Appl Intell (2014) 40:44–53
DOI 10.1007/s10489-013-0444-6

Population based Local Search for university course timetabling problems

Anmar Abuhamdah · Masri Ayob · Graham Kendall · Nasser R. Sabar

Published online: 28 June 2013

Abstract Population based algorithms are generally better at exploring a search space than local search algorithms (i.e. searches based on a single heuristic). However, the limitation of many population based algorithms is in exploiting the search space. We propose a population based Local Search (PB-LS) heuristic that is embedded within a local search algorithm (as a mechanism to exploit the search space). PB-LS employs two operators. The first is applied to a single solution to determine the force between the incumbent solution and the trial current solution (i.e. a single direction force), whilst the second operator is applied to all solutions to determine the force in all directions. The progress of the search is governed by these forces, either in a single direction or in all directions. Our proposed algorithm is able to both diversify and intensify the search more effectively, when compared to other local search and population based algorithms. We use university course timetabling (Socha benchmark datasets) as a test domain. In order to evaluate the effectiveness of PB-LS, we perform a comparison between the performances of PB-LS with other approaches drawn from the scientific literature. Results demonstrate that PB-LS is able to produce statistically significantly higher quality solutions, outperforming many other approaches on the Socha dataset.

Keywords Course timetabling problem · Metaheuristics · Population based algorithm · Hybrid methods · Gravitational emulation

A. Abuhamdah
Department of Computer Science, College of Computer Science and Engineering, Taibah University, 41411 Madinah, Kingdom of Saudi Arabia
e-mail: aabuhamdah@taibahu.edu.sa

A. Abuhamdah · M. Ayob · N.R. Sabar (✉)
Data Mining and Optimisation Research Group (DMO), Center for Artificial Intelligence Technology, Universiti Kebangsaan Malaysia, 43600 Bangi, Selangor, Malaysia
e-mail: naserdolayme@yahoo.com

M. Ayob
e-mail: masri@ftsm.ukm.my

G. Kendall
ASAP Research Group, School of Computer Science, The University of Nottingham, Nottingham NG8 1BB, UK
e-mail: Graham.Kendall@nottingham.ac.uk

G. Kendall
The University of Nottingham Malaysia Campus, Jalan Broga, 43500 Semenyih, Selangor Darul Ehsan, Malaysia
e-mail: Graham.Kendall@nottingham.edu.my

1 Introduction

University course timetabling problems involve 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 the hard constraints and attempt to accommodate the soft constraints as much as possible (in order to produce a high-quality timetable). All hard constraints must be satisfied in order to obtain a feasible timetable, whilst soft constraints can be violated if necessary, but each is penalized. The smaller the overall penalty value, the better the quality of the timetable. University course timetabling problems have 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 have been applied to university course timetabling problems. These approaches include great deluge [3], simulated annealing [4], tabu search [5], randomized descent [6] and honey bee mating algorithm [7]. The disadvantage of the local search methodologies is that they are incapable of escaping from local optima [6, 8–10].

In [11], we proposed an adaptive randomized descent algorithm (ARDA) that employs an adaptive criterion to escape from local optima. We then hybridized MPCA (multi-neighbourhood particle collision algorithm [12]) and ARDA to complement their strengths and weaknesses [13]. However, MPCA-ARDA 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 [14]. Generally, the limitation of many population based algorithms is in exploiting the search space [14], as most of them are good at exploration rather than exploitation. In this work, we propose a population based heuristic (Population Based Local Search, PB-LS) to overcome the limitations of a population based algorithm by increasing the intensification mechanism. Our proposed algorithm is motivated by the idea of Gravitational Emulation Local Search (GELS), which was proposed and developed by Webster [15].

It is also motivated by ideas drawn from memetic algorithms and hyper-heuristics. In the scientific literature, population based algorithms are often hybridized with local search algorithms (i.e. searches based on a single heuristic). The term memetic algorithm is often used to refer to a genetic algorithm hybridized with a local search algorithm [14]. Hyper-heuristics [16–20] are an emergent research trend that aims to control the selection of which heuristic should be applied at each decision point. The proposed PB-LS share some features with both memetic algorithms and hyper-heuristics. PB-LS operate on single solutions, as well as a population of solutions. Thus, it could be classified as a memetic algorithm. The main difference between PB-LS and a memetic algorithm is that PB-LS does not apply the local search at every iteration. PB-LS also utilizes a round-robin strategy to select which neighborhood to explore. Choosing different neighborhoods helps the search to diversify. In this regard, PB-LS has some similarities to hyper-heuristics.

