Experimental Result of Particle Collision Algorithm for Solving Course Timetabling Problems

Anmar Abuhamdah[1],and Masri Ayob[2]

Data Mining & Optimization Group (DMO), Centre of Artificial intelligence Technology (CAIT)
[12]Faculty of Computer science, Universiti Kebangsaan Malaysia, Selangor, 43600 Malaysia

Summary
This work presents a Particle Collision Algorithm (PCA) to solve university course timetabling problems. The aim is to produce an effective algorithm for assigning a set of courses, lecturers and students to a specific number of rooms and timeslots, subject to a set of constraints. PCA approach that was originally introduced by Sacco for policy optimization. PCA always accepts improved solution but adaptively accepts worse solution based on the quality of the solution. PCA differs from Simulated Annealing and other meta-heuristic approaches where, before accepting the trial solution (although we obtain good-quality solution), PCA attempts to further enhance the trial solution by exploring different neighbourhood structures. Therefore, PCA could be able of escaping from local optima. We evaluate the effectiveness of PCA. This testing it on standard test benchmark course timetabling datasets which were introduced by Socha. Results show that PCA significantly outperformed Simulated annealing (SA) and Great Deluge approach in some instances. Results also show that PCA is able to produce good quality solutions, which are comparable to other work in the literature.

Key words:
Course Timetabling Problem; Meta-Heuristics; Particle Collision Algorithm; Simulated Annealing; Great Deluge.

1. Introduction

The university course timetabling problem involves assigning a set of courses, lecturers and students to a specific number of rooms and timeslots [1]. The goal is to produce high-quality timetable that satisfies all hard constraints and attempts to accommodate soft constraints as much as possible. A feasible timetable must satisfy all hard constraints, and soft constraints can be violated if necessary. However, each violation of the soft constraints will increase the penalty cost. A good quality timetable should have a small penalty value. A course timetabling problem is an NP-complete problem [2-4], to which it is difficult to find an optimal solution in a reasonable time. Finding good-quality solutions to these problems depends on the technique itself and the neighbourhood structure employed during the search. Various approaches and techniques have been intensively applied to the course timetabling problems over the last few years. For further information on previous works, please refer to the following survey/overview papers: [5-9].

Manuscript received September 5, 2009
Manuscript revised September 20, 2009

In this work, we investigate the effectiveness of applying a Particle Collision Algorithm to solving university course timetabling problems. Particle Collision Algorithm (PCA) was originally introduced by Sacco et al. [10]. The structure of PCA resembles the simulated annealing structure. The algorithm starts with solution generated by any constructive heuristic method; the solution will be iteratively improved [10]. A basic difference is that, it does not rely on user-defined parameters. PCA always accept improving solution. However, worse solution will probability be accepted based on the quality of solution. This may avoid being trap in local optima [10].

The aim of this work is to investigate the performance of applying PCA for solving course timetabling problems. In order to evaluate the effectiveness of the enhanced algorithm, we tested it on eleven standard benchmark datasets that were introduced by Socha et al. [11] and made comparison with the simulated annealing and great deluge algorithm that were applied on examination timetabling problem by Burke et al. [12] with the same parameters as those employed in Burke et al. [12], then we compare the PCA with other approaches in the literature.

2. Problem Description

In this work, we use eleven datasets introduced by Socha et al [11] in the first international timetabling competitions. These datasets tackle the students' satisfaction only for the course timetabling problem. The problem consists of:

- A set of events to be scheduled in 45 timeslots (5 days of 9 hours each).
- A set of rooms in which events can take place.
- A set of students who attend the events.
- A set of features characterizing the rooms and required by events.

These eleven datasets are categorized into three groups, small (S1,S2,S3,S4,S5) , medium (M1,M2,M3,M4,M5) and large (L) (see Table 1).

Table 1. Eleven Datasets (Socha ET al -2002)

Datasets	# Events	#Rooms	# Features	# Students
S1	100	5	5	80
S2	100	5	5	80
S3	100	5	5	80
S4	100	5	5	80
S5	100	5	5	80
M1	400	10	5	200
M2	400	10	5	200
M3	400	10	5	200
M4	400	10	5	200
M5	400	10	5	200
L	400	10	10	400

Table 1, shows the attributes of the eleven datasets, for each individual dataset, either for small, medium or large datasets. Each dataset has a different number of conflicts even if they have the same number of (events, rooms, features, students). These datasets were collected from various real-world university course timetabling problems (see details description in [11]). In order to produce a feasible timetable, all hard constrains must be satisfied. Soft constraints should be satisfied as much as possible. These datasets that were introduced in international timetabling competitions include (Hc1, Hc2 and Hc3) hard constraints and (Sc1, Sc2 and Sc3) soft constrains, as follows:

a) Hard constrains.
   Hc1: No student attends more than one event at the same time.
   Hc2: The room is big enough for all the attending students and has all the features required by the event.
   Hc3: Only one event takes place in each room at any timeslot.

b) Soft constrains
   Sc1: A student should not have a class in the last slot of the day.
   Sc2: A student should not have more than two classes consecutively.
   Sc3: A student should not have a single class on a day.

