Australian Journal of Basic and Applied Sciences, 5(10): 1393-1397, 2011
ISSN 1991-8178

Designing A Generic Layout For Benchmark Scheduling Problems

[1]Aftab Ahmed, [2]Mazhar Ali, [3]Mirza Aamir Mehmood, [4]Abdul Hussain Shah Bukhari

[1]Department of Computer Science, BUITEMS, Quetta.
[2]Department of Information Technology, BUITEMS, Quetta.
[3]Department of Computer Science, BUITEMS, Quetta.
[4]Faculy of Information and Communication Tech., BUITEMS, Quetta.

Abstract: scheduling the academic events is a significant branch of combinatorial optimization and which is specifically based on resources allocation. The laborious problem indeed provides a broad scope of research applicability. A neatly originated scheduling may be greatly supportive to streamline an academic tenure. A range of solving techniques have been successfully developed and employed over scheduling problems in past years; however their effectiveness primarily depends on efficient layout designing and problem representation. Throughout entire computing process the events move back and forth within layout. This Research work is inclined to develop a layout or container, in order to accommodate the scheduling events according to predefined constraints conveniently. Computationally the layout serves as groups of integrated subspaces to shape a complete search space. Consequential advantages of research work are to setup a well-organized events deployment and maxim utilization of resources.

Key words: Scheduling Problem, Layout, Constraints.

INTRODUCTION

Scheduling the events is a tremendously vital job for regulating the educational cycle in all the academic institutes. It is a prominent NP-hard problem in the domains of Operation Research and Artificial Intelligence. There is direct proportion between the time & space complexity and dimensions of problem. Classical solving techniques or manmade solutions need plenteous time and hard work to accomplish the task because of dependant and diverse parameters which are believed to be satisfied anyhow. More or less in the real world scheduling cases, it is somewhat not possible to come up with entirely constraints free solution because one contented constraint may cause of violation to any other soft or hard constraint. However, each adapted solving technique has to negotiate over a small amount of unsolved violations. Solving Scheduling problem thus entail a concerted brainstorming, skillfulness and years of experience in the relevant field. Definitely the research field demands more study, investigation and classification of the precise techniques to formulate more effectual and proficient automated University scheduling. On the surface, scheduling is gridiron demonstration of academic resources containing faculty, Curriculums, and enrolled groups of students situated together in crisscrossed slots and columns usually titled by session-time and venue. Scheduling provides an ordered chain of events where resources come to congregate in fixed time of intervals.

The events container or scheduling layout is very significant for detecting and solving all type of constraints. In fact, it provides a frame of reference from one event to other. The layout furnishes stand alone platforms or partial state-spaces for distinguished heuristics to shape up a complete search space. A number of vital objectives have been achieved by designing specific layout. Following sections are discussed for the mathematical and logical aspects of layout designing.

Related Work:

The typical timetabling (Maciej Norberciak, Oct 2006) problem includes assigning a set of activities, actions, events (e.g. work shifts, duties, classes) to a set of resources (e.g. physicians, teachers, rooms) and time periods, fulfilling a set of constraints of various types. Furthermore, Goltz (H.J. Goltz and Matzke, 1999) defines Timetabling, It is also a difficult constraint satisfaction optimization problem and in fact a NP-complete problem. In addition, Cooper and Kingston illustrated the timetabling problem is known to be an NP-hard optimization problem (Cooper and Kingston, 1996).

