An Adaptive Flex-Deluge Approach to University Exam Timetabling

Edmund K. Burke
School of Electronic Engineering and Computer Science,
Queen Mary University of London, Mile End Road, London, E1 4NS, United Kingdom
e.burke@qmul.ac.uk
Yuri Bykov
The University of Nottingham, Business School,
Jubilee Campus, Wollaton Road, Nottingham, NG8 1BB
mail@yuribykov.com

This paper presents a new methodology for university exam timetabling problems, which draws upon earlier work on the Great Deluge metaheuristic. The new method introduces a “flexible” acceptance condition. Even a simple variant of this technique (with fixed flexibility) outperforms the original Great Deluge algorithm. Moreover, it enables a run-time adaptation of an acceptance condition for each particular move. We investigate the adaptive mechanism where the algorithm accepts the movement of exams in a way that is dependent upon the difficulty of assigning that exam. The overall motivation is to encourage the exploration of a wider region of the search space. We present an analysis of the results of our tests of this technique on two international collections of benchmark exam timetabling problems. We show that 9 out of 16 solutions in the first collection and 11 out of 12 solutions in the second collection produced by our technique have a higher level of quality than previously published methodologies.

Keywords: metaheuristics, optimisation, analysis of algorithms, education systems: timetabling.

1. Introduction

The overall goal in university exam timetabling is to assign all given exams into a limited number of timeslots (and often into available rooms) so that no one student has to sit two (or more) exams at the same time in addition to satisfying a number of other constraints. For example, if room assignment is required, then the number of students seating in a room should not exceed its capacity. The constraints, which have to be satisfied under any circumstances are called *hard*. Constraints, which are desirable but not essential are called *soft*. Timetables which satisfy all the hard constraints are called *feasible*. In most situations it is not possible to satisfy all soft constraints. Therefore, the goal is to minimise the amount of violation penalty. The soft constraints can be quite varied across different universities (Burke et al, 1996). The most common soft requirement is that students should have an appropriate amount of time between exams. However, university administrators will usually express additional preferences, such as “larger exams should be scheduled earlier” or “avoid exams with different durations in the same room”.

Examination timetabling has been very widely studied over the last decades and a broad range of techniques from across Operations Research and Artificial Intelligence have been investigated and adapted for the problem. A selection of the methods which have been explored for this important problem includes: Tabu Search (Di Gaspero and Schaerf, 2001), (White and Xie, 2001), Greedy Randomised Adaptive Search Procedure (Casey & Thomson, 2003), Simulated Annealing (Thompson and Dowsland, 1996a), (Thompson and Dowsland, 1996b), Ahuja-Orlin’s large neighbourhood search (Abdullah et al, 2007), (Meyers & Orlin, 2007), Adaptive ordering (Burke and Newall, 2004), (Abdul Rahman et al, 2014), Case Base Reasoning (Burke et al, 2006), (Perovic et al, 2007), Cell biology developmental approach (Pillay, 2009), Evolutionary methods (Erben, 2001), (Ulker et al, 2007), (Al-Betar et al, 2014), Multi-criteria methods (Paquete and Fonseca, 2001), (Petrovic and Bykov, 2003), (Bykov, 2003), (Cheong et al, 2009), Constraint Based Methods (David, 1998), Fuzzy methods (Petrovic et al, 2005), (Asmuni et al, 2009) and multi-processor (grid) computing (Gogos et al, 2010). In addition, there have been different variants of hybrid techniques (Merlot et al, 2003), (Caramia et al, 2008), (Burke et al, 2010), (Abdullah and Alzaqebah, 2014) and hyper-heuristic methods (Burke et al, 2007), (Pillay, 2016) and Late Acceptance Hill Climbing (Ozcan et al, 2009).

Early examination timetabling methods were reviewed in (Carter, 1986) and a follow up paper was published by (Carter and Laporte, 1996). A more recent survey of examination timetabling techniques is presented in (Qu et al, 2009). Other papers, which review and overview the timetabling literature include (de Werra, 1985), (Bardadym, 1996), (Burke et al, 1997), (Schaerf, 1999), (Burke and Petrovic, 2002), (Petrovic and Burke, 2004), (McCollum, 2007) and (Lewis, 2008).

