Sindh Univ. Res. Jour. (Sci. Ser.) Vol.43 (2) 261-168 (2011)

SINDH UNIVERSITY RESEARCH JOURNAL (SCIENCE SERIES)

A Comparative Study On Three Hyper-Heuristic Approaches For Solving Benchmark Scheduling Problems

A. AHMED, A. H. S. BUKHARI * AND I. A. ISMAILI**

Department of Computer Science, BUITEMS, Quetta
Corresponding Author: Aftab Ahmed, E-mail: aftabshaikhs@hotmail.com, Cell# +92-333-7815354

Received 03rth August 2011and Revised 19th October 2011)

Abstract: The research work compares the outcome and solving capabilities of three prominent algorithms. Each algorithm are separately implemented as higher level heuristic to manage the group of low level heuristics (LLHs) in order to solve the benchmark university scheduling instances. The study comprises over Particle Swarm Optimization (PSO), Genetic Algorithms (GA) and Evolutionary Algorithm (EA). All these optimization techniques are highly appraised for their skills to handle the complex problems. A number of classical operators and parameters have been examined with each hyper-heuristics due to high diversity in datasets. Secondly, Domain specified Low Level Heuristics have been designed under several operational classifications. In addition, obtaining effective deployment and utilization of the academic resources to the greatest extent are counted as supplementary but essential advantages of the research work.

Keywords: *Hyper-Heuristic, Benchmark Scheduling Problems, Constraints*

1. INTRODUCTION

The academic scheduling is an essential component of the educational calendar in order to regulate the events. Tough constraints stand the scheduling problem in the group of NP-hard classification. So, traditional problem solving methods sometimes cannot perform desirably. Hyper-heuristic is very successful and state-of-art technique to solve the scheduling problem. In this research work, three prominent algorithms are applied as higher level mechanism managing domain specific low level heuristics in order to examine managerial capabilities of each.

Particle swarm optimization is a significant member of swarm Intelligence. It emulates the behavior of a birds swarm which looks for an optimal food source. In PSO, each particle tries to reach the best ever position of swarm using individual and collective intelligence. Therefore the entire swarm is converged by using information sharing approach. Basic PSO as introduced by Kennedy and Eberhart (J. Kennedy and Eberhart 1995) which is briefly described in equation 1 and 2, where *x* represents an individual, *i* stand for the identification of particle, *k* shows iteration, and *v* is the particle's velocity. $c_1 r_1 (p_k^i -$: Refers to the best location of the individual ever traversed (Individual Intelligence), and $c_2 r_2 (p_k^g -$: points to the best position of any particle in the swarm ever traversed (Collective intelligence). While the r_j represents weight of inertia which directly effects on velocity, c_j make influence over individual and swarm's knowledge respectively. r_j are random numbers ensures the process not to be trapped into local optima.

$$x_k^i = x_k^i + v_{k+1}^i \quad (1)$$

$$v_{k+1}^i = \omega v_k^i + c_1 r_1 (p_k^i - x_k^i) + c_2 r_2 (p_k^g - x_k^g) \quad (2$$

The term of Genetic algorithms was coined by John Holland in his book *"Adaption in Natural and Artificial Systems"* published in 1975. The fundamental notion is to mimic the natural system in order to emerge optimal solving method. In fact, the Genetic Algorithm is a computing version of evolution theory. GA converges the chromosomes in shape of partial/tentative solutions to reach optimal level. Offspring of each succeeding genome nominates by predefined selection criteria. The core reason of using GA is its accurate approach and computational maturity to obtained promising results.

Most of the features are common between Genetic and Evolutionary algorithms, so a general review is given here. Generation is a combination of chromosomes where each chromosome further divides into genes. Probably the most common pattern of encoding the chrome is using binary string. The specific bits on each position stand for particular features or sometimes entire binary string is used for solo meaning. There are also several encoding methods, mostly depends on nature of problem. Other encoding schemas contain integer, real and permutations representations.

- *Initialization* - Initialization is set of basic values assigned to first genome.
- *Selection* - is to refine the fitness level of individuals and discard the off springs having weaker fitness. Chromosomes are chosen from the one generation to other for reproduction. Roulette selection is one of the prominent classic GA/EA selection methods. The fittest chromosomes are

*Faculty of Information and Communication Technology, BUITEMS, Quetta
** Sindh University Campus, Mirpur Khas

likely to be selected. Roulette selection is based on linear search with the slots in the wheel marked by fitness weight.

- *Crossover* - Crossover reproduces the offspring of selected parent chromosomes, expecting that population becomes enriched with better individuals. In single point crossover, the operator chooses one crossover points and copy slice of data of both sides than swaps with each other. **(Fig.1)** shows the single point crossover in detail.
- *Mutation* - Mutation prevents population falling into local optimum. Mutation alters the offspring on random or predefined point. **(Fig. 2)** demonstrates the mutation.