The designed Layout is very supportive for low level heuristics. By and Large, sequential heuristics are remarkably straightforward to implement for all kind of scheduling problem although they also are observed not to reach up to an optimal solution with respect to the satisfy the possible amount of soft constraints. In order to maintain the quality level, hybridization of various techniques has been raised up in current research trend. For instance, Burke *et al*., (E.K. Burke *et al*., 1998) produced partial solution by the evolutionary algorithm with heuristic ordering. Experiments reveal systematic initialization of a population provided better

Corresponding Author: Aftab Ahmed, Department of Computer Science, BUITEMS, Quetta.
Cell# +92-333-7815354
E-mail: aftab.ahmed@buitms.edu.pk,

results than random initialization. Carter and Johnson(Carter and Johnson., 2001) implemented the clique initialization method and subsequently used sequential heuristics to schedule rest of events. Carter *et al*. (M.W. Carter *et al.*, 1994) investigated the work efficiently of sequential heuristics incorporating backtracking procedure. The backtracking procedure locates the empty slot with almost compete accuracy in a reasonable amount of time. Carter *et al*., (M.W. Carter and Laporte, 1996) extended the idea and drawn the results, they had proved that the use of Backtracking Search can decrease the number of timeslots required for the schedule with compare to standalone sequential heuristics.

The Layout designing plays very important role for solving all type constraints under various academic environments. Aftab and Li (Aftab Ahmed and Li, 2010a,b) successfully have solved the real world dataset using efficient layout designing and proper deployment of dataset. They have used various low level heuristics including Backtracking Search and Min-Conflict for solving Hard and Soft Constraints separately. In (Aftab Ahmed *et al.*, 2011) used all the novel features of well designed layout and solved the bunch of prominent benchmark scheduling instances with various complexity scales. The Hyper-Heuristics was employed over set of low level operators. Generic Layout supports tremendously each method with equal efficiency to wipe out the constraints from problem instances.

Layout Designing:

An efficiently designed scheduling layout not just makes dataset comprehensible apparently, but greatly helps to converge error-free order of events. Each event-slot is a cross point between *period* and *location.* In that consequence, there is no chance of resources replication constraint e.g. Single class room allocation for two courses.

Layout DesigningLogic :

The Layout algorithm generates container or compound storage positions for schedule. The logic is built upon various mathematical equations applied on counter variable that generates three assorted iterating sequences which helps to manage the dataset. The Fig. **1** illustrates the values generated by Layout Algorithm the day number is repeated in all sessions (columns) up to certain number of range in rows than increases from top to bottom in sequential order. The day sequence has been generated by the step (3) of the layout algorithm. Second variable PEROID varies from left to right throughout the layout representing same span of time. Step (4) is responsible for this iterative sequence with the help of modulo operation.

Algorithm: Designing layout for Benchmark Scheduling.

```
Def  layout( ):
1.    For Counter  In range(self.N):
2.    Days =  (1+ Co nter  $\div \sum_1^n$ Sessions $\times \sum_1^n$ Rooms)
3.    Periods =  (1 + (Counter  $\div \sum_1^n$ Rooms) % $\sum_1^n$ Sessions)
4.    Class_Rooms = (1 + Counter % $\sum_1^n$ Rooms)
5.    Layout [Days, Periods, Class_Rooms]  = None
6.    End
```

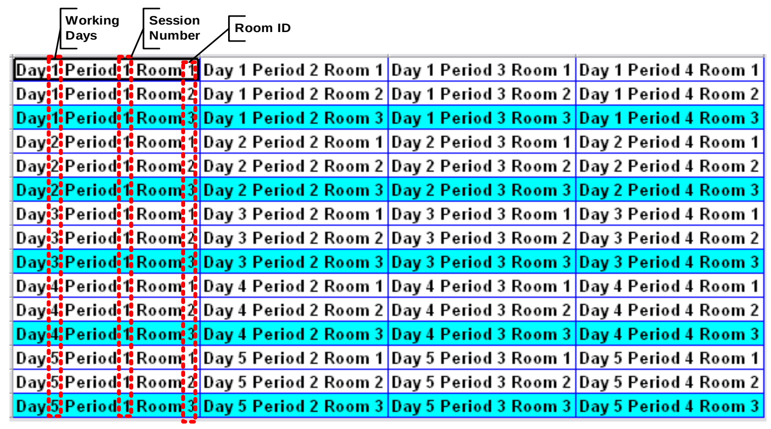
From all sequences emerges to construct a unique compound index number for a single entity. Technically there are two major parts of layout designing, first the composite but unique index number or subscript that can contain immutable information regarding days, sessions and room number. Python Dictionary is used that can work well for such kind of complicated and compound piece of information which also works as subscript or index in data structure.