In the last decade, a particular interest in exam timetabling has focused on the Great Deluge algorithm (GDA), introduced by Dueck (1993). The method operates with a control parameter, called a “*level*”, which represents the upper bound of an acceptable cost function. At the beginning of the search, the level is equal to an initial cost function and with every step it is lowered by some chosen decay rate. At each iteration, a candidate solution is accepted, if its cost function is lower than the level or lower than the current cost. The process is continued until no further improvement is detected for an appropriate amount of time.

The Great Deluge algorithm was investigated and adapted for exam timetabling problems in (Bykov, 2003) and (Burke et al, 2004a). Several multiobjective formulations of the original method have also been explored in (Petrovic and Bykov, 2003) and (Bykov, 2003). The method produced some of the best results in the literature (at the time of publication) on a range of exam timetabling benchmark instances (Burke et al, 2004a). It was also applied to course timetabling in (Burke et al, 2003). In the same year, this algorithm won 3rd place in the first International Timetabling Competition (ITC2002), which had an additional impact on the growth of its popularity. Since that time, a number of modifications and adaptations of the GDA have been proposed for timetabling problems. For example, (Burke and Newall 2003) considered starting the algorithm by hybridizing with an adaptive approach. Obit et al (2009) and Jaddi and Abdullah (2014) investigated non-linear level lowering. An adaptive reheating-like mechanism was suggested by (McMullan 2007). In addition, the GDA was hybridized with other methods (Turabeih and Abdullah 2011), (Fong et al, 2014) and widely used in hyper-heuristic studies (Ozcan et al 2010), (Sin and Kham 2012). The high practical effectiveness of the GDA was confirmed in the second International Timetabling Competition (ITC2007) where the GDA-based entry methods won the 1st place prizes simultaneously in two (over three) tracks: in the examination timetabling track and in the curriculum based course timetabling track.

The properties of GDA, which provide its high level of practical effectiveness were highlighted in (Burke et al 2004a). In addition to the simplicity in implementation, this algorithm represents one of the most transparent techniques in respect of parameter setting. Its single parameter (decay rate) is problem-independent and has a clear physical meaning (dimensionality): it expresses the amount of cost reduction per iteration. Thus, the decay rate could be thought of as a "search speed". Correspondingly, knowing the initial and the final costs, the search speed can be easily calculated in order to pass the given cost interval in a required number of iterations. Taking into account that GDA has a very clear convergence and proposing an approximate final cost, the complete search procedure can easily fit into a required amount of computing time. The simplicity of the search time management is important for practical applications (including competition entries) where the available computing time is limited and should be utilized effectively. It is known that for many problems (especially for larger-sized ones) the prolongation of the search procedure can lead to better results (see Burke et al 2004a). However, such a prolongation should be controllable in order to avoid situations where the available time is elapsed before the search achieves the best possible result. In practical exam timetabling (which is often not a time critical problem) proper time management could provide a significant improvement in the quality of solution by employing more computational time where appropriate. Here it is perfectly feasible to leave an algorithm running for an extended period of time (say, overnight or even for a weekend).

The above reasoning suggests that a major practical advantage of GDA is its high reliability in respect of parameterization, i.e. its performance is robust in terms of incorrect parameter setting. Correspondingly, when proposing further modifications and adaptations of GDA, it is important to keep its practical effectiveness, i.e. not to weaken the discussed advantages.

2. A Flex-Deluge Algorithm

2.1. The flexible acceptance condition in GDA

The significance of preserving the distinct GDA advantages (overviewed in the previous section) has motivated us into developing an alternative conception of the improvement of this technique. Its essence is that the original GDA shape of the level lowering is kept intact, while the *increase in the search efficiency is achieved by employing a more advanced acceptance condition*. This idea was first introduced in an abstract at the PATAT conference in (Burke and Bykov 2006). In the current study, we expand this idea and present its detailed analysis.

The research development of the addressed acceptance condition is based on the following assumptions: Great Deluge algorithm (as with many other types of search method) accepts all downhill (penalty improving in the context of minimization) moves and a limited number of uphill (penalty worsening) ones. The search without uphill moves (greedy Hill-Climbing) has a very weak level of performance. So, the mechanism for accepting uphill moves underpins the algorithm's strength. Also, the quality of the results can be improved by the slowing of the downhill motion of a prospective search. Combining these two observations, it can be proposed that the algorithm's performance might be increased when the uphill motion of a prospective search is also slowed.