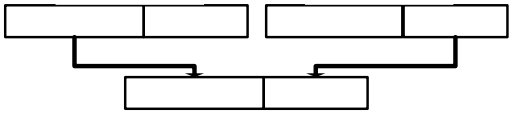

Fig. 1 Crossover between two chromes

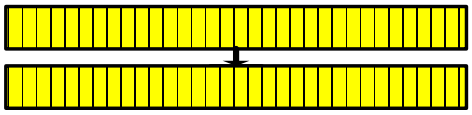

Fig. 2 Mutation of selected bits

2. RELATEDWORK

A plenty of research approaches have been explored over the investigating of automated course/examination scheduling for a number of decades. A broad survey of the classical scheduling techniques is held by (Burke *et al.*, 2004). A plenty of researchers exercise the scheduling problem using Genetic algorithm/ Evolutionary Method (GA) (Ahmed *et al.*, 2011; Burke *et al.*, 1994; Mahdi *et al.*, 2003). (Ahmed and Li 2010) applied hybrid approach using GA along with local search to solve successfully course scheduling problem.

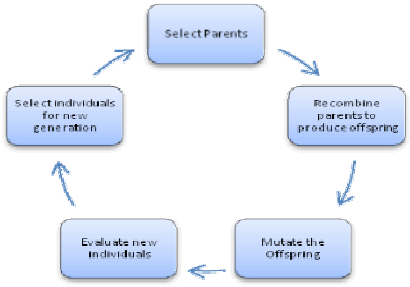

Fig. 3 Processing Cycle of GA

(Fig. 3) depicts processing cycle for Genetic and evolutionary algorithm for hyper heuristics framework. (Angeline 1998) show and that the PSO obtained the optimal solution considerably quicker than a range of other optimization techniques. Additionally Shi (Shi and Eberhart 1999) described the fast conversion and varied dataset scalability of PSO. In (Kennedy and Spears 1998) PSO demonstrate notable performance and objective achievement capacity greatly in sophisticated binary as well as continuous problems. In the research work (Shi and Eberhart 1998) the authors appreciated sturdiness of PSO for solving dynamic and non-linear problems. As for as, its scope is concerned for scheduling problems, Particle Swarm Optimization is decidedly competent for combinatorial optimization especially for academic scheduling. PSO is proved to be a reliable replacement of other classical methods. It established to be extremely efficient for finer detection and solving competence over scheduling problem instances (Parsopoulos and Vrahatis 2002). The terminology of hyper-heuristics was firstly introduced by Cowling *et al., ("The hyper-heuristics manage the choice of which lower-level heuristic method should be applied at any given time, depending upon the characteristics of the region of the solution space currently under exploration."* Hyper heuristics are widely identified as intelligent selection of the correct heuristic or operator in a given situation (Burke *et al.*,2003). Terashima-Marín, *et al.*,(Terashima-Marin *et al.*, 2005) successfully implemented GA based Hyper-Heuristic on 2D Cutting Stock problem.

3 PROBLEM FORMULATION

Timetabling schema is categorized into two main types, hard & soft constraints. In general, hard constraints satisfaction is fundamental qualification of any solution. While reasonable number of soft constraints removal, definitely scale up the solution quality.

3.1 Hard Constraints

In fact, any academic scheduling is normally recognized at feasible level if all the hard constraints violations have been eliminated, however not a single hard violation can be tolerated in real world or benchmark solution.

3.1.1 HC_1 *(Group Conflict):* student(s) should not be notified for two or more simultenous events, a cnflict will be counted in such case.

3.1.2 HC_2 *(Lecture Conflict):* double lectures mst not be sheduled for any teacher on same time.

3.1.3 HC_3 *(Room Occupancy):* mostly in nomral size of class room only single event can be assigned therefore dual fixtures on same vaune and time must be counted a hard violation.

3.1.4 HC_4 *(Room Suitability):* seating capacity or class room size must be equal greter than enrollment number.

3.1.5 HC_5 *(Teacher Availability):* Lecture assignment must be avoided if teacher cannot make availability on a specific time due to any valid reason.

3.2 Soft Constraints: Soft constraints facilitate the resource persons and students to perform daily routines conveniently. In fact, all the soft constraint cannot be removed from the search space in almost all the real world and benchmarking cases however state of the art techniques may ensure the better quality. So that soft conflicts are directly proportional to quality scale.

3.2.1 *SC_1 (Room Stability):* subsequent lecture events are required to be fixed on single venue.
3.2.2 *SC_2 (Windows):* Time gape or window should be avoided among/between events.

3.2.3 *SC_3 (Student Max):* exhausting number of consective events unremitting must not be scheduled on same day.

3.2.4 *SC_4 (Student Min):* the number of events meets atleast the lower bond recommended per day for each group.

3.2.5 *SC_5 (Teacher Load):* Teaching load is highly requisited to be in prdeirned limit.
3.2.6 *SC_6 Min Working Day:* The sessions of each course are supposed to be scattered throughout working days. Each day bellow the required scale is counted as violation.