The quality of timetable is measured by penalizing equally each violation of the Soft constraints, where each violation will be penalized '1' for each student who involve in this situation.

3. Construction Algorithm

In this work, we use a constructive heuristic was proposed by Landa-Silva, and Obit [13]. We obtained the feasible solution by adding and removing courses from the timetable while using the tabu list until we had a feasible timetable (satisfy hard constraints), as follow:

- Step 1 (Highest Degree Heuristic Step): After we assign all the events that do not have conflict, we assign randomly the unassigned event with the highest number of conflicts (not satisfying the hard constraints) to timeslots. Then, assigning all events to timeslots, we assign each event to a room by using the maximum matching algorithm for bipartite graph (see [28]). Even after ending this step, there is no guarantee that the timetable satisfies the hard constrains (no guarantee for feasible timetable). Otherwise if not feasible, we move to step 2.
- Step 2 (Local Search Step): Two neighbourhood structure moves (move 1, move 2) were applied to construct a feasible timetable that we could not obtain to the Highest Heuristic Degree. The move will be accepted if it is more satisfying of hard constraints, to find the feasibility. After 10 iterations, if the quality of solution does not improve, then this step is terminated. Otherwise if not feasible, we move to step 3.
- Step3 (Tabu Search Step): Tabu search had been applied using move 1 only. "The tabu list contains events that were assigned less than tl iterations before calculated as $tl = ran(10) + \delta \times nc$, where ran(10) is a random number between 0 and 10, nc is the number of events involved in hard constraint violations in the current timetable, and $\delta = 0.6$" [28]. After 500 iterations, if the quality of solution does not improve, this step will be terminated.

  We repeat the local search step and the Tabu search step until we find a feasible solution.

4. The Algorithms

In this work, we applied three algorithms Simulated Annealing (SA), Great Deluge (GD) and Particle Collision Algorithm (PCA) to evaluate the performance of PCA. We applied Simulated Annealing algorithm and Great Deluge algorithm with the same parameters as those employed to both algorithm in Burke et al. [12]. Particle Collision Algorithm (PCA) was introduced by Sacco et al. [10] in 2005. This is a stochastic optimization algorithm based on simulated annealing approach. PCA structure resembles a simulated annealing structure, but the basic difference that is does not have cooling schedule and it does not rely on user-defined parameters. Since the acceptance criteria of PCA can probably accept worst solution, it is capable of avoiding being trapped in local optima [10].

After generate feasible time table by any construction heuristic then select a random R neighbourhood structure N from different neighbourhood and apply to the current solution to generate new feasible solution. if the new solution not feasible then we apply the same

neighbourhood that had been selected NR to generate new feasible solution and so on.

4.1 PCA Pseudo code

In this work, we apply Hill Climbing Search [14] in PCA approach in the *Exploration Phase''*, in PCA there is an exploration of the boundaries searching for an even better solution (i.e. the function *''Exploration''* performs local search generating a small stochastic perturbation of the solution inside a loop) [10]. The function *''Scattering''* in PCA consider as the acceptance criteria, where the algorithm is more likely to accept little bit worse solution to solve university course timetabling problem. The scattering is equal to the [penalty of the new solution divided by the penalty of best solution] minus one. Then we check whether the scattering value is less than the random number between zero and one. If yes, then we accept that worst solution and enhance it by exploration that will apply Hill Climbing search, but if greater than the random number between zero and one, we will not accept that solution. Fig.1 shows the PCA Pseudo code.

```
Generate an initial solution So (current      solution)
Calculate the initial cost function value, Fitness (So);
Set BestSol = So
Set # of n.iteration
Set # of m.iteration
Set index=0;
for n = 0 to n.iterations
Generate new solution (S*) by Applying Random
neighbourhood from different neighbourhood structure to
                                    the current solution So

 if    Fitness(S*) < Fitness (So)
   So = Exploration (S*);
   BestSol = So;
 else
   S1=Scattering (S*, BestSol);
   So =S1;
      if    Fitness (S1) < Fitness (BestSol)
         BestSol = So;
      end
 end
end
-------------------------------------------------------------------------
Scattering (S*, BestSol)
                          Fitness(S*)
Scattering =                                -1;
                       Fitness (BestSol)

 if    Scattering > random (0, 1)
    So = BestSol;
else
    So = Exploration (S*);
end
Return So
-------------------------------------------------------------------------
Exploration (S*)
 for m = 0 to m. Iterations
 Generate new solution (S1*) by applying neighbourhood
 structure to the solution (S*)
  if    Fitness (S1*) < Fitness(S*)
     S*= S1*;
  end
 end
  Return S*
```