The aim of our work is to investigate that if PB-LS with MPCA-ARDA has more capability in intensifying and diversifying the search, then it will produce better quality solutions compared to other local search (single based). In order to evaluate the effectiveness of PB-LS we compare its performance with MPCA, ARDA, MPCA-ARDA as well as other approaches. We use Socha's university course timetabling datasets, in line with many other researchers [21, 22].

2 Problem description

The eleven standard benchmark instances that were introduced by Socha et al. [21] are used in this work, 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 characterize the rooms.
- A set of Students S who attend the events.

These datasets are categorized into three groups: small (*small 1, small 2, small 3, small 4* and *small 5*), medium (*medium 1, medium 2, medium 3, medium 4* and *medium 5*) and large (*large*) (see Table 1 and for a detailed description). Table 1 also shows the number of students, events, rooms and features as well as the conflict density (*CD*) for each dataset (representing the complexity), an approximation of the number of students enrolled in each event (Students/Events), and an approximation of the number of available rooms for each event (Rooms/Events) which are calculated as in [23].

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 a timetable is measured based on the number of soft constraint violations (penalty cost). Each violation of a soft constraint will be penalized '1' for each student who is involved in this situation [24]. 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

We propose a population based local search algorithm (PB-LS) for university course timetabling problems. PB-LS

Table 1 The course timetabling datasets

Dataset	# Students	# Events	# Rooms	# Features	CD	Students/Events	Rooms/Events
Small 1	80	100	5	5	10.96	4.98	0.82
Small 2	80	100	5	5	13.92	5.36	0.79
Small 3	80	100	5	5	9.71	4.65	1.00
Small 4	80	100	5	5	7.16	3.45	1.39
Small 5	80	100	5	5	15.10	5.99	1.17
Medium 1	200	400	10	5	37.38	8.85	2.23
Medium 2	200	400	10	5	37.66	8.84	1.91
Medium 3	200	400	10	5	40.44	8.85	1.91
Medium 4	200	400	10	5	37.50	8.81	1.88
Medium 5	200	400	10	5	28.27	8.66	1.37
Large	400	400	10	10	45.57	8.92	0.76

starts with an initial solution and iteratively explores its neighbour solutions, seeking 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 [25] to generate initial solutions for the course timetabling problems. The constructive heuristic has three phases: largest degree heuristic [26], neighborhood search and tabu search. The constructive heuristic starts with an empty timetable and successively invokes 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 is 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 feasible timeslot-rooms, we ignore phases 2 and 3. Otherwise, phases 2 and 3 are invoked to achieve feasibility.

Phase 2, employs a hill climbing algorithm to reduce the number of hard constraint 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 in a similar way to phase 2, but it also maintains a tabu list to prevent certain moves being made for a certain number of iterations. The size of the tabu list is calculated by $tl = rand(10) + \delta \times nc$, 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 [25]. This phase will stop after 1,000 non-improving iterations. If the generated solution is infeasible, we re-execute the constructive heuristic, until a feasible solution is returned.

3.2 Neighbourhood structures

The proposed algorithm starts with an initial solution and iteratively improves it by generating a neighbour solution using a set of neighbourhood structures. We use eight neighbourhood structures (NS1–NS8) which have been widely used in the literature. These neighbourhood structures are classified into two groups: (i) common neighbourhoods (NS1–NS5 as in [27]) and (ii) shaking neighbourhoods (NS6 and NS8 as in [27], and NS7 as in [28]). Five of them (NS1–NS5) are the same neighbourhoods utilised in ARDA [11]. 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) [27].

NS2: Randomly select two timeslots and swap all the courses in one timeslot with all the courses in the other timeslot [27].

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) [27].

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 another timeslot. Abdullah [27] used this neighbourhood (NS5) but with 10 % selection. The reason behind for 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 diversify 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 then apply a kempe chain move [28] to move the select even from $t1$ to $t2$. The idea of the kempe chain neighbourhood is to move a chain of courses within the timetable while maintaining the feasibility by swapping conflicting courses between the selected timeslots until achieving feasibility.