In order to explore the above idea, we introduce a "*flexibility*" coefficient k_f ($0 \le k_f \le 1$). Now, at every iteration, the algorithm accepts a new candidate solution with penalty C' if it satisfies Inequality (1):

$$\begin{aligned} C' &\le C + k_f(B - C) && \text{when } C < B \\ C' &\le C && \text{when } C \ge B \end{aligned} \qquad (1)$$

where C and B are the current penalty and the current value of the level of the Great Deluge algorithm, respectively. According to this formula, the increase in the penalty of the accepted candidate should not be greater than the difference between C and B multiplied by k_f .

This mechanism works in the following way. Although B is the absolute upper limit for the penalty value of all solutions, the upper limit for the current candidate is always lower than B. When this uphill move is accepted, the current upper limit becomes higher (closer to B) and so on. Thus, the penalty for a current solution cannot exceed B and can approach it only by several uphill moves. We now introduce the idea that the lowering of B forces the lowering of the current penalty using some kind of "flexible bumper" and k_f denotes the degree of its flexibility. Following this reasoning, we have called this method the Flex-Deluge algorithm (FDA).

When $k_f = 0$ (the inflexible case), the algorithm degenerates into the greedy Hill Climbing (HC) method. "*Infinite* flexibility" ($k_f = 1$) corresponds to the original Great Deluge. With the flexibility defined in the middle of this interval we have an algorithm with *moderate* characteristics, which lies inbetween the extremes. It is not so "greedy" as HC but not so "generous" as GDA. This property of FDA is reminiscent of the Peckish strategy proposed by Corne and Ross (1996). Their technique is the *intermediate* between greedy Hill-Climbing and Random Ordering. Although the purpose, implementation and performance of our method and the Peckish approach is different, both of them employ a parameter, which provides a smooth transformation of one heuristic into another and enables the setting up of a preferable level of “greediness” for the algorithm.

It could be argued that, on the one hand, HC does not enable *escape* from local minima. In contrast, the original GDA supports the ability to escape from anywhere, irrespective of whether it is a local minimum or a promising region of the search space. The Flex-Deluge method enables the escaping, but it forces a prospective solution to stay, for some time, near the achieved minimum to explore its surroundings in the search space. In his paper (Dueck 1993), Dueck associated the Great Deluge algorithm with walking on the water’s edge during a deluge in order to finally find the top of the highest hill. Using the flexible variant, one tries to "walk" a little higher than the edge. As a result, when reaching the top of the hill, there is still the chance to walk down and check whether there is a higher top nearby.

Note, that when introducing the flexibility, we keep all the properties of the original Great Deluge, i.e. in the proposed technique, the decay rate, ΔB, also corresponds to the search speed and the computational time of the search procedure can be predefined in the same way. All other details (initialisation, termination) are the same as in the original version (Burke et al, 2004a). The pseudo-code for the Flex-Deluge approach can be outlined in Figure 1.

```
Produce an initial solution $s$
Calculate initial cost function $C(s)$ and initial level $B:=C(s)$
Specify decay rate $\Delta B:=?$
Specify the flexibility $k_f:=?$
While further improvement is impossible
   Construct a candidate solution $s*$
   Calculate $C(s*)$
   If $C(s) \geq B$
   Then If $C(s*) \leq C(s)$
        Then accept the candidate $s:= s*$
   Else If $C(s*) \leq C(s)+k_f(B-C(s))$
        Then accept the candidate $s:= s*$
   Lower the level $B:= B-\Delta B$
```

Figure 1: Pseudo code for FDA, the Flex-Deluge Algorithm

Obviously, the introduction of a new parameter (k_f) in FDA requires some additional effort on its tuning. However, we expect that the benefit from the employment of this parameter could prevail over the nuisance of extra parameterisation. First of all, the specification of a proper value for k_f could be relatively straightforward. Factually, the flexibility coefficient can have any value from the specified interval with just one restriction: it should not be assigned extremely close to 0 (in this case the greedy behaviour could dominate over the performance of FDA). Our experiments below demonstrate that starting from some reasonable point (for example, from 0.005) any arbitrary value of k_f for FDA provides a better performance when compared to the original Great Deluge. Of course, the best (optimal) setting of the new parameter is problem-dependent but, *even a non-optimal setting still improves the performance of the initial algorithm.* Moreover, some extra effort spent on adjusting the best value of k_f can yield a relatively significant increase in the overall effectiveness of the search. It should be noted that the additional time expense (if it leads to better results) is tolerable for tasks such as practical exam timetabling, where the quality of results is a primary goal and the computing time is not so critical.