Fig. 1 shows the layout example; it can be noticed, for the single entry of event there is only one available placement. The storage side also contains some merged values, a dictionary (Hash table) is used as layout, consisting information [Day, Session, Room]. Another Python data structure the LIST supports event substance, EventList [Groups, Teacher, Course, Related Information]. Gradually, each element from event List move into layout in case of finding valid placement.

Fig. 2 portrays the information contains each cell, Row label accumulates Day and Room information, in contrast column label shows the sessions or periods separate each stack of events from each other. Information can be selected to display on layout as per requirement.

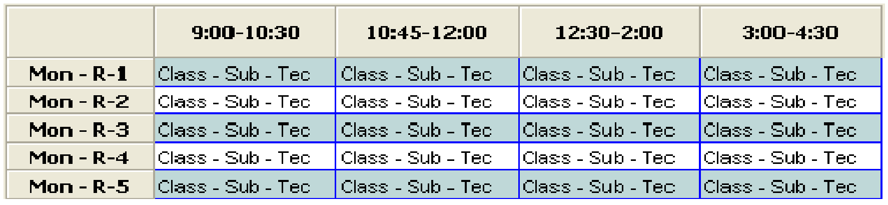
Layout Characteristics :

The designed Layout is result of vast study, exploration and scope of problem. The design phase integrates several features as described below.


Working Days
Session Number
Room ID

Day 1 Period 1 Room 1	Day 1 Period 2 Room 1	Day 1 Period 3 Room 1	Day 1 Period 4 Room 1
Day 1 Period 1 Room 2	Day 1 Period 2 Room 2	Day 1 Period 3 Room 2	Day 1 Period 4 Room 2
Day 1 Period 1 Room 3	Day 1 Period 2 Room 3	Day 1 Period 3 Room 3	Day 1 Period 4 Room 3
Day 2 Period 1 Room 1	Day 2 Period 2 Room 1	Day 2 Period 3 Room 1	Day 2 Period 4 Room 1
Day 2 Period 1 Room 2	Day 2 Period 2 Room 2	Day 2 Period 3 Room 2	Day 2 Period 4 Room 2
Day 2 Period 1 Room 3	Day 2 Period 2 Room 3	Day 2 Period 3 Room 3	Day 2 Period 4 Room 3
Day 3 Period 1 Room 1	Day 3 Period 2 Room 1	Day 3 Period 3 Room 1	Day 3 Period 4 Room 1
Day 3 Period 1 Room 2	Day 3 Period 2 Room 2	Day 3 Period 3 Room 2	Day 3 Period 4 Room 2
Day 3 Period 1 Room 3	Day 3 Period 2 Room 3	Day 3 Period 3 Room 3	Day 3 Period 4 Room 3
Day 4 Period 1 Room 1	Day 4 Period 2 Room 1	Day 4 Period 3 Room 1	Day 4 Period 4 Room 1
Day 4 Period 1 Room 2	Day 4 Period 2 Room 2	Day 4 Period 3 Room 2	Day 4 Period 4 Room 2
Day 4 Period 1 Room 3	Day 4 Period 2 Room 3	Day 4 Period 3 Room 3	Day 4 Period 4 Room 3
Day 5 Period 1 Room 1	Day 5 Period 2 Room 1	Day 5 Period 3 Room 1	Day 5 Period 4 Room 1
Day 5 Period 1 Room 2	Day 5 Period 2 Room 2	Day 5 Period 3 Room 2	Day 5 Period 4 Room 2
Day 5 Period 1 Room 3	Day 5 Period 2 Room 3	Day 5 Period 3 Room 3	Day 5 Period 4 Room 3

Fig. 1: Layout Positions Generated.



