Australian Journal of Basic and Applied Sciences, 5(9): 190-199, 2011
ISSN 1991-8178

Hyper-Heuristic Approach For Solving Scheduling Problem: A Case Study

[1]Aftab Ahmed, [2]Abdul Wahid Shaikh, [3]Mazhar Ali and [4]Abdul Hussain Shah Bukhari

[1]Department of Computer Science, BUIETMS, Quetta
[2]Department of Computer Science, SALU, Kharpur
[3]Department of Information Technology, BUIETMS, Quetta
[4]Faculty Information and Communication Technology, BUIETMS, Quetta

Abstract: Academic scheduling is tremendously significant and demanding problem which is largely exercised in research community. Contemporary research approaches are focused to enhance geniality so divergent problem instances can be tackled using minimum efforts. In this research work, a three layer hyper-heuristic approach is illustrated, on top most level Genetic Algorithm is used to manage middle layer of constraints solving operators, on the other hand bottom level is occupied by problem domain. The technique intends a performance based selection of low level heuristics by GA. The proposed framework is implemented over the real world dataset to validate the research direction.

Key words: Hyper-Heuristic, Course Timetabling Problem, Genetic algorithm.

INTRODUCTION

The timetable is an integral part of the academic calendar in order to commence the curriculum activities. The timetabling belongs to the group of NP-hard problems. Therefore, conventional problem solving techniques might not perform well. The set of inversely proportional constraints (one constraint does not allow another to satisfy and vice versa) makes the timetabling a vicious circle. Accordingly, the manual practices require several hours of workor even days and extensive brainstorming. Sometimes the problem becomes further complicated specifically when scheduling is required in hierarchical environment, i.e., a faculty has several departments and each of them shares instructors, rooms and other resources. Thus, it is a vital and compelling research area to acquire further investigation for designing efficient automated schedules.

The curricula timetabling is noteworthy in many ways, primarily to commence the educational term and secondary to streamline the curriculum events in the rest of the semester. The university timetabling is classified into two prominent groups,i.e., course and examination scheduling. Both have deviations in group of constraints as well difference in operational continuance. Nevertheless, many of mutual features are also available. In most of the universities the beginning and ending of the semester depends on the course scheduling. The timetable is legible matrix where academic activities usually are intersected by resources and time periods. Each working day comprises *N* number of periods pointed by fixed timescale.The timeperiods serve well-organized stack of events, where an event is a set of unified information of courses, students and teachers, assigned specifically over timeslot and location. Figure-1 depicts the conventional timetabling characteristics.

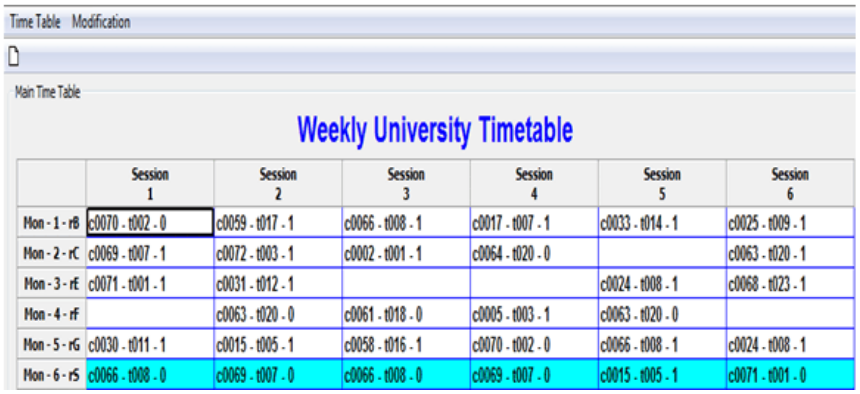
The hard constraints are supposed not to be violated under all situations, and it is compulsory for each feasible solution to qualify this basic criterion. The soft constraints are highly desired to be solved but not necessarily indeed. Most of the time, it is impossible to wipe out all soft constraints.In order to formulate the problem, dynamic penalty cost is used to assign constraint violations so that quality can be measured. The solution may be identified as nearest to optimal in case of all hard constraints removal and presence of none or minimum soft constraint violations as much as possible.

