Descending (stage 1). As shown in Table 11, all HO manage to achieve 0 violation for both instances.

Table 8 The number of soft constraint (SC1) violations of the proposed VND algorithm with different HO (semester 2's instance) with 31 timeslots.

	LD Ascending	LE Ascending	(LD+LE) Ascending	LD Descending	LE Descending	(LD+LE) Descending	Random
Initial solution	38	38	38	38	38	38	38
NS1	11	12	10	8	8	8	10
NS1 + NS2	7	8	9	8	8	8	7
NS1 + NS2 + NS3	6	5	7	5	5	5	7
NS1 + NS2 + NS3 + NS4	5	5	5	5	5	5	6
NS1 + NS2 + NS3 + NS4 + NS5	2	1	2	1	1	1	1

Table 9 The number of soft constraint (SC1) violations of the proposed VND algorithm with different HO (semester 2's instance) after the number of allocated timeslots is increased to 32.

	LD Ascending	LE Ascending	(LD+LE) Ascending	LD Descending	LE Descending	(LD+LE) Descending	Random
Initial solution	39	39	39	39	39	39	39
NS1	9	12	8	8	8	8	8
NS1 + NS2	6	8	7	8	8	8	7
NS1 + NS2 + NS3	5	5	5	5	5	5	7
NS1 + NS2 + NS3 + NS4	4	5	4	5	5	5	6
NS1 + NS2 + NS3 + NS4 + NS5	1	0	1	1	0	0	1

Table 10 The number of soft constraint (SC1) violations of the proposed VND algorithm with different HO (semester 2's instance) with 31 timeslots when NS are applied in different order.

	LD Ascending	LE Ascending	(LD+LE) Ascending	LD Descending	LE Descending	(LD+LE) Descending	Random
Initial solution	38	38	38	38	38	38	38
NS2	6	8	8	5	5	7	6
NS2 + NS4	3	6	6	3	3	3	6
NS2 + NS4 + NS1	3	6	6	3	3	3	6
NS2 + NS4 + NS1 + NS3	3	5	6	3	3	3	4
NS2 + NS4 + NS1 + NS3 + NS5	0	2	2	1	1	1	2

Table 11 The number of soft constraint (SC1) violations of the proposed VND algorithm with different HO. N= 30 runs.

HO	Instance			
	Semester 1 (35 timeslots)		*Semester 2 (31 timeslots)*	
	Best	Average	Best	Average
LD Ascending	0	1.13	0	1.53
LD Descending	0	0.63	0	1.07
LE Ascending	0	1.23	0	1.57
LE Descending	0	0.40	0	0.77
(LD + LE) Ascending	0	1.00	0	1.57
(LD + LE) Descending	0	0.80	0	1.03
Random	0	0.47	0	1.17

6. Conclusion

We address the UCTTP at the FCSIT, UNIMAS utilising a 2-stage approach. In stage 1, HO is used to find a feasible solution. In stage 2, a hybrid of HO and VND is used to improve the quality of the solution. The proposed algorithm is tested on real-world data instances for semester 1 and 2 of 2019/2020.

LD Descending and (LD+LE) Descending ordering manage to generate feasible solutions for both the instances when the number of allocated timeslots is increased to 35, which is less compared to the number of allocated timeslots (48) required by the existing timetabling software.

We also compare different HO and NS in VND. VND works best with LE Ascending, LD Descending, LE Descending, (LD+LE) Descending and random ordering for both the instances by executing single iteration. The proposed algorithm manages to achieve soft constraint (split a course with 4 continuous lecture hours over minimum working days) violations of 0 and 1 for instances for semesters 1 and 2, respectively. However, all HO manage to yield 0 violation for both instances after 30 iterations of the proposed algorithm. Results show that the order of NS also will impact the number of soft constraint (SC1) violations achieved in this study. Future research may focus on other soft constraints such as one-hour lunch break and minimising isolated events, which are also the concern of most universities.

Acknowledgements

The work was supported by Hadiah Latihan Persekutuan (HLP) scholarship from Ministry of Higher Education Malaysia.

References

Akkan, C., & Gülcü, A. (2018). A bi-criteria hybrid Genetic Algorithm with robustness objective for the course timetabling problem. Computers and Operations Research, 90, 22–32. https://doi.org/10.1016/j.cor.2017.09.007

Assi, M., Halawi, B., & Haraty, R. A. (2018). Genetic Algorithm Analysis using the Graph Coloring Method for Solving the University Timetable Problem. Procedia Computer Science, 126, 899–906. https://doi.org/10.1016/j.procS.2018.08.024

Babaei, H., Karimpour, J., & Hadidi, A. (2015). A survey of approaches for university course timetabling problem. Computers and Industrial Engineering, 86, 43–59. https://doi.org/10.1016/j.cie.2014.11.010

Borchani, R., Elloumi, A., & Masmoudi, M. (2017). Variable neighborhood descent search based algorithms for course timetabling problem: Application to a Tunisian University. Electronic Notes in Discrete Mathematics, 58, 119–126. https://doi.org/ 10.1016 /j.endm.2017.03.016

Burke, E., Curtois, T., Post, G., Qu, R., Veltman, B., Burke, C. E., Curtois, T., Post, G., Qu, R., & Veltman, B. (2005). University of Nottingham Jubilee Campus Computer Science Technical Report No . NOTTCS-TR-2005-9 A Hybrid Heuristic Ordering and Variable Neighbourhood Search for the Nurse Rostering Problem A Hybrid Heuristic Ordering and Variable Neighbourhood Search for.

