Population based Local Search for University Course Timetabling Problems

Anmar Abuhamdah[1], Masri Ayob[1], Graham Kendall[2] and Nasser R. Sabar[1]

[1]Data Mining and Optimisation Research Group (DMO), Center for Artificial Intelligence Technology, Universiti Kebangsaan Malaysia, 43600 UKM, Bangi Selangor, Malaysia
anmar@ftsm.ukm.my, masri@ftsm.ukm.my, naserdolayme@yahoo.com
[2]ASAP Research Group, School of Computer Science, The University of Nottingham, Nottingham NG8 1BB, UK.
gxk@cs.nott.ac.uk

Abstract. This work proposes a population based heuristic that can be embedded within any local search algorithm to solve university course timetabling problems. Population based Local Search (PB-LS) employs two main operators. The first is applied to a single solution to determine the force between that solution and the current solutions, whilst the second is applied to all solutions to determine the force in all directions. The progress of the search through the search space is governed by the forces either in a single direction or in all directions. The aim of our work is to produce an effective algorithm, which can increase the diversity of local search algorithms and overcome the limitations of other population based algorithms. In order to evaluate the effectiveness of PB-LS, we perform a comparison between the performance of PB-LS with other approaches in the scientific literature. We use university course timetabling benchmark datasets as a test domain. Results show that, PB-LS is able to produce statistically significant higher quality solutions that outperforms many other approaches (in some instances with regards to Socha dataset).

Keywords: Course Timetabling Problem; Particle Collision Algorithm; Adaptive Randomized Descent Algorithm; Population Based Algorithm.

1 Introduction

Generally, university course timetabling problems involves assigning a set of courses (events), teachers and students to a fixed number of timeslots and rooms subject to a variety of constraints [1]. Constraints in a timetabling problem can be classified as *hard* and *soft* [1]. The goal, when solving timetabling problems, is to satisfy all hard constraints and attempt to accommodate the soft constraints as much as possible (in order to produce high-quality timetable). All hard constraints must be satisfied to obtain a feasible timetable, whilst soft constraints can be violated if necessary, but each violation of the soft constraints will be penalized. The smaller the penalty value, the better the quality of the timetable. University course timetabling problem has been classified as an NP-hard problem; therefore it is difficult (in general) to find an optimal solution (for larger size instances) in a reasonable time [2]. Finding good-quality

solutions to these problems depends on the methodology used and the problem representation employed during the search [2].

In recent years, various approaches had been applied to solve university course timetabling problems. These approaches include great deluge [3], simulated annealing [4] tabu search [5] and randomized descent method [6].

A new approach to solve course timetabling problems based on multi-neighbourhood particle collision algorithm (MPCA) [7] was recently proposed. However, MPCA is a descent heuristic. The disadvantage of these methodologies is that, they are incapable of escaping local optima [6]. Therefore, we extend our previous work [7] by proposing an adaptive randomized descent algorithm (ARDA) [8] that employs an adaptive criterion to escape from the local optima. We also investigated hybridizing MPCA and ARDA [9] to overcome the limitations of MPCA.

However, MPCA-ARDA still has no diversification strategy, which motivated us to use the concept of a population based algorithm due to their ability to explore wider areas of the search space than local search algorithms [10]. Generally, the limitation of many population based algorithms is in exploiting the search space [10], as most of them are good at exploration rather than exploitation. Therefore, in this work, we propose a population based heuristic (i.e. Population Based Local Search, PB-LS) to overcome the limitations of a population based algorithm by increasing the ability of the intensification process. This algorithm is also motivated by the idea of Gravitational Emulation Local Search (GELS), which was proposed and developed by Webster and Bernhard [11].

The aim of our work is to investigate the performance of applying the PB-LS with MPCA-ARDA for solving university course timetabling problems. In order to evaluate the effectiveness of the MPCA-ARDA, we make a comparison between the performance of MPCA, ARDA, MPCA-ARDA as well as other approaches on Socha's university course timetabling datasets [12].

2 Problem Description

In this work, the eleven standard benchmark test datasets instances that were introduced by Socha et al. [12] are used, which seek to optimise the students satisfaction for the university course timetabling problem. The problem consists of:

- A set of Rooms *R* in which events can take place.
- A set of Events (courses) *E* to be scheduled in 45 timeslots (5 days of 9 hours each with one hour for each timeslot).
- A set of Features *F* characterizing the rooms.
- A set of Students *S* who attend the events.

These datasets are categorized into three groups, small (*S1, S2, S3, S4, S5*), medium (*M1, M2, M3, M4, M5*) and large (*L*) (see Table 1 and [12] for a detailed description). Table 1 also shows, the number of students, events, room and features as well as the conflict density (*CD*) for each dataset (representing the complexity), approximate number of students enrolled in each event (Students/ Events), and the approximate

number of available rooms for each event (Rooms/ Events) which are calculated as in [13].

