Hybridizing Plant Propagation and Local Search for uncapacitated exam scheduling problems

Meryem Cheraitia, Salim Haddadi

LabSTIC, 8 Mai 1945 University, Guelma, Algeria
E-mail: haddadi.salim(cheraitia.meryem)@univ-guelma.dz

Abdellah Salhi

Department of Mathematical Sciences, University of Essex, Wivenhoe Park, Colchester CO4 3SQ, UK E-mail: as@essex.ac.uk

Abstract The uncapacitated exam scheduling problem (UESP) is a well-known computationally intractable combinatorial optimization problem. It aims at assigning exams to a predefined number of periods, avoiding conflicts over the same period, and spreading exams as evenly as possible. Here, we suggest a new hybrid algorithm combining the Plant Propagation algorithm (PPA) and Local Search (LS) for it. PPA is a population-based metaheuristic that mimics the way plants propagate. To the best of our knowledge, this is the first time this idea is exploited in the context of UESP. Extensive testing on the University of Toronto benchmark dataset, and comparison against a large number of new as well as well-established methods shows that this new metaheuristic is competitive and represents a substantial addition to the arsenal of tools for solving the problem.

Keywords: Uncapacitated exam scheduling; plant propagation algorithm; local search; hybridization.

Reference to this paper should be made as follows: Cheraitia, M., Haddadi, S. and Salhi, A. (201X) 'Hybridizing Plant Propagation and Local Search for uncapacitated exam scheduling problems', *International Journal of Services and Operations Management*, Vol. x, No. x, pp.xxx–xxx.

Biographical notes:

Meryem Cheraitia is preparing her PhD in Computer Science at the University of Guelma. Her research interests include issues related to image recognition, data mining, and operational research. She coauthored, with S. Haddadi, three papers published at international journals, and several conference papers.

Salim Haddadi received a PhD in Computer Science from Blaise Pascal University, Clermont-Ferrand, France. He is professor of operations research at the Computer Science department, University of Guelma. His areas of interest include combinatorial optimization, computational complexity, and scheduling. He has published several research papers at international journals and a book edited by Hermes/Lavoisier.

Abdellah Salhi is Professor of Operational Research. He obtained his PhD from the University of Aston in Birmingham, UK. His research interests are in the design, analysis, implementation and application of mathematical programming algorithms. Application areas include optimization, scheduling, decision making under partial information, data-mining and forecasting. Recently, he led a project on workforce scheduling in container terminals, in conjunction with the Port of Felixstowe, UK. He has introduced the Plant Propagation Algorithm for global optimization, a heuristic inspired by the way strawberry plants propagate using

runners. He has published over sixty papers in refereed journals and conference proceedings.

1 Introduction

Timetabling is a repetitive, hard and tedious task. It is met almost everywhere: in educational establishments for course timetabling (Kaviani et al., 2014) and exam scheduling as we shall see in a while, in hospitals for nurse rostering (Nasiri and Rahvar, 2017) or health care management (Jittamai and Kangwansura, 2016), in fire stations (Alutaibi et al., 2015), in public transport (Azadeh et al., 2012), in seafaring (Ammar et al., 2014), in military training (Lee et al., 2009). Timetabling can be seen as both an assignment and a scheduling problem, since it aims at assigning resources (teachers, nurses, rooms, ...) to tasks (student classes, patients, ...) and scheduling them over time. Timetabling problems are theoretically, as well as computationally, intractable.

Exam scheduling is a special timetabling problem. It strives to assign a predefined set of exams to a fixed number of periods so as to satisfy a given set of constraints. One of the characteristics of exam scheduling is that the constraints involved vary considerably from an institution to another. Some of the constraints, qualified as hard, are essential and should be fully satisfied. These constraints often relate to the physical limitations of the real world. For example, a student cannot sit two distinct exams in the same period. A timetable satisfying the hard constraints is called feasible. The rest of the constraints are qualified as soft, which means that it is desirable to satisfy them, but they can be violated or bypassed. Teachers' wishes and preferences are an example of such constraints. Generally, soft constraints cannot be completely satisfied due to their conflicting nature. Therefore, the aim is usually to minimize their violation. The quality of a timetable, however, is determined by the extent to which it satisfies the soft constraints.

Every term of the academic year, educational institutions across the world face the exam scheduling problem. In the early days, timetables were generated manually, requiring a lot of hard work. These efforts often resulted in timetables of unsatisfactory quality due to the complex nature of the problem particularly when the number of exams and students increased. Nowadays, automated timetabling is common place often producing high quality timetables in reasonable times.

In this paper, we deal with UESP with no capacity constraints. This is the simplest exam scheduling problem since it involves only two constraints, one hard and one soft. Note, however, that low number of constraints do not, in general, imply that a problem is easy to solve.

1.1 UESP

Assume that m students have to take n exams over p periods. Let $(c_{ij})_{n\times n}$ be the conflict matrix where c_{ij} is the number of students taking both exams i and j. We consider UESP, where only one hard constraint and one soft constraint are addressed. The hard constraint stipulates that exams with common students should not be assigned to the same period while the soft constraint requires spreading the exams over the periods so as to reduce the number of students sitting exams in close proximity.

Consider the graph $G = (X, E)$ where X is a set of n nodes, corresponding to the exams, and where E is a set of edges such that there is an edge connecting nodes i and j if $c_{ij} \neq 0$. A p-colouring of the graph G is an assignment of p colours to the nodes in such a way that every node is coloured and any two adjacent nodes are coloured differently. Hence, a p-colouring of the graph G partitions the set of exams into periods $\pi_1, \cdots, \pi_p$ and yields a feasible timetable (periods and colours correspond one to one). The solution space of UESP is thus the set of all p-colourings of the graph G, which, although finite, increases exponentially with the graph size.

