Solving University Scheduling Problem Using Hybrid Approach

AFTAB AHMED SHAIKH*, ABDUL HUSSAIN SHAH BUKHARI**, AND ZUBAIR AHMED MEMON***

RECEIVED ON 21.05.2010 ACCEPTED ON 03.01.2011

ABSTRACT

In universities scheduling curriculum activity is an essential job. Primarily, scheduling is a distribution of limited resources under interrelated constraints. The set of hard constraints demand the highest priority and should not to be violated at any cost, while the maximum soft constraints satisfaction mounts the quality scale of solution. In this research paper, a novel bisected approach is introduced that is comprisesd of GA (Genetic Algorithm) as well as Backtracking Recursive Search. The employed technique deals with both hard and soft constraints successively. The first phase decisively is focused over elimination of all the hard constraints bounded violations and eventually produces partial solution for subsequent step. The second phase is supposed to draw the best possible solution on the search space. Promising results are obtained by implementation on the real dataset. The key points of the research approach are to get assurance of hard constraints removal from the dataset and minimizing computational time for GA by initializing pre-processed set of chromosomes.

Key Words: **Course Timetabling Problem, GA, Backtracking Recursive Search.**

1. INTRODUCTION

The importance of curriculum scheduling for academic calendar in universities cannot be underrated. The curricula timetabling is important in two ways, firstly to initiate the academic term and secondly to stabilize the curriculum activities in rest of the semester. University scheduling is categorized into two distinguished formats, course and examination timetabling. Both have variation in constraints and different operational timescale; However, many of mutual features are available as well. Probably in all universities, beginning of semester and its ending depends on these scheduling. As appearance timetable is legible matrix of academic activities usually intersected by available resources. Each working day of weekly timetable includes N number of sessions or periods pointed by fixed length of time. Time periods serve well-organized stacks of events whilst event is set of interrelated information about subject, group of students and teacher, assigned specifically over timeslot and venue.

University scheduling is a well known NP hard problem; its complexity exponentially increases with respect to size of problem instance. Conventional computational methods

* Ph.D. Scholar, School of Computer Science & Engineering, Beihang University Beijing, China.
** Dean, Faculty of Information & Communication Technology, Blochistan University of Information Technloogy, Engineering & Management Sciences, Quetta.
*** Assistant Professor, Department of Electrical Engineering, Mehran University of Engineering and Technology, Jamshoro, Pakistan.

or manmade solutions require plenty of efforts and time-span due to large number of diverse parameters which are supposed to satisfy. Almost in all the real world cases it is impossible to obtain the perfect timetabling solution because once satisfied constraint might become the cause of violation for another constraint. However, the methodology has to compromise over small number of violations. Solving timetabling problem therefore requires an extensive brainstorming, skills and working experience of the related field. This area requires further investigation and identifying the accurate methods to design more effective and efficient automated timetabling.

The work described in this research paper is partial implementation of the large scaled ongoing project. The core idea is to wipe out all the hard constraints by applying backtracking recursive algorithm. The first step does not touch any of the soft constraints and focuses to eliminate all the hard constraint violation swinging the penalized slots back and forth. The succeeding step consists of GA applied over semi-matured set of solutions produced by previous phase. GA is supposed to obtain whatever the best possible solution is available on the search space. The subsequent step eventually brought up with optimal timetabling solution. Dividing the task between two effective methods save reasonable amount of computing time and provids robust and promising results.

The research paper is organized as follows: The Section 2 looks at the few prominent research methods employed successfully. The Section 3 illustrates the problem specification by various constraints and variables. Section 4 is consisted of two subsections. First one depicts solving the hard constraints by Backtracking Recursive Search. Second subsection explains the implementation of GA with respect to required parameters and operators for maximizing the soft constraints satisfaction. The results are discussed in Section 5, and lastly, conclusion and future work presented in Section 6.

2. PREVIOUS WORK

Timetabling problem has gained attention by the scientific community for many decades. The solution came out from diverse disciplines such as Operation Research; Computational Intelligence etc. However, producing high quality effective solution requires extensive working experience and involvement in real university timetabling system.

In Abdullah, et. al. [1] an integrated approach is developed and convergence is made faster by incorporating local search with GA. A repair function has also been applied for checking hard constraints violations in new offspring. The method is applied over diverse datasets. [2] Testifies GA and explores a range of crossover parameters for producing high quality outcome. [3] Introduces slightly different approach of using genetic programming for the evolution of hyper-heuristics for the uncapacitated examination timetabling problem. The system has been tested on large number of benchmark examination timetabling problems. Kanoh, et. al. [4] elaborate their knowledge based GA. Solution methodology is based on a GA which uses an installed knowledge base and an infection operator. The knowledge base was set of candidate solutions assembled from timetables used in past years and based on some queries response asked for quality assurance. Fen, et. al. [5] investigated the constraint-based reasoning approaches tested over two different datasets of real world. Selecting the resources on their suitable priority value and backtracking method are the main elements of the approach. Landa-Silva, et. al. [6] proposes a modified version of the great deluge algorithm in which the decay rate of the water level is non-linear. The proposed method produces new best results in 4 of the 11 course timetabling problem instances used in experiments.