3.2.7 *SC_7 (Isolated Lectures):* Only single fixture on a day of specific curriculum group is supposed to be soft violation.

3.2.8 *SC_8 (Travel Distance):* Students are not supposed to move from one building to other within consecutive events. All the assignments of the group should be fixed in a same building; travel between two buildings for successive assignments in a single day would be counted a violation.

3.2.9 *CS_9 (Room Suitability):* Lectures should be assigned in well-equipped room as per requirement of course for example Multimedia projector, sound system

3.2.10 SC_{10} (Double Lectures): Some courses require adjacent lectures per week especially in executive or professional classes on demand of visiting faculty members; failure of such requirement is considered a soft violation.

A. Dynamic Penalty Scale

Penalty scale decides the weight of violation occurred, basic penalty costs are given in **(Table 1)** which shows the variation on each classified complexscale. The penalty cost scheme is illustrated by Alex Bonutti *et al.*, (Bonutti 2010). In this research work, one higher scale is added in all bench mark problems for testing efficiency of research approach. Each Scale comes up with a different set of changeable weights and intensity under different schemas. Penalty cost raises with every additional violation occurrence. In addition, number of constraints and variables increases from easy to a hard problem level. Such variations make each dataset instance different from other. The constraints represented with variables HC (Hard Constraint) in Table 1 are mandatory for datasets while rests of the conditions are soft constraints. The SC group of constraints is highly required to scale up the quality and flexibility of the solution. Constraints sketch the entire blueprint for workable scheduling and facilitate the students and teachers in many ways. For example, SC_3 makes sure that, students not to waste their time between travelling from home to institute for inadequate number of lecturers.

Var.		Constraint Label	S_1	S_2	S_3	S_4	S_5	S_6
HrC_1	Hard Con	Lectures	∞	∞	∞	∞	∞	∞
HrC_2		Conflicts	∞	∞	∞	∞	∞	∞
HrC_3		Room Occupancy	∞	∞	∞	∞	∞	∞
HrC_4		Availability	∞	∞	∞	∞	∞	∞
HrC_5		Room Suitability	NA	NA	3	HC	NA	NA
SC_1	Soft Constraints	Room Capacity	1	1	1	1	1	NA
SC_2		Min Working Days	5	5	NA	1	5	5
SC_3		Isolated Lectures	1	2	NA	NA	1	2
SC_4		Windows	NA	NA	4	1	2	1
SC_5		Room Stability	NA	1	NA	NA	NA	2
SC_6		Student Min Max Load	NA	NA	2	1	2	1
SC_7		Travel Distance	NA	NA	NA	NA	2	NA
SC_8		Double Lectures	NA	NA	NA	1	NA	NA
SC_9		Teaching Max Load	NA	NA	NA	NA	NA	5

Table 1 Benchmark Dataset specification (A. Ahmed et al. 2011)

B. Higher Level Mechanism

A generic hyper-heuristic framework is briefly explained in Figure 4. A higher level of heuristic manages a group of low level perturbative heuristics (LLHs). First, A tentative solution *S* is formed, later the LLH which shown the highest fitness has been selected (at first random selection was made) and applied on dataset. Fitness function endorses the acceptance or rejection of new move. In case of acceptance the newly produced outcome swaps the existing one otherwise the new tentative solution use to cast off and process moves to succeeding iteration. Following Algorithm depicts the overall logic.

Algorithm: Hyper-Heuristic Procedure

1. Produce tentative set of Solutions (S)
2. set $S^p \leftarrow S$
3. **Do Loop**
   a. Selection from pool of Low Level Heuristic
   b. Implementation of Low Level Heuristic
   c. Evaluate *fitness*(S^p) of each LLHs
   d. Produce tentative solutions S^{new}
   e. **IF** S^{new} is improved **Than** set $S \leftarrow S^{new}$
   f. **Else** $f(S^{new}) < f(S^P)$ **Than** set $S \leftarrow S^p$
4. **Until** Exit Criteria
5. **return** (S)