Given a p-colouring, let $t_i \in \{\pi_1, \cdots, \pi_p\}$ be the period attributed to exam i. Now, the goal of UESP is to find a p-colouring satisfying the soft constraint. Carter et al. (1996) suggested a clever and widely adopted model: find a p-colouring of the graph G so as to minimize

$$\frac{1}{m} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c_{ij} \times prox(t_i, t_j) \tag{1}$$

where

$$prox(t_i, t_j) = \begin{cases} 2^5/2^{|t_j - t_i|} & \text{if } 1 \leq |t_j - t_i| \leq 5 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

The objective function (1) represents more or less the soft constraint violation. The proximity costs $prox(t_i, t_j)$ penalize the proximity of two conflicting exams. If i and j are conflicting exams assigned to adjacent periods t_i and t_j (*i.e.* $|t_j - t_i| = 1$) then the penalty is the higher, 16. But if there is one (resp. two, three, four) free day(s) between the two periods t_i and t_j ($|t_j - t_i| = 2$, resp. $3, 4, 5$), then the penalty decreases to 8, resp. $4, 2, 1$. Otherwise, the penalty $prox(t_i, t_j)$ is null.

Stated in this way, UESP is clearly a combinatorial optimization problem. If X is the set of all p-colourings of the graph G, UESP is of the general form: find $x^* \in X$ such that $f(x^*) = min_{x \in X} f(x)$ where $f(x)$ is the objective function in (1). UESP is computationally intractable even if we omit the soft constraint since, in this case, it reduces to the decision problem of whether there exists a feasible timetable which is the well-known NP-complete p-colouring problem.

1.2 Our contributions

PPA is a new plant inspired paradigm. The successful applications (Selamoglu and Salhi, 2016; Zhou et al., 2016) it has had in combinatorial optimization motivate our attempt to use it in the context of UESP.

Since UESP is intractable, exact solution approaches often require prohibitive amounts of time. It is, therefore, desirable to settle for a practical and reasonable approach in the form of a heuristic capable of finding good timetables in acceptable computing times. The heuristic we have in mind and which we put forward in this work consists in hybridizing PPA with LS. Our contributions can be summarized as follows.

- We introduce some improvements to PPA and show how to map it onto UESP.
- LS is hybridized to improve the quality of the timetables iteratively generated.
- Extensive testing is realized on the University of Toronto benchmarks.

- Computational comparison of our hybrid method with thirteen recent metaheuristics is performed.

1.3 Outline of the paper

Section 2 is an overview of the state-of-the-art. Section 3 recalls some basic facts about PPA. Sections 4 and 5 constitute the core of our contribution. The first presents our solution methodology, and the last is devoted to the computational experiments where our results are presented.

2 Related work

Due to its practical relevance, UESP is one of the most studied combinatorial optimization problems. For the same reason, new paradigms and new methods are continuously discovered and designed.

As mentioned earlier, exact methods are often unsuitable for combinatorial problems, including UESP (MirHassani, 2006). This means that new solution approaches are almost exclusively metaheuristics.

A metaheuristic is a high-level method whose purpose is to drive a simple low-level heuristic for finding good solutions to an optimization problem, without guaranteeing optimality, by sampling the solution space which is too large to be completely searched. Metaheuristics proceed by first generating an initial solution (LS-based approaches) or an initial set of solutions (population-based approaches), to which one or more improving procedures are applied.

Metaheuristics are often capable of generating satisfactory solutions in reasonable computing times, even when the problem instances under consideration are large. An important number of metaheuristics for UESP have been proposed and investigated, covering almost all possible paradigms (Qu et al., 2008). In the following, we review some of the most recent.

In their study, Abdul-Rahman et al. (2011) divided the exams into easy and difficult. Ordering and shuffling are then incorporated into a decomposition and ordering strategy to place the exams according to their difficulty.

Al-Betar et al. (2013) studied different cases that combine the main components of Harmony Search in order to investigate their effectiveness. Three main groups of criteria are considered: memory, randomness and pitch adjustment. Each of these criteria is associated with recombination, randomness and neighbourhood structures.

Alzaqebah and Abdullah (2015) also used the Artificial Bee Colony algorithm. They introduced three selection strategies (tournament, rank and disruptive selection) for the follower bees to select a recruiter and a self-adaptive mechanism to select a neighbourhood structure.

Arogundade et al. (2010) used a simple Genetic Algorithm on a real-world exam timetabling problem arising at the University of Agriculture, Abeokuta, Nigeria. Bullnheimer (1998) modelled the exam timetabling at the University of Magdeburg as a quadratic assignment problem. Simulated Annealing is then used to search for solutions based on two neighbourhood structures (period move and exam move).

Burke et al. (2011) focused on Graph Colouring heuristics and emphasize their attractiveness. They introduced a new strategy of using linear combinations of these heuristics

whose main role is to order the nodes of the graph before colouring them. Caramia et al. (2008) proposed new LS approaches. Many variants of the problem are investigated including that where the objective is to minimize the number of periods.

Cheraitia and Haddadi (2016) used Simulated Annealing as an iterative heuristic for better spreading the conflicting exams. Three neighbourhood structures are searched alternatively in the improvement phase. Eley (2006) compared a simple Ant Colony system and max-min Ant system for exam timetabling. He reported that, unlike for the Travelling Salesman problem or the Quadratic Assignment problem, the max-min Ant system was the worst.

Pais and Amaral (2011) implemented a Decision Expert system, namely the Fuzzy Inference Rule Based System for tuning the parameters of a simple version of Tabu Search (More precisely for tuning the tabu tenure) based on the concepts of frequency and inactivity. These concepts are related respectively to the number of times a move is introduced in the tabu list and the last time the move is prevented. The initial solution is obtained by using the saturation degree heuristic. Finally, new solutions are obtained using two neighbourhood structures. The duration of the tabu status for each move is determined individually using the Fuzzy Inference Rule Based System.