NS8: Rotate two timeslots: Randomly select two timeslots (t_i and t_j), where i and j are the timeslot index, $i < j$ and the timeslots are ordered $t_0, t_1, \dots, t_n$ where n is the total number of timeslots. 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 in [27].

4 Population based Local Search algorithm

This work is motivated by the strength of population based algorithms to explore wider areas of the search space than when using a local search algorithm [14]. However, some limitations of population based algorithms are [29]:

- 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 require 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:

(a) Increasing the ability of the intensification process by using a population of solutions in the local search.
(b) Using a memory to guide PB-LS to search more promising regions of the search space which also contain some elite solutions and useful information about them.
(c) Systematically update the population.

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 [15]. 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 are 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 [15].

Both versions of GELS use the same gravitational force equation (see Eq. (1)) but in slightly different ways [15]. The first version applies the equation 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 equation to all solutions (vectors) within the neighbourhood and calculates the gravitational force between each of them and the current solution.

GELS simulates Newton's equation of gravitational force between two objects [15] (Eq. (1)).

$$F = G(CU - CA)/R^2 \tag{1}$$

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 incumbent solution, $CA =$ objective function value of the trial solution and R is the value of the parameter radius, representing the middle point in the distance between two objects (which requires parameter tuning).

Therefore, this work proposes the PB-LS heuristic that eliminates the radius parameter and G to overcome the parameter tuning issue. This gives rise to a new equation (Eq. (2)), based on Eq. (1), but without R^2 and G.

$$F = CU - CA \tag{2}$$

where F is the Force value (assuming a minimization problem), $CU =$ objective function value of the incumbent solution and $CA =$ objective function value of the trial solution.

Equation (2) differs from Eq. (1), as GELS obtains a real force value, whilst Eq. (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. We also omitted parameter R (radius) in the gravitational force equation since in PBLS the gravitational force is increased or decreased solely based on the quality of solution.

Figure 1 shows the pseudo code for the PB-LS approach. Let S_0 be a given initial solution, S_{best} be the best obtained solution, $f(S_{best})$ be the quality (penalty cost) of S_{best}; $f(S_0)$ be the quality of S_0; $N.iters$ 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, vv_i be the ith velocity vector and N be the number of shaking neighbourhoods. In addition, we will also need to initialize the required parameters for the local search method (in our case, we use MPCA-ARDA, see Table 2).

PB-LS starts from zero velocity (i.e. direction value is zero) and is 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 Eq. (2). We use MPCA-ARDA (as a local search) to intensify the search since we have already proposed MPCA-ARDA in [13]. Therefore, we can compare the result of MPCA-ARDA against the PB-LS with MPCA-ARDA in order to demonstrate the effectiveness of our proposed PB-LS approach.

In the initialization phase, we initialize all the parameters (see Table 2), generate initial velocity vectors (by applying shaking neighbourhoods; NS6, NS7 and NS8 on S_0). For our 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 (Step 2 in Fig. 1), at each iteration, we rearrange the solutions in vv in descending order based on their direction values. Higher direction value indicates better potential for improving the solution rather than other solutions in vv. If all direction values are different (Step 2.1 in Fig. 1), we choose the solution that has the largest direction value, vv_1 (e.g. see Fig. 2(b)) and set $S_0 = vv_1$. The local search (in our work, we use MPCA-ARDA) is applied on S_0 to produce S_0^* until stopping condition is met. In MPCA-ARDA, we use common neighbourhood structures (NS1–NS5). The force value for the solution returned by the local search is calculated using Eq. (2) with $CU = S_0$ and $CA = S_0^*$; and is added (positive or negative) to the direction of vv_1.

The S_{best} will be updated with S_0^* if $f(S_0^*)$ is better than $f(S_{best})$. If S_0^* has better quality than S_0, we replace vv_1 with S_0^*. Otherwise, we will increase the unimproved counter of the selected solution ($UnImprove_1$) by one. If $UnImprove_1$ 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 zero and replace vv_1 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 optimum and diversify the search.