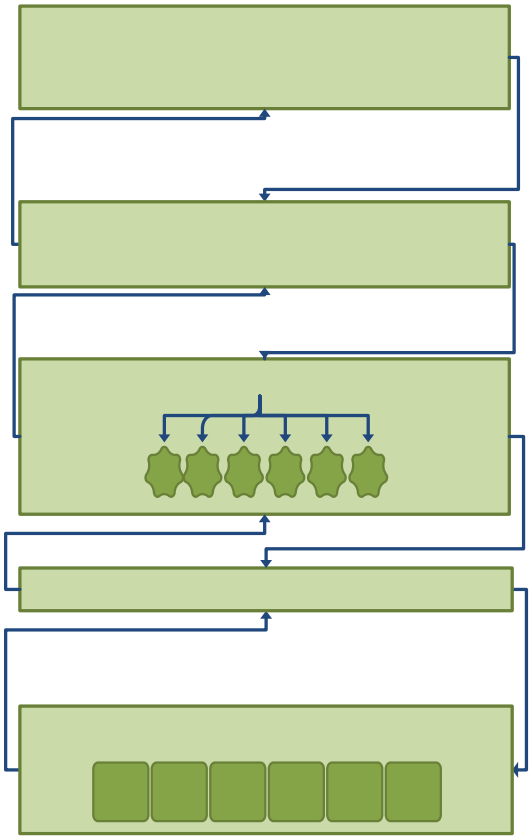

Fig. 4 Hyper-Heuristic Framework

GA/EA as top level heuristic(s) has produced stunning results; The GA/EA evolves initial results or chromosomes. Each generation is made converging of outcome to optimal point. Selection of chromosomes for every successive generation made on the basis of predefined fitness criteria. **(Fig. 5)** depicts the GA prospective of Hyper-heuristic in detail. Each individual chrome in generation is sequence of heuristics and their specification details which containing invoking Identification, Fitness rank etc. **(Fig. 6)** is briefly explaining chrome representation. Initialization for first generation is made by random function where ach chromosome differs from other in order of sequence. The Crossover operator chooses the LLHs genes from chromosomes and assembles slightly more fit offspring. The single bit crossover is implemented, this method set randomly crossover point in two chromosomes and move the elements into new one. Following the crossover operation the mutation takes place. The mutation prevents the outcome from local optima trap. The operator makes random alteration in offspring. For example in binary encoding it switches on or off few bits in chromosome.On the other hand in Particle Swarm Optimization, every particle works as counterpart of chromosome. Each particle gradually converges to *PBest* and *Gbest* in order to obtained optimal solution. **(Fig. 7)** is explaining the mechanism of PSO based hyper-heuristic.