Some other recent and prominent approaches for solving timetabling have been explored including fuzzy logic reasoning [7]. Hybrid approaches have produced good

quality results [8]. Various types of Neighborhood search algorithms have achieved certain degree of success as well [9-12]. Other techniques include Case-Based Reasoning [13-14] and Ant Algorithms [15].

The approach discussed in this research paper is slightly distinguished because the blend of algorithms and operators are very effective and efficient for solving scheduling problems. The computational stuff is result of several experiments and brainstorming sessions.

3. PROBLEM FORMULATION

The Timetabling is Combinatorial Optimization problem necessiting proper allocation as well as deployment of inadequate resources under predefined global as well specific conditions known as constraints. Typical Timetabling consists of Teacher, Room, Subject, Time period, Student Group and Constraints. The research approach has been successfully experimented over the extremely complex dataset of Department of Computer Science, Balochistan University of Information Technology, Engineering and Management Sciences, Quetta, Pakistan [16]. The department requires regulating the schedule for spring and fall semesters respectively. Dataset comprises of 8x2 sections for each department. Moreover, remedial sections sometimes can be offered depending upon the number of enrolments. Approximately, five courses are offered to each group considering the number of credit hours. Most of the lectures are scheduled twice a week. Table 1 illustrates the available resources and requirements. In the dataset, there are more or less five fully dedicated rooms plus two rooms for partial usage and ten teachers. A working week is consisted of six continuous days ended by one holiday. While a working day includes four sessions of 90 minutes each followed by 15 minutes recess.

TABLE 1. DATASET SPECIFICATION

No.	Depiction	Quantity
01	Class Rooms	5+2
02	Courses	50
03	Sessions per Week	120
04	Working Days	06
05	Periods per Day	04
06	Teaching Faculty	10
07	Courses per Teacher	>3

Therefore, overall 4x6=24 periods have to be managed for 50x2+20=120 sessions in a week. Inadequate resources and high complexity of constraints make scheduling problem a challenging to be solved. Timetabling constraints that have to be satisfied for solution are typically divided into two distinguished groups known as hard and soft constraints. Hard constraints are extremely significant and supposed to be fulfilled strictly. A single hard constraint violation makes timetabling impracticable. At the same time soft constraints satisfaction is crucial to the reliability and optimality of solution. By and large, many of the constraints are commonly immersed. Nevertheless, bespoke version of constraints is also available in all universities associated to their academic priorities and convenience.

3.1.1 Constraints

The constraints are usually categorized into hard constraints and soft constraints. Set of hard constraints demands the highest priority to be processed while maximum soft constraints satisfaction shoots up the overall quality of solution. Hard and Soft constraints are described in Tables 2-3.

Variables and their finite domains used in constraints formulation are defined as:

E	Total number of events in timetable
G	Denotes all sets of students
T	Overall events in a specific session
C	Required capacity of the room
D_i	Showing the working week $i \in \{1, \cdots, 6\}$
K	Represents the sessions of fix length time
t	Smallest unit of placement for event
Z	Represents all the resource persons
R	Available rooms for scheduling
S_{ij}	All day events for a group where $i \in \{1, \cdots, G\}$ and $j \in \{1, \cdots, D\}$
L_{ij}	Teaching load per day where $i \in \{1, \cdots, N\}$ and $j \in \{1, \cdots, D\}$

4. GOAL FORMULATION

The first phase deals with the vanishing of all the hard constraints from search space and sets up various instances of partial solutions for the second phase. Moreover in order to pack the events together the algorithm keenly converges the dataset to focal point.

4.1 Layout Designing

A well designed timetabling layout not only makes data readable to a certain extent but also supports to put up events in order. Each data chamber is intersection point between session time and class room thus hard constraints belonging to repeated allocation become extinct by default. Fig. 1 shows the layout example; it can be noticed, for single entry of event that is only one available placement. A multi-dimensional array is used as layout, consisting information [Day, Session, Room, Penalty Cost]. Another array supports event stuff Event List = [Class, Course, Teacher]. One by one value from event array immerses into layout in case of finding valid placement. During the process, events repeatedly swap with each other until solution gets ripened. The ending subscript number represents penalty cost of that slot. Accumulating interrelated information in single unit shapes the layout more applicable, editable and capable for evaluating the data.