Fig. 1. PCA Pseudo code

4.2 SA Pseudo code

In this work, we applied the simulated annealing algorithm with the same parameters as those employed in Burke et al. [12], where the initial temperature IT is equal to 5000, the final temperature FT is equal to 0.05 and the number of iterations, IN is set to be 10,000,000 but in our work we set the number of iterations, IN is set to be 200,000.
At the beginning of the search, the initial temperature (Temp) is set to be T.I At every iteration, Temp is decreased by α where α is defined as:

$$\alpha = (\log (T.I) - \log (T.F)) / N.I)$$

In the do-while loop, we generate a new solution by random neighbourhood selection of a course and assigning it to a randomly selected valid timeslot and room. A worse solution could be accepted if the random number is less then $e^{-\delta/Temp}$, where δ = Fitness (S*) - Fitness (So). Then the current solution is update into a best solution. This process continues until the temperature (Temp) is less than the final temperature FT. Fig.2 shows the SA Pseudo code.

```
Generate an initial solution So (current solution)
Calculate the initial cost function value, Fitness (So);
Set BestSol = So;
Set # of iterations, IN;
Set initial temperature IT;
Set final temperature FT ;
Set decreasing temperature rate as α where
                    α = (log (IT) - log (FT))/IN;
Set Temp = IT;
do while (Temp > FT)
Generate new solution (S*) by Applying Random
neighbourhood from different neighbourhood structure to the
current solution So
   Calculate Fitness (S*);
if (Fitness (So*) < Fitness (BestSol))
   So = S*;
   BestSol = S*;
else
   δ = Fitness (S*) - Fitness (So)
   Generate a random number called RN;
   if (RN ≤ e-δ/Temp )
   So = S*;
   Temp = Temp-Temp*α;
end while;
```

Fig. 2. SA Pseudo code

4.3 GD Pseudo code

In this work, we define a number of iterations as N.I and an estimated quality of the final solution as e.q, a decreasing rate, β (which is calculated using the Formula , adopted from Burke et al.[12]:

$$\beta = (\text{Fitness (Sol)} - \text{Fitness (estimatedquality)}) / (\text{NumOfIte})$$

at the beginning of the search we set the level is equal to penalty cost of the initial solution, Fitness (So) and a decreasing Value by β. In the do-while loop, some of neighbours are defined then randomly selecting a course and assigning it to a valid timeslot and room. A worse solution is accepted if the cost function value of the new solution, Fitness (S*) is lower than the level. The current solution is updated and the process continues until the number of iterations increases to the maximum number of iterations, N.I, or until there is no improvement for a certain number of iterations, referred to as not_ improving_ length_ GDA (in the pseudo-code). Fig.3 shows the GD Pseudo code.

```
Generate an initial solution So (current solution) Calculate the
initial cost function value, Fitness(So);
Set BestSol = So;
Set estimated quality of final solution, e.q;
Set # of iterations, N.I;
Set initial level: level= Fitness (Sol);
Set decreasing rate
    β = ((Fitness (So)-estimatedquality)/(N.I);
Set iteration = 0;
Set not_improving_counter = 0;
do while (iteration < N.I)
Generate new solution (S*) by Applying Random
neighbourhood from different neighbourhood structure
                                    to the current solution So
Calculate Fitness (S*);
 if (Fitness (S*) < Fitness (BestSol))
   So = S*;
   BestSol = S*;
   not_improving_counter = 0;
else   if (Fitness (S*)≤ level)
            So = S*;
          not_improving_counter = 0;
       else
            Increase not_improving_counter by 1;
           if (not_improving_counter == not_improving_
                                              length_GDA)
          exit;
level = level - β; Increase iteration by 1;
end do;
```

Fig. 3. GD Pseudo code

5. Neighbourhood Structures

We implemented to the PCA, GD, SA algorithms the same four different neighbourhood structure (N1-N4) to evaluate the performance and effectiveness of each algorithm with fair comparison. These different neighbourhood structures for solving the university course timetabling problem are:

N1: Randomly select two courses and swap their timeslots (and rooms if feasible).

After selecting the two random courses as shown in Fig. 4, we swap the timeslots as shown in Fig. 5. If the room's features and size between the random selected courses are suitable, then we swap the rooms as shown in Fig. 6 if feasible.