In Qu and Burke (2009), the authors tried to unify the vast research directions in hyper-heuristics. Thompson and Dowsland (1998) applied a two-stage method. In the first stage, a feasible solution is generated, which is then improved in the next stage using Simulated Annealing. Kempe chain neighbourhood is searched. The experiments conducted conclude that adaptive cooling produces in general better results than geometric cooling. It is also shown that the solution quality depends not only on the choice of the cooling schedule and the neighbourhood structure, but also on the way the neighbourhood is sampled.

Abdullah et al. (2009) presented a hybrid approach that starts with a random generated population of timetables. Electromagnetic-like mechanism is applied to compute the force for each solution. The Great Deluge algorithm then used this force value to compute the decreasing rate parameter. Abdullah et al. (2010) hybridized Tabu List and the Memetic Algorithm. The initial population is generated using the Saturation Degree heuristic. Then the crossover and mutation operators are applied to produce significant improvements in solution quality. The Tabu List is used to penalize ineffective solutions.

Ahandani et al. (2012) hybridized a new LS method, called the two-stage Hill-Climbing, with the Discrete Particle Swarm Optimization algorithm using three strategies of hybridization. Kolonias et al. (2014) suggested hybridizing a Genetic Algorithm and a simple LS method (the greedy Steepest Descent). The resulting metaheuristic is implemented on a multi-threaded GPU.

Burke et al. (2012) presented a hyper-heuristic where the low-level heuristics consist of searching the Kempe chain and period swapping neighbourhoods. A hyper-heuristic is presented by Qu et al. (2014). It is a constructive method based on a simple Estimation Distribution algorithm for driving appropriately the low-level heuristics. Sabar et al. (2011) investigated a hyper-heuristic where four low-level Graph Coloring heuristics are hierarchically hybridized.

3 PPA: an overview

Real-world search/optimization problems are notoriously hard to solve. Fortunately, equivalent problems occur in Nature. For instance, foraging for food is a search problem, propa-

gating and multiplying for perenniality and survival are optimization problems. These are well solved in Nature by all living things otherwise they would have gone extinct. Many successful heuristics are inspired by life processes. PPA is one of the most recent such heuristics.

Salhi and Fraga (2011), among others, observed that plants, and in particular the strawberry plant, exhibit what can possibly be referred to as intelligent behavior. They noticed that this behavior can be exploited to solve real world problem. Subsequently, they suggested an algorithm, PPA, that emulates the way the strawberry plant propagates. PPA belongs to a family of algorithms named "Plant intelligence based" (Akyol and Alatas, 2016) or "Plant-inspired" (Brabazon et al., 2015) algorithms.

3.1 Basic principle

The basic principle of PPA is set out in its pseudo-code, shown in Figure 1. All plants have an underlying propagation strategy. Most of them propagate through seeds. Some, such as the commercial hybrid strawberry plant, however, rely on runners. Recall that hybrids, generally do not bear offspring through mating or seeds. A runner is a branch (rhizome), arising from the parent plant, which grows over ground. When it touches the ground, it produces roots which then give rise to another plant. Some plants use runners to explore the landscape where they are to find good places to grow and propagate. Typically, a good place is one which is sunny, has enough nutrients and humidity. To improve its chances of survival, a strawberry plant implements this basic strategy:

- If it is in a good spot, it sends many short runners;

- if it is in a poor spot, it sends few long runners.

The rationale behind this strategy is the following: when a plant grows in a good spot, it gives chance to its descendants to take full advantage of the favourable conditions, by generating many of them in its vicinity. On the contrary, when the plant is in a poor spot, there is no point in keeping its descendants too close; it, therefore, sends long runners to explore the land further afield searching for better conditions. Moreover, it can only send few of these long runners since the resources at its disposal are limited, and the search for better conditions is not guaranteed to succeed.

Exploitation and exploration, and more importantly the balance between them, are essential features of any successful heuristic. Exploitation and exploration manifest themselves in a contradictory way. While exploitation concentrates the search to around a local optimum, exploration allows the search to escape from the attraction region of the local optimum.

Exploitation and exploration play a central role in PPA. Exploitation is implemented through sending short runners from parent plants which are in profitable places, while exploration is implemented through sending long runners from parent plants living in less profitable places.

Although PPA can be started with a single plant, it is better initialized with a population of plants randomly spread throughout the search space (Line 1 of Figure 1). Sorting in Lines 2 and 6 is necessary to distinguish between the s plants in good spots and the $N-s$ plants in less good spots, and for rejecting "bad" plants from the tail of the sorted population in Line 7. Notice that there is no cooling schedule or equivalent for controlling the algorithm. Only a percentage of the generated plants and their parents is accepted in Line 7 of Figure 1. This

selection has two reasons. First, it keeps the population with a manageable size. Second, as in nature, it gives the opportunity to the best to reproduce by sending runners. The absence of a cooling schedule for instance makes PPA more robust than cooling-schedule based approaches like Simulated Annealing, Great Deluge, and Threshold Accepting.

1. Sort plants $p_1, \cdots, p_N$ in ascending order according to fitness
2. Repeat until stopping criterion met
3. $s \leftarrow \lceil \%Good \times N \rceil$
4. For $i = 1, \cdots, s$ plant p_i sends r short runners
5. For $i = s+1, \cdots, N$ plant p_i sends R long runners
6. Sort the new larger population
7. Remove a percentage of plants from the tail
8. // Let N be the size of the accepted population
9. Output p_1

Figure 1 Basic PPA

PPA, like most metaheuristics, requires customization through a set of parameters. There are parameters for which generic decisions are to be made: the size N of the initial population, the proportion $\%Good$ of plants considered to be in good spots, the number r of short runners to be generated from good parents, and the number R of long runners that need to be generated from parents in less good spots, ($r > R$). Other parameters need problem-specific decisions: solution space, cost function (fitness), the "distance"a runner, either short of long, must travel.