Burke, E. K., & Petrovic, S. (2002). Recent research directions in automated timetabling. European Journal of Operational Research, 140(2), 266–280. https://doi.org/10.1016/S0377-2217(02)00069-3

Erdeniz, S. P., & Felfernig, A. (2018). OCSH : Optimized Cluster Specific Heuristics for The University Course Timetabling Problem. 0–5.

Goh, S. L., Graham, G., Sabar, N. R., & Abdullah, S. (2020). An effective hybrid local search approach for the post enrolment course timetabling problem. OPSEARCH. https://doi.org/10.1007/s12597-020-00444-x

Goh, S. L., Kendall, G., & Sabar, N. R. (2017). Improved local search approaches to solve the post enrolment course timetabling problem. European Journal of Operational Research, 261(1), 17–29. https://doi.org/10.1016/j.ejor.2017.01.040

Goh, S. L., Kendall, G., & Sabar, N. R. (2019). Simulated annealing with improved reheating and learning for the post enrolment course timetabling problem. Journal of the Operational Research Society, 70(6), 873–888. https://doi.org/10.1080/01605682.2018.1468862

Habashi, S. S., Yousef, A. H., Salama, C., & Fahmy, H. M. A. (2018). Adaptive Diversifying Hyper-Heuristic Based Approach for Timetabling Problems. 2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), 259–266.

Hansen, P., & Mladenovi´c, N. (2014). Variable neighborhood search. In Edmund K. Burke & G. Kendall (Eds.), Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques (pp. 313–338). Springer New York Heidelberg Dordrecht London. https://doi.org/10.1007/978-1-4614-6940-7_12

Hansen, P., Mladenovi´c, N., Brimberg, J., & Pérez, J. A. M. (2018). Variable Neighborhood Search. In International Series in Operations Research & Management Science (Vol. 272). Springer New York LLC. https://doi.org/10.1007/978-3-319 -91086-4_3

Matias, J. B., Fajardo, A. C., & Medina, R. P. (2019). A hybrid genetic algorithm for course scheduling and teaching workload management. 2018 IEEE 10th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management, HNICEM 2018, 1–6. https://doi.org/10.1109/HNICEM.2018.8666332

Muklason, A., Irianti, R. G., & Marom, A. (2019). Automated course timetabling optimization using tabu-variable neighborhood search based hyper-heuristic algorithm. Procedia Computer Science, 161, 656–664. https://doi.org/10.1016/j.procs. 2019.11.169

Pillay, N., & Özcan, E. (2019). Automated generation of constructive ordering heuristics for educational timetabling. Annals of Operations Research, 275(1), 181–208. https://doi.org/10.1007/s10479-017-2625-x

Schaerf, A. (1999). Survey of automated timetabling. Artificial Intelligence Review, 13(2), 87–127. https://doi.org/10.1023/A:1006576209967

Tan, J. S., Leng, S., 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(May 2020), 113943. https://doi.org/10.1016/j.eswa.2020.113943

Tan, J. S., Say, T., Goh, L., Sura, S., Kendall, G., & Sabar, N. R. (2020). Hybrid particle swarm optimization with particle elimination for the high school timetabling problem. Evolutionary Intelligence, 0123456789. https://doi.org/10.1007/s12065-020-00473-x

Thepphakorn, T., & Pongcharoen, P. (2020). Performance Improvement Strategies on Cuckoo Search Algorithms for Solving the University Course Timetabling Problem. Expert Systems With Applications, 113732. https://doi.org/10.1016/j.eswa.2020.113732

Vianna, D. S., Martins, C. B., Lima, T. J., Vianna, M. de F. D., & Meza, E. B. M. (2020). Hybrid VNS-TS heuristics for University Course Timetabling Problem. Brazilian Journal of Operations & Production Management, 17(1), 1–20. https://doi.org/10.14488/bjopm.2020.014

AUTHOR BIOGRAPHIES

Mei Ching Chen is a PhD student in the Faculty of Computer Science & Information Technology, Universiti Malaysia Sarawak. She obtained her Master of Education in Technical and Vocational Study and Bachelor of IT in Computational Science. Her current research interest is hybrid algorithms with specific interest in timetabling problems.

Dr. San Nah Sze is presently working as a senior lecturer at the Faculty of Computer Science & Information Technology, Universiti Malaysia Sarawak (UNIMAS). She received her Doctor of Philosophy (PhD) in University of Sydney in 2011. She has more than 10 years of experience in research and development (R&D). Her research
interests include educational timetabling, vehicle routing, heuristics and meta-heuristics. She had presented at numerous conferences and published her work in ISI and Scopus publications.

Dr. Say Leng Goh is an academic with the Faculty of Computing and Informatics, Universiti Malaysia Sabah, Malaysia. He graduated with a first class B.I.T. degree from Universiti Malaysia Sabah. He received his M.Sc. degree and Ph.D. degree from Imperial College London and University of Nottingham, respectively. He has published in various top-tier journals such as European Journal of Operational Research, The Operational Research Society, Expert Systems with Applications etc. His current research interests include artificial intelligence, operational research, discrete optimisation, meta-heuristics, timetabling and scheduling.

Dr. Lau Sei Ping is a senior lecturer at Faculty of Computer Science & Information Technology (FCSIT), Universiti Malaysia Sarawak (UNIMAS). He received his Doctor of Philosophy (PhD) from University of Southampton, UK at 2015. His in-depth knowledge in computer science had given him the trust to handle wide varieties of course in UNIMAS including commercial postgraduate program courses. Besides the teaching and learning activities, he also actively participates and had more than 15 years experience in research and development (R&D) activities. The area of interests includes wireless sensor networks, applied computing, and cybersecurity.