If direction values are the same, vv_2 and vv_3 in Fig. 2(c), we perform Step 2.1(a) in Fig. 1 on the appropriate vv (e.g. vv_2 and vv_3 in Fig. 2(c)), to differentiate that directions for all solutions that have similar direction value. This attempts to maintain a set of diverse solutions.

5 Experimental results and discussion

We run our algorithm 20 times across 11 instances that were introduced by Socha et al. [21]. The algorithm is run on PC with an Intel dual core 1.8 MHz, 1 GB RAM. PB-LS parameters are shown in Table 2. For the small datasets, PB-LS obtain results within 2 to 10 minutes. Whilst for the medium and large datasets, PB-LS took between 10 to 13 hours to achieve the results.

Table 2 shows that PB-LS employs four parameters as follows: the first parameter ($N.iters$) is determined based on the literature [30], whereas the second parameter ($reset.iter$) is determined based on preliminary experiments. In this experiment, we performed 5 runs for values of 5, 10, 15 and 20 and found that the $reset.iter = 10$ produces the best results. The third parameter (N) is determined based on the number of shaking neighborhoods (in our case, 3 neighborhoods were used: NS6, NS7, and NS8). The fourth parameter is the selection of local search method.

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

Table 3 shows that for small instances (Small 1 to Small 5), the performance of both algorithms (PB-LS and MPCA-ARDA) are the same (p-value greater than 0.05). This is because small instances are easy to solve and both algorithms manage to solve them effectively. However, PB-LS with MPCA-ARDA outperformed MPCA-ARDA in all medium and large datasets (p-value is less than 0.05).

```
Procedure PB-LS
Step 1: Initialization Phase
    Given an initial candidate solution $S_0$, $f(S_0)$ = the quality of $S_0$;
    Set $S_{best} = S_0$, $f(S_{best}) = f(S_0)$, $N$ = number of shaking neighbourhood;
    Generate $N$ neighbours of $S_{best}$ (one neighbour from each shaking neighbourhood);
    Assign each generated neighbour to the velocity vector, $vv$;
    Set all the directions and un-improve counters ($UnImprove$) for each vector in $vv = 0$;
    Set $N.iters$; //stopping condition (see Table 2);
    Set $reset.iter$; //the number of iterations to reset the direction & update solutions (see Table 2);
    Initialize the required parameters for the Local Search Method (e.g. see Table 2); //depend on what local search is used
Step 2: Improvement (Iterative) Phase
iterations = 1;
repeat
    Arrange vector $vv$ descending order based on their direction value;
    2.1 If the direction is clear //No duplication in the direction values
            Set $S_0 = vv_1$; //Select the best direction
            Set $k = 1$;
      (a) Apply Local Search on $S_0$ to produce $S_0^*$; //any local search can be applied
              Calculate the force value for the $vv_k$ using Eq. (2); //positive or negative force with $S_0$ as a current solution
              and $S_0^*$ as a candidate solution.
              Update the direction value by adding force value to the direction of the $vv_k$;
              If $f(S_0^*)$ is better than $f(S_{best})$, then $S_{best} = S_0^*$;
              If $f(S_0^*)$ is better than $f(S_0)$, then set $vv_k = S_0^*$;
              Else
                  Increase the $UnImprove_k$ by one;
                  If $UnImprove_k == reset.iter$;
                      Set the direction of $vv_k = 0$;
                      $UnImprove_k = 0$;
                      Randomly generate $N$ neighbours of $S_{best}$ (one neighbour from each shaking neighbourhood)
                      and set $vv_k = S_{best}^*$ (the best neighbour of $S_{best}$);
                  End If
              End Else;
        End if
    2.2 Else; //Improve all solutions that have a similar direction value
            While (direction not clear) //some vv have similar direction value
                Select the $vv^*$ (the $vv$ that have similar direction values with other $vv$), set $k$ = index of $vv^*$,
                set $S_0 = vv^*$ and apply Step 2.1(a);
            End while
        End Else;
iterations++;
until iterations > $N.iters$ (termination condition is met)
Step 3: Termination phase ($N.iters$ termination condition is met)
    Return the best found solution $S_{best}$
```

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

Figure 3 shows the box and whisker plot of the PB-LS with MPCA-ARDA. It shows that in most cases, PB-LS is capable of producing good quality solutions in all datasets (the medians are close to the best solutions), except in small 3, small 4 and medium 2 datasets, the median is close to the worst. For example, in small 2, more than 75 % of runs obtained solutions that are close to the minimum penalty value.