II. Related Work:

Lot of research approaches have been emerged for investigating the automated course/exam timetabling since last two decades. A comprehensive survey of the initial approaches is held by Burke *et al*. (2004). A noticeable number of researchers solved the problem using Genetic algorithm (GA) (Mahdi, 2003; Burke, 1994). S. Abdullah (2008) uses hybrid approach using GA with sequential local search to solve course

Corresponding Author: Aftab Ahmed, Department of Computer Science, BUIETMS, Quetta.
E-mail: aftabshaikhs@hotmail.com

timetabling problem. Tabu search (Burke, 2003) is another prominent method of the area due to its accuracy and efficiency.


Time Table Modification

Main Time Table

Weekly University Timetable

	Session 1	Session 2	Session 3	Session 4	Session 5	Session 6
Mon - 1 - rB	c0070 - t002 - 0	c0059 - t017 - 1	c0066 - t008 - 1	c0017 - t007 - 1	c0033 - t014 - 1	c0025 - t009 - 1
Mon - 2 - rC	c0069 - t007 - 1	c0072 - t003 - 1	c0002 - t001 - 1	c0064 - t020 - 0		c0063 - t020 - 1
Mon - 3 - rE	c0071 - t001 - 1	c0031 - t012 - 1			c0024 - t008 - 1	c0068 - t023 - 1
Mon - 4 - rF		c0063 - t020 - 0	c0061 - t018 - 0	c0005 - t003 - 1	c0063 - t020 - 0	
Mon - 5 - rG	c0030 - t011 - 1	c0015 - t005 - 1	c0058 - t016 - 1	c0070 - t002 - 0	c0066 - t008 - 1	c0024 - t008 - 1
Mon - 6 - rS	c0066 - t008 - 0	c0069 - t007 - 0	c0066 - t008 - 0	c0069 - t007 - 0	c0015 - t005 - 1	c0071 - t001 - 0

Fig. 1: Example of University Scheduling.

The hyper-heuristic techniques have been discussed vaguely in the research literature since the early 1960s (Fisher, 1961) but paradigm has been shifted since past decade. At the first, hyper-heuristic term was coined in 2000 by Cowling *et al.* (2000) where a heuristic is chosen from a set of heuristic by predefined criteria. In Fang *et al*. (1993; 1994) initiated the term `evolving heuristic choice' to improve the performance quality of GA. The approach has been applied on benchmark job-shop scheduling problems and outperformed on several instances. Cowling *et al*. (2002) put forward a GA based hyper-heuristic which they called `HyperGA'. In their work chromosome is represented as sequences of localsearch heuristics for geographically distributed training staff and course scheduling. The set of heuristics or chromosomes were applied to search space and the better solutions became the input for the next generation. The entire generation evaluated according to the overall improvement obtained. The parameters were depended upon variance of CPU time. Overall success of the generation was used to tune the probability of mutation. The approach was implemented by four versions of the hyper-GA along with a number of simple heuristics. The method testified on five test data. E. Burke *et al*. (2005) also reported efficiency of using graph based hyper heuristics for solving exam timetabling problem.

III. Problem Formulation:

A. Dataset Specification:

The research has been experimentally verified over comprehensive real-world timetabledataset ofComputer ScienceDepartment, Balochistan University of Information Technology, Engineering and Management Sciences, Quetta. Particularly, the dataset BUIETMS-1 (Aftab Ahmed, 2010; Aftab Ahmed, 2010) presented here is real data for undergraduate semester I year 2011. The department runs two academic semesters for spring and fall semesters per annum. The dataset comprises over eight semesters data, occasionally remedial sessions become also the part of planning. Typically, five to six courses are supposed to be offered to each group with justified necessary credit hours. The courses are separated into two sessions per week. Regardless of under graduation exams, the curriculum courses are also offered simultaneously to several students from diverse departments.

Table 1 illustratesthe dataset specification in detail. Approximately four rooms are reserved for theory classes and one lab for practical. Seven regular faculty members and very small number of parttime teachers are also hired. The regular events are stretched in five uninterrupted days of week whereasthe remedial sessions are settled down on weekends. The working day consists of four sessions where each session is one and half hour long. There is short recess between two assigned events. The limited resources, number of events and extremely intricate constraints stands the problem incredibly tricky.