TABLE 2. HARD CONSTRAINTS DESCRIPTION

Variable	Hard Constraints	Penalty
HC1	Each group must not be scheduled for two or more events in same period $\sum_{i=1}^{k\times D}\sum_{j=1}^{T} gijk \le 1 \text{ for } \forall k \in \{1\ldots G\}$	1000
HC2	No more than one lecture can be assigned to each teacher $\sum_{i=1}^{k\times D}\sum_{j=1}^{L} Iijk \le 1 \text{ where } \forall j \in \{1\ldots L\}$	1000
HC3	Only one event can be allotted in class room $\sum_{i=1}^{R} eij \le 1 \text{ for } \forall j \in \{1\ldots E\}$	1000
HC4	Capacity of class room should be enough to accommodate the group of students $\sum_{j=1}^{E} riek \le C \text{ where } \forall j \in \{1\ldots R\}$	1000

4.2 Eliminating the Hard Constraints

The first computing segment removes all the hard constraints without consideration of any soft constraint violation. This phase also scatters the entire dataset over predesigned layout. List of events is an array of strings but backtracking method traverses as tree in recursive order along with pruning test. It starts from the root and branches downward. Here, root is empty first slot and

TABLE 3. HARD CONSTRAINTS DESCRIPTION

Variable	Soft Constraints	Penalty
SC1	Succeeding event(s) of each group should be assigned on same venue $\sum_{i=1}^{S} \alpha i \le \text{r where } r \in R$	05
SC2	Day events for each students group should be fixed in succession. $\sum_{i=1}^{S} \lvert ti - ti+1 \rvert = 1$	06
SC3	Scheduling three plus unremitting events are supposed to be avoided $\sum_{i=1}^{G} Si > 1$	07
SC4	At least two to three events are recommended to be fixed for all groups $\sum_{i=1}^{G} Si \le 3$	08
SC5	Number of lecturers per teacher should not go over two per day $\sum_{i=1}^{Z} Li \le 2$	09

child nodes are neighboring sessions of day. The algorithm checks whether the event can be placed in the session by verifying its status with existing events with respect to group of students and teacher. If it conflicts, the branch or session becomes pruned and programming control enumerates another sibling session. The algorithm navigates nodes down in depth-first manner. Each failure in placement of event extends the sub-tree one step further until leaves of the search tree come in reach. Likewise, every event from the list is supposed to move back and forth in search tree and avail the validate placement in sessions. The Backtracking search method is very accurate and fast for most of the datasets. Fig. 2 depicts the overall mechanism.

Day	Room No.	09:00-10:30	10:45-12:00	12:30-14:00	15:00-16:30
Monday	1	Clas-Sub-Tech	Clas-Sub-Tech	Clas-Sub-Tech	Clas-Sub-Tech
	2				
	3				
	4				
	5				