2.2. The integration of a self-adaptive technique into FDA

In the previous section, we have proposed that the overall performance of a search procedure can be improved by fixing k_f at some particular value (between two extremes). However, we see a further advantage behind the introduction of the flexibility coefficient which is that this coefficient can be varied during the search. I.e. instead of selecting the best *fixed value* of k_f we should develop the best *variation rule* for the flexibility. For example, it might be beneficial to perform the first phase of the search close to Hill Climbing, but the second phase close to Great Deluge (or vice versa). The number (gradation) of the phases can be much higher, up to the case where each iteration is performed with its own value of flexibility. It is important to note that, in all cases, a variation of k_f does not affect (similar to the fixed-flexibility version) the major properties of the Great Deluge algorithm, and correspondingly, there are no formal restrictions on a suitable variation method. Now, any idea of the automatic variation of k_f can be thought of as the implementation of an additional *heuristic* integrated into an *adaptive* metaheuristic search. With this in mind, the proposed technique emerges as an Adaptive Flex-Deluge Algorithm (AFDA), which can play the role of a universal engine for testing and employing various self-adaptive heuristics.

Of course, the development of a variation rule, which provides further improvement on the performance of the search (compared to the fixed flexibility variant) represents a greater challenge than just setting up the fixed value of a parameter. This requires a particular expertise in the algorithmic behaviour in conjunction with the properties of specific problems. In addition, some helpful ideas here can be borrowed from the area of self-adaptive methods, which represent a subject of significant study across different optimization problems. In the examination timetabling literature there are several methodologies, which automatically adjust an algorithm to the properties of a particular instance. Examples of such studies can be found in (White & Xie, 2001), (Casey & Thomson, 2003), (Burke and Newall, 2004), (Burke et al, 2007). By drawing upon this body of work, we suggest to use the variation of k_f to adapt the search procedure to instance-specific properties.

In this paper, we present an example of the self-adaptive heuristic for exam timetabling, which can be integrated into the Flex-Deluge algorithm. This heuristic can be thought of as an "*activity-based*" method by drawing upon the idea of White and Xie (2001), who ranked the exams as being "*light*" and "*heavy*", based upon their *activity*. It was suggested that the movement of *overactive* exams could be suspended in order to give more consideration to less *active* exams.

The above idea is reasonable in situations where a search procedure demonstrates different behaviours in respect of different moves. In most real-world timetabling problems, the moving of some (heavy) exams more frequently yields infeasibility and/or induces a higher average increase in the cost function than it does when other (light) exams are moved. Therefore, the light moves are more frequently accepted (with the same acceptance condition for all moves). In this case, the search can be thought of as being biased towards seeking an improved solution in respect of a limited number of variables (exams) but that it might miss the opportunity to explore significant parts of the search space thereby never finding even better solutions.

This situation is illustrated in Figure 2, where a vertical axis represents the cost function and the current cost is shown by the dotted line. Possible moves are shown by double vertical arrows. Of course, the current cost function can be either increased or decreased by the current move. We can see that for heavy moves the increase/decrease can be much higher than for light ones.