Hard Constraints:

The hard constraints exceptionally should remain unbreakable under all circumstances. The chance of conflict likely rises when two or more events require a shared resource concurrently. The following hard constraints are common in a wide range of academic scheduling problems.

Table 1: Dataset Specifiation.

No.	Explanation	Quantity
01	Rooms	04
02	Labs	01
03	Teaching Faculty	07
04	Sessions Per Day	4
05	Working day	5
06	Courses	45
07	Courses per Week	>90

Terminologies:

E	Over all events in week,$E= \{e_1,e_2,e_3...e_n\}$
EvtUB	Daily limit (Upper) of events per curricula
EvtLB	Daily limit (Lower) of events per curricula
EvtDayLmt	Represents the desired number of events per week
L	All faculty staff
G	Groups of students
T	Overall events in a specific session
C	Shows the desired capacity of the class room
D_i	Shows the working day $i \in \{1, \cdots, 5 \text{ or } 6\}$
K	Stand for the fix length time of any session
t	is placement of single event
V	The buildings/venue having class rooms
Z	All the resource persons
R	Available rooms in department, $R= \{r_1,r_2...r_n\}$
Q	Required equipments for course
S	is number of courses offered to students group
F	is total number teachers in faculty
M	Matrix of availability for each teacher
S_{ij}	A single dayfixtures for a student's group where $i \in \{1, \cdots, G\}$and $j \in \{1, \cdots, D\}$
L_{ij}	Coursed load per day for a teacher where $i \in \{1, \cdots, N\}$ and $j \in \{1, \cdots, D\}$

Hard Constraint No. 1 (HC1):

Multiple events allocation in the same period cannot not be permissible. A hard violation will be recorded if any dual assignment is observed in a single period.

$$\sum_{i=1}^{K\times D} \sum_{j=1}^{T} e_i e_j \leq 1 \text{ for } \forall j \in \{1 \cdots G\} \tag{3.1}$$

Hard Constraint No. 2 (HC2):

Any teacher must not be assigned multiple lectures at the same time. The conflicting fixtures represent a violation

$$\sum_{i=1}^{K\times D} \sum_{j=1}^{L} L_{ij} \leq 1 \text{ where } \forall j \in \{1 \cdots L\} \tag{3.2}$$

Hard Constraint No. 3 (HC3):

At the moment just one event can be allotted in a class room, double lecturers at the same venue and the same time counts a hard violation.

$$\sum_{i=1}^{R} e_{ij} \leq 1 \text{ for } \forall j \in \{1 \cdots E\} \tag{3.3}$$

Hard Constraint No. 4 (HC4):

Room capacity must be adequate to accommodate enrolled students.

$$\sum_{j=1}^{E} r_i e_j \leq C \text{ for} \forall j \in \{1 \cdots R\} \tag{3.4}$$

Soft Constraints:

Usually the low penalty cost is associated with soft constraints. Minimum numbers of soft violations decide the overall quality and work efficiency. The fixed penalty values are adopted for the constraints violation and penalties scale is deposited based on common practice. Following soft constraints are frequently occurred in timetabling.

Room Stability- Soft Constraint No.1 (SC1):

Subsequent event(s) of any group are believed to be assigned over the same placement row.

$$\sum_{i=1}^{S} \alpha_i \leq r \text{ where } r \in R \tag{3.5}$$

Student Max – Soft constraint No. 2 (SC2):

Scheduling the events beyond the given range is supposed to be avoided.

$$\sum_{i=1}^{D} \sum_{j=1}^{G} e_{ij} \leq EvtUB \tag{3.6}$$

Teacher Load – Soft constraint No. 3(SC3):

Total lecturersper teacher should not go over upper bond limit.

$$\sum_{i=1}^{Z} L_i \leq 2 \tag{3.7}$$

Fitness Function:

The fitness function operates over the dataset is described is as under,

$$Maximize\ f(x) = \frac{1}{1 + \sum_{i=1}^{n} \alpha H_i + \sum_{j=1}^{n} \beta S_j} \tag{3.8}$$

Where α and β stand for penalty coefficient, and S_i, and H_j refer to the specific soft and hard constraints respectively.