Table 1. Eleven Datasets [12; 13]

Dataset	# Students	# Events	# Rooms	# Features	CD	Students/ Events	Rooms/ Events
S1	80	100	5	5	10.96	4.98	0.82
S2	80	100	5	5	13.92	5.36	0.79
S3	80	100	5	5	9.71	4.65	1.00
S4	80	100	5	5	7.16	3.45	1.39
S5	80	100	5	5	15.10	5.99	1.17
M1	200	400	10	5	37.38	8.85	2.23
M2	200	400	10	5	37.66	8.84	1.91
M3	200	400	10	5	40.44	8.85	1.91
M4	200	400	10	5	37.50	8.81	1.88
M5	200	400	10	5	28.27	8.66	1.37
L	400	400	10	10	45.57	8.92	0.76

These datasets have three hard constraints (Hc1, Hc2 and Hc3) and three soft constraints (Sc1, Sc2 and Sc3), as follows:

(a) Hard constraints.

Hc1: No student attends more than one event at the same time.

Hc2: The room has to be large enough for all the attending students and has all the features required by the event.

Hc3: Only one event takes place in each room in any timeslot.

(b) Soft constraints

Sc1: A student should not have a class in the last timeslot 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 based on the number of soft constraints violations. Each violation of the soft constraints will be penalized '1' for each student who is involved in this situation [12]. All hard constraints must be satisfied since we only deal with feasible solutions, which is usually the case for the majority of research in this domain.

3 Methodology overview

This works proposes a population based local search algorithm (PB-LS) for university course timetabling problems. PB-LS starts with an initial solution and iteratively explores its neighbour solutions, seeking for a better one. The neighbour solution is obtained by modifying the current solution using one or more neighbourhood structures.

3.1 Initial Solution

In this work, we use a constructive heuristic that was proposed in [16] to construct initial solutions for course timetabling problems. The constructive heuristic has three

phases: largest degree heuristic, neighborhood search and tabu search. The constructive heuristic starts with an empty timetable and successively invoke the three phases to generate a feasible timetable. In the first phase, all unscheduled courses are sorted based on the number of students they have in conflict with other courses. Then, the one that has the highest number of conflicts compared to the other courses is selected first. The selected course is then assigned to a feasible timeslot-room which are selected randomly. If there is no feasible room for this course, it will be assigned to any room. If all courses have been scheduled to timeslot-rooms, we ignore phases 2 and 3. Otherwise, phases 2 and 3 are invoked to achieve feasibility.

Phase 2, employs a simple decent algorithm to reduce the number of hard constrain violations. The neighbourhood solution is generated by either moving one course from its current timeslot-room into another timeslot-room, selected randomly, or it randomly selects two courses and swaps their timeslots and rooms. In both cases, the new solution is accepted if the move does not violate any hard constraints and the quality of the generated timetable in terms of hard constraints violation is better than the previous solution. Phase 2 is terminated after ten non improving iterations. If the solution is feasible, we ignore phase 3, otherwise, phase 3 is invoked.

Phase 3, employs a tabu search algorithm that explores neighbouring solutions similar to phase 2, but it also maintains a tabu list to prevent certain move been made for a certain number of iteration. The size of the tabu list is calculated by $tl = rand(10)+\delta * nc$, where $rand(10)$ is a random number between 0 and 10, nc is the number of events that violate the hard constraints and δ is a constant which is set to 0.6 [16]. This phase will stop after 1000 non improving iterations.

3.2 Neighbourhood Structures

The proposed algorithm starts with an initial solution and iteratively improves it by generating a neighbourhood solution using a set of neighbourhood structures. In this work, we use eight neighbourhood structures (NS1-NS8) which have been widely used in university course timetabling problem. These neighbourhood structures are classified into two groups: i) common neighbourhoods (NS1-NS5 as in [14]) and ii) shaking neighbourhoods (NS6 and NS8 as in [14], and NS7 as in [15]). Five of them NS1-NS5 [14] are the same neighbourhoods utilised in ARDA [8]. In addition, we use three other neighbourhood structures (NS6-NS8) to diversify the search (shake the timetable). The neighbourhood structures are:

NS1: Randomly select two courses and swap their rooms and timeslots if feasible. Otherwise swap their timeslots only, if feasible [14].

NS2: Randomly select two timeslots and swap all the courses in one timeslot with all the courses in the other timeslot [14].

NS3: Randomly select four courses and swap timeslots and rooms of the first and second courses with the third and fourth courses, if feasible.

NS4: Randomly select a course, timeslot and room, and then move the course (reassign) to the new timeslot and room if feasible [14].

NS5: Randomly choose a course from the top 15% of the list (ordered based on the penalty value in descending order) and randomly assign to other timeslot. Abdullah [14] used this neighbourhood (NS5) but with 10% selection. The reason behind using 15% is to widen the search space.

NS6: Choose 15% of the courses in the list and randomly assign them to other feasible timeslots. We use the same idea as NS5 but the difference is that, in NS5 we assign one course only whereas in NS6 we assign 15% of the courses to diverse the search (shake the timetable).