FIG. 1. TIMETABLING LAYOUT

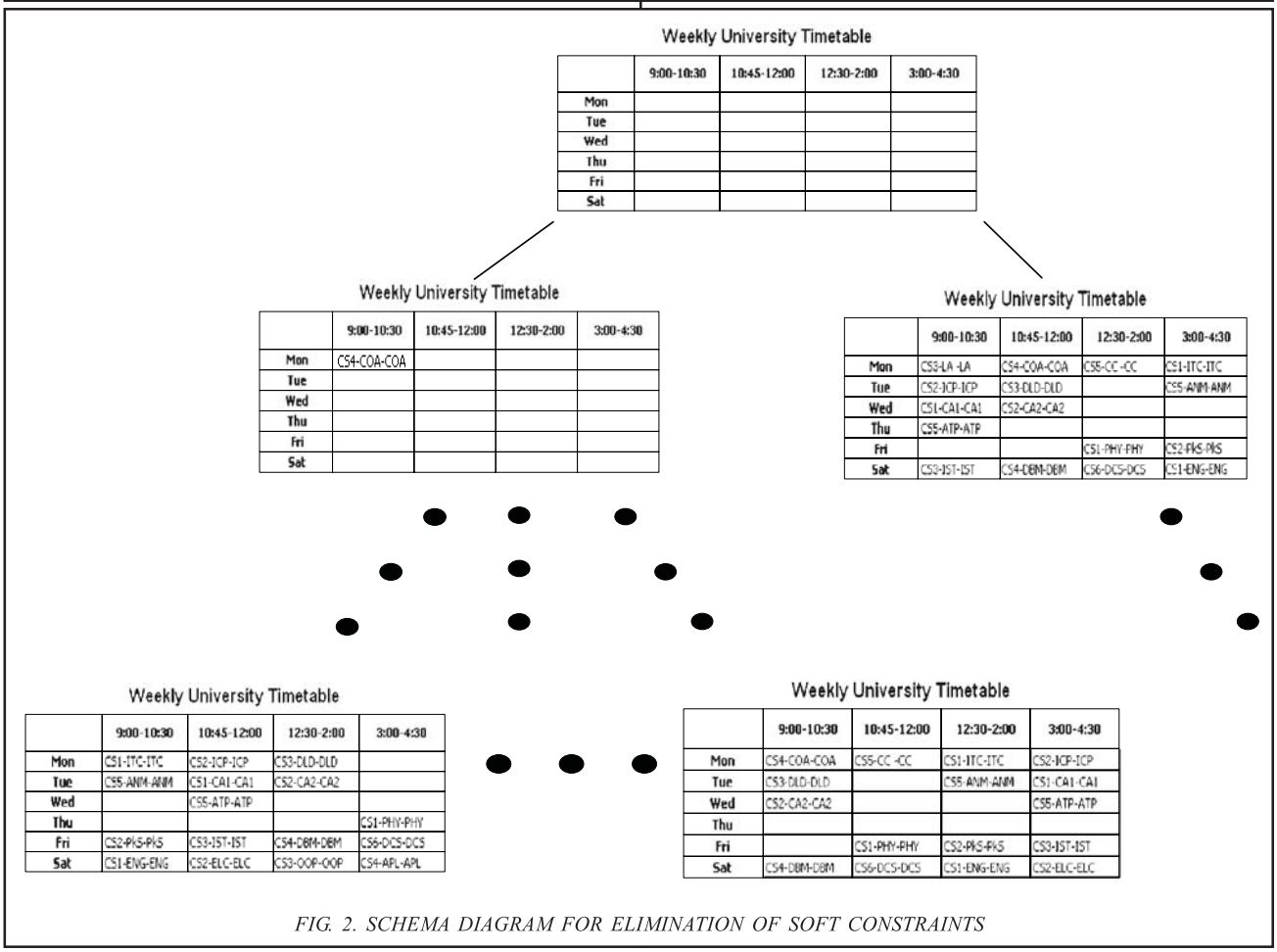

FIG. 2. SCHEMA DIAGRAM FOR ELIMINATION OF SOFT CONSTRAINTS

In Algorithm-1, the sub-procedure VERIFY performs the prune test and perceives the hard constraint clashes for duplication of students group or teacher on the sub tree. The process repeats before each entry ensures the validity. The mechanism proceeds in depth first order. In case of any violation backtracking mechanism rollbacks from pruned session and gets way to neighboring session. Moreover, VERIFY procedure also leave some slides intentionally vacant at rear end of each day so that events can be exchanged conveniently. As a result, it preemptively reduces the number of occurrences of soft constraints.

Algorithm-1 Solving Hard Constraints

```
1.    Procedure BTS(ListofEvents):
2.    While (True):
3.    IF All Events are Assigned Than Exit
4.    For All the Periods
5.    IF Verify(New Event, Period ) == TRUE
6.    THAN
7.     Assign ListofEvents[Counter] to Empty Slot
8.    Call BTS(ListofEvents[Counter + 1])
9.    ELSE
10.   select Succeeding Period
11.   End For
12.   End While
13.   End Procedure
14.   Sub-Procedure Verify(Event, Period ):
15.   For All Slots in Period
16.   IF Event [Class] OR Event [Teacher] Exists
17.   THAN return FALSE
18.   End For
19.   Return TRUE
20.   End Sub-Procedure
```

4.3 Phase II: Dealing with soft Constraints

This phase is decisively focused to deal with soft constraints in order to achieve high quality operational solution. The phase is needed for not to violate any hard constraint settled from pervious phase, eventually it decreases the number of options for shuffling events interpedently.

(i) *Genetic Algorithm:* The GAs are computational inspiration of Darwin's evolutionary theory. GAs evolve the set of partial solutions (chromosomes) called genome or precisely population. The newly generated population is supposed to be converged towards target. Offspring for each new generation are selected by some decisive fitness criteria. The second phase investigates approach partially based on GA for university scheduling problems. The reason for using Genetic Algorithm approach is its maturity and promising results. Fig. 3 portrays all about GA based timetabling.

(ii) *Representation:* Generation is a combination of chromosomes where each chromosome is partial timetabling solution. Each chromosome further divides into genes; the detail about event consists of [Class, Subject, Teacher]. Fig. 4 illustrates the construction of chromosomes and set of genes.

(iii) *Initialization:* Initialization is proposed to inject input from first phase. Each chromosome is different from each other due to diverse placement of genes as result proved a robust and converging initial population.

(iv) *Crossover:* Crossover selects genes from parent chromosomes and creates a new offspring. For example to choose randomly some crossover points and copy slice of data of both sides than swapping with each other.

(v) *Mutation:* After a crossover is performed, mutation takes place. This is to prevent population falling into a local optimum of solved problem. Mutation alters offspring on random or predefined point.

(vi) *Fitness Function:* The fitness function evaluates the quality of chromosomes in each generated population. Expectedly, hard constraints summation will return zero while minimum total of soft constraints violation is appreciated and believed right conversion for next generation. Typically fitness function can return value between 0 and 1 due to formulation.