3.2 Applying PPA to combinatorial optimization

Given finite set X (solution space), any $x \in X$ is called a solution. The above strategy of the strawberry plant can be implemented into PPA to search for the best solution $x^* \in X$ according to objective function $f(x)$, $f(x^*) = \min_{x \in X} f(x)$. Call this combinatorial optimization problem COP.

Plants in PPA correspond to solutions for COP, and fitness to cost function $f(x)$. Changes in state, or neighbourhood moves, correspond to the concept of sending runners. The concept of goodness of spots or plants is clearly captured by the concept of fitness, whereas the distance a runner travels from its parent plant is captured by the notion of perturbation or move. A short runner is obtained from its parent by a small perturbation, while a long runner is obtained by a substantial change. The rationale, in terms of COP, is that good solutions can be obtained, either from good solutions by small perturbations, or from bad solutions but by substantial changes.

Short runners exploit the solution space in the vicinity of their parent searching for local optima. Long runners explore the solution space in order to escape from local optima. The strawberry plant strategy of survival in its environment can thus be seen as that of searching for points in the solution space which give rise to better fitness values, and hopefully ultimately to the best one.

4 Solution methodology

We begin our discussion by showing, in subsection 4.1, how an initial population of timetables is generated. Then, in subsection 4.2, we show how LS is incorporated to locally improve the generated timetables. We conclude this section by presenting the proposed algorithm in subsection 4.3.

4.1 Generating an initial population

In the context of UESP, every individual of the initial population (see Line 1 of Figure 3) is a feasible timetable. It is generated by employing a graph colouring heuristic. Recall that a feasible timetable is a legal p-colouring of the graph whose nodes represent exams, and whose edges indicate that the end-points are conflicting and should not be scheduled in the same period. Graph colouring heuristics are often able to quickly generate feasible solutions. Many such heuristics have been suggested (Largest Degree, Largest Weights Degree, Largest Enrolment).

In this work, Saturation Degree with Backtracking, a well-known and widely used heuristic, is employed (Ahandani et al., 2012; Carter et al., 1996). It consists of colouring firstly exams with smallest number of available legal colours. Ties are broken by Largest Degree First strategy. Observe that, in every iteration, the number of legal colours available for the uncoloured nodes needs to be recalculated. When no valid period is available for some exam, backtracking considers freeing exams already assigned to make room for it. The backtracking process consists of giving an ordering index after every node colouring, then determining the minimum index of nodes adjacent to the current node, and finally backtracking to this index. The process terminates when all nodes are coloured. Rules should be imposed to avoid cycling.

4.2 Hybridizing LS

The LS framework demonstrates outstanding achievements in solving combinatorial optimization problems. There are different options for LS. The option selected in this hybridization is Steepest Descent which always selects the best improvement (when it exists). The purpose of LS, in this work, is to improve every timetable generated by the algorithm, from the point of view of the cost function in (1).

The most important feature of LS is the definition of the neighbourhood structure. In this study, two structures, already used in Cheraitia and Haddadi (2016), are proposed for search.

1) ExamShifting: a neighbour of the current solution is obtained by shifting a given exam, *i.e.* by ejecting the exam from the period it belongs to, and reassigning it to another period. Clearly, this is not always possible. For exam number k to incorporate period π, it must satisfy $c_{ki} = 0$ for all $i \in \pi$, which means that it is compatible with all the exams in period π.

2) KempeMove: given a timetable, consider two arbitrary periods π_i and π_j. Let $B = (\pi_i, \pi_j, E)$ be a bipartite graph where E is a set of edges connecting any two conflicting exams. Since exams in the same period are compatible, if two exams are conflicting then one must belong to π_i and the other to π_j. Kempe chain refers to any connected component $C_i \cup C_j$ where $C_i \subset \pi_i$ and $C_j \subset \pi_j$. In this structure, a neighbour of the

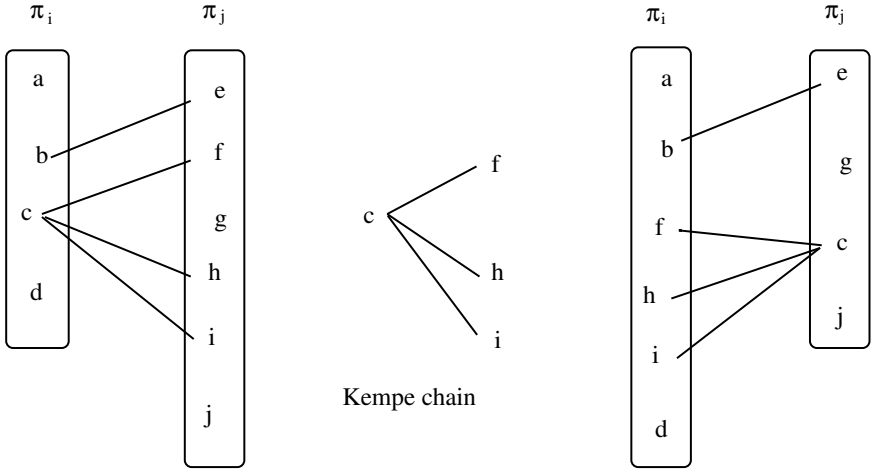
current timetable is obtained by swapping the sets C_i and C_j. An example of KempeMove is given in Figure 2 with $C_i = \{c\}, C_j = \{f, h, i\}$. It is easy to see that this move preserves feasibility.