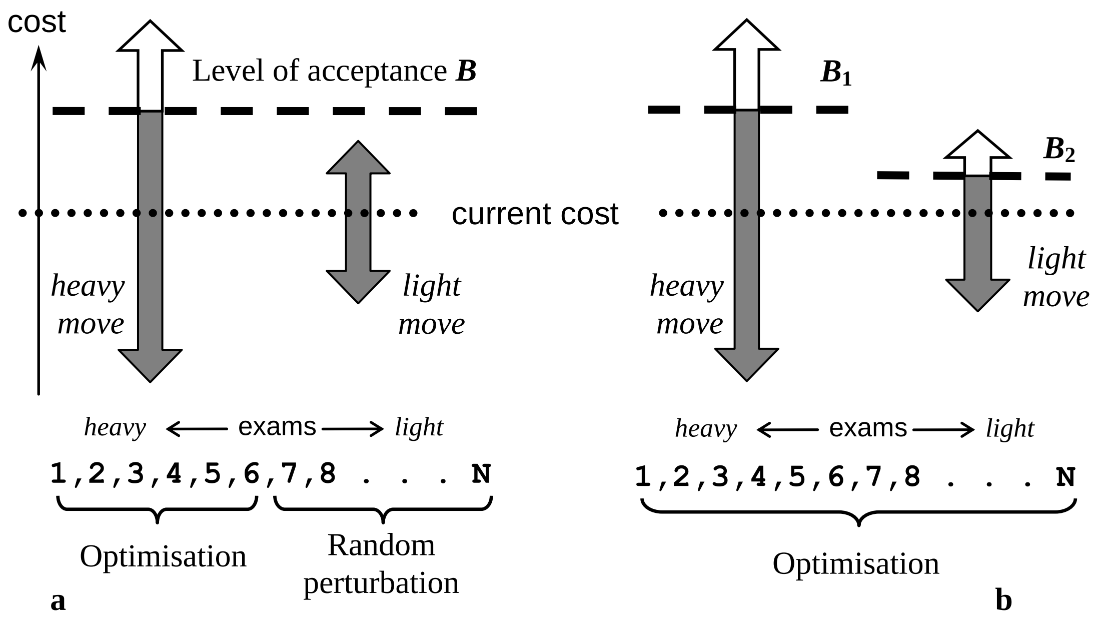

Figure 2: Heavy and light moves accepted with the same (a) and different (b) acceptance conditions.

If we have the same level of acceptance *B* for both heavy and light moves (Figure 2(a)), we impose the restriction (represented by the unshaded part of the double arrow) to heavy moves only while accepting all light moves. This can cause a biasing tendency, i.e. a complete set of exams is divided into two subsets where intelligent computational search is carried out with the first subset only, while the second subset is just randomly perturbed. It is proposed that the procedure can be improved by adding a mechanism, which limits this tendency.

To build this mechanism into the flexible algorithm introduced here, we suggest the use of different values of k_f when moving exams with a different activity. It may be beneficial to accept the most active exams with less flexibility and vice versa. Thus, in Figure 2(b), the acceptance level for heavy exams B_1 is higher than for light exams B_2. Now, the restriction affects all moves (heavy and light ones) and the search process involves the complete set of exams.

Of course, it could be expected that the performance of the activity-based AFDA heuristic outlined here is dependent on the properties of a particular type of move, which is a matter of implementation and is dependent on the particular formulation of the exam timetabling problem. Therefore, to demonstrate the universality of the proposed approach, this paper presents two examples of its application to exam timetabling problems of different types.

3. The application of FDA to exam timetabling

To evaluate the methodologies presented in this paper we use the university exam timetabling benchmark datasets from two collections where this problem is formulated in different ways. The first one is the university of Toronto collection. This is available publicly on the web at http://www.cs.nott.ac.uk/~rxq/data.htm. These problems were studied in a number of papers, starting with (Carter et al, 1996) who originated these datasets and published the first results (produced by different graph colouring heuristics with backtracking). The second collection was launched at the Examination Timetabling Track of the 2nd International Timetabling Competition (ITC2007) and is available at http://www.cs.qub.ac.uk/itc2007/. Two papers that have addressed these problems were published by the methods that came in 1st and 2nd place (Muller, 2008), (Gogos et al, 2008). A further study of these datasets was presented in (McCollum et al, 2009) and (Gogos et al 2010). We compare our results against the other methods in the literature which tackle these instances later on in the paper.

3.1 Application to Toronto benchmark problems

The university of Toronto collection contains real-world exam timetabling datasets. It has recently been adapted and corrected (see (Qu et al, 2009)). Different problem instances had circulated (under the same name) over the years and this had generated some confusion in the academic literature. The different versions have now been re-named and the situation is discussed and clarified in (Qu et al, 2009). Five of the problems had two different versions. Both versions were given distinct names in (Qu et al, 2009) and the suggested notation is also employed here. A more detailed discussion of this situation and a more detailed description of the problem can be seen in (Qu et al, 2009). The new collection contains 18 real-world university exam timetabling problems. Their identifiers and characteristics are given in Table 1.