Fig. 2 Example showing initializing velocity vectors in PB-LS

Velocity vector	Penalties	Directions value
vv_1	400	5
vv_2	405	4
vv_3	420	1

(a)

Velocity vector	Penalties	Directions value
vv_1	400	0
vv_2	405	0
vv_3	420	0

(b)

Velocity vector	Penalties	Directions value
vv_1	400	5
vv_2	405	4
vv_3	420	4

(c)

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
N	The number of shaking neighbourhood structures = 3 (i.e. NS6, NS7 and NS8)
Local Search	MPCA-ARDA (number of iterations = 10)

In general, the results in Table 3 shows that PB-LS outperformed MPCA-ARDA across all instances (with regard to *fmin* and *favg*). This demonstrates that the use of a population of solutions helps the algorithm in diversifying the search space to obtain better quality solutions.

Table 4 shows the comparison between our PB-LS and other metaheuristic searches that were tested on the 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 outperforms other methods on medium 1, medium 2, and medium 3 instances. The best result of medium 4 is obtained by Turabieh and Abdullah [31], whilst the best results of medium 5 and large instance is obtained by Turabieh et al. [30]. If we consider an individual comparison with these two methods, our algorithm outperformed Turabieh and Abdullah [31] on 5 instances and ties on the small instances (obtained the same results for all small instances) and outperformed Turabieh et al. [30] on 3 instances and also ties on the small instances (obtained same results for all small instances with regard to the best obtained solutions). Also, the percentage deviation (Δ (%)) of our algorithm for medium 4, medium 5 and large are 0.21, 0.122 and 0.13, which are very close to the best known results for these instances (third column in Table 4).

In order to find out whether the performance of the PB-LS is different in terms of solution quality when compared to other methods in the literature, again we carried out a Wilcoxon test between PB-LS and other methods (Garcia et al. [47]). Since none of the compared methods report the full details of the runs, the reported average value by other methods were used in our test. Please note that only those that 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).

According to the p-values in Table 5, for almost all compared algorithms, the reported p-value is less than 0.05, which means that the performance of PB-LS is significantly different from other methods (except M4 [24] and M15 [31]). Although our algorithm is not better than M4 and M15, according to the Wilcoxon test, the results reported in Table 5 show that in terms of solution quality, our method outperformed M4 on 6 instances and ties on the small instances (obtained same results for all small instances) and outperformed M15 on 5 instances and ties on the small instances (obtained same results for all small instances).

Table 3 Statistical analysis of the results obtained by PB-LS with MPCA-ARDA algorithm and MPCA-ARDA algorithm out of 20 runs

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. The value for *fmin* and *favg* are the minimum and average penalty cost, respectively

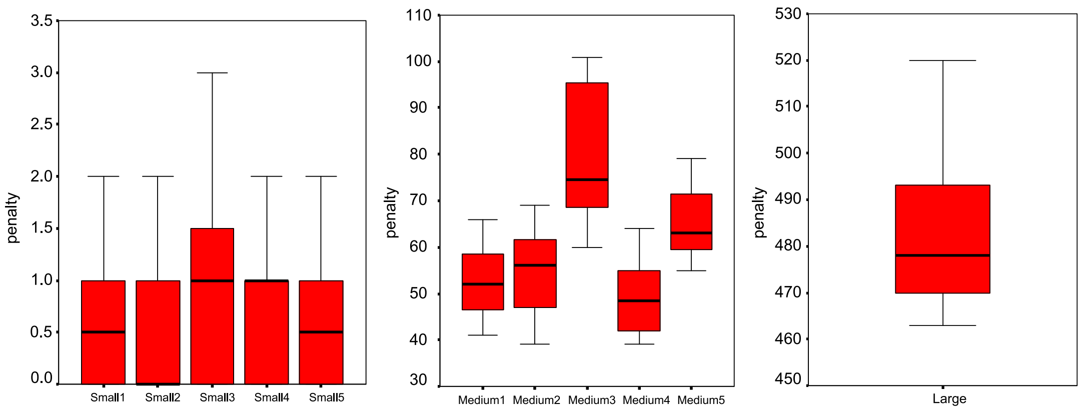

Fig. 3 Box and whisker plot of results obtained by PB-LS (for 20 runs) for all datasets