The fitness function examines the most fitted chromosomes in generated populations respectively. The penalty summation of hard constraints is required to be nullified while the lowest number of soft violation is greatly desired. In generalthe fitness function may return the outcome in fractional values between 0 and 1. Generic Hyper Heuristic Framework

A typical hyper heuristic framework contains following components

- A top level Metaheuristic which mange overall activities.
- A set of wellknown domain specific low level heuristics.

- The fitness functions for each problem instance.

Figure 2 depicts the abstract level of the proposed framework. First two layers do not contain any problem specified flow of information which makes higher level more generic. Third layer includes the set of problem specified low level heuristics which operates on problem domain directly.

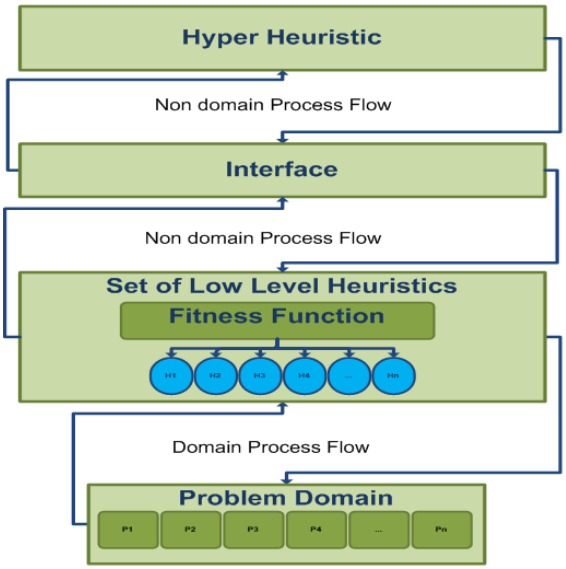


Fig. 2: The Generic Hyper Heuristic Framework.

Higher Level Mechanism:

The GA is computational inspiration and of prominent Darwin's evolutionary theory. The GA evolves the group of partial solutions (chromosomes) called genome or populationin particular. The generated population is desired to be converged towards optimal point of solution. The offspring for each new generation are selected by some decisive fitness criteria. The Hyper-GA written bellow is adapted for this research work.

Algorithm (GA): Hyper Genetic Algorithm

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

The generation is a set of chromosomes in which each individual is complete sequence of calling heuristics for specific timetabling problem. At the same time as chromosome further split into the details of local heuristic with respect to its characteristics including Identification, Fitness level, rate of conversion and time related matters. Figure 3 is illustrating the data representation of chromosomes and genes.

Initialization:

The GA gets initialization directly from the first phase ofthe set of partial solutions. Each chromosome slightly differs from others due to diverse placements of genes resulting in a robust and converging initial population.

Crossover:

The Crossover selects genes from parent chromosomes and creates a new offspring. The simplest way of

implementing this operator mark randomly some crossover point in parent chromosomes, afterward copy the selected partition from both points into child to shape up new generation.

Mutation:

After a crossover is performed mutation takes place. The operator is used to prevent population from local optimum type. The mutation randomly changes in new offspring. For binary encoding we can switch on/off few randomly chosen bits from 1 to 0 or from 0 to 1.