Room / T.Slots	R1	R2	R3	R4	R5
T1	E22	E13	E66	E40	E78
T2	E6	E36	E44	E23	E69
T3	E1	E98	E14	E11	
.	.	.	.	.	.
.		.	.	.	.
.	.	.	.	.	.
T43		E97	E25		E88
T44	E10	E100	E91	E9	E66
T45	E82	E48	E5		E59

Fig. 4. N1.1

Room / T.Slots	R1	R2	R3	R4	R5
T1	E22	E13	E66	E40	E78
T2	E6	E36	E44	E23	E69
T3	E1	E98	E14		E88
.	.	.	.	.	.
.		.	.	.	.
.	.	.	.	.	.
T43		E97	E25	E11	
T44	E10	E100	E91	E9	E66
T45	E82	E48	E5		E59

Fig. 5. N1.2

Room / T.Slots	R1	R2	R3	R4	R5
T1	E22	E13	E66	E40	E78
T2	E6	E36	E44	E23	E69
T3	E1	E98	E14	E88	
.	.	.	.	.	.
.		.	.	.	.
.	.	.	.	.	.
T43		E97	E25		E11
T44	E10	E100	E91	E9	E66
T45	E82	E48	E5		E59

Fig. 6. N1.3

N2: Randomly select two timeslots and simply swap all the courses in one timeslot with all the courses in the other timeslot. In Fig. 7, for example, we select the courses in one timeslot (T3) and swap with the courses in the other timeslot (T44).

Room / T.Slots	R1	R2	R3	R4	R5
T1	E22	E13	E66	E40	E78
T2	E6	E36	E44	E23	E69
T3	E1	E98	E14	E11	E23
.					
.					
.					
T43		E97	E25		E88
T44	E10	E100	E91	E9	E66
T45	E82	E48	E5		E59

Fig. 7. N2

N3: Randomly select four courses and swap their timeslots (and rooms if necessary). In Fig. 8, for example, we select four random courses; we swap the 1st course (E98) with the 4th course (E25) and swap the 2nd course (E11) with the 3rd course (E88) if feasible.

Room / T.Slots	R1	R2	R3	R4	R5
T1	E22	E13	E66	E40	E78
T2	E6	E36	E44	E23	E69
T3	E1	E98	E14	E11	E23
.	.	.	.	.	.
.		.	.	.	.
.	.	.		.	
T43		E97	E25		E88
T44	E10	E100	E91	E9	E66
T45	E82	E48	E5		E59

Fig. 8. N3

N4: Randomly select a course, feasible timeslot and feasible room, and move the course to the new timeslot (and move the course to the new room if necessary). For example, we select a course (E14) and assign to timeslot (T45) as shown in Fig. 9, and assign to the room (R4) if feasible as shown in Fig. 10.

Room / T.Slots	R1	R2	R3	R4	R5
T1	E22	E13	E66	E40	E78
T2	E6	E36	E44	E23	E69
T3	E1	E98	E14	E11	E23
.	.	.	.	.	.
.		.	.	.	.
.	.	.		.	
T43		E97			E88
T44	E10	E100	E91	E9	E66
T45	E82	E48			E59

Fig. 9. N4.1

Room / T.Slots	R1	R2	R3	R4	R5
T1	E22	E13	E66	E40	E78
T2	E6	E36	E44	E23	E69
T3	E1	E98	E14	E11	E23
.	.	.	.	.	.
.		.	.	.	.
.	.	.	.	.	
T43		E97	E25		E88
T44	E10	E100	E91	E9	E66
T45	E82	E48	E5		E59

Fig. 10. N4.2

6. Experimental Result

In this work, we tested our algorithm on a PC with an Intel dual core 1800 MHz, 1GB RAM. These results were obtained out of 11 runs and 200,000 iterations. We use Socha benchmark test datasets (see Socha [11] for more details of the problem description) to test the performance of our approaches.

Table 2 shows the comparison between PCA, SA and GD searches that we applied on socha benchmark datasets under 200,000 iterations with the same neighbourhood structures.

Table 2. Comparison between PCA and SA and GD

Data Set	PCA		SA		GD	
	Min	Avg	Min	Avg	Min	Avg
S1	1	2.09	1	2.72	**0**	2.45
S2	1	1.54	1	2.72	**0**	1.72
S3	1	2.36	1	2.54	1	1.69
S4	1	1.63	1	2.09	**0**	2.45
S5	**0**	1.54	**0**	2.63	**0**	1.72
M1	**136**	156.09	143	174	151	169.36
M2	**138**	152.18	148	168.09	148	160.90
M3	**165**	177.72	191	211.63	174	187.45
M4	143	160.09	152	168	**137**	150.09
M5	135	165.09	158	181.45	**121**	145.18
L	789	834	772	838.90	**734**	808

Table. 2, shows that the results obtained indicates that PCA outperformed SA and GD in some instances such as medium1, medium2 and medium 3 datasets and the other datasets were within the range of the SA and GD results and obtained good-quality solutions for all small, medium and large datasets. Also Table. 2, shows that GD outperformed PCA and SA in some instances such as small1, small2, small3, small4, medium4, medium5 and large and the other datasets were within the range of the PCA and SA. Also Table. 2, shows that SA obtained result better than PCA in the large dataset and better than GD in medium 1 dataset. Fig. 11, Fig. 12., Fig. 13, shows the box and whisker plot that summarise the results of 11 runs on Socha benchmark datasets for each of PCA, SA and GD algorithms respectively.