	9:00-10:30	10:45-12:00	12:30-2:00	3:00-4:30
Mon - R-1	Class - Sub - Tec	Class - Sub - Tec	Class - Sub - Tec	Class - Sub - Tec
Mon - R-2	Class - Sub - Tec	Class - Sub - Tec	Class - Sub - Tec	Class - Sub - Tec
Mon - R-3	Class - Sub - Tec	Class - Sub - Tec	Class - Sub - Tec	Class - Sub - Tec
Mon - R-4	Class - Sub - Tec	Class - Sub - Tec	Class - Sub - Tec	Class - Sub - Tec
Mon - R-5	Class - Sub - Tec	Class - Sub - Tec	Class - Sub - Tec	Class - Sub - Tec

Fig. 2: Layout Demonstration of Single Day.

Efficiency:

Table 1 describes the hard and soft constrains violation in test dataset. The events were initialized in random manner. It is clear from picture that all the constraints have been violated as per dataset complexity. The null value of HC2 (Conflict Constraint) is evidence of designing efficiency of layout despite random input.

Table 1: Random Data Initialization over Layout.

No.	Constraint	Random Initialization
1	SC_1	25
2	SC_2	22
3	SC_3	17
4	SC_4	15
5	SC_5	13
6	HC_1	12
7	HC_2	00
8	HC_3	11
Soft Volitions		92
Hard Volitions		23
Soft Penalty		613
Hard Penalty		230000
Total Penalty		230613

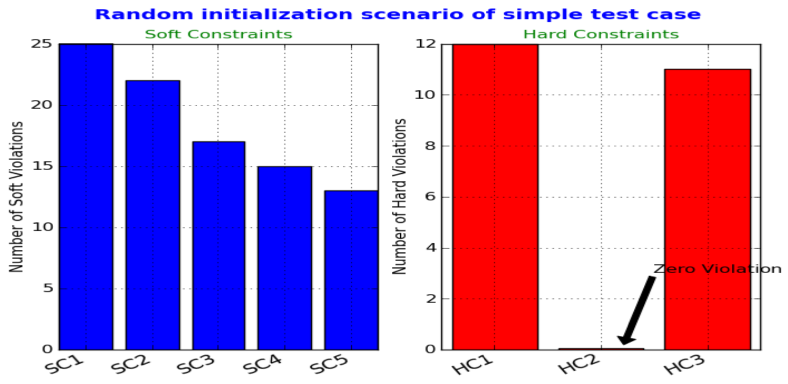
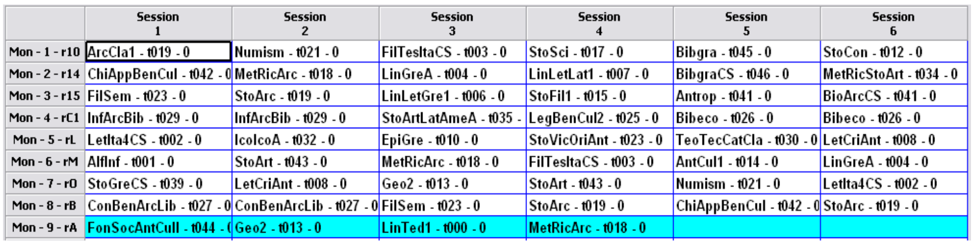


Fig. 3: Evaluation Results of Random Initialization of test dataset.

Fig. 3 shows an alternative way of presenting the random distribution of sample dataset. Subplot named 'Soft Constraints' depicts a visible difference with respect to frequency of soft violation occurrence. On the other hand subplot 'Hard Constraints' in each Column illustrates the hard violations. The second column represents the zero HC2 (Venue Conflict) violation on the layout. In fact, preemptively the hard constraint (HC2) has been handled by single placement in each layout cell.

Fig. 4 shows the extent of dataset distribution.