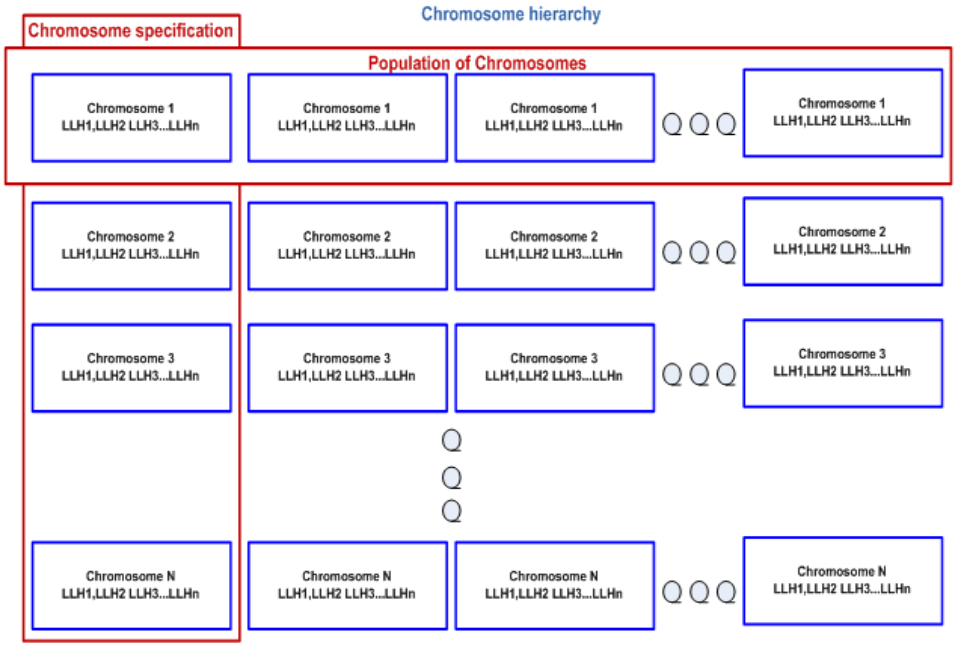


Fig. 3: Chromosomes Representation.

Low Level Heuristics:

The set of low level heuristics (LLH) is very essential element of hyper heuristic model. All low level heuristics are domain specific where each heuristic converges the problem instance for making some constructive changes.

Table 2: Set of Low Level Heuristics.

ID	Algorithm Name	Scope	Funct-ion	Interact	LOC
LLH_1	MaxPenalizedDay	Day	Shift	Semi –R	89
LLH_2	ToLessSituDay	Day	Shift	Semi –R	60
LLH_3	WithDayConstImp	Period	Shift	Progress	86
LLH_4	InColumn	Period	Swap	Progress	87
LLH_5	NeighboringPeroid	Period	Shift	Random	62
LLH_6	Swap_InRows	Period	Swap	Progress	67

Table 2 shows the different properties of LLH employed in this paper. The LLH perform either shift or swap operation upon violated events. The shift operation requires vacant place on target side whereas swap operation is exchange of two events especially in saturated problem instances. The LLHs are designed to reduce the total penalty cost effectively. The random heuristics are used to shuffle the events throughout layout so the gaps can be inserted among events for convenient shuffling. The semi random (Semi-R)type of LLH operators pick out the violated event under some criteria and shift to range of places randomly. On the other hand progressive LLHs necessarily make some positive changes into problem instance or retain it on previous state. The scope of LLHs tells the working range of subspace, i.e., some LLHs are effective on the day level and some on the period level.in the last column of Table 2 total number of coding lines for each procedure is given. Figure 4 reveals the extent of low level heuristics over problem instances.These LLHs are directly interacting with problem domain and are managed by top level strategy.

Selection Mechanism of Local Heuristic:

In this research work genetic evolving technique is used for choosing local heuristics. The idea is depicted as a utility rank/penalty weight assigned to each local heuristic. If a selected heuristic shows improved performance, the penalty rank is increased by a given rate (which is based on calculated parameters). The utility values are allowed to change within an interval of (min, max). At each iteration a set of local heuristic used to select decisively based on the rank/weight values.

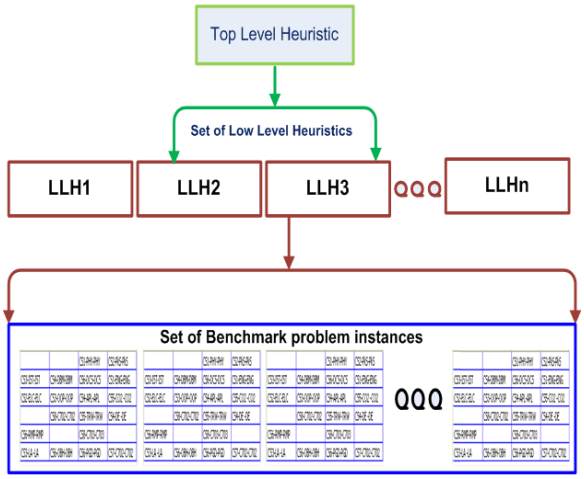


Fig. 4: Interaction with Low Level Heuristics.