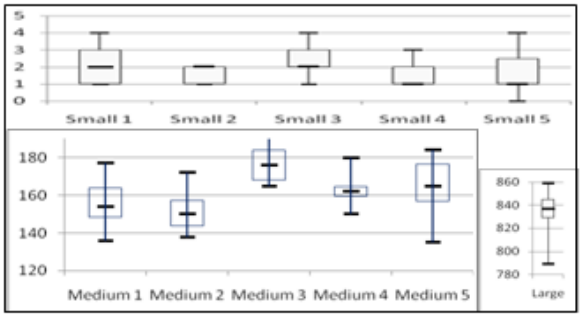

Fig. 11. Box and whisker plot of PCA for all datasets

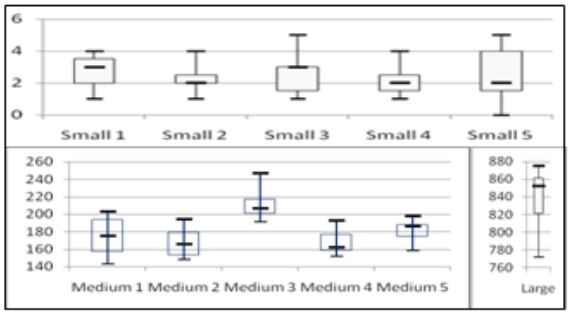

Fig. 12. Box and whisker plot of SA for all datasets

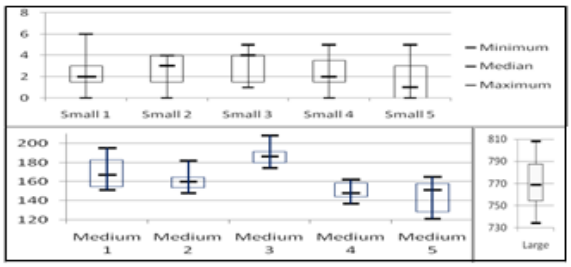

Fig. 13. Box and whisker plot of GD for all datasets

From Fig. 11, Fig. 12, Fig. 13 above shows that PCA In small1, small3, small4, small5, medium1, medium2, medium3 and medium4 dataset the median more close to the best than worst of these runs.

In addition, Fig .11 shows that the algorithm is stable and consistent and most of the time can produce very good quality solution. Also shows SA in small2, small4, small5, medium2, medium3 and medium4 dataset, we can see that the median more close to the best than worst of these runs. These indicate that the algorithm is stable and consistent but sometimes can produce good quality solution. Also shows that GD in small1, small4, small5, medium1, medium2, medium3, medium4 and large dataset, we can see that the median more close to the best than worst of these runs. These indicate that the algorithm most of the time can produce very good quality solution

Table 3, Table 4, Table 5 illustrates the statistical analysis of applying PCA, SA and GD algorithms respectively on the Socha benchmark datasets for more understanding about the robustness and consistency. The statistical readings are based on the following performance indicators: the best score (fmin), the average score (favg) , the standard deviation (σ std). If there are multiple hits on the best solution in each independent runs (11 runs for small, 11 runs for medium, and 11 runs for the large instances), the values listed in the table are the average over these multiple best.

Table 3. Statistical analysis of our algorithm (PCA) applied to Socha benchmark datasets

Data Set	*fmin*	*favg*	*Std. Dev.(σ)*
Small 1	1	4	1.22
Small 2	1	2	0.52
Small 3	1	4	0.92
Small 4	1	3	0.81
Small 5	0	4	1.29
Medium 1	136	177	12.64
Medium 2	138	172	10.68
Medium 3	165	199	11.31
Medium 4	143	180	11.53
Medium 5	135	184	15.08
large	789	859	19.04

Table 4. Statistical analysis of our algorithm (SA) applied to Socha benchmark datasets

Data Set	*fmin*	*favg*	*Std. Dev.(σ)*
Small 1	1	4	1.01
Small 2	1	4	1.01
Small 3	1	5	1.29
Small 4	1	4	0.94
Small 5	0	5	1.63
Medium 1	143	203	22.11
Medium 2	148	194	16.38
Medium 3	191	247	16.44
Medium 4	152	193	12.77
Medium 5	158	198	11.81
large	772	875	34.47

Table 5. Statistical analysis of our algorithm (GD) applied to Socha benchmark datasets

Data Set	*fmin*	*favg*	*Std. Dev.(σ)*
Small 1	0	6	1.63
Small 2	0	4	1.50
Small 3	1	5	1.58
Small 4	0	5	1.63
Small 5	0	5	1.79
Medium 1	151	195	16.06
Medium 2	148	182	10.87
Medium 3	1474	208	10.65
Medium 4	137	162	8.083
Medium 5	121	165	16.19
large	734	808	23.74

From Table 3, Table 4 and Table 5 above, shows the average score of PCA better than SA in all datasets and equal small1 dataset, also better than GD in all datasets except medium4, medium5 and large datasets. Also Table 3, Table 4 and Table 5, shows that the standard deviation of PCA better than SA in all datasets and except small1 dataset, also better than GD in all datasets except medium3, medium4 datasets.