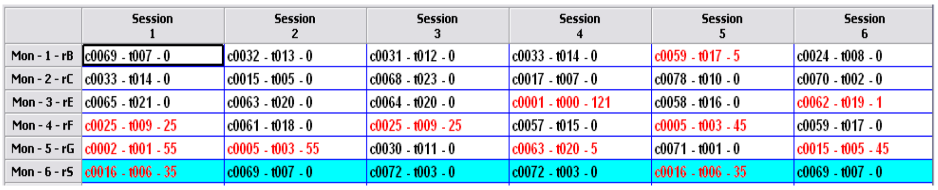


	Session 1	Session 2	Session 3	Session 4	Session 5	Session 6
Mon - 1 - r10	ArcCla1 - t019 - 0	Numism - t021 - 0	FilTesltaCS - t003 - 0	StoSci - t017 - 0	Bibgra - t045 - 0	StoCon - t012 - 0
Mon - 2 - r14	ChiAppBenCul - t042 - 0	MetRicArc - t018 - 0	LinGreA - t004 - 0	LinLetLat1 - t007 - 0	BibgraCS - t046 - 0	MetRicStoArt - t034 - 0
Mon - 3 - r15	FilSem - t023 - 0	StoArc - t019 - 0	LinLetGre1 - t006 - 0	StoFil1 - t015 - 0	Antrop - t041 - 0	BioArcCS - t041 - 0
Mon - 4 - rC1	InfArcBib - t029 - 0	InfArcBib - t029 - 0	StoArtLatAmeA - t035 -	LegBenCul2 - t025 - 0	Bibeco - t026 - 0	Bibeco - t026 - 0
Mon - 5 - rL	LetIta4CS - t002 - 0	IcoIcoA - t032 - 0	EpiGre - t010 - 0	StoVicOriAnt - t023 - 0	TeoTecCatCla - t030 - 0	LetCriAnt - t008 - 0
Mon - 6 - rM	AlfInf - t001 - 0	StoArt - t043 - 0	MetRicArc - t018 - 0	FilTesltaCS - t003 - 0	AntCul1 - t014 - 0	LinGreA - t004 - 0
Mon - 7 - rO	StoGreCS - t039 - 0	LetCriAnt - t008 - 0	Geo2 - t013 - 0	StoArt - t043 - 0	Numism - t021 - 0	LetIta4CS - t002 - 0
Mon - 8 - rB	ConBenArcLib - t027 - 0	ConBenArcLib - t027 - 0	FilSem - t023 - 0	StoArc - t019 - 0	ChiAppBenCul - t042 - 0	StoArc - t019 - 0
Mon - 9 - rA	FonSocAntCull - t044 - (	Geo2 - t013 - 0	LinTed1 - t000 - 0	MetRicArc - t018 - 0		

Fig. 4: Room Conflict elimination through Layout Designing.

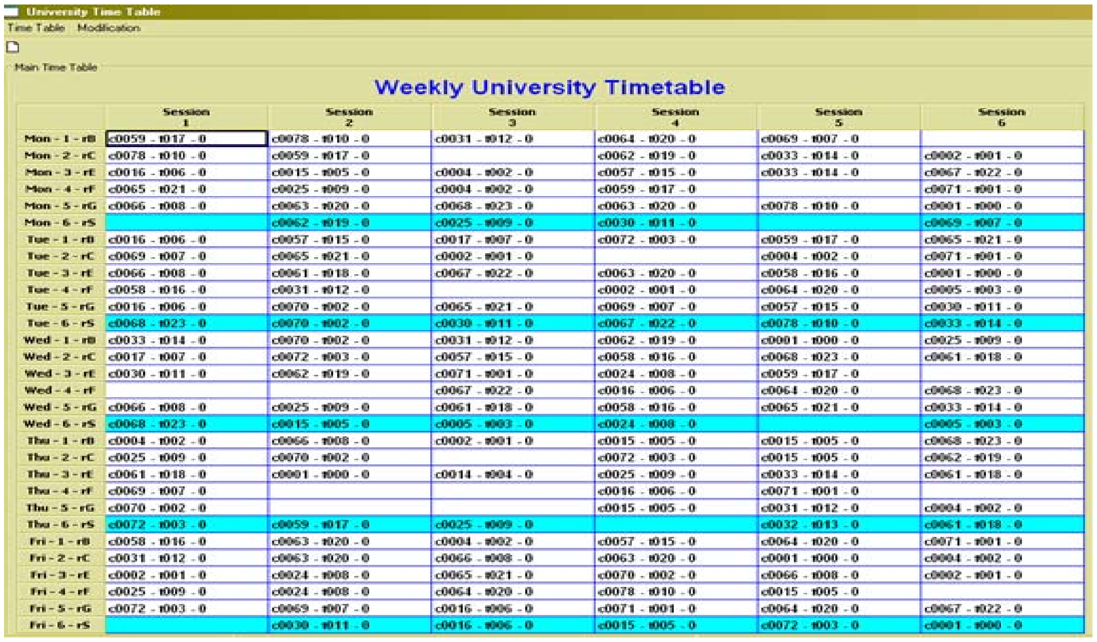
Effectiveness:

The most prominent aspect of layout is its effectiveness and usefulness for evaluation and detection of constraints violation. Violated slots can be traced out with high accuracy, shows in Fig. 5 entire search space (Layout) is functionally divided into subspaces (Days and Sessions) separately. Second important factor is greater capability for frequent data flow. The solving process enormously needs the shuffling and swapping operation over slots. Throughout the computing, event-elements frequently exchange their placements reciprocally, until solution becomes mature enough. The tale ending index/subscript (shown in fig. 5 is representing penalty cost of the slot. Accumulating interrelated information in single unit shapes the layout more applicable, editable and capable for evaluating the data.



	Session 1	Session 2	Session 3	Session 4	Session 5	Session 6
Mon - 1 - rB	c0069 - t007 - 0	c0032 - t013 - 0	c0031 - t012 - 0	c0033 - t014 - 0	c0059 - t017 - 5	c0024 - t008 - 0
Mon - 2 - rC	c0033 - t014 - 0	c0015 - t005 - 0	c0068 - t023 - 0	c0017 - t007 - 0	c0078 - t010 - 0	c0070 - t002 - 0
Mon - 3 - rE	c0065 - t021 - 0	c0063 - t020 - 0	c0064 - t020 - 0	c0001 - t000 - 121	c0058 - t016 - 0	c0062 - t019 - 1
Mon - 4 - rF	c0025 - t009 - 25	c0061 - t018 - 0	c0025 - t009 - 25	c0057 - t015 - 0	c0005 - t003 - 45	c0059 - t017 - 0
Mon - 5 - rG	c0002 - t001 - 55	c0005 - t003 - 55	c0030 - t011 - 0	c0063 - t020 - 5	c0071 - t001 - 0	c0015 - t005 - 45
Mon - 6 - rS	c0016 - t006 - 35	c0069 - t007 - 0	c0072 - t003 - 0	c0072 - t003 - 0	c0016 - t006 - 35	c0069 - t007 - 0

Fig. 5: Constraints Violation.