Experimental Results:

The computational approach is investigated on benchmark datasets, comparative results revealed the broad potential and capacity of adapted methodology. All the experiments are performed on AMD® Turion 64 2 Technology, TL-58, 2.0 GB RAM, ATI Redeon Xpress 1250. The Python language version 2.6 is used to implement all the coding.

Table 3 briefly represents the selected parameters which were observed. At the time of initialization, it isendeavoreddeliberately to spread up the events over the scheduling layout.Figure 5 depicts the each constraint proceeding individually, lessening number of violations mainlywith respect to hard constraints can be observed. Hard Constraints HC_3 as well as HC_4has very small presenceintable 3 because HC_5 is entirelyeradicated by layout-design plan however HC_4 is came into account during construction of event elements.

Table 3: GA Parameters details.

No.	GA Parameter	Value
01	Crossover Method	Order Crossover
02	Selection Method	Rank Selector
03	Genome size	35
04	Total Generations	1000
05	Crossover Rate	0.08
06	Mutation Rate	0.0125

Figure 5 depictsthe overall progress precisely, it illustrates stable decreasing penalty cost of constraints separately. In first segment, penalty cost drops off with low rate because low level heuristics has been chosen randomly until their selection scale is to be established. On the other hand, second segment is identifiable by steady and sharp decline of penalty cost. The hyper-GA approaches to stable state of results after finite generations. The occasional rise and fall of penalty cost for constraint SC_1 is noticeable due to swapping of penalized timeslots from other sessions by rest of the solver. The situation is repeated for SC_2 as well. Figure 6 illustrates the total penalty cost in hyper-GA computing phase. The gradually increasing saturation is cause of low rate of dropping off particularly in ending, however very small number of violations is stayed unsettled. Figure 7 refers the fitness function value generated throughout the process. At the last the value is close to 1 that indicates most of constraints have been solved. Table 4 is giving overview of all generations with respect to constraints decreasing order.

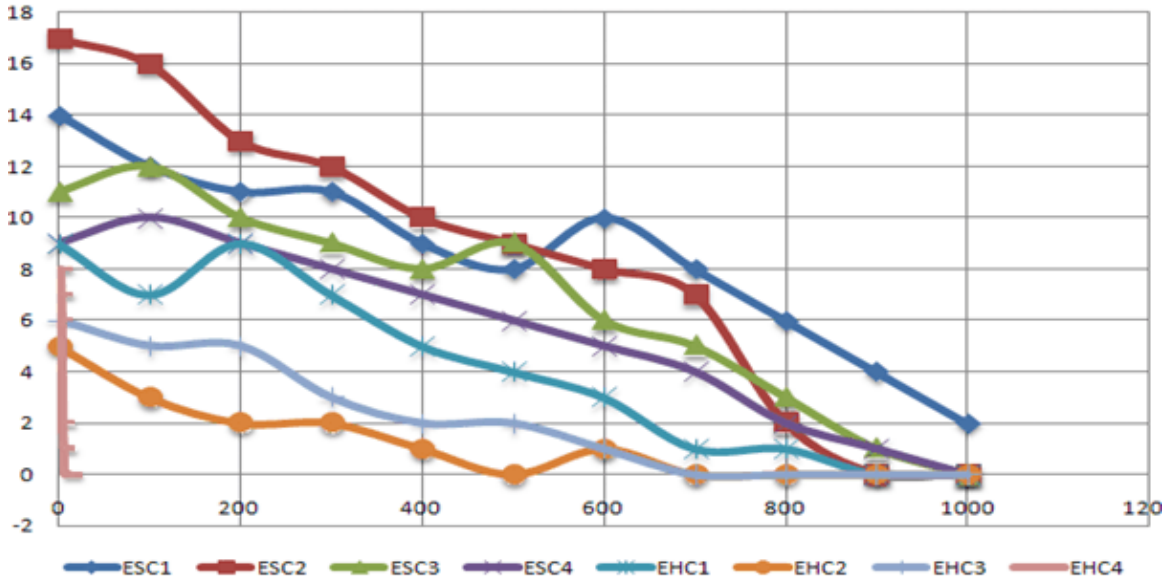


Fig. 5: Constraints Individually.