NS7: Randomly select two timeslots (t1 and t2) based on the largest enrolled (conflict) events. Select the most conflicting event in *t1* and *t2* and then apply a kempe chain move [15]. The main idea of the kempe chain neighbourhood is to move a chain of course within the timetable while maintain the feasibility by swapping conflicting courses until achieving feasibility.

NS8: Rotate two timeslots: Randomly select two timeslots (t_i and t_j), where $t_i > t_j$ and the timeslots are ordered t_0, t_1, …, t_{44}. Take all the courses in t_i and allocate them to t_j. Now take the courses that were in t_j and allocate them to t_{j-1}. Then allocate those that were in t_{j-1} to t_{j-2} and so on until those courses that were in t_{i+1} are allocated to t_i. This neighbourhood structure was introduced by Abdullah [14].

4 Population based Local Search Algorithm

This work is motivated by the concept of population based algorithms due to their ability to explore wider areas of the search space than using a local search algorithm [10]. However, some limitations of population based algorithms are [17]:

- Ineffective exploitation of the solution space (intensification process), in which there is no significant solution improvement.
- Solution combination methods to generate new solutions (e.g. crossover in genetic algorithm) and mutation operator usually rely on randomization and need a repair mechanism to use them effectively in constrained problems.
- The process of updating the population is usually performed randomly.
- Some algorithms do not have a memory as guidance for the search (e.g. genetic algorithm and memetic algorithm).

The idea of the PB-LS is to enhance the performance of population based algorithms by increasing the ability of the intensification process. This heuristic is motivated by the idea of gravitational emulation local search (GELS).

The GELS algorithm is based on the natural principles of gravitational attraction [11]. The reason for using gravity is to cause objects to be pulled towards each other. The more massive an object, the more gravitational “pull” it exerts on other objects. Also the closer two objects to each other, the stronger the gravitational forces are between them. This means that a given object will be more strongly attracted to larger and closer objects.

Previously the GELS algorithm was known as gravitational local search algorithm (GLSA) and consisted of two versions: one that was based on the gravitational attraction between two objects, and allowed navigation only to adjacent positions within the solution space; the other was based on gravitational field attractions among objects and allowed navigation to non-adjacent positions [11].

Both versions of GELS utilize the same gravitational force formula (see equation 1) but in slightly different ways [11]. The first version applies the formula to a single solution (vector) within the local search neighbourhood to determine the gravitational force between that solution and the current solutions. Whilst, the second version applies the formula to all solutions (vectors) within the neighbourhood and calculates the gravitational force between each of them and the current solution.

GELS simulates Newton's formula of gravitational force between two objects [11] (equation 1).

$$F= G (CU - CA) / R^2 \quad (1)$$

Where F is the Force value representing the value of the gravitational force between two objects (i.e. to enhance the difference between the quality of solutions), G = 6.672, CU = objective function value of the current solution, CA = objective function value of the candidate solution and R is the value of the parameter radius, representing the middle point in the distance between two objects.

GELS has an intensification mechanism (improvement of a solution using a local search). In GELS, there are two parameters to be tuned:

- Radius – sets the radius value in the gravitational force formula. It is used to determine how quickly the gravitational force can increase or decrease.
- Iterations – defines the number of iterations for the algorithm before it is terminated.

Therefore, in this work, we propose the PB-LS heuristic which overcomes some of the limitation in GELS. In PB-LS, we propose a new formula to calculate the force value as shown in formula 2 to overcome the issue of parameter *Radius* in GELS.

$$F= CU - CA \quad (2)$$

Where F is the Force value (in minimization problem), CU = objective function value of the current solution and CA = objective function value of the candidate solution.

Formula 2 differs from the formula 1, as GELS obtains real force value, whilst, formula 2 does not employ any static parameters. Indeed there is no relation between *G* (i.e. 6.672) and university course timetabling problem or the problems that GELS has been applied to.

PB-LS start from zero velocity (i.e. direction value is zero) but will be updated during the search process. In PB-LS, we only apply the first method. The procession of the search through the search space is governed by the forces in a single direction as determined by formula 2. In this work, we use MPCA-ARDA (as a local search) to

intensify the search since we already proposed MPCA-ARDA in [9]. Therefore, we can compare the result of MPCA-ARDA against the PB-LS with MPCA-ARDA in order to demonstrate the effectiveness of PB-LS approach.

Fig. 1 shows the pseudo code for PB-LS approach. Let S_o be a given initial solution, S_{best} be the best obtained solution, $f(S_{best})$ be the quality of S_{best}; $f(S_o)$ be the quality of S_o; *N.iters* be the number of maximum iterations; *reset.iter* be the number of non-improving iterations required to reset the directions and update solutions; *force* be the force value. In addition, we will also need to initialize the required parameters for the local search method (in our case, we use MPCA-ARDA).

PB-LS starts with an initialization phase, where we initialize all the parameters (see table 2), generate initial velocity vectors (by applying shaking neighbourhoods on S_o). In our work, for the first iteration vv_1, vv_2 and vv_3 are equal to the solutions generated by NS6, NS7 and NS8 accordingly. Initially, the direction for all velocity vectors is reset to 0 (see example in Fig. 2-a).