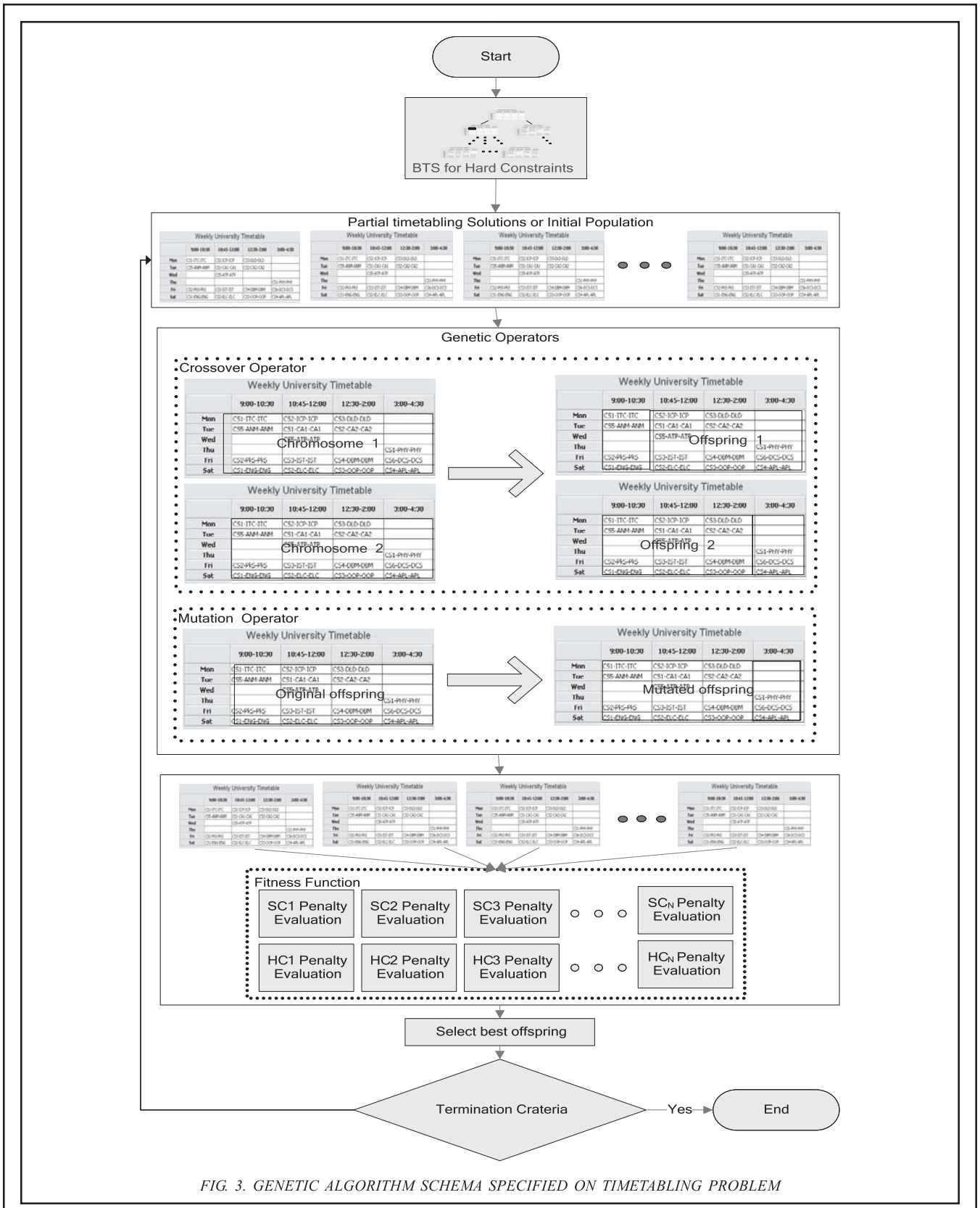

FIG. 3. GENETIC ALGORITHM SCHEMA SPECIFIED ON TIMETABLING PROBLEM

$$Maximizef(x) = \frac{1}{1 + \sum_{i}^{n} \alpha_i h_i + \sum_{i}^{n} \beta_j s_j}$$

Where α and β represent penalty cost while s_i, and h_j denote the soft and hard constraints respectively.

Algorithm-2 Solving Soft Constraints using GA

1. [Start] Input the population of n Timetabling chromosomes From BTS Algorithm
2. Repeat while (Convergence Criteria OR N populations)
   - (a) [Fitness] Examine the fitness f(x) of all chromosome x in the population
   - (b) [Following operators will converge the population to desirable results]
   - (c) [Selection] Rank-based Selector choose parent timetabling chromosomes from a population according to their fitness
   - (d) [Crossover] Order Crossover (OX) operator crosses over the parents to form a new offspring.
   - (e) [Mutation] Mutation operator mutates new offspring with objective inclination towards better solution.
   - (f) [Acceptance] repair and put new offspring in a next generation
   - (g) [Regenerate] replace new generated population with old one
3. [End While]
4. Exit

Fig. 5 illustrates the penalized slots for a day. Penalties 5 and 6 can only be eliminated by swinging slots back and forth on same day. On the other hand slots having penalties 7-9 definitely have to shift over another day. In addition, these slots need to scan entire domain looking for unoccupied position. The search is associated with availability of similar event pair along-with avoiding of preemptive violations. However, such shuffling may reason for collateral damage of low level violations on either day side. Slot marked due to HSC5 and HSC6 violations actually have a small number of suitable placement options since these constraints require adjacency of relevant slots.