Table 1: Benchmark problems from the university of Toronto collection

Dataset	Exams	Students	Student's exams	Density	Timeslots
Car-f-92	543	18419	55522	0.14	32
Car-s-91	682	16925	56877	0.13	35
Ear-f-83I	190	1125	8109	0.27	24
Ear-f-83IIc	189	1108	8092	0.27	24
Hec-s-92I	81	2823	10632	0.42	18
Hec-s-92II	80	2823	10625	0.42	18
Kfu-s-93	461	5349	25113	0.06	20
Lse-f-91	381	2726	10918	0.06	18
Pur-s-93	2419	30032	120681	0.03	42
Rye-f-92	486	11483	45051	0.07	23
Sta-f-83I	139	611	5751	0.14	13
Sta-f-83IIc	138	549	5689	0.14	35
Tre-s-92	261	4360	14901	0.06	23
Uta-s-92I	622	21266	58979	0.13	35
Uta-s-92II	638	21329	59144	0.13	35
Ute-s-92	184	2749	11793	0.09	10
Yor-f-83I	181	941	6034	0.29	21
Yor-f-83IIc	180	919	6012	0.29	21

The specification of the hard and soft constraints that are employed in this scenario can be presented as follows. The solution is feasible when no conflicting exams (having common students) are assigned to the same timeslot (a room allocation is not required here). The goal is to minimize the "proximity cost" C (penalty). For a problem with N exams and S students the penalty C is calculated as a double sum of all pairs of exams expressed by Formula (2):

$$C = \frac{\sum_{i=1}^{N-1} \sum_{j=i+1}^{N} s_{ij} p_{ij}}{S} \tag{2}$$

where N is the total number of exams, s_{ij} is the number of students having both exam i and exam j and p_{ij} is the proximity coefficient of the pair of exams (i, j), which depends on their assignment to timeslots. If two exams are allocated into neighbouring timeslots, then the proximity coefficient is equal to 16. When they are separated by one timeslot, the proximity coefficient has value 8. If there are two timeslots between them then $p_{ij} = 4$, for three timeslots the value is 2 and for a gap of four timeslots $p_{ij} = 1$. In all other cases $p_{ij} = 0$. Finally, the sum is divided by the total number of students S.

The experiments with the University of Toronto datasets were run using our exam timetabling search algorithm described in (Burke et al, 2004a). Here all algorithmic details (initialisation, stopping criteria, etc.) are not altered except for the following two additions.

- The algorithm of Burke et al. (2004a) performed only one type of move: the replacement of a random exam into a new randomly chosen timeslot. If this move caused an infeasible solution then it was rejected. The new algorithm does not reject such a move immediately but uses the Kempe chain procedure (as suggested for exam timetabling problems by Thompson and Dowsland (1996)), to replace the exam's neighbours in order to obtain a feasible solution.

- Our algorithm also performs the swapping of two randomly chosen timeslots (together with all their exams) by drawing upon the work of Di Gaspero (2002). At each iteration, the type of move is chosen randomly. However, the algorithm tries to keep the average timeslot swapping to 20% of all the moves.

The activity-based AFDA heuristic proposed in this paper is applied to the first type of moves: the placement of an exam into a different timeslot. It could be observed that here the activity of an exam is highly dependent on the number of exams having common students (i.e. the *degree* of exam in terms of graph colouring (see Burke et al, 2004b)). Obviously, the exams with the largest degree (LD) are often the most difficult to move (the heaviest) ones and vice versa, the lightest exams are often those, which have the lowest degree.

To implement the above idea (AFDA/LD adaptive heuristic), we define an appropriate interval of the variation of k_f, where the upper bound corresponds to exam(s) with the largest degree and the lower bound to exam(s) with the smallest degree. The remaining flexibilities are distributed inside this interval. The maximum possible interval is (0,1). However, we have found (see experiments below) that a smaller interval can provide a better overall performance of the search. We suggest the assignment of a lower bound very close to 0 in order to guarantee the absence of biasing tendencies. Thus, only the upper bound is considered as the input parameter of this algorithm. In our approach, the flexibility coefficients for all exams k_{fi} ($i = 1 \ldots N$) are calculated by formula (3)

$$k_{f\,i} = k_f^{\max} \frac{d_i}{d_{\max}} \tag{3}$$

where k_f^{max} is an upper bound of the variation, d_i is the degree of ith exam and $d_{\max}$ is the maximum degree. In this mechanism, the acceptance condition for the lightest exams is very close to that for the Hill Climbing rule. However, k_f cannot be equal to 0 as the placement of exams without common students does not represent a key step and they are excluded from the search.