In the improvement phase (*Step2.1 in Fig. 1*), at each iteration, we rearrange the solutions in *vv* in descending order based on their direction values. Higher direction value indicates the better potential for improving the solution rather than other solutions in *vv*. If all directions values are different (see Step 2-1 in Fig.1), we choose the solution that has the largest direction value (e.g. vv_1 in Fig. 2-b) vv_k and set is as S_0 to be improved by local search (in our work, we use MPCA-ARDA) to produce S_0* until the stopping condition is met. In MPCA-ARDA, we use common neighbourhood structures (NS1-NS5). If the solution S_0* is accepted by the local search, the force value is calculated using equation (2) is added (positive or negative) to the selected direction and the selected solution in *vv* is replaced with the accepted solution. Otherwise, we will increase the unimproved counter of the selected solution ($UnImprove_k$) by one. The best solution found S_{best} will be updated with S_0* if the quality of S_0* is better than S_0. If the $UnImprove_k$ is equal to the predetermined successive unimproved iterations, *reset.iter* (i.e. 10 in this work), then we reset the direction of vv_k to '0' and replace vv_k with the best neighbour generated from S_{best} by randomly generating some neighbours (i.e. 5 in this work) from shaking neighbourhoods. Otherwise, we proceed with the next iteration. This mechanism attempts to escape from a local optima and to diversify the search.

If some direction values are the same (*Step2.2 in Fig. 1*), e.g. vv_2 and vv_3 in Fig. 2-c, we then perform step 2.1a) in Fig.1 to differentiate that directions for all solutions that has similar direction value. This attempts to maintain a set of diverse solution.

The limitation of PB-LS approach is that, we need to determine the number of *reset.iter* to reset the directions and update the solutions. Smaller *reset.iter* indicate more exploration, whilst, biggest value indicate more exploitation.

```
Procedure PB-LS
Step 1: Initialization Phase
  Given an initial candidate solution So with f(So) as the quality of So;
  Set Sbest = So , f(Sbest)= f(So);
  Generate one neighbour solution of Sbest from each shaking neighbourhood;
  Initialize the velocity vector (vv), where vv1,vv2, vv3 ...   vvN+1 =
       solution generated by each shaking neighbourhood;
  Set all the directions and un-improve counters (UnImprovek) for each vector in vv = 0;
  Set N.iters; (stopping condition);
  Set reset.iter (the number of iterations to reset the direction & update solutions);
  Initialize the required  parameters  for the Local Search Method;

Step 2: Improvement (Iterative) Phase
repeat
     Arrange the vector in vv in descending order based on their direction value;
     2.1 If   the direction is clear  // No duplication in the direction values
            Set So  equal to the first vector in vv;  // Select the best direction
            Set k=1;

            2.1a) Apply Local Search on So to produce So*;
                  Calculate the force value for the vvk using equation 2; //positive or negative
                      force with So as a current solution and So*as a candidate solution.

                  Update the direction value by adding force value to the direction of the vvk;
                  If f(So*) is better than f(Sbest), then Sbest= So*;
                  If f(So*) is better than f(S0), then set vvk = So*;
                  Else
                     Increase the UnImprovek of the vvk by one;
                     If UnImprovek = reset.iter;
                         Set the direction of vvk = 0;
                         UnImprovek=0;
                         Randomly generate one neighbour solution of Sbest from each shaking
                         neighbourhood and replace vvk with the best neighbour;
                     End If
                  End Else;
          End if
     2.2 Else  ; // Improve all solutions that has similar direction value
                     Select all solutions in vv that have similar values and apply Step 2.1a) by
                     setting So and k index appropriately..
          End Else;
    iterations=iterations+1;
until  iterations> N.iters (termination condition is met)
Step 3: Termination phase (N.iters termination condition is met)
       Return the best found solution Sbest
```

Fig. 1. Pseudo code for PB-LS approach to solve university course timetabling problem

Velocity Vector	Penalties	Directions Value
vv_1	400	0
vv_2	405	0
vv_3	420	0

(a)

Velocity Vector	Penalties	Directions Value
vv_1	400	5
vv_2	405	4
vv_3	420	1

(b)

Velocity Vector	Penalties	Directions Value
vv_1	400	5
vv_2	405	4
vv_3	420	4

(c)

Fig. 2. Example on PB-LS approach

5 Experimental Results and Discussion

In this work, we have run our algorithm 20 times across 11 instances that were introduced by Socha et al. [12]. The algorithm is run on PC with an Intel dual core 1.8 MHz, 1GB RAM. PB-LS parameters are shown in Table 2. For the small datasets, PB-LS with MPCA-ARDA obtain results within 2 to 10 minutes. Whilst for the medium and large datasets, PB-LS took within 10 to 13 hours to achieve the results.