5. RESULTS

The experimental results demonstrated here are partial implementations of ongoing research project. Initially the proposed methodology is testified over single department dataset and prominent results validate the proper direction and potential of adopted research mechanism. Table 4 represents the adopted parameters for GA. Genome is collection of partially solved timetabling instances.

TABLE 4. SELECTED PARAMETERS FOR GENETIC ALGORITHM

Parameter	Value
Genome Size	100
Total Generations	500
Crossover Rate	0.06
Mutation Rate	0.01
Selection Method	Rank Selector
Crossover Method	Order Crossover

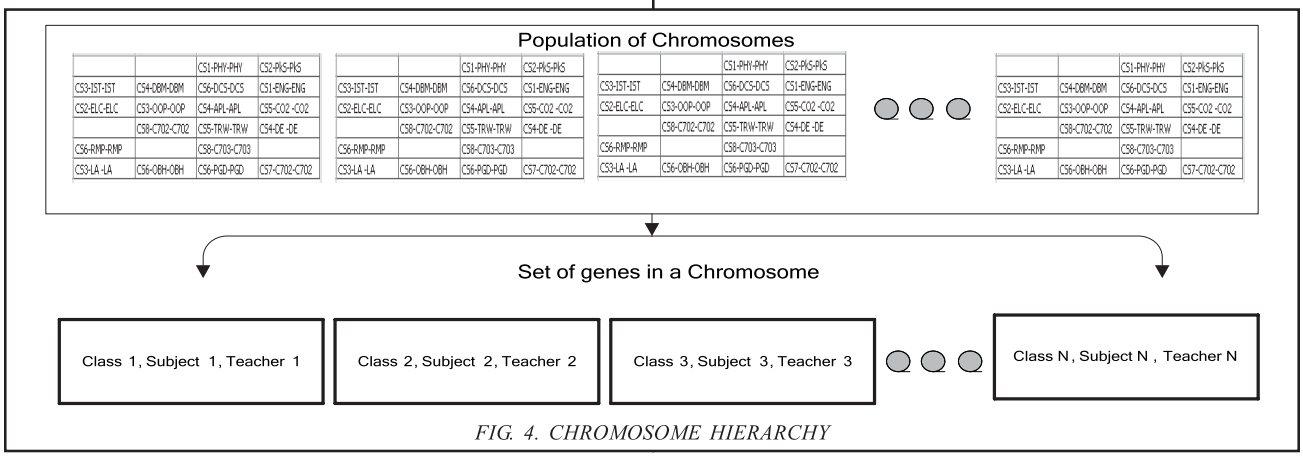

FIG. 4. CHROMOSOME HIERARCHY

Table 5 illustrates the set of partial solutions come out from first phase. It can be observed that all the solution instances are free of hard constraint violations. Fig. 6 Illustrates soft constraints and their steady descending penalty cost individually. For SC1 and SC2 constraints steep decreases followed by occasional rises. As various constraints handlers shuffle the penalized slots to other sessions, the activity causes increasing of SC1 and SC2 penalties sometimes. Escalating saturation might be cause of low rate of improvements with ending generations. Very small number of penalties remains unsolved at the bottom due to NP-hard nature of problem. Fig. 7 depicts the overall progress specifically. A stable decrease of total penalty verifies the promising success of solving technique. After starting with sharp drop off for few steady steps it is noticeable that progress becomes smaller due to increasing complexity. Valid placements eventually produce the saturation for rest of events. At one stage results show no more improvements or conversion that leads to terminate the program execution.

5. CONCLUSIONS

Scheduling the events is an extremely crucial job for regulating the academic cycle for every university. Timetabling belongs to combinatorial optimization, based on resources allocation. The laborious and tedious problem provides wide range of algorithmic applicability. A clearly formulated scheduling can be much supportive to run an academic session in a well organized way.

The work introduced in this research paper is an adoption of novel approach consisting two distinguished phases. First one is totally concentrated to resolve all the hard

Day	Room No.	09:00-10:30		10:45-12:15		12:30-14:00		15:00-16:30	
Monday	1	CS1, S13, T02	6	CS2, S25, T11	5	CS1, S16, T14	9		
	2	CS2, S23, T04				CS3, S36, T13		CS3, S31, T15	
	3	CS4, S42, T07	6	CS9, S94, T12	7	CS4, S44, T11		CS4, S41, T06	
	4	CS5, S53, T05		CS5, S55, T12		CS5, S51, T05			
	5	CS6, S65, T13		CS8, S83, T02	5	CS6, S61, T115			
	6	CS7, S74, T08		CS7, S73, T10		CS7, S71, T12			
	7	CS8, S81, T14		CS6, S62, T14	5	CS8, S84, T08		CS8, S85, T03	8