Since the number of exams is n and the number of periods is p, the size of ExamShifting is $n \times p$. The size of KempeMove is more problematic. There are $\theta(\theta-1)/2$ manners of choosing two periods, but the number of connected components in the underlying bipartite graph is not expected to be known beforehand since it is instance-dependent. In the context of UESP, Dowsland and Thompson (2012) and Cheraitia and Haddadi (2016) conclude their computational investigation by pointing that KempeMove is the best choice.

A question that is often asked is: why do some neighbourhoods lead to good results which are often independent of the initial solution, while others lead to lesser solutions although highly correlated to the initial one? If we were to refer to the first kind of neighbourhoods as good, what are the characteristics of a good neighbourhood? What makes a structure better than another?

All of these outstanding questions are raised and addressed in an excellent reference (Lü et al., 2011). Using a curriculum-based course timetabling as a case study, and three evaluation criteria, the search capability of four neighbourhoods is investigated. The research concludes that KempeMove is one of the most powerful structures.

Since a local minimum in one neighbourhood is not necessarily a local minimum in another one, combining different neighbourhoods can be attractive. Consequently, another question raised by Lü et al. (2011), concerns the combination of neighbourhoods, and how to search them. Again, the study concludes that ExamShifting and KempeMove searched with token-ring strategy (Di Gaspero and Schaerf, 2005) constitutes a good option. Token-ring search consists of searching a neighbourhood, and when a local optimum is achieved, the search is continued with this incumbent but in another neighbourhood. These choices, recommended in Lü et al. (2011), are closely adhered to in this paper.



(a) Configuration before move

(b) Configuration after move

Figure 2 Example of KempeMove

4.3 Proposed algorithm

4.3.1 Pseudo-code

The pseudo-code of the proposed method is given in Figure 3. The termination criterion depends on the time spent by the algorithm. As explained in Section 4.2, using a graph colouring heuristic, it starts in Line 1 by generating an initial population. Every timetable generated (in Lines 9 and 14), is improved (in Lines 10 and 15) by LS. The resulting improved timetable is inserted in the population in the appropriate place (regarding its cost function) to avoid resorting the population.

```
// Initialization
 1.   Generate N distinct timetables T_1, ..., T_N
 2.   For i = 1, ..., N improve T_i // Let P = {τ_1, ..., τ_N} be the improved population
 3.   Sort P in ascending order according to objective function values
 4.   s ← %Good × N
 5.   Q ← P
// Loop
 6.   While (time < Timelimit)
 7.       For i = 1, ..., s
 8.           For j = 1, ..., r
 9.               - Generate timetable ρ_j by moving from timetable τ_i
10.               - Improve ρ_j
11.               - Q ← Q ∪ {ρ_j} // Insert timetable ρ_j in the appropriate place
12.       For i = s + 1, ..., N
13.           For j = 1, ..., R
14.               - Generate timetable ρ_j by moving from timetable τ_i
15.               - Improve ρ_j
16.               - Q ← Q ∪ {ρ_j} // Insert timetable ρ_j in the appropriate place
17.       Keep in Q only N timetables according to some acceptance criterion
18.       P ← Q  // Let P = {τ_1, ..., τ_N}
19.   End While
20.   Output τ_1
```

Figure 3 Hybrid algorithm

4.3.2 Generation of short and long runners

Short runners are implemented using a small perturbation: ExamShifting. Here only one exam is concerned. Given a feasible timetable ρ_j in Line 9 (parent plant), a new feasible timetable τ_j (short runner) is obtained from its parent by shifting a randomly chosen exam from its position in some period to another period. Obviously the move must maintain feasibility by avoiding conflicts between the chosen exam and the exams belonging to the new period it should be shifted to.

Long runners are generated using a substantially larger perturbation: PeriodSwapping. Given a feasible timetable ρ_j in Line 14, a new feasible timetable τ_j (long runner) is obtained

from its parent by randomly choosing two periods, and swapping them. In this case, many exams, those belonging to the two periods, are perturbed. It comes without saying that this move does not affect feasibility.

4.3.3 Acceptance criterion or selection

The content of this section constitutes a departure from previous versions of PPA. First, the population size is kept constant. Moreover, the original version of PPA accepts only good timetables from the top of the sorted population. This rule selects only the best to reproduce. Geneticists would argue that this rule reduces the diversity of the population. Indeed, "bad" parents can still generate "good" children (long runners) by substantial perturbations. From this observation, a more common selection rule is adopted, namely the Roulette wheel. Hence, newly generated timetables are accepted or rejected proportionally to their fitness.

5 Computational experiments

Subsection 5.1 presents the experimental design. In subsection 5.2, we describe how were set the parameters of our method. Subsection 5.3 displays the results obtained and Subsection 5.4 deals with the comparison of our method with the state-of-the-art.

5.1 Experimental design

Our method is coded in C and run on an Intel Core I3 (2.4 GHz). Tests are carried out on the University of Toronto benchmarks, whose characteristics are shown in Table 1 (see Carter et al. (1996) for more information about this dataset).

Table 1 Characteristics of the benchmark instances

	Number of exams	Number of students	Number of periods	Conflict density
CAR91	682	16,925	35	0.13
CAR92	543	18,419	32	0.14
EAR83	189	1,125	24	0.27
HEC92	81	2,823	18	0.42
KFU93	461	5,349	20	0.06
LSE91	381	2,726	18	0.06
RYE93	486	11,483	23	0.07
STA83	139	611	13	0.14
TRE92	261	4,360	23	0.18
UTA92	622	21,266	35	1.13
UTE92	184	2,750	10	0.08
YOR83	181	941	21	0.29

5.2 Parameters setting

It is well known that the efficiency of metaheuristics relies heavily on parameters setting. Unfortunately, optimally setting the parameters a priori is difficult and even impossible when they are instance-dependent. The most common way to set parameter values begins

by considering a small sample of instances; values are then assigned to the parameters, and the problem instances are solved under these settings. The parameters are adjusted accordingly for better results. These values are then adopted as default parameter values for future use. It goes without saying that finding this default values is tedious and potentially time consuming.