From Fig. 11, Fig. 12, Fig. 13, Table 2, Table 3, Table 4 and Table 5, the result in Table 2 indicate that PCA obtained good quality solution in the all medium datasets. Meanwhile the result in Box and whisker plot and the Statistical analysis shows that PCA obtained good quality solution for all datasets comparable with SA and GD.

Table 6 shows the comparison between PCA with other local hybrid meta-heuristic searches. The best results are presented in bold. The observation of the results obtained indicates that PCA performed well and obtained a good-quality solution for all the small, medium and large datasets. The results of all small datasets scored zeros, the same as some of those published results; whilst for medium and large datasets our result are comparable to other published results.

Table 6. Comparison between PCA and other local hybrid meta-heuristic search results on course timetabling problem

Data Set	PCA		R1	R2	R3	R4	R5
	Min	Avg					
S1	1	2.09	**0**	**0**	**0**	**0**	5
S2	1	1.54	**0**	**0**	**0**	**0**	5
S3	1	2.36	**0**	**0**	**0**	**0**	3
S4	1	1.63	**0**	**0**	**0**	**0**	3
S5	**0**	1.54	**0**	**0**	**0**	**0**	**0**
M1	136	1569	338	175	**80**	221	176
M2	138	152.18	326	197	**105**	147	154
M3	165	177.72	384	216	**139**	246	191
M4	143	160.09	299	149	**88**	165	148
M5	135	165.09	307	190	**88**	130	166
L	789	834	-	912	730	**529**	798

Data Set	R6	R7	R8	R9	R10	R11
S1	10	2	**0**	6	3	1
S2	9	4	**0**	7	4	3
S3	7	2	**0**	3	6	1
S4	17	**0**	**0**	3	6	1
S5	7	4	**0**	4	**0**	**0**
M1	243	254	242	372	140	195
M2	225	258	161	419	130	184
M3	249	251	265	359	189	248
M4	285	321	181	348	112	164.5
M5	132	276	151	171	141	219.5
L	1138	1027	-	1068	876	851.5

Note:
R1. VNS-TS [15]. R2. ELM-GD[16]. R3. EGD [17].
R4.HEA [18]. R5. DHC-OABA[19]. R6. FMHA [20].
R7. GA-LS [21]. R8. RII-C N [22]. R9. GBHH [23].
R10. NLGD [13]. R11. AMMA-LS [24].

Abdullah et al. [15] reproduced the used of eight neighbourhoods in Randomized Iterative Improvement with Composite Neighborhood (RII-CN) [22] along with three additional neighbourhoods had been used in Variable Neighbourhood Structure with Tabu Search (VNS-TS) algorithm for clarity and completeness.

The Result for Electromagnetic like Mechanism and Great Deluge (ELM-GD) from S1 to M5 datasets obtained within 10 hours and 200,000 iterations for each [16].

Extended Great Deluge (EGD) has produced the best results of all datasets except the large dataset and each best result has been obtained within a range between 15 to 60 seconds for the small datasets, and 200,000 iterations with 10 runs for each to obtain an average value [17].

Hybrid Evolutionary Algorithm (HEA) obtained feasible results for the whole datasets, and 8 best results out of 11 for among Evolutionary Algorithm along with beating only 1 dataset (large) among local search methods; their termination criterion is set to 200,000 iterations, and 10 hours to obtain each result [18].

Graph-Based Hyper-Heuristic (GBHH) produced only one good quality and competitor result for the 5th medium datasets. They considered the number of iterations as 5 multiply by number of events [23].

Asmuni et al. [20] provided a comparison of solution quality and rescheduling procedure (required for producing solutions) between the Fuzzy Multiple Heuristic Approach (FMHA) and single heuristics ordering in solving the problem [20]. The single heuristic ordering is a basic version of the timetabling problem and can be considered as a graph coloring problem, which means, does not handle the soft constraints. The used single heuristics ordering in their work are: Largest Degree first, Largest Enrolment first, Least Saturation Degree first, Largest Colored Degree first and for large dataset).

Die Hard Co-Operative Ant Behaviour Approach (DHC-OABA) [19] obtained a feasible competitor results to others in some datasets.

The hybrid meta-heuristics Non-Linear Great Deluge (NLGD)performed well and obtained good quality results for all medium datasets, and large dataset as well, beside one optimal solution for the S5 dataset with a fixed computational time in seconds (3600 for small datasets, 4700 for medium, 6700 for the large dataset), and run the algorithm 10 times for each dataset[13].