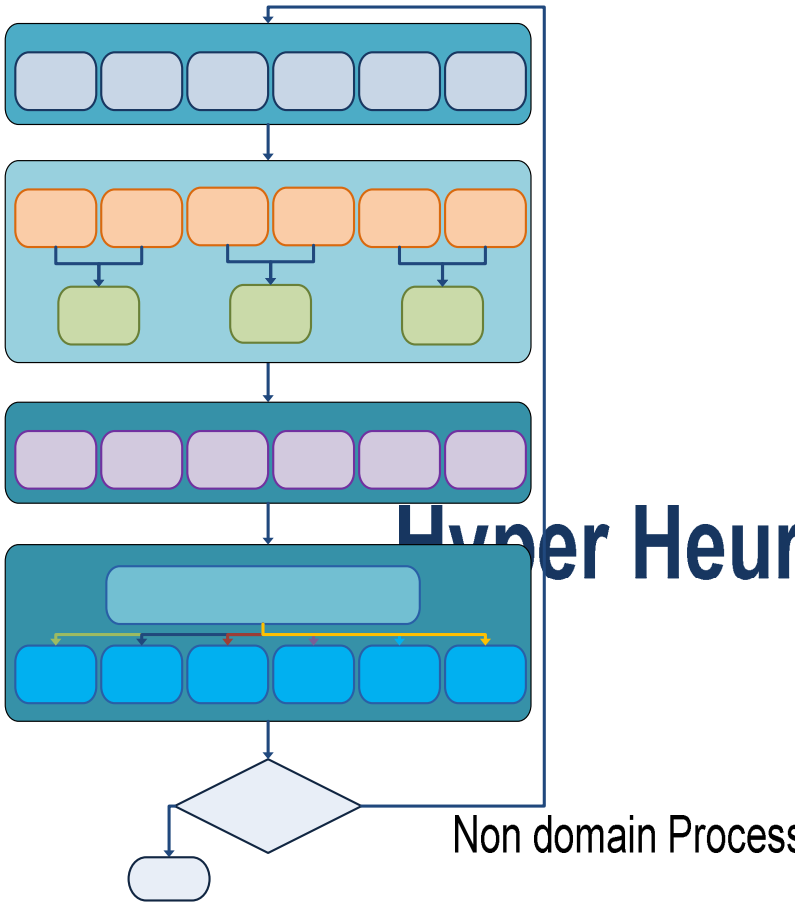

Fig. 5 GA as Hyper-Heuristic

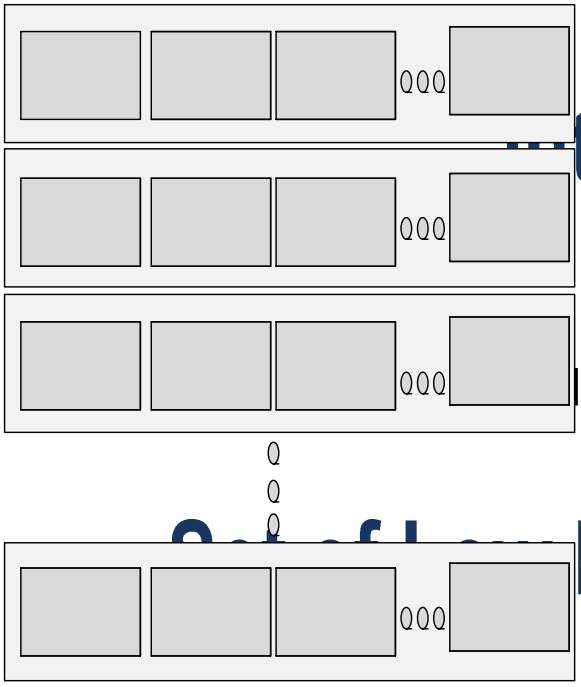

Fig. 6 Chromosomes Representation

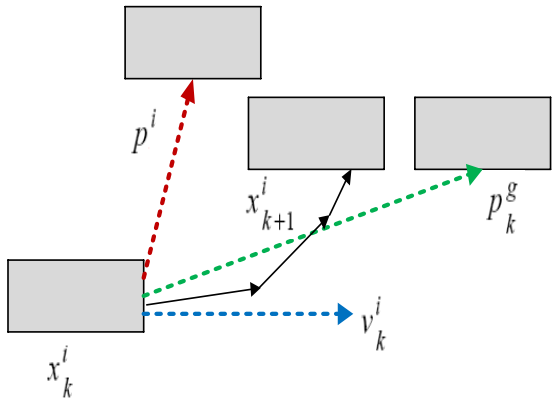

Fig. 7 PSO framework

A. *Low Level Heuristics*

The group of low level heuristics (LLH) is core component of hyper heuristic framework. In general low level heuristics are domain specified and each LLH is intended to make progressive change(s) in problem instance. Table 2 illustrates the various features of LLHs applied on the benchmark instances. Functionally LLHs carry out either swap/shift operation on datasets. The shift operator needs empty place on destination while swap operator is interchange between two events, this operator is more effective specifically in highly saturated problems. The designing aim of LLHs is lessen the penalty cost rapidly.

ID	Algorithm Name	Scope	Fun	Interaction
LLH_1	MaxPenalizedDayToTail	Day	Shift	Semi –R
LLH_2	MaxSituratedDayToTail	Day	Shift	Semi –R
LLH_3	LessSituratedDay	Day	Shift	Semi –R
LLH_4	DayConstToLessSituDay	Day	Shift	Semi –R
LLH_5	WithDayConstImpro	Period	Shift	Progressive
LLH_6	Swap_InDays	Period	Swap	Semi –R
LLH_7	RandDayImprovment	Day	Shift	Progressive
LLH_8	Swap_InColumn	Period	Swap	Progressive
LLH_9	NeighboringPeroid	Period	Shift	Random
LLH_{10}	Swap_InRows	Period	Swap	Progressive

Table 2 Specification of Low Level Heuristics

The randomly interactive LLH disperses the events on entire layout in order to make the vacuums among events. Similarly the semi-random (Semi-R) heuristics choose the conflicting event using predefined schemas and moves it suitable empty slot. In contrast, progressive/constructive heuristics make optimistic changes in dataset or preserve and restore its earlier status. The classification under SCOPE shows the operating range of particular LLH, for example a few LLHs are useful upon day-level and some can work better on the session-level. Figure 4 exposes the scope of LLHs over problem domain.

D. *Selection Mechanism of Local Heuristic*

In this work, evolving strategy is applied for selecting local heuristics. According to it a utility rank/weight is to each LLH after performance. In case of showing improvement in performance the rank increases by predefined calculated value. The utility values are chosen from range of (min, max) intervals.

4. EXPERIMENTAL RESULTS

The empirical results of comparative study are presented here, many of the parameters and operators have been tested and those found effective in their performance are given here. **(Tables 3, 4, and 5)** show the optimal parameters found experimentally effective on most of the datasets. Although some particular combination of operators comparatively perform more efficiently over some specific cases but this study is conducted to explore generic features of algorithms for benchmark datasets. All the experiments are performed Lenovo® Intel CORE ™ *i3*, 2.27 GHz, 2.00 GB RAM. The Python language ver. 2.6 is adapted to code the research work.