Table 2. Parameters settings used in our PB-LS

Parameter	Value
N.iters	Termination condition (Number of Iterations) = 500,000.
reset.iter	The number of iterations to reset the directions and update the solutions= 10
VV_N	The number of shaking neighbourhood structures = 3 (i.e. NS6, NS7 and NS8)
Local Search	MPCA-ARDA (number of iterations=?? as a stopping condition)

Table 2 shows that, PB-LS employ four parameters as follows: the first parameter (*N.iters*) is determined based on the literature [18]. Whereas, the second parameter (*reset.iter*) is determined as '10' based on preliminary experiments. In the preliminary experiments, we performed 5 runs for each of *reset.iter* equal to 5, 10, 15 and 20 and found that the *reset.iter*=10 produce the best results. The third parameter (VV_N) is

determined as '3' based on number of shaking neighbourhood (in our case three neighbourhood used are NS6, NS7 and NS8). The fourth parameter is the selection of local search method.

As can be seen from Table 3, both algorithms obtained same results for small instances (S1 to S5). This is because small instances are easy to solve. However, PB-LS with MPCA-ARDA outperformed MPCA-ARDA in all medium and large datasets.

In order to investigate the performance differences between PB-LS with MPCA- and MPCA-ARDA, a Wilcoxon test is carried out with 95% confident level. The null hypothesis assumes there is no difference between the compared methods. The p-value less than 0.05 mean there is a significant difference between these methods. Table 3 shows the comparison between MPCA-ARDA and PB-LS with MPCA-ARDA. It illustrates the best score (fmin), the average score (favg), the standard deviation (σ std) for PB-LS with MPCA-ARDA algorithm and MPCA-ARDA algorithm as well as the p-value of PB-LS- MPCA-ARDA (abbreviated as PB-LS) vs. MPCA-ARDA.

Table 3. Statistical analysis of PB-LS with MPCA-ARDA algorithm and MPCA-ARDA algorithm.

Data Set	***fmin***		***favg***		***Std. Dev.(σ)***		**PB-LS vs. MPCA-ARDA**
	PB-LS	***MPCA-ARDA***	***PB-LS***	***MPCA-ARDA***	***PB-LS***	***MPCA-ARDA***	***p-value***
Small 1	0	0	0.65	1.00	0.75	0.86	0.071
Small 2	0	0	0.55	1.00	0.83	0.79	0.007
Small 3	0	0	0.95	1.15	0.89	0.81	0.392
Small 4	0	0	0.75	0.90	0.72	0.79	0.414
Small 5	0	0	0.60	0.95	0.68	0.83	0.008
Medium 1	41	64	52.75	75.35	7.55	6.99	**0.000**
Medium 2	39	65	54.80	78.05	9.01	8.04	**0.000**
Medium 3	60	91	80.85	106.00	14.73	8.65	**0.000**
Medium 4	39	66	49.50	79.35	7.74	8.32	**0.000**
Medium 5	55	89	65.05	104.05	7.54	9.99	**0.000**
large	463	576	483.20	593.50	16.69	11.89	**0.000**

Note: Bold in p-value indicate that PB-LS is significantly better than MPCA-ARDA.

From Table 3 one can see that, for all tested instances the p-value is less than 0.05, which means that PB-LS is performed better than MPCA-ARDA, except in small 3 and small 4 instances the difference is not significant. Again, this is because small instances are easy and most of proposed method obtained very good results on these instances. Fig. 3 shows the box and whisker plot details of the basic PB-LS with MPCA-ARDA.

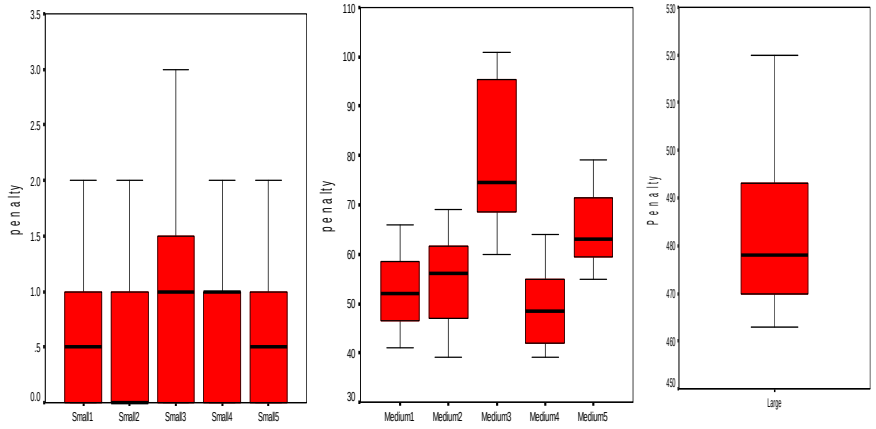

Fig. 3. Box and whisker plot of PB-LS with MPCA-ARDA for all datasets

Fig. 3 shows that in most cases, PB-LS with MPCA-ARDA was capable of producing good quality solutions, in all datasets (the medians are close to the best runs solutions except in medium 2 and small4 datasets the median is close to the worst).