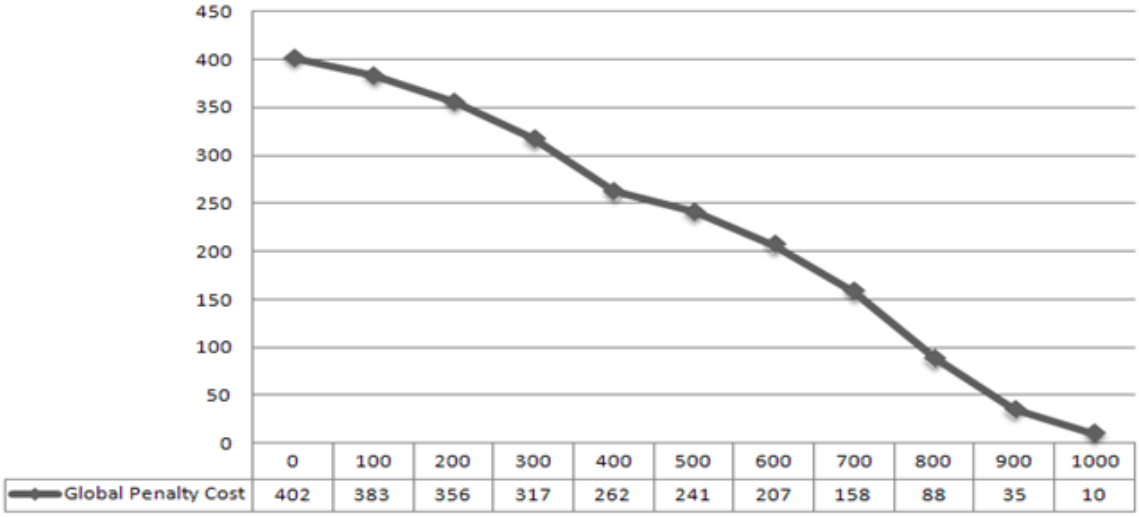


Fig. 6: Total Penalty Cost.

Table 4: Constraints Penalty per Generations.

Generations	SC_1	SC_2	SC_3	SC_4	HC_1	HC_2	HC_3	HC_4
0	14	17	11	9	9	5	6	8
100	12	16	12	10	7	3	5	7
200	11	13	10	9	9	2	5	6
300	11	12	9	8	7	2	3	2
400	9	10	8	7	5	1	2	1
500	8	9	9	6	4	0	2	0
600	10	8	6	5	3	1	1	0
700	8	7	5	4	1	0	0	0
800	6	2	3	2	1	0	0	0
900	4	0	1	1	0	0	0	0
1000	2	0	0	0	0	0	0	0