FIG. 5. DIAGRAM IS EXEMPLIFIED BY SOFT CONSTRAINTS VIOLATIONS

TABLE 5. HARD CONSTRAINTS FREE INITIAL POPULATION FOR GA

Seed	SC1	SC2	SC3	SC4	SC5	HC	Total Penalty
01	30	17	18	19	16	0	669
02	25	19	17	18	10	0	592
03	29	22	13	16	09	0	584
04	31	28	16	14	11	0	646
05	22	21	15	13	08	0	517
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
80	28	23	17	12	13	0	602

constraints form search space and second one minimizes the soft constraints related violations and produces highly quality solution. This approach has drawn much better and accurate result in less computational time than conventional GA approaches. The objectives are expanded to maximize utilization and efficient deployment of the resources that satisfy a lot of requirement measures.

The next pace of research tends to sustain the optimality of solution for interactive post-processed user modifications in solution. An additional computational step using Min-Conflict Algorithm is proposed as repairing mechanism. The implementation circle is supposed to be expanded up to faculty level dataset.

ACKNOWLEDGEMENT

This work is supported by the Balochistan University of Information Technology, Engineering & Management Sciences, Quetta, Pakistan.

REFERENCES

[1] Abdullah, S., and Turabieh, H., "Generating University Course Timetable Using Genetic Algorithms and Local Search", Third International Conference on Convergence and Hybrid Information Technology, Volume 1, pp. 254-260, 2008.