Following this general procedure, we obtained the following values for our parameters. To generate an execution time based on the instance to be solved, we let $Timelimit = \max\{n \times \theta, 10800\}$. This means that, in general, the code runs during $n \times \theta$ seconds. But some instances are of so small size that this limit is insufficient. In this case, 10800 seconds are provided. The size N of the population is kept constant with value 10 throughout the run. The percentage $\%Good$ is fixed to 20%. This means that the first 20% of the plants in the sorted population is considered as living in good spot. Every plant in a good spot sends $r = 3$ short runners, and every other parent plant sends $R = 1$ long runner.

5.3 Results

The computational results are provided in Table 2. Every instance is run 10 times under the same settings. The labels of the columns are self-explanatory. The average cost in the third column is obtained over 10 runs. The number of generations refers to the number of times the while loop is executed.

Table 2 Results of the hybrid algorithm on benchmark instances

	Cost function		
	Best	Mean	Number of generations
CAR91	5.22	5.29	7
CAR92	4.34	4.39	9
EAR83	33.36	34.25	70
HEC92	10.08	10.19	693
KFU93	13.45	13.71	6
LSE91	10.76	11.25	7
RYE93	8.68	8.85	6
STA83	157.03	157.03	99
TRE92	8.14	8.43	25
UTA92	3.47	5.56	9
UTE92	24.88	24.98	95
YOR83	35.39	36.29	146

5.4 Comparison with the state-of-the-art

Our algorithm is compared with thirteen recent published methods (see Table 3). The comparison is on the accuracy point of view and is recorded in Table 4, where the best values are in bold. From this perspective, it can be seen that the two best methods are due to Caramia et al. (2008) and Alzaqebah and Abdullah (2015), immediately followed by our approach, obtaining respectively 5, 4 and 2 best results. This conclusion is confirmed by another criterion. The last row of Table 4 gives the average deviation from best. From this standpoint also our method wins the third rank behind the same two methods. Pertinent questions follow:

is this conclusion mathematically justified ? Are there significant differences between the methods ?

Trying to give an answer, we take the four first ranked methods and use Friedman test. The results of the test are provided by XLSTAT (Addinsoft, 2015) in appendix (Tables 5 and 6). The test concludes with the negative. There are no significant differences between the efficiency of the four first ranked methods. Therefore, according to this test, any difference is simply due to chance.

Interestingly, the proposed hybrid method outperforms a careful implementation of simulated annealing (Cheraitia and Haddadi, 2016).

Table 3 Heuristics in competition

#	Methodology	Reference
0	Our approach	
1	Constructive approach	Abdul-Rahman et al. (2011)
2	Hybrid particle swarm optimization	Ahandani et al. (2012)
3	Memetic technique	Al-Betar et al. (2013)
4	Hybrid bee colony optimization	Alzaqebah and Abdullah (2015)
5	Linear combinations of heuristics	Burke et al. (2011)
6	Adaptive selection of heuristics	Burke et al. (2012)
7	Local-Search-Based Approaches	Caramia et al. (2008)
8	Simulated annealing	Cheraitia and Haddadi (2016)
9	Hybrid evolutionary algorithm running on a GPU	Kolonias et al. (2014)
10	Tabu search	Pais and Amaral (2011)
11	Hybridizations within a graph-based hyper-heuristic	Qu and Burke (2009)
12	Hybridizing heuristics within an estimation distribution algorithm	Qu et al. (2014)
13	Graph colouring constructive hyper-heuristic	Sabar et al. (2011)

Table 4 Comparison with the state-of-the-art

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
CAR91	5.22	5.17	5.22	4.99	4.38	5.03	5.19	6.60	4.34	5.24	5.46	**4.16**	4.95	5.14
CAR92	4.34	4.74	4.67	4.29	**3.88**	4.22	4.31	6.00	5.26	4.47	4.57	5.16	4.09	4.70
EAR83	33.36	40.91	35.74	34.42	33.34	36.06	35.79	**29.30**	34.17	34.41	33.50	35.86	34.97	37.86
HEC92	10.08	12.26	10.74	10.40	10.39	11.71	11.19	**9.20**	10.43	10.39	10.52	11.94	11.11	11.90
KFU93	13.45	15.85	14.47	13.50	**13.23**	16.02	14.51	13.80	14.36	13.77	14.05	14.79	14.09	15.30
LSE91	10.76	12.58	10.76	10.48	10.52	11.15	10.92	**9.60**	11.25	11.06	-	11.15	10.71	12.33
RYE93	8.68	10.11	9.95	8.79	8.92	9.42	-	**6.80**	9.47	8.61	9.11	-	9.20	10.71
STA83	**157.03**	158.12	157.10	157.04	157.06	158.62	157.18	158.20	157.13	157.05	157.29	158.33	157.64	160.12
TRE92	8.14	9.30	8.47	8.16	**7.89**	8.37	8.49	9.40	8.47	8.51	8.71	8.60	8.27	8.32
UTA92	3.47	3.65	3.52	3.43	**3.13**	3.37	3.44	3.50	3.56	3.63	3.71	3.42	3.33	3.88
UTE92	24.88	27.71	25.86	25.09	25.12	27.99	26.70	**24.40**	25.71	24.87	25.18	28.30	26.18	32.67
YOR83	**35.39**	43.98	38.72	35.86	35.49	39.53	39.47	36.20	36.92	37.15	39.08	40.24	37.88	40.53
	0.90	3.36	1.77	1.04	0.78	2.29	1.82	0.75	1.42	1.49	1.52	2.25	1.53	3.29

6 Conclusion

A new hybrid heuristic which combines PPA and LS is put forward. The most important concept introduced in this research is that of short and long runners, implemented by PPA mimicking the way the strawberry plant propagates. This concept parallels that of exploitation/exploration, vital for any metaheuristic. In this paper, it has been implemented to solve

the notoriously difficult UESP. The results obtained on benchmark instances show that this approach, in conjunction with simple LS but implemented with an effective strategy for searching the neighbourhoods of the solution space, produces high quality timetables. Comparison with a large number of known methods demonstrates the relative efficiency of the proposed hybrid algorithm.