In general the result in Table 3 shows that, PB-LS with MPCA-ARDA outperformed the MPCA-ARDA across all instances (with regard to *fmin* and *favg*). Indeed, the standard deviation shows that PB-LS with MPCA-ARDA is more consistent than MPCA-ARDA (in small1, small3, small5, medium4 and medium5 datasets). Generally, we can conclude that, the result in Fig. 3 and Table 3 showed that PB-LS with MPCA –ARDA obtained good quality solution compared to MPCA-ARDA. This demonstrates that the use of population of solution helps the algorithm in diversifying the search space to get a better quality solution.

Table 4 shows the comparison between our PB-LS (with MPCA-ARDA) and other meta-heuristic searches that were tested on Socha benchmark datasets and we also report the rank of our algorithm compared to the others. The best results are presented in bold.

Results in Table 4 shows that of our algorithm outperformed other methods on medium1, medium2, and medium3 instances. The best result of medium4 is obtained by Abdullah and Turabieh [35] whilst, the best results of medium5 and large instance is obtained by Turabieh et al. [18]. If we consider an individual comparison with these two methods, our algorithm outperformed Abdullah and Turabieh [35] on 5 out of 6 instances and tieing with small instances (obtained the same results for all small instances) and outperformed Turabieh et al. [18] on 3 out of 6 instances and also tieing with small instances (obtained same results for all small instances with regard to the best obtained soltuions). Also, the percentage deviation of our algorithm for medium4, medium5 and large are 0.21, 0.122 and 0.13 which are very close to the best known results for these instances.

Table 4. Comparison between our PB-LS with MPCA-ARDA and other approaches in the literature

Data Set	**Rank**	**PB-LS with MPCA-ARDA**	[23]	[24]	[25]	[26]	[27]	[28]	[29]	[30]	[31]	[14]	[12]	[32]	[33]	[34]	[35]	[36]	[37]	[37]	[38]	[18]	[39]	[40]	[7]	[8]	[9]
Small1	Same	**0**	**0**	**0**	**0**	**0**	5	10	2	**0**	6	3	1	1	8	5	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
Small2	Same	**0**	**0**	**0**	**0**	**0**	5	9	4	**0**	7	4	3	2	11	3	**0**	1	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
Small3	Same	**0**	**0**	**0**	**0**	**0**	3	7	2	**0**	3	6	1	**0**	8	2	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
Small4	Same	**0**	**0**	**0**	**0**	**0**	3	17	**0**	**0**	3	6	1	1	7	3	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
Small5	Same	**0**	**0**	**0**	**0**	**0**	**0**	7	4	**0**	4	**0**	**0**	**0**	5	1	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
Medium1	1	**41**	317	175	80	221	176	243	254	242	372	140	195	146	199	316	55	126	71	88	168	45	84	117	105	82	64
Medium2	1	**39**	313	197	105	147	154	225	258	161	419	130	184	173	202.5	243	70	123	82	88	160	40	82	108	108	78	65
Medium3	1	**60**	357	216	139	246	191	249	251	265	359	189	248	267	-	255	102	185	137	112	176	61	123	135	156	136	91
Medium4	3	39	247	149	88	165	148	285	321	181	348	112	164.5	169	177.5	235	**32**	116	55	84	144	35	62	75	84	73	66
Medium5	2	55	292	190	88	130	166	132	276	151	171	141	219.5	303	-	215	61	129	106	103	71	**49**	75	160	141	103	89
large	3	463	926	912	730	529	798	1138	1027	-	1068	876	851.5	1166	-	-	653	821	777	915	417	**407**	690	589	719	680	576

In order to find out whether the performance of the PB-LS are different in term of solution quality when compared to other methods in the literature, again we carried out a Wilcoxon test between PB-LS and other methods. Since none of the compared method report the full details of the runs, the reported average value by other method were used in our test. Please note that only those reported the average values are considered in the comparisons. All methods are compared by means of pairwise comparisons. The confidence level is 95%. The null hypothesis assumes there is no difference between the compared methods. A p-value less than 0.05 means that there is a significant difference between these methods. Table 5 show the results of Wilcoxon test (P-value).

Table 5 Wilcoxon test results (P-value)

PB-LS Vs.	P-value
[25]	**0.010**
[26]	0.110
[29]	**0.003**
[12]	**0.010**
[32]	**0.004**
[35]	0.110
[37]	**0.026**
[7]	**0.003**
[8]	**0.004**
[9]	**0.003**

According to the p-value in Table 5, for all compared algorithm, except [26] and [35], the reported p-value are less than 0.05 which mean our algorithm outperformed other methods. Although, our algorithm is not better than [26] and [35], according to Wilcoxon test, the results reported in Table 5 shows that in term of solution quality, our method outperformed [26] on 6 out of 6 instances and tieing with small instances (obtained same results for all small instances) and outperformed [35] on 5 out of 6 instances and tieing with small instances (obtained same results for all small instances).