Weekly University Timetable

	Session 1	Session 2	Session 3	Session 4	Session 5	Session 6
Mon - 1 - rB	c0059 - t017 - 0	c0078 - t010 - 0	c0031 - t012 - 0	c0064 - t020 - 0	c0069 - t007 - 0	
Mon - 2 - rC	c0078 - t010 - 0	c0059 - t017 - 0		c0062 - t019 - 0	c0033 - t014 - 0	c0002 - t001 - 0
Mon - 3 - rE	c0016 - t006 - 0	c0015 - t005 - 0	c0004 - t002 - 0	c0057 - t015 - 0	c0033 - t014 - 0	c0067 - t022 - 0
Mon - 4 - rF	c0065 - t021 - 0	c0025 - t009 - 0	c0004 - t002 - 0	c0059 - t017 - 0		c0071 - t001 - 0
Mon - 5 - rG	c0066 - t008 - 0	c0063 - t020 - 0	c0068 - t023 - 0	c0063 - t020 - 0	c0078 - t010 - 0	c0001 - t000 - 0
Mon - 6 - rS		c0062 - t019 - 0	c0025 - t009 - 0	c0030 - t011 - 0		c0069 - t007 - 0
Tue - 1 - rB	c0016 - t006 - 0	c0057 - t015 - 0	c0017 - t007 - 0	c0072 - t003 - 0	c0059 - t017 - 0	c0065 - t021 - 0
Tue - 2 - rC	c0069 - t007 - 0	c0065 - t021 - 0	c0002 - t001 - 0		c0004 - t002 - 0	c0071 - t001 - 0
Tue - 3 - rE	c0066 - t008 - 0	c0061 - t018 - 0	c0067 - t022 - 0	c0063 - t020 - 0	c0058 - t016 - 0	c0001 - t000 - 0
Tue - 4 - rF	c0058 - t016 - 0	c0031 - t012 - 0		c0002 - t001 - 0	c0064 - t020 - 0	c0005 - t003 - 0
Tue - 5 - rG	c0016 - t006 - 0	c0070 - t002 - 0	c0065 - t021 - 0	c0069 - t007 - 0	c0057 - t015 - 0	c0030 - t011 - 0
Tue - 6 - rS	c0068 - t023 - 0	c0070 - t002 - 0	c0030 - t011 - 0	c0067 - t022 - 0	c0078 - t010 - 0	c0033 - t014 - 0
Wed - 1 - rB	c0033 - t014 - 0	c0070 - t002 - 0	c0031 - t012 - 0	c0062 - t019 - 0	c0001 - t000 - 0	c0025 - t009 - 0
Wed - 2 - rC	c0017 - t007 - 0	c0072 - t003 - 0	c0057 - t015 - 0	c0058 - t016 - 0	c0068 - t023 - 0	c0061 - t018 - 0
Wed - 3 - rE	c0030 - t011 - 0	c0062 - t019 - 0	c0071 - t001 - 0	c0024 - t008 - 0	c0059 - t017 - 0	
Wed - 4 - rF			c0067 - t022 - 0	c0016 - t006 - 0	c0064 - t020 - 0	c0068 - t023 - 0
Wed - 5 - rG	c0066 - t008 - 0	c0025 - t009 - 0	c0061 - t018 - 0	c0058 - t016 - 0	c0065 - t021 - 0	c0033 - t014 - 0
Wed - 6 - rS	c0068 - t023 - 0	c0015 - t005 - 0	c0005 - t003 - 0	c0024 - t008 - 0		c0005 - t003 - 0
Thu - 1 - rB	c0004 - t002 - 0	c0066 - t008 - 0	c0002 - t001 - 0	c0015 - t005 - 0	c0015 - t005 - 0	c0068 - t023 - 0
Thu - 2 - rC	c0025 - t009 - 0	c0070 - t002 - 0		c0072 - t003 - 0	c0015 - t005 - 0	c0062 - t019 - 0
Thu - 3 - rE	c0061 - t018 - 0	c0001 - t000 - 0	c0014 - t004 - 0	c0025 - t009 - 0	c0033 - t014 - 0	c0061 - t018 - 0
Thu - 4 - rF	c0069 - t007 - 0			c0016 - t006 - 0	c0071 - t001 - 0	
Thu - 5 - rG	c0070 - t002 - 0			c0015 - t005 - 0	c0031 - t012 - 0	c0004 - t002 - 0
Thu - 6 - rS	c0072 - t003 - 0	c0059 - t017 - 0	c0025 - t009 - 0		c0032 - t013 - 0	c0061 - t018 - 0
Fri - 1 - rB	c0058 - t016 - 0	c0063 - t020 - 0	c0004 - t002 - 0	c0057 - t015 - 0	c0064 - t020 - 0	c0071 - t001 - 0
Fri - 2 - rC	c0031 - t012 - 0	c0063 - t020 - 0	c0066 - t008 - 0	c0063 - t020 - 0	c0001 - t000 - 0	c0004 - t002 - 0
Fri - 3 - rE	c0002 - t001 - 0	c0024 - t008 - 0	c0065 - t021 - 0	c0070 - t002 - 0	c0066 - t008 - 0	c0002 - t001 - 0
Fri - 4 - rF	c0025 - t009 - 0	c0024 - t008 - 0	c0064 - t020 - 0	c0078 - t010 - 0	c0015 - t005 - 0	
Fri - 5 - rG	c0072 - t003 - 0	c0069 - t007 - 0	c0016 - t006 - 0	c0071 - t001 - 0	c0064 - t020 - 0	c0067 - t022 - 0
Fri - 6 - rS		c0030 - t011 - 0	c0016 - t006 - 0	c0015 - t005 - 0	c0072 - t003 - 0	c0001 - t000 - 0