Genetic Algorithm and Local Search (GA-LS) [21] perform separate Local Search routine with genetic algorithm but the result were not comparable, Ant MAX-MIN Algorithm (AMMA)with a separate Local Search routine (10 runs and 200,000 iterations, and a limited given time: 90sec for small datasets, 900sec for medium datasets, and 9000sec takes approximately 10 hours for each datasets)[24].

The result also shows that PCA capable of producing feasible solution for all datasets with high quality solutions that are comparable with the best-known results obtained in the literature. Results of the PCA are seems to be quite good for all the datasets (best results obtained out of 11 runs, 200,000 iterations).

7. Conclusions

The overall goal of this work is to investigate the effectiveness of applying PCA as a meta-heuristic search for solving the university course timetabling problem. In

order to evaluate the effectiveness of our approach, we test it on Socha benchmark test datasets. PCA start by generate solution by any constructive heuristic then apply random neighbourhood selection to generate new solution and a function *''Exploration''* performs local search to enhance the solution. In the *Exploration''* phase we apply Hill Climbing search. In addition the function *''Scattering''* in PCA consider as the acceptance criteria, where the algorithm is more likely to accept little bit worse solution based on scattering formula. Results indicate that using the Particle Collision Algorithm approach and defining the suitable Neighbourhood Structure (N1-N4) is comparable with the other approaches in the literature in small, medium and large datasets and is particularly suitable for solving course timetabling problems. The PCA approach is capable of producing high quality solutions in reasonable time. Our future work is to improve the PCA with other optimisation methods and to improve PCA to be capable with all kind of datasets.

7. References

[1] S. Petrovic, and E.K. Burke, "University timetabling" , Ch. 45 in the Handbook of Scheduling: Algorithms, Models, and Performance Analysis (eds. J. Leung), Chapman Hall/CRC Press, 2004.

[2] A. Schaerf, "A Survey of Automated Timetabling", Technical Report CS-R9567, CWI, Amsterdam,NL, 1995.

[3] S. Even, A. Itai and A. Shamir, "On the Complexity of Timetable and Multi commodity Flow Problem", SIAM J. Comput., 5(4):691-703, 1976.

[4] R. Lewis, B. Paechter and B. McCollum," Post Enrolment based Course Timetabling": A Description of the Problem Model used for Track Two of the Second International Timetabling Competition, Cardiff Working Papers in Accounting and Finance A2007-3. Prifysgol Caerdydd/ Cardiff University, Wales. ISSN: 1750-6658, v 1.0, 2007.

[5] D. De Werra, "An Introduction to Timetabling". European Journal of Operations Research, 19, pp 151-162, 1985.

[6] V. Bardadym, Computer-Aided School and University Timetabling: The New Wave. In Edmund Burke and Peter Ross, editors, "The Practice and Theory of Automated Timetabling" : Selected papers from the 1st International Conference, Springer Lecture Notes in Computer Science Vol 1153, pp 22-45, (1996)

[7] A. Schaerf, "A Survey of Automated Timetabling". Artificial Intelligence Review 13(2), pp 87-127, 1999.

[8] E. K. Burke, J. Kingston and D. de Werra , " Applications to Timetabling". Handbook of Graph Theory, (editors, J. Gross and J. Yellen), Chapman Hall/CRC Press, pp 445-474, 2004.

[9] D. Costa, "A tabu search for computing an operational timetable". European Journal of Operational Research, 76, pp 98-110, 1994.

[10] W. F. Sacco and B. de Oliveira, "A new stochastic optimization algorithm based on particle collisions". In: Transactions of the American Nuclear Society, vol. 92, 2005,ANSAnnualMeeting, San Diego, CA, June, 2005.

[11] K. Socha, J. Knowles and M. Samples "A max-min ant system for the university course timetabling problem". Proceedings of the 3rd International Workshop on Ant Algorithms, ANTS 2002, Springer Lecture Notes in Computer Science Vol 2463 (10), pp 1-13, 2002.

[12] E. K. Burke, Y. Bykov, J. Newall and S. Petrovic, " A time-predefined approach to course timetabling". Yugoslav Journal of Operations Research (YUJOR), 13(2), pp 139-151, 2003.

[13] D. Landa-Silva and J.H. Obit, "Great Deluge with Non-linear Decay Rate for Course timetabling Problems", 2008 4th International IEEE Conference Intelligent Systems, 978-1-4244-1739-1/08/, 2008.

[14] H.H. Hoos and T. Stutzle, "Stochastic Local Search: Foundations and Applications", Elsevier/Morgan Kaufmann, San Francisco. Mathematical Methods Of Operation Research, ISBN 1-55860-872-9658, Vol. 63, No. 1/ February 2006, pp. 193-194. 2006.