The positive result reported in Tables 4 and 5 reveals that our method obtains competitive results compared to the best known methodologies and also outperforms them on some instances (medium 1 to medium 3). As a result, we conclude that the use of a population helps PB-LS in obtaining good quality solutions. Indeed, PB-LS did not employ a complex operator such as a crossover operator which usually requires a repair mechanism to maintain feasibility.

6 Conclusions and discussion

This work has proposed a new population based local search algorithm (PB-LS) that is motivated by the idea from Gravitational Emulation Local Search algorithm (GELS). PB-LS was embedded with MPCA-ARDA (multi-neighbourhood particle collision algorithm and adaptive randomized descent algorithm) as a local search to overcome the limitations of population based algorithms (in fact, it can be embedded within any local search algorithm). In order to evaluate the effectiveness of PB-LS with MPCA-ARDA, we tested PB-LS with MPCA-ARDA on the Socha course timetabling benchmark dataset. Results indicate that PB-LS with MPCA-ARDA outperform MPCA-ARDA and some other methods in the literature, the results are generally statistically significant. Thus we can conclude that PB-LS with MPCA-ARDA has more capability in intensifying and diversifying the search.

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 larger values promote more exploitation.

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

Data Set	Rank	Δ (%)	PB-LS with MPCA-ARDA	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12	M13	M14	M15	M16	M17	M17	M18	M19	M20	M21	M22	M23	M24
Small 1	Same	0	**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**
Small 2	Same	0	**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**
Small 3	Same	0	**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**
Small 4	Same	0	**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**
Small 5	Same	0	**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**
Medium 1	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
Medium 2	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
Medium 3	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
Medium 4	3	0.21	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
Medium 5	2	0.122	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	0.13	463	926	912	730	529	798	1138	1027	–	1068	876	851.5	1166	–	–	653	821	777	915	417	**407**	690	589	719	680	576

Note: M1: (Abdullah et al. [32]). M2: (Abdullah and Turabieh [33]). M3: (McMullan [24]). M4: (Abdullah et al. [34]). M5: (Ejaz and Javed [35]). M6 (Asmuni et al. [36]). M7: (Abdullah and Turabieh [37]). M8: (Abdullah et al. [38]). M9: (Burke et al. [39]). M10: (Landa-Silva and Obit [25]). M11: (Socha et al. [21]). M12: (Burke et al. [40]). M13: (Socha et al. [22]). M14: (Al-Betar et al. [41]). M15: (Turabieh and Abdullah [31]). M16: (Landa-Silva and Obit [42]). M17: (Obit et al. [43]). M18: (Al-Betar et al. [44]). M19: (Turabieh et al. [30]). M20: (Jaradat and Ayob [45]). M21: (Shaker and Abdullah [46]). M22: (Abuhamdah and Ayob [12]). M23: (Abuhamdah and Ayob [11]). M24: (Abuhamdah and Ayob [13]). The value for PL-LS, M1, M2, etc. are the minimum penalty cost obtained by each approach

Table 5 Wilcoxon test results (p-value) between PB-LS and other approaches with 95 % confidence level

PB-LS vs.	p-value
M3	0.010
M4	0.110
M7	**0.003**
M11	**0.010**
M12	**0.004**
M15	0.110
M17	**0.026**
M22	**0.003**
M23	**0.004**
M24	**0.003**