A possible limitation of our approach is that, like all nature-inspired algorithms, it suffers from two drawbacks: a limited theory to underpin it and many arbitrarily set parameters.

Although it may seem surprising that PPA with LS could solve an intractable problem such as UESP, it is less so when we know that PPA has been shown to be effective in continuous global optimization (Sulaiman et al., 2014). In fact, it has already been shown to be effective in combinatorial optimization in the context of the Travelling Salesman problem (Selamoglu and Salhi, 2016) and Path Planning (Zhou et al., 2016). We do not yet have a formal proof of this. But, the success of plants and their ability to optimize their growth and disperse their offspring effectively across their environment is testament to the effectiveness of any heuristic, such as PPA, based on their propagation strategies.

Finally, because our method is population-based, a future extension of this work may investigate the parallel/distributed execution of our code in order to reduce CPU runtime.

Acknowledgement

Many thanks to the anonymous referees whose suggestions and comments contribute to better clarity and readability of the manuscript.

References

Abdul-Rahman, S., Burke, E.K., Bargiela, A., McCollum, B. and Özcan, E. (2011) 'A constructive approach to examination timetabling based on adaptive decomposition and ordering', *Annals of Operations Research*, Vol. 218 No. 1, pp. 3–21.

Abdullah, S., Turabieh, H. and McCollum, B. (2009) 'A hybridization of electromagnetic-like mechanism and great deluge for examination timetabling problems', in International Workshop on Hybrid Metaheuristics, Springer Berlin Heidelberg, pp. 60–72.

Abdullah, S., Turabieh, H., McCollum, B. and McMullan, P. (2010) 'A tabu-based memetic approach for examination timetabling problems', in International Conference on Rough Sets and Knowledge Technology, Springer Berlin Heidelberg, pp. 574–581.

Addinsoft (2015) XLSTAT V. 2015.4.01.22368 : Data analysis and statistics software for Microsoft Excel.

Ahandani, M.A., Vakil Baghmisheh, M.T., Zadeh, M.A.B. and Ghaemi, S. (2012) 'Hybrid particle swarm optimization transplanted into a hyper-heuristic structure for solving examination timetabling problem', *Swarm Evolutionary Computation*, Vol. 7, pp. 21–34.

Akyol, S. and Alatas, B. (2016) 'Plant intelligence based metaheuristic optimization algorithms', *Artificial Intelligence Review*, Vol. 47 No. 4, pp. 417–462.

Al-Betar, M.A., Khader, A.T. and Abu Doush, I. (2013) 'Memetic techniques for examination timetabling', *Annals of Operations Research*, Vol. 218 No. 1, pp. 23–50.

Alutaibi, K., Alsubaie, A., and Marti, J. (2015) 'Allocation and scheduling of firefighting units in large petrochemical complexes', in International Conference on Critical Infrastructure Protection, Springer International Publishing, pp. 263–279.

Alzaqebah, M. and Abdullah, S. (2015) 'Hybrid bee colony optimization for examination timetabling problems', *Computers and Operations Research*, Vol. 54, pp. 142–154.

Ammar, M.H., Benaissa, M. and Chebchoub, H. (2014) 'Seafaring staff scheduling', *International Journal of Services and Operations Management*, Vol. 19 No. 2, pp. 229–249.

Arogundade, O.T., Akinwale, A.T. and Aweda, O.M. (2010) 'A genetic algorithm approach for a real-world university examination timetabling problem', *International Journal of Computer Applications*, Vol. 12 No. 5, pp. 1–4.

Azadeh, A., Faghihroohi, S. and Izadbakhsh, H.R. (2012) 'Optimisation of train scheduling in complex railways with imprecise and ambiguous input data by an improved integrated model', *International Journal of Services and Operations Management*, Vol. 13 No. 3, pp. 310–328.

Brabazon, A., O'Neill, M. and McGarraghy, S. (2015) 'Plant-inspired algorithms', in Natural Computing Algorithms, Springer Nature, pp. 455–477.

Bullnheimer, B. (1998) 'An examination scheduling model to maximize students' study time', in International Conference on the Practice and Theory of Automated Timetabling (PATAT 1997), Springer Berlin Heidelberg, pp. 78–91.

Burke, E.K., Pham, N., Qu, R. and Yellen, J. (2011) 'Linear combinations of heuristics for examination timetabling', *Annals of Operations Research*, Vol. 194 No. 1, pp. 89–109.

Burke, E.K., Qu, R. and Soghier, A. (2012) 'Adaptive selection of heuristics for improving exam timetables', *Annals of Operations Research*, Vol. 218 No. 1, pp. 129–145.

Caramia, M., Dell'Olmo, P. and Italiano, G.F. (2008) 'Novel local-search-based approaches to university examination timetabling', *INFORMS Journal on Computing*, Vol. 20 No. 1, pp. 86–99.

Carter, M.W., Laporte, G. and Lee, S.Y. (1996) 'Examination timetabling: algorithmic strategies and applications', *Journal of the Operational Research Society*, Vol. 47 No. 3, pp. 373–383.