The positive results reported in Table 4 and 5, revealed that our method obtained competitive results compared to the best knows method and also outperformed them on some instances (medium1 to medium3). As a result, we may conclude that the use of population based helped PB-LS in obtaining good results. Also, PB-LS dose not employ a complex operator such as crossover operator which is usually need a repair mechanism to maintain the feasibility.

6 Conclusions and discussion

This work proposed a new population based (PB-LS) that is driven from Gravitational Emulation Local Search algorithm (GELS) idea. PB-LS can be embedded within any local search algorithm to overcome the limitation of population based algorithms. In order to evaluate the effectiveness of PB-LS with MPCA-ARDA, we tested PB-LS

with MPCA-ARDA on Socha course timetabling benchmark dataset (Socha et al. 2002). Results indicate that, PB-LS with MPCA-ARDA outperformed MPCA-ARDA and some other methods in the literature (with regards to method tested on Socha's datasets). This is indicates that PB-LS is suitable for solving university course timetabling problems.

References

1. S. Petrovic and E. K. Burke, "eds. J. Leung, University timetabling, Ch. 45 in the Handbook of Scheduling: Algorithms, Models, and Performance Analysis," Chapman Hall/CRC Press, 2004.
2. A. Schaerf A, "A survey of automated timetabling," Artificial Intelligence Review, 13(2):87-127, 1999.
3. E. K. Burke, Y. Bykov, J. Newall, and S. Petrovic, "A time-predefined approach to course timetabling," Yugoslav Journal of Operations Research (YUJOR), volume 13, number 2, pp. 139-151, 2003.
4. MAS. Elmohamed, P. Coddington, and G. Fox, "A comparison of annealing techniques for academic course scheduling," Selected Papers from 2nd International Conference on the Practice and Theory of Automated Timetabling (PATAT II), Toronto, Canada, Lecture Notes in Computer Science 1408, Springer-Verlag., pp. 92-112, 1998.
5. D. Costa, "A tabu search for computing an operational timetable," European Journal of Operational Research, 76, pp 98-110, 1994.
6. A. Schaerf, "Local search techniques for large high-school timetabling problems," systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on Volume 29, Issue 4, Jul 1999 pp. 368-377, 1999.
7. A. Abuhamdah and M. Ayob, "Multi-neighbourhood particle collision algorithm for solving course timetabling problems," Proceeding in 2009 2nd Conference On Data Mining and Optimization, October, pp.21-27, Selangor, Malaysia, IEEE, 2009.
8. A. Abuhamdah, and M. Ayob, "Adaptive Randomized Descent Algorithm for Solving Course Timetabling Problems," in the International Journal of the Physical Sciences (IJPS - 2010), volume 5(16), pp.2516-2522, December 2010.
9. A. Abuhamdah, and M. Ayob, "MPCA-ARDA for Solving Course Timetabling Problems," Proceeding of the 3nd Conference On Data Mining and Optimization (2011), pp, 171-177, June 2011.
10. C. Blum and A. Roli, "Metaheuristics in combinatorial optimization: Overview and conceptual comparison," ACM Computing Surveys (CSUR), volume 35, issue 3, pages. 268-308, ACM, 2003.
11. B.L. Webster, "Solving Combinatorial Optimization Problems Using a New Algorithm Based on Gravitational Attraction," Thesis submitted to the College of Engineering at Florida Institute of Technology in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Science, 2004.
12. 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.
13. M. Chiarandini, M. Birattari, K. Socha, and O. Rossi-Doria, "An effective hybrid algorithm for university course Timetabling," proceeding in Journal of Scheduling, Volume 9, Number 5 / October, 2006, Springer Netherlands, pp. 403-432, 2006.
14. S. Abdullah, "heuristic approaches for university timetabling problems," PhD Thesis, The University of Nottingham. The School of Computer Science and Information Technology. 17-18, 2006.