References

1. Petrovic S, Burke E (2004) Educational timetabling. In: Handbook of scheduling: algorithms, models, and performance analysis, pp 41–45
2. Schaerf A (1999) A survey of automated timetabling. Artif Intell Rev 13(2):87–127
3. Burke E, Bykov Y, Newall J, Petrovic S (2003) A time-predefined approach to course timetabling. Yugosl J Oper Res 13(2):139–151. ISSN: 0354-0243, EISSN: 2334-6043
4. Elmohamed MS, Coddington P, Fox G (1998) A comparison of annealing techniques for academic course scheduling. In: Practice and theory of automated timetabling II. Springer, Berlin, pp 92–112
5. Costa D (1994) A tabu search algorithm for computing an operational timetable. Eur J Oper Res 76(1):98–110
6. Schaerf A (1999) Local search techniques for large high school timetabling problems. IEEE Trans Syst Man Cybern, Part A, Syst Hum 29(4):368–377
7. Sabar NR, Ayob M, Kendall G, Qu R (2012) A honey-bee mating optimization algorithm for educational timetabling problems. Eur J Oper Res 216(3):533–543
8. Ayvaz D, Topcuoglu H, Gurgen F (2012) Performance evaluation of evolutionary heuristics in dynamic environments. Appl Intell 37(1):130–144. doi:10.1007/s10489-011-0317-9
9. Lwin K, Qu R (2013) A hybrid algorithm for constrained portfolio selection problems. Appl Intell. doi:10.1007/s10489-012-0411-7, pp 1–16
10. Rajabalipour Cheshmehgaz H, Desa M, Wibowo A (2013) Effective local evolutionary searches distributed on an island model solving bi-objective optimization problems. Appl Intell 38(3):331–356. doi:10.1007/s10489-012-0375-7
11. Abuhamdah A, Ayob M (2010) Adaptive randomized descent algorithm for solving course timetabling problems. Int J Phys Sci 5(16):2516–2522
12. Abuhamdah A, Ayob M (2009) Multi-neighbourhood particle collision algorithm for solving course timetabling problems. In: 2nd conference on data mining and optimization, 2009. DMO'09. IEEE Press, New York, 21–27
13. Abuhamdah A, Ayob M (2011) MPCA-ARDA for solving course timetabling problems. In: 3rd conference on data mining and optimization (DMO). IEEE Press, New York, pp 171–177
14. Blum C, Roli A (2003) Metaheuristics in combinatorial optimization: overview and conceptual comparison. ACM Comput Surv 35(3):268–308
15. Webster BL (2004) Solving combinatorial optimization problems using a new algorithm based on gravitational attraction. PhD thesis, College of Engineering at Florida Institute of Technology