Cheraitia, M. and Haddadi, S. (2016) 'Simulated annealing for the uncapacitated exam scheduling problem', *International Journal of Metaheuristics*, Vol. 5 No. 2, pp. 156–170.

Di Gaspero, L. and Schaerf, A. (2005) 'Neighborhood portfolio approach for local search applied to timetabling problems', *Journal of Mathematical Modeling and Applications*, Vol. 5 No. 1, pp. 65–89.

Dowsland, K.A. and Thompson, J.M. (2012) 'Simulated annealing', in Rozenberg, G. et al (Eds.), Handbook of Natural Computing, Springer, Berlin, pp. 1623–1655.

Eley, M. (2006) 'Some experiments with ant colony algorithms for the exam timetabling problem', in Ant Colony Optimization and Swarm Intelligence, Springer Nature, pp. 492–499.

Jittamai, P. and Kangwansura, T. (2016) 'A hospital admission planning model for operating room allocation under uncertain demand requirements', *International Journal of Services and Operations Management*, Vol. 23 No. 2, pp. 235–256.

Kaviani, M., Shirouyehzad, H., Sajadi, S.M. and Salehi, M. (2014) 'A heuristic algorithm for the university course timetabling problems by considering measure index: a case study', *International Journal of Services and Operations Management*, Vol. 18 No. 1, pp. 1–20.

Kolonias, V., Goulas, G., Gogos, C., Alefragis, P. and Housos, E. (2014) 'Solving the Examination Timetabling Problem in GPUs', *Algorithms*, Vol. 7 No. 3, pp. 295–327.

Lee, S.Y., Kim, Y.D. and Lee, H.J. (2009) 'Heuristic algorithm for a military training timetabling problem', *Asia Pacific Management Review*, Vol. 14 No. 3, pp. 289–299.

Lü, Z., Hao, J.K. and Glover, F. (2011) 'Neighborhood analysis: a case study on curriculum-based course timetabling', *Journal of Heuristics*, Vol. 17 No. 2, pp. 97–118.

MirHassani, S.A. (2006) 'Improving paper spread in examination timetables using integer programming', *Applied Mathematics and Computation*, Vol. 179 No. 2, pp. 702–706.

Nasiri, M.M. and Rahvar, M. (2017) 'A two-step multi-objective mathematical model for nurse scheduling problem considering nurse preferences and consecutive shifts', *International Journal of Services and Operations Management*, Vol. 27 No. 1, pp. 832–101.

Pais, T.C. and Amaral, P. (2011) 'Managing the tabu list length using a fuzzy inference system: an application to examination timetabling', *Annals of Operations Research*, Vol. 194 No. 1, pp. 341–363.

Qu, R. and Burke, E.K. (2009) 'Hybridisations within a graph-based hyper-heuristic framework for university timetabling problems', *Journal of the Operational Research Society*, Vol. 60 No. 9, pp. 1273–1285.

Qu, R., Burke, E.K., McCollum, B., Merlot, L.T.G. and Lee, S.Y. (2008) 'A survey of search methodologies and automated system development for examination timetabling', *Journal of Scheduling*, Vol. 12 No. 1, pp. 55–89.

Qu, R., Pham, N., Bai, R. and Kendall, G. (2014) 'Hybridising heuristics within an estimation distribution algorithm for examination timetabling', *Applied Intelligence*, Vol. 42 No. 4, pp. 679–693.

Sabar, N.R., Ayob, M., Qu, R. and Kendall, G. (2011) 'A graph colouring constructive hyper-heuristic for examination timetabling problems', *Applied Intelligence*, Vol. 37 No. 1, pp. 1–11.

Salhi, A. and Fraga, E.S. (2011) 'Nature-inspired optimisation approaches and the new plant propagation algorithm', Proceedings of the International Conference on Numerical Analysis and Optimization, pp. K2.1–K2.8.

Selamoglu, B.I. and Salhi, A. (2016) 'The plant propagation algorithm for discrete optimisation: the case of the travelling salesman problem', in Nature-Inspired Computation in Engineering, Springer Nature, pp 43–61.

Sulaiman, B.I., Salhi, A., Selamoglu, B.I. and Kirikchi, O.B. (2014) 'A plant propagation algorithm for constrained engineering optimisation problems', *Mathematical Problems in Engineering*, Vol. 2014, 10 pages.

Thompson, J.M. and Dowsland, K.A. (1998) 'A robust simulated annealing based examination timetabling system', *Computers and Operations Research*, Vol. 25 No. 7-8, pp. 637–648.

Zhou, Y., Wang, Y., Chen, X., Zhang, L. and Wu, K. (2016) 'A novel path planning algorithm based on plant growth mechanism', *Soft Computing*, Vol. 21 No. 2, pp. 435–445.

Appendix

A Results of Friedman test (Comparison of k samples)

Test interpretation (as provided by XLSTAT 2015.4.01.22368):

- Null hypothesis: samples come from the same population;
- Alternative hypothesis: samples come from different populations.

Since the p-value is greater than the statistical signification level $\alpha = 0.05$, we cannot reject the null hypothesis (ties have been detected and appropriate corrections applied).

Table 5 Descriptive statistics

Method	Size	Minimum	Maximum	Mean	Standard deviation
Caramia et al (7)	12	3.500	158.200	26.083	42.875
Alzaqebah and Abdullah (4)	12	3.130	157.060	26.113	42.712
Our method (0)	12	3.470	157.030	26.233	42.622
Al-Betar et al. (3)	12	3.430	157.040	26.371	42.657

Table 6 Friedman test

Q (Observed value)	1.900
Q (Critical value)	7.815
DDL	3
p-value (bilateral)	0.593
α	0.05