[15] S. Abdullah, E.K. Burke and B. McCollum, "An Investigation of Variable Neighbourhood Search for Course Timetabling". In: The Proceedings of the 2nd nnnMultidisciplinary International Conference on Scheduling: Theory and Applications (MISTA 2005), New York, USA, July 18th-21st, pages 413-427, 2005.

[16] S. Abdullah and H. Turabieh, "Electromagnetic Like Mechanism and Great Deluge for Course Timetabling Problems", In the First 2008 Seminar on Data Mining and Optimization DMO, vol. I, ISBN 9778-967-5048-36-4, 21-25, 2008.

[17] P. McMullan "An Extended Implementation of the Great Deluge Algorithm for Course Timetabling", ICCS International Conference of Computational Science, Part I, LNCS Lecture Note in Computer Science, vol. 4487, pp. 538-545, Springer-Verlag Berlin Heidelberg, Germany, 2008.

[18] S. Abdullah, E.K. Burke and B. Mccollum, "A hybrid evolutionary approach to the university course timetabling problem", In Proceedings of the IEEE Congress on Evolutionary Computation. Singapore, September, 2007.

[19] N. Ejaz and M. Javed, "Approach for Course Scheduling Inspired by Die-Hard Co-Operative Ant Behavior", Proceedings of the IEEE International Conference on Automation and Logistics August 18 - 21, 2007, Jinan, China , 2007.

[20] H. Asmuni, E. K. Burke and J. M. Garibaldi, "Fuzzy Multiple Heuristic Ordering for Course Timetabling", In: The Proceedings of the 5th United Kingdom Workshop on Computational Intelligence (UKCI05), London, UK, September 5th-7th, 302-309, 2005.

[21] S. Abdullah and H. Turabieh, "Generating University Course Timetabling Using Genetic Algorithm and Local Search", In the Third 2008 International Conference on Convergence and Hybrid Information Technology ICCIT, vol. I, 254-260, 2008.

[22] S. Abdullah, E.K. Burke and B. McCollum "An investigation of a variable neighbourhood search approach for course timetabling", The Proceedings of the 2nd Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA 2005), New York, USA, pp. 413-427, 2006.

[22] S. Abdullah, E.K. Burke and B. McCollum, "Using a Randomised Iterative Improvement Algorithm with Composite Neighbourhood Structures for the University Course Timetabling Problem". In Metaheuristics - Progress in Complex Systems Optimization (Doerner K. F., Gendreau M., Greistorfer P., Gutjahr W. J., Hartl R. F., and Reimann M., Eds.), Computer Science Interfaces Book Series, Springer Operations Research, ISBN-13:978-0-387-71919-1, Vol 39, pp 153-169, 2007.

[23] E. K. Burke, B. McCollum, A. Meisels, S. Petrovic and R. Qu, "A Graph-Based Hyper-Heuristic for Educational Timetabling Problems", European Journal of Operational Research, 2006.

[24] K. Socha, M. Samples and M. Manfrin, "Ant algorithm for the university course timetabling problem with regard to the state-of-the-art", The Proceedings of the 3 European Workshop on Evolutionary Computation in Combinatorial Optimisation, Essex, UK, Lecture Notes in Computer science 2611, Springer-Verlag, pp. 334-345, 2003.

AnmarAbuhamdah received the B.Sc from Princess Sumaya University for Technology in 2003. After working as a programmer and consultant (from 2003) in Quest Scope Company. He received the Master of Science (Intelligent System) from Universiti Utara Malaysia in 2006. After working as lecturer in Almjd Quality. After doing Phd in Universiti Kebangsaan Malaysia from 2007) with Data Mining & Optimization Group (DMO), Centre of Artificial intelligence Technology (CAIT). His research interest includes Timetabling scheduling, real time system, software development. He is a member of IEEE (2001).

Masri Ayob received the Bachelor of Engineering degree from Universiti Kebangsaan Malaysia in 1990. After working as lecturer in Butterworth Institute of Technology (BIT) From (1990), researcher in MIMOS BHD (from 1992). She received her Master of Electrical (MEng) from Universiti Teknologi Malaysia in 1996. After working as lecturer in UKM (Faculty Of Information Science & Technology) (from 1997), Post Doctoral Researcher in University of Nottingham (From 2005).She received her Phd (Computer Science) from The University of Nottingham in 2005. After she working as Senior lecturer in UKM (Faculty Of Information Science &Technology) (from 2006), Deputy Director(Application System and Information Management)in UKM (Center of Information Technology) (from 2006). Her research interest includes Automated Scheduling - Machine Optimisation, Timetabling, Job Shop Scheduling, Nurse Scheduling, etc, Microprocessor/Microcontroller Board Design, SCADA System , Assembly Language Programming, Programmable Logic Controller, Process Automation. She is is a member of CISched 2007, ISDA 2007 AIPR 07, ISDA 2006, ICTAI 06.