16. Burke EK, Hyde M, Kendall G, Ochoa G, Özcan E, Woodward JR (2010) A classification of hyper-heuristic approaches. In: Handbook of metaheuristics. Springer, Berlin, pp 449–468
17. Burke EK, Hyde MR, Kendall G (2012) Grammatical evolution of local search heuristics. IEEE Trans Evol Comput 16(3):406–417
18. Ren Z, Jiang H, Xuan J, Luo Z (2012) Hyper-heuristics with low level parameter adaptation. Evol Comput 20(2):189–227
19. Sabar N, Ayob M, Qu R, Kendall G (2012) A graph coloring constructive hyper-heuristic for examination timetabling problems. Appl Intell 37(1):1–11. doi:10.1007/s10489-011-0309-9
20. Soghier A, Qu R (2013) Adaptive selection of heuristics for assigning time slots and rooms in exam timetables. Appl Intell. doi:10.1007/s10489-013-0422-z
21. Socha K, Knowles J, Sampels M (2002) A max-min ant system for the university course timetabling problem. In: Ant algorithms. Springer, Berlin, pp 1–13
22. Socha K, Sampels M, Manfrin M (2003) Ant algorithms for the university course timetabling problem with regard to the state-of-the-art. In: Applications of evolutionary computing. Springer, Berlin, pp 334–345
23. Chiarandini M, Birattari M, Socha K, Rossi-Doria O (2006) An effective hybrid algorithm for university course timetabling. J Sched 9(5):403–432
24. Mcmullan P (2007) An extended implementation of the great deluge algorithm for course timetabling. In: Computational science—ICCS 2007. Springer, Berlin, pp 538–545
25. Landa-Silva D, Obit JH (2008) Great deluge with non-linear decay rate for solving course timetabling problems. In: 4th international IEEE conference intelligent systems, 2008. IS'08. IEEE Press, New York, pp 8-11–8-18
26. Abbasian R, Mouhoub M (2013) A hierarchical parallel genetic approach for the graph coloring problem. Appl Intell. doi:10.1007/s10489-013-0429-5, pp 1–19
27. Abdullah S (2006) Heuristic approaches for university timetabling problems. PhD thesis, School of Computer Science, The University of Nottingham
28. Thompson JM, Dowsland KA (1996) Variants of simulated annealing for the examination timetabling problem. Ann Oper Res 63(1):105–128
29. Hoos HH, Stützle T (2004) In: Stochastic local search: foundations & applications. The Morgan Kaufmann series in artificial intelligence
30. Turabieh H, Abdullah S, McCollum B, McMullan P (2010) Fish swarm intelligent algorithm for the course timetabling problem. In: Rough set and knowledge technology. Springer, Berlin, pp 588–595
31. Turabieh H, Abdullah S (2009) Incorporating tabu search into memetic approach for enrolment-based course timetabling problems. In: 2nd conference on data mining and optimization, 2009. DMO'09. IEEE Press, New York, pp 115–119
32. Abdullah S, Burke EK, Mccollum B (2005) An investigation of variable neighbourhood search for university course timetabling. In: The 2nd multidisciplinary international conference on scheduling: theory and applications (MISTA), pp 413–427
33. Abdullah S, Turabieh H (2009) Electromagnetic like mechanism and great deluge for course timetabling problems. Paper presented at the first 2008 seminar on data mining and optimization DMO
34. Abdullah S, Burke EK, McCollum B (2007) A hybrid evolutionary approach to the university course timetabling problem. In: IEEE Congress on evolutionary computation, 2007. CEC. IEEE Press, New York, pp 1764–1768.
35. Ejaz N, Javed MY (2007) A hybrid approach for course scheduling inspired by die-hard co-operative ant behavior. In: 2007 IEEE international conference on automation and logistics, 2007. IEEE Press, New York, pp 3095–3100
36. Asmuni H, Burke EK, Garibaldi JM (2005) Fuzzy multiple heuristic ordering for course timetabling. In: Proceedings of the 5th United Kingdom workshop on computational intelligence (UKCI 2005), pp 302–309. Citeseer
37. Abdullah S, Turabieh H (2008) Generating university course timetable using genetic algorithms and local search. In: Third international conference on convergence and hybrid information technology, 2008. ICCIT'08. IEEE Press, New York, pp 254–260
38. Abdullah S, Burke EK, McCollum B (2007) Using a randomised iterative improvement algorithm with composite neighbourhood structures for the university course timetabling problem. In: Metaheuristics. Springer, Berlin, pp 153–169
39. Burke EK, McCollum B, Meisels A, Petrovic S, Qu R (2007) A graph-based hyper-heuristic for educational timetabling problems. Eur J Oper Res 176(1):177–192
40. Burke EK, Kendall G, Soubeiga E (2003) A tabu-search hyper-heuristic for timetabling and rostering. J Heuristics 9(6):451–470
41. Al-Betar MA, Khader AT, Gani TA (2008) A harmony search algorithm for university course timetabling. In: 7th international conference on the practice and theory of automated timetabling (PATAT 2008), Montreal, Canada, August 18–22.
42. Landa-Silva D, Obit JH (2009) Evolutionary non-linear great deluge for university course timetabling. In: Hybrid artificial intelligence systems. Springer, Berlin, pp 269–276
43. Obit J, Landa-Silva D, Ouelhadj D, Sevaux M (2009) Non-linear great deluge with learning mechanism for solving the course timetabling problem. In: MIC 2009: the VIII metaheuristics international conference, Hamburg, Germany, pp 1–10
44. Al-Betar MA, Khader AT, Liao IY (2010) A harmony search with multi-pitch adjusting rate for the university course timetabling. In: Recent advances in harmony search algorithm. Springer, Berlin, pp 147–161
45. Jaradat GM, Ayob M (2010) An elitist-ant system for solving the post-enrolment course timetabling problem. In: Database theory and application, bio-science and bio-technology. Springer, Berlin, pp 167–176
46. Shaker K, Abdullah S (2010) Controlling multi algorithms using round robin for university course timetabling problem. In: Database theory and application, bio-science and bio-technology. Springer, Berlin, pp 47–55
47. García S, Fernández A, Luengo J, Herrera F (2010) Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: experimental analysis of power. Inf Sci 180(10):2044–2064