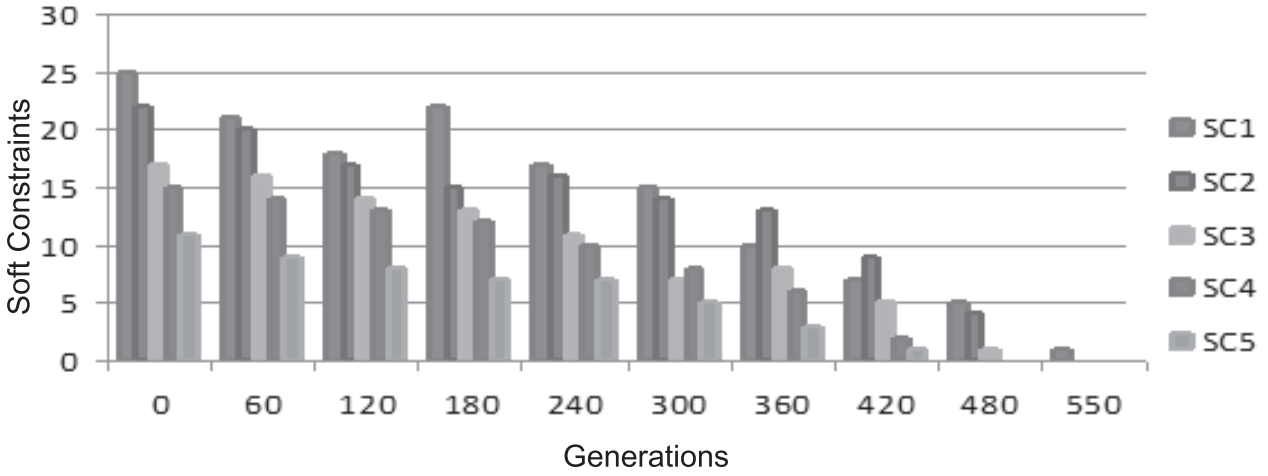

FIG. 6. AVERAGE PENALTY COST OF EACH SOFT CONSTRAINT

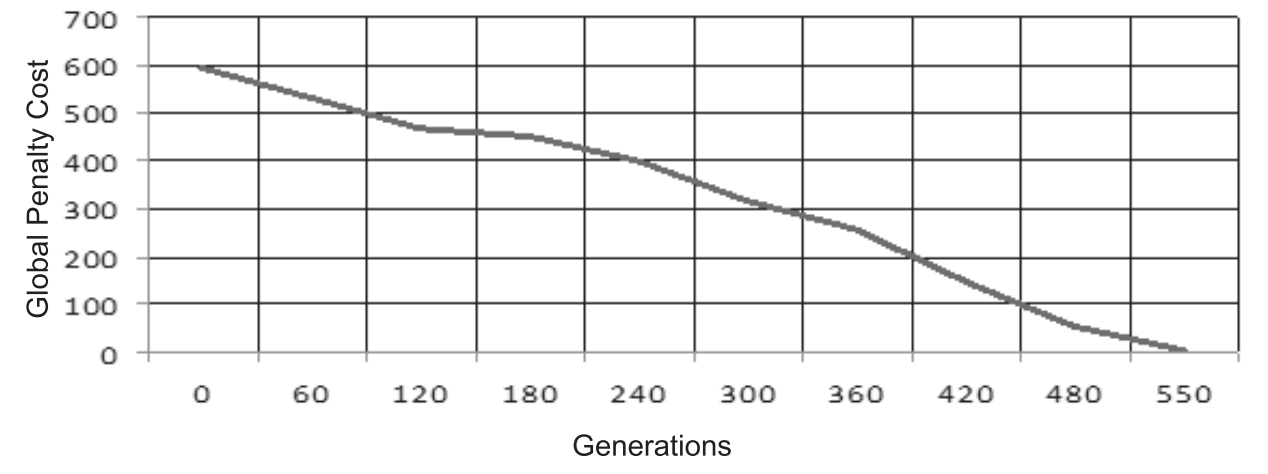

FIG. 7. TOTAL PENALTY COST THROUGHOUT PROGRAM EXECUTION

[2] Khonggamnerd, P., and Innet, S., "On Improvement of Effectiveness in Automatic University Timetabling Arrangement with Applied Genetic Algorithm", Fourth International Conference on Computer Sciences and Convergence Information Technology, pp. 1266-1270, 2009.

[3] Pillay, N., "An Analysis of Representations for Hyper-Heuristics for the Uncapacitated Examination Timetabling Problem in a Genetic Programming System", Annual Conference of South African Institute of Computer Scientists and Information Technologists on IT Research in Developing Countries., Volume 338, pp. 188-192, Wilderness, South Africa, 2008.

[4] Kanoh, H., and Sakamoto, Y., "Knowledge-Based Genetic Algorithm for University Course Timetabling Problems", International Journal of Knowledge-based and Intelligent Engineering Systems, Volume 12, No. 4, pp. 283-294, 2008.

[5] Fen, H.S., Deris, I.S., and Hashim, S.Z., "Investigating Constraint Based Reasoning for University Timetabling Problems", International Multi Conference of Engineers and Computer Scientists, Volume 1, 2009.

[6] Landa-Silva, D., and Obit, J.H., "Great Deluge with Nonlinear Decay Rate for Solving Course Timetabling Problems", Proceedings of EEE Conference on Intelligent Systems, IEEE Press, 8.11-8.18, 2008.

[7] Asmuni, H., Burke, E.K., and Garibaldi, J., "Fuzzy Multiple Ordering Criteria for Examination Timetabling", Burke, E.K., and Trick, M., (Editors), Selected Papers from the 5th International Conference on the Practice and Theory of Automated Timetabling, Lecture Notes in Computer Science 3616, pp. 334-353, 2005.

[8] Caramia, M., DellOlmo, P., and Italiano, G., "New Algorithms for Examination Timetabling", Naher, S., Wagner, D., (Editors), Algorithm Engineering, Lecture Notes in Computer Science 1982, pp. 230-241, 2001.

[9] Abdullah, S., and Ahmadi, S., Burke, E.K., and Dror, M., "Investigating Ahuja-Orlin's Large Neighbourhood Search for Examination Timetabling", Technical Report NOTTCS-TR-2004-8, School of CSiT, University of Nottingham, UK, 2004.

[10] Meyers, C., and Orlin, J.B., "Very Large-Scale Neighborhood Search Techniques", Burke, E.K., and Rudova, H., (Editors), Selected Papers from the 6th International Conference on the Practice and Theory of Automated Timetabling, Lecture Notes in Computer Science 3867, pp. 24-39, 2007.

[11] Abdullah, S., Burke, E.K., and McCollum, B., "Using a Randomised Iterative Improvement Algorithm with Composite Neighborhood Structures for University Course Timetabling", Doerner, K.F., Gendreau, M., Greistorfer, P., Gutjahr, W.J., Hartl, R.F., and Reimann, M., (Editors), Computer Science Interfaces Book Series, Springer Operations Research, 2006.

[12] Abdullah, S., Burke, E.K., and McCollum, B., "An Investigation of a Variable Neighborhood Search Approach for Course Timetabling", Proceedings of the 2nd Multidisciplinary Conference on Scheduling: Theory and Applications, pp. 413-427, NY, USA, 18-21 July, 2005.

[13] Burke, E.K., Dror, M., Petrovic, S., and Qu, R., "Hybrid Graph Heuristics within a Hyper-heuristic Approach to Exam Timetabling Problems", Golden, B.L., Raghavan S., and Wasil, E.A., (Editors), The Next Wave in Computing, Optimization and Decision Technologies, pp. 79-91. Springer, 2005.

[14] Burke, E.K., Petrovic, S., and Qu, R., "Case Based Heuristic Selection for Examination Timetabling", Journal of Scheduling, Volume 9, No. 2, pp. 99-113, 2006.

[15] Eley, M., "Ant Algorithms for the Exam Timetabling Problem", Burke, E.K., and Rudova, H., (Editors), Selected Papers from the 6th International Conference on the Practice and Theory of Automated Timetabling, Lecture Notes in Computer Science 3867, pp. 364-382, 2007.

[16] Ahmed, A., and Li, Z., "A Biphasic Approach for University Timetabling Problem", 2nd International Conference on Computer Engineering and Technology, Volume 1, pp. 192-197, Chengdu, China, 2010.