15. J. M. Thompson, and K. A. Dowsland, "Variants of Simulated annealing for the examination timetabling problem," Proceeding Annals of Operations Research, volume 63, issue 1, pages. 105–128, 1996.
16. 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, 2008.
17. H.H. Hoos, and T. Stutzle, "Stochastic Local Search: Foundations and Applications," Proceeding Mathematical Methods Of Operation Research, volume 63, number 1, February, pages. 193-194, Elsevier/Morgan Kaufmann, San Francisco, 2006.
18. H. Turabieh, S. Abdullah, B. McCollum, and P. & McMullan, "Fish swarm intelligent algorithm for the course timetabling problem," proceeding Rough Set and Knowledge Technology Conference (RSKT), Lecture Notes in Computer Science, volume 6401/2010, pp.588-595, 2010.
19. R.A., Fisher, "Book of Statistical Methods for Research Workers. Chapter 6, the correlation coefficient," Edinburgh: Oliver and Boyd, pages. 43, 1925.
20. D. R. Anderson, D. J. Sweeney, and T. A. Williams, "Statistics for Business and Economics (with Student CD-ROM, iPod Key Term, and InfoTrac)," South-Western College Publishing, 2005.
21. M. Abramowitz and I. A. Stegun, "Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables," ninth dover printing, tenth gpo printing edn. New York: Dover, 1964.
22. A. A. Afifi and S. P. Azen, "Statistical Analysis: A Computer Oriented Approach," Orlando, FL, USA: Academic Press, Inc, 1979.
23. S. Abdullah, E.K. Burke, and B. McCollum, "An Investigation of Variable Neighbourhood Search for University Course Timetabling," In. The 2nd Multidisciplinary Conference on Scheduling: Theory and Applications (MISTA), July 18th-21st, pages. 413–427. New York, USA, 2005.
24. 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, volume. I, ISBN 9778-967-5048-36-4, pages. 21-25, 2008.
25. P. McMullan, "An extended implementation of the great deluge algorithm for course timetabling," ICCS. Proceedings of the 7th International Conference of Computational Science, Part I, LNCS Lecture Note in Computer Science, Springer-Verlag Berlin Heidelberg, Germany, volume 4487, pp.538-545, , 2007.
26. 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 (CEC 2007), Singapore, pages. 1764–1768, 2007.
27. N. Ejaz and M. Javed, "A Hybrid Approach for Course Scheduling Inspired by Die-Hard Co-Operative Ant Behavior," Proceedings of the IEEE International Conference on Automation and Logistics, August, pages. 3095 – 3100, Jinan, China, 2007.
28. 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, pages. 302-309, 2005.
29. S. Abdullah and H. Turabieh, "Generating University Course Timetable Using Genetic Algorithms and Local Search," In Proceedings of the 2008 Third International Conference on Convergence and Hybrid Information Technology , volume 1, pages. 254-260, 2008.
30. S. Abdullah, E. K. Burke and B. McCollum, "Metaheuristics - Progress in Complex Systems Optimization. Operations Research/Computer Science Interfaces, chapter Using a Randomised Iterative Improvement Algorithm with Composite Neighbourhood Structures for the University Course Timetabling Problem," volume 39, pages. 153–169. Springer US, 2007.
31. E. K. Burke, A. Meisels, S. Petrovicand R. Qu, "A Graph-Based Hyper-Heuristic for Timetabling Problems," European Journal of Operational Research, volume 176, issue 1, January, pages. 177-192, 2007.
32. E. K. Burke, G. Kendall and E. Soubeiga, "A tabu-search hyperheuristic for timetabling and rostering," In Journal of Heuristics, volume 9, number 6, pages. 451–470, 2003.

33. K. Socha, M. Sampels, and M. Manfrin, “Ant Algorithms for the University Course Timetabling Problem with Regard to the State-of-the-Art,” Proceeding in the Third European Workshop on Evolutionary Computation in Combinatorial Optimization (EvoCOP 2003), Lecture Notes in Computer Science, 2003, volume 2611/2003, pages. 334-345, 2003.
34. M. A. Al-Betar, A. T, Khader and T. A. Gani, “A harmony search algorithm for university course timetabling,” Proceedings the 7th International Conference on the Practice and Theory of Automated Timetabling (PATAT 2008), Montreal, Canada, August 18-22, 2008.
35. H. Turabieh and S. Abdullah, “Tabu based Memmetic approach, Incorporating Tabu Search into Memetic approach for enrolment-based course timetabling problems (TS-MA),” Proceeding in 2009 2nd Conference On Data Mining and Optimization, October, pages. 115-119, , Selangor, Malaysia, IEEE, 2009.
36. D. L-. Silva and J. H. Obit, “Evolutionary nonlinear great deluge for university course timetabling,” Proceedings of the 2009 International Conference on Hybrid Artificial Intelligence Systems (HAIS 2009), Hybrid Artificial Intelligence Systems, Lecture Notes in Computer Science, volume 5572/2009, pages.269-276, 2009.
37. J. H. Obit, D. L-. Silva, D. Ouelhadjand M. Sevaux, “Non-Linear Great Deluge with Learning Mechanism for Solving the Course Timetabling Problem,” MIC 2009: The VIII Metaheuristics International Conference, Hamburg, Germany, pages.id1-id10, 2009.
38. M. A. Al-Betar, A.T. Khader, and I.Y. Liao, “A harmony search with multi-pitch adjusting rate for the university course timetabling,” Annals of Operations Research, Recent Advances In Harmony Search Algorithm Studies in Computational Intelligence, volume 270, pp.147-161, 2010.
39. G. M. Jaradat and M. Ayob, “An elitist-ant system for solving the post-enrolment course timetabling problem. In the 2010 International Conference on Database Theory and Application (DTA 2010), Lecture Notes in Computer Science, Springer-Verlag Berlin Heidelberg, pp.167-176, December 2010.
40. K. Shaker, and S. Abdullah, “Controlling multi algorithms using round robin for university course timetabling problem. In the 2010 International Conference on Database Theory and Application (DTA 2010), Lecture Notes in Computer Science, Springer-Verlag Berlin Heidelberg, pp.47-55, December 2010.