Particle

LLH1,LLH2 LLH3...LLHn

Table 3 GA Parameters

	Parameter	Value
1	Selector	Rank selection
2	Max Evaluations	1000
3	Number of elites	2
4	Population Size	30
5	Chromosome size	8 bits
6	Mutation Operator	Single bit
7	Crossover Operator	Single Point
8	Mutation rate	0.09
9	Crossover rate	0.90
10	Termination Criteria	Converging + Max iterations

P

LLH1,LLH

Table 4 PSO Parameters

	Parameter	Value
1	topology	Ring topology
2	Neighborhood size	30
3	Max Evaluations	1000
4	Number of elites	0
5	Population Size	30
6	evaluator	Binary
7	Termination Criteria	Converging + Max iterations

Table 5 EA Parameters

	Parameter	Value
1	Max Generations	1000
2	Number of elites	1
3	Population Size	30
4	Selector	Tournament selection
5	Tournament size	Population Size / 3
6	Number of Selection	2
7	Crossover Operator	Single Point
8	Mutation rate	0.02
9	Termination Criteria	Converging + Max iterations

Particle

LLH1,LLH2 LLH3...LLHn

(Fig. 9 to 12) depict the investigation summary (applied over five benchmark datasets) of prominent optimization methods including Genetic Algorithm, Particle Swarm Optimization and Evolutionary Algorithm. The study have look on comparative features of each algorithm as Hyper-Heuristics which operates on similar fitness function, set of low level heuristics and benchmark datasets. Almost all methods have produced the quality outcome with meager differences. Genetic Algorithm

however seems much closer to optimal solution. Graphs given below illustrates initial global penalty of each scale return by fitness function (shown in x-axis) and bars are showing solution achievement scale (GA, PSO, EA) and (Scale1…Scale6) respectively are number of solved constraints at the end.

Tables (6 to 10) are showing comprehensive and two dimensional look of entire comparative study, each scale of complexity scale is evaluated by different top-level strategy but other ingredients (fitness function and set of low level heuristics) remained same. Small amount of difference in performance quality has been observed, as it is visible in outcome tables.

Table 11, illustrates the comparison of adapted research approach to other well-known methods. It has been noticed that the central objective of working in this research direction is to increase the level of generality. The adapted method proved the effectiveness over various instances especially with less saturated datasets. The bold text is showing the finest results.

Table 6 Comparison using Comp 01 Dataset

	Initial Solution	GA	PSO	EA	Avg.	Std. Dev
Scale1	172	4	9	1	4.67	4.04
		168	163	171		
Scale2	287	7	1	2	3.33	3.21
		280	286	285		
Scale3	207	7	3	3	4.33	2.31
		200	204	204		
Scale4	240	7	12	9	9.33	2.52
		233	228	231		
Scale5	231	14	23	10	15.67	6.66
		217	208	221		
Scale6	295	11	14	12	12.33	1.53
		284	281	283		
Total		50	62	37		
Avg.		8.33	10.33	6.17		
Std. Dev		3.56	7.99	4.71		

Table 7 Comparison using Comp 02 Dataset

	Initial Solution	GA	PSO	EA	Avg.	Std. Dev
Scale1	426	11	15	9	11.67	3.06
		415	411	417		
Scale2	939	23	19	28	23.33	4.51
		916	920	911		
Scale3	437	21	24	37	27.33	8.50
		416	413	400		
Scale4	489	32	31	20	27.67	6.66
		457	458	469		
Scale5	530	153	212	167	177.33	30.83
		377	318	363		
Scale6	956	14	12	18	14.67	3.06
		942	944	938		
Total		254	313	279		
Avg.		42.33	52.17	46.50		
Std. Dev		54.71	78.59	59.79		

Table 8 Comparison using Comp 03 Dataset

	Initial Solution	GA	PSO	EA	Avg.	Std. Dev
Scale1	814	37	55	46	46	9
		777	759	768		
Scale2	1237	55	51	42	49.33	6.66
		1182	1186	1195		
Scale3	805	25	24	31	26.67	3.79
		780	781	774		
Scale4	852	327	321	411	353.00	50.32
		525	531	441		
Scale5	936	167	188	160	171.67	14.57
		769	748	776		
Scale6	809	33	30	41	34.67	5.69
		776	779	768		
Total		644	669	731		
Avg.		107.33	111.50	121.83		
Std. Dev		119.84	119.07	149.65		

Table 9 Comparison using Dataset Comp 04

	Initial Solution	GA	PSO	EA	Avg.	Std. Dev
Scale1	373	21	32	20	24.33	6.66
		352	341	353		
Scale2	686	39	45	34	39.33	5.51
		647	641	652		
Scale3	378	4	9	23	12.00	9.85
		374	369	355		
Scale4	421	15	25	14	18.00	6.08
		406	396	407		
Scale5	596	89	99	106	98.00	8.54
		507	497	490		
Scale6	421	14	23	17	18.00	4.58
		407	398	404		
Total	Total	182	233	214		
Avg.	Avg.	30.33	38.83	35.67		
Std. Dev	Std. Dev	30.98	31.74	35.14		

Table 10 Comparison using Dataset Comp 05

	Initial Solution	GA	PSO	EA	Avg.	Std. Dev
Scale1	828	226	251	221	232.67	16.07
		602	577	607		
Scale2	1092	289	290	301	293.33	6.66
		803	802	791		
Scale3	979	293	309	290	297.33	10.21
		686	670	689		
Scale4	1540	257	260	250	255.67	5.13
		1283	1280	1290		
Scale5	1758	575	570	590	578.33	10.41
		1183	1188	1168		
Scale6	1618	105	98	99	100.67	3.79
		1513	1520	1519		
Total	Total	1745	1778	1751		
Avg.	Avg.	290.83	296.33	291.83		
Std. Dev	Std. Dev	155.33	153.50	163.06		