Fig 6: Complete Layout View.

Readability:

The simple but comprehensive outcome look for end-users is a significant feature of the layout. The rooms, day and session are intersected among event ingredients, view focus can be changed for specific group of students, teacher or venue. Fig. 6 depicts the major features of scheduling layout. A single row illustrates the events of day for single venue. On the other hand column show entire stack of events in time intervals. Entire view seems very precise and readable for end-user.

Conclusions:

Formulating scheduling layout involves years of manmade relevant experience. Intelligently designed layout not only used to accommodating the events but also causes to accelerate the accuracy, enhances the readability and computing capability. The core contributions of this research work are to make diligence of organized event placement, an absolute hard constraints removal, reducing computational time and optimal resources exploitation. The further prospective of research study is the investigation and deployment of various benchmark as well as real-world datasets over the layout.

REFERENCES

Aftab Ahmed, Abdul Wahid Shaikh, Mazhar Ali, Bukhari, A.H.S., 2011. Hyper-GA for Solving Benchmark Scheduling Problems. Australian Journal of Basic and Applied Sciences, 5: 1657-1667.

Aftab Ahmed, Li, Z., 2010a. A Biphasic Approach for University Timetabling Problem. The 2nd International Conference on Computer Engineering and Technology (ICCET 2010), Chengdu, Sichuan, China, pp: 192-197.

Aftab Ahmed, Li, Z., 2010b. Solving Course Timetabling Problem Using Interrelated Approach. 2010 IEEE International Conference on Granular Computing IEEE, San Jose, California, USA, pp: 651-655.

Carter, M.W., D.G. Johnson., 2001. Extended clique initialisation in examination timetabling. Journal of the Operational Research Society, 52: 538-544.

Cooper, T.B., J.H. Kingston, 1996. The complexity of timetable construction problems. In: (Eds), E.K.B.a.P.R. (Ed.), Practice and Theory of Automated Timetabling. Springer-Verlag, pp: 283-295.

Burke, E.K., J.P. Newall, R.F. Weare, 1998. Initialisation strategies and diversity in evolutionary timetabling. Evolutionary Computation , 6: 81-103.

Goltz, H.J., D. Matzke, 1999. Combined interactive and automatic timetabling. Practical Application of Constraint Technology and Logic Programming, London, pp: 529-535.

Carter, M.W., G. Laporte, J.W. Chinneck., 1994. A general examination scheduling system. Interfaces 24: 109-120.

Carter, M.W., G. Laporte, 1996. Recent developments in practical examination timetabling. In: E.K. Burke, Ross, P. (Eds.), The Practice and Theory of Automated Timetabling I: Selected Papers from 1st International Conference on the Practice and Theory of Automated Timetabling (PATAT I). Springer-Verlag, pp: 3-21.

Maciej Norberciak, 2006. Universal Method for Timetable Construction based on Evolutionary Approach. Proceedings of world academy of science, engineering and technology.