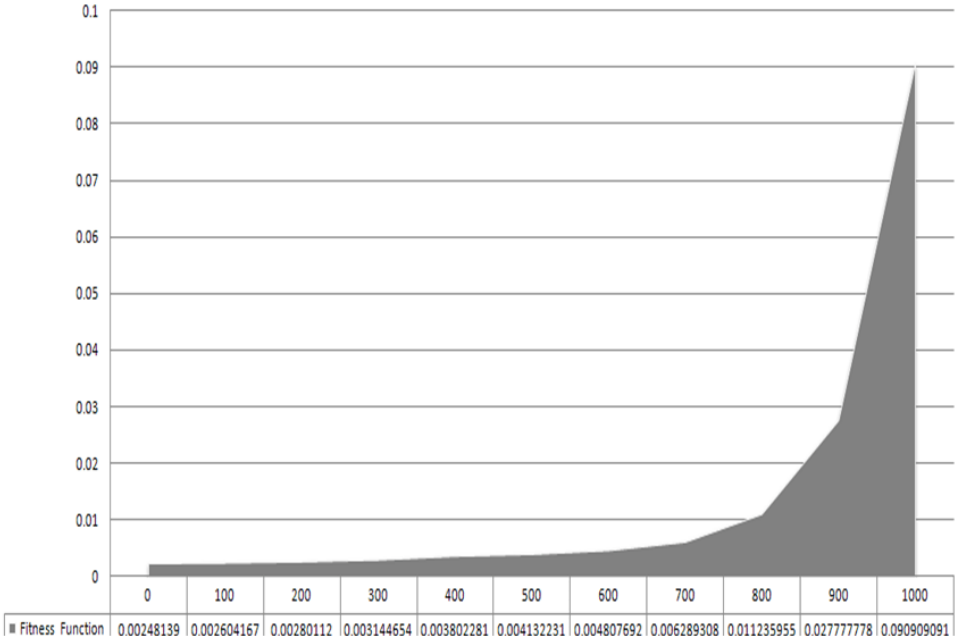


Fig. 7: Fitness Function.

Conclusions:

The research in this reach paper is proposed to investigate the hyper-heuristic approach that proved to be exceptionally competent of providing superior solutions over academic course and examination timetabling problems. A hyper-heuristic is a state-of-art and high level sophisticated problem solving methodology that operates over the of low level heuristics domain. The proposed mechanism in fact, evolves the solver rather than solutions itself on a higher level of abstraction. Secondly the generic framework can easily be applied on other instances of the same class of the problem with minor changes. Research is focusing on efficient local heuristic selection and move acceptance ways to select local heuristics.

REFERENCES

Abdullah, S. and H. Turabieh, 2008. "Generating university course timetable using genetic algorithms and local search," presented at the Third 2008 International Conference on Convergence and Hybrid Information Technology ICCIT.

Aftab Ahmed and L. Zhoujun, 2010. "Solving Course Timetabling Problem Using Interrelated Approach," in *2010 IEEE International Conference on Granular Computing* San Jose, California, USA, pp: 651-655.

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

Burke, E.K. *et al.*, 2004. "A Time-Predefined Local Search Approach to Exam Timetabling Problems," *IIE Transactions,* pp: 509-528.

Burke, E.K. *et al.*, 1994. "A Genetic Algorithm Based University Timetabling System," presented at the East-West Conference on Computer Technologies in Education, Crimea, Ukraine.

Burke, E.K. *et al.*, 2003. "A Tabu-Search Hyperheuristic for Timetabling and Rostering," Journal of Heuristics, 9(6): 451-470.

Burke, E. *et al.*, 2005. "Hybrid graph heuristics in a hyper-heuristic approach to exam timetabling problems," in *The Next Wave in Computing, Optimization and Decision Technologies*, pp: 79-92.

Cowling, P. *et al.*, "A hyperheuristic approach for scheduling a sales summit," presented at the In Selected Papers of the Third International Conference on the Practice and Theory of Automated Timetabling, PATAT 2000, Konstanz, Germany.

Cowling, P. *et al.*, 2002. "An investigation of a hyperheuristic genetic algorithm applied to a trainer scheduling problem," in *Proceedings of the Congress on Evolutionary Computation 2002*, pp: 1185-1190.

Fang, H.L. *et al.*, 1993. "A promising genetic algorithm approach to job shop scheduling, rescheduling and open-shop scheduling problems," in *Proceedings of the Fifth International Conference on Genetic Algorithms*, San Mateo, pp: 375-382.

Fang, H.L. *et al.*, 1994. "A promising hybrid ga/heuristic approach for open-shop scheduling problems.," in *Proceedings of ECAI 94: 11th European Conference on Articial Intelligence*, pp: 590-594.

Fisher, H. and G.L. Thompson, 1961. "Probabilitistic learning combinations of local job-shop scheduling rules.," Factory Scheduling Conference, Carnegie Institute of Technology.

Mahdi, O. *et al.*, 2003. "Using a genetic algorithm optimizer tool to generate good quality timetable," in *Proc. of the 10th IEEE International Conference*, pp: 1300-1303.