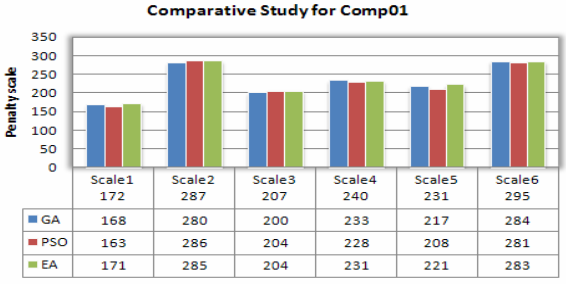

Fig. 8 Instance 1 with six scales of complexity

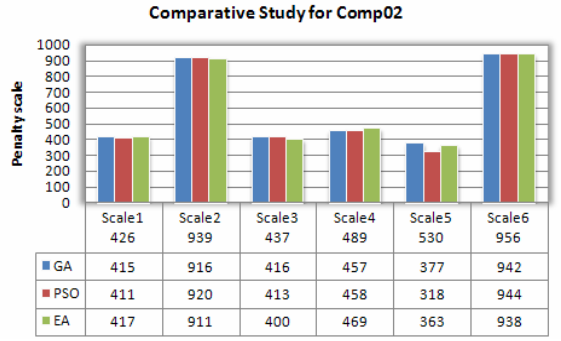

Fig. 9 Instance 2 with six scales of complexity

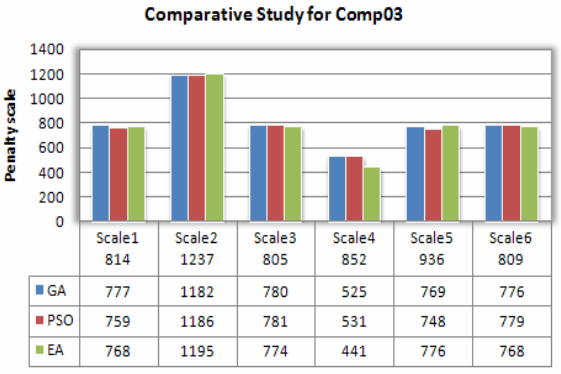

Fig. 10 Instance 3 with six scales of complexity

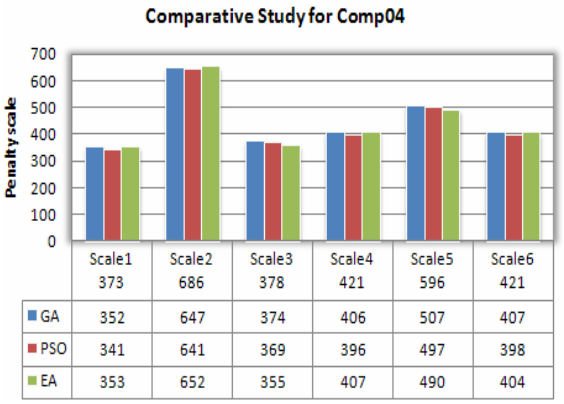

Fig. 11 Instance 3 with six scales of complexity

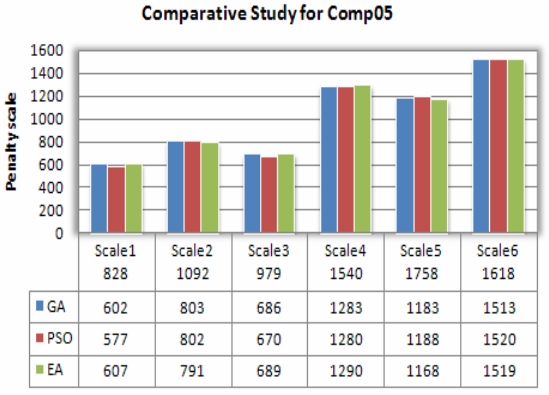

Fig. 12 Instance 5 with six scales of complexity

Table 11 Results comparisons with other techniques

		Evol. Comp	Tabu Search	SA	HC	Hyper-GA
Comp01	Scale1	**4**	**4**	5	5	**4**
	Scale2	**5**	**5**	8	8	7
	Scale3	~	8	7	12	**7**
	Scale4	~	**6**	**6**	21	7
	Scale5	~	11	12	23	14
Comp02	Scale1	20	24	13	12	**11**
	Scale2	58	56	25	**22**	23
	Scale3	~	22	**21**	20	**21**
	Scale4	~	32	35	**24**	32
	Scale5	~	170	145	171	**153**
Comp03	Scale1	41	39	39	**35**	37
	Scale2	82	79	59	56	**55**
	Scale3	~	29	30	**24**	25
	Scale4	~	362	**315**	321	327
	Scale5	~	191	201	**165**	167
Comp04	Scale1	20	18	25	**21**	**21**
	Scale2	39	38	40	**38**	39
	Scale3	~	**2**	8	9	4
	Scale4	~	15	21	**14**	15
	Scale5	~	90	99	91	**89**
Comp05	Scale1	253	240	230	229	**226**
	Scale2	318	291	295	**281**	289
	Scale3	~	324	301	311	**293**
	Scale4	~	260	**251**	278	257
	Scale5	~	601	567	581	**575**

3. CONCLUSIONS

This Research is indented to investigate the variation of Hyper Heuristics framework under diverse prominent solving approaches and parameters. It is has been revealed that there is very meager difference in outcome of various top level heuristic approaches however Genetic Algorithm proved efficiency on multiple occasions.

Future work is planned to incorporate reinforcement learning using TABU search to make inefficient chromosomes or particles restricted /suspended using TABU logic.

REFERENCES:

Ahmed, A., A. W. Shaikh, M. Ali, and A.H.S., Bukhari, (2011a) Hyper-GA for Solving Benchmark Scheduling Problems. Australian Journal of Basic and Applied Sciences (**5**): 1657-1667.

Ahmed, A., I. A. Mehmood, and A.H.S.Bukhari, 2011b. Particle Swarm Optimizatin Based Hyper-Heuristic For Tackling Real World Examinations Scheduling Problem. Australian Journal of Basic and Applied Sciences (**5**): 1406-1413.

Ahmed, A. and Z. Li, (2010) Solving Course Timetabling Problem Using Interrelated Approach. 2010 IEEE International Conference on Granular Computing IEEE, San Jose, California, USA, 651-655.

Aftab A., Z. and Li, A.H.S..Bukhari, (2011c) Triphasic solving approach for scheduling problem. Seventh International Conference on Natural Computation (ICNC), 2011 IEEE, Shanghai, China 2358 - 2363

Alex B., F. De Cesco, L. Di Gaspero, and A. Schaerf, (2010) Benchmarking curriculum-based course timetabling: formulations, data formats, instances, validation, visualization, and results Annals of Operations Research 179Pp.

Angeline, P.J., 1998. Evolutionary Optimization Versus Particle Swarm Optimization: Philosophy and Performance Differences., Proceedings of the Seventh Annual Conference on Evolutionary Programming Springer, San Diego, 601-610.

Burke, E.K., Y. Bykov, J. Newall, and S. Petrovic, (2004) A Time-Predefined Local Search Approach to Exam Timetabling Problems IIE Transactions 509–528.

Burke,EK., E. Hart, G. Kendall, G. Newall, P. Ross, and S. chulenburg., (2003) Hyper-heuristics: An emerging direction in modern search technology.

Ghaemi, S., Y. Vakili, and A. cghagolzadeh, (2007) Using a genetic algorithm optimizer tool to solve Univ. timetable scheduling problem. ISSPA 07. IEEE.

Kennedy, J., and R.C., Eberhart, (1995) Particle swarm optimization. Proceedings of the IEEE International Conference on Neural Networks, 1942-1948.

Kennedy, J., and W.M. Spears, (1998) Matching Algorithms to Problems: An Experimental Test of the Particle Swarm and Some Genetic Algorithms on the Multimodal Problem Generator., Proc. 1998 IEEE World Congress on Computational Intelligence. IEEE, Alaska, 74-77.

Cowling, P., G. Kendall, and L. Han (2002) An investigation of a hyperheuristic genetic algorithm applied to a trainer scheduling problem. Proceedings of the Congress on Evolutionary Computation (2002) 1185-1190,.

Cowling, P., G. Kendall, and E. Soubeiga, (2000) A hyperheuristic approach for scheduling a sales summit. In Selected Papers of the Third International Conference on the Practice and Theory of Automated Timetabling, PATAT 2000. Springer, Konstanz, Germany, 176–190.

Parsopoulos, K.E., and M.N.Vrahatis, (2002) Initializing the Particle Swarm Optimizer Using the Nonlinear Simplex Method. Advances in Intelligent Systems, Fuzzy Systems, Evolutionary Computation, 216-221.

Shi, Y., and R.C. Eberhart, (1998) A Modified Particle Swarm Optimizer. Proceedings of the IEEE International Conference on Evolutionary Computation. NJ: IEEE Press, Piscataway, 69-73.

Shi, Y., and R.C. Eberhart, (1999) Empirical Study of Particle Swarm Optimization. Proceedings of the 1999 Congress on Evolutionary Computation. NJ: IEEE Service Center, Piscataway, 1945-1950.

Terashima-Marin, H. Moran-Saavedra, A. and P. Ross, (2005) Forming hyper-heuristics with GAs when solving 2D-regular cutting stock problems. Proceedings of the congress on evolutionary computation 1104–1110.