Note that the AFDA/LD heuristic described above is not applicable to the "timeslot swapping" moves because we cannot associate a notion of degree with a timeslot. Therefore, the adaptation is not used here and these moves are accepted with the fixed flexibility equal to k_f^{max}. The pseudo code for the final Self-Adaptive Flex-Deluge Algorithm is presented as follows:

```
Produce an initial solution $s$
Calculate initial cost function $C(s)$ and initial level $B:=C(s)$
Calculate degree of all exams $D_i$ and their maximum $D_{max}$
Specify decay rate $\Delta B:=?$
Specify the upper bound of flexibility interval $k_f^{max}:=?$
While further improvement is impossible
   Select randomly a type of move
{exam replacing move}
   Select randomly exam $i$ and new timeslot $j$
   Construct a candidate solution $s*$
   If $s*$ is infeasible
   Then If Kempe chain does not provide feasibility
        Then reject $s*$ and return
   Set $k_f := k_f^{max} * D_i / D_{max}$
{timeslot swapping move}
   Select randomly two timeslots $j_1$ and $j_2$
   Construct a candidate solution $s*$
   Set $k_f = k_f^{max}$
{for all types of moves}
   Calculate $C(s*)$
   If $C(s) \geq B$
   Then If $C(s*) \leq C(s)$
        Then accept the candidate $s := s*$
   Else If $C(s*) \leq C(s) + k_f(B - C(s))$
        Then accept the candidate $s := s*$
   Lower the level $B := B - \Delta B$
```

Figure 3: Pseudo code of AFDA/LD for exam timetabling problem

3.2 Application to the ITC2007 benchmark problems

One of the purposes of the alternative ITC2007 (McCollum et al, 2010) collection was to "create better understanding between researchers and practitioners by allowing emerging techniques to be developed and tested on real-world models of timetabling problems". This was provided by EventMAP Ltd. and contains twelve datasets taken from their client Institutions. The problem instance characteristics are given in Table 2.

Table 2: Benchmark problems from the ITC2007 collection

Dataset	Exams	Students	Student's exams	Density	Timeslots	Rooms
Exam_1	607	7891	32380	0.05	54	7
Exam_2	870	12743	37379	0.012	40	49
Exam_3	934	16439	61150	0.026	36	48
Exam_4	273	5045	21740	0.15	21	1
Exam_5	1018	9253	34196	0.0087	42	3
Exam_6	242	7909	18466	0.062	16	8
Exam_7	1096	14676	45493	0.019	80	15
Exam_8	598	7718	31374	0.046	80	8
Exam_9	169	655	2532	0.078	25	3
Exam_10	214	1577	7853	0.05	32	48
Exam_11	934	16439	61150	0.026	26	40
Exam_12	78	1653	3685	0.18	12	50

These instances have a more complex specification than those of the Toronto collection. Firstly, in addition to the assignment of exams to timeslots, they require the allocation of exams into available rooms. Secondly, the timeslots are associated with particular dates and time intervals. Correspondingly, these problems impose a wider variety of hard constraints, such as:

- An exam can be placed only in those timeslots where the duration is not less than the duration of the exam.
- The total number of students taking all exams scheduled in a room should not exceed the capacity of the room.
- Some exams can require an exclusive room usage.
- Some pairs of exams can be coincident (must be allocated to the same timeslot) or exclusive (must not be allocated into the same timeslot) or consecutive (one exam should be scheduled before/after another).

In addition to this, the cost function is composed of a higher number of soft constraints (including room-related components). The following situations can be penalized:

- A student sits two exams in a row in one day or just two exams in the same day or two exams within a given time interval.

- The largest exams (i.e. those having the most number of students) are scheduled in later periods.
- Exams of different durations are allocated into the same room.
- The scheduling of exams into particular periods or rooms can be inappropriate in some situations.

The complete specification of different hard and soft constraints for such a problem can be found in (McCollum et al, 2010) or on the original ITC2007 web site.

To show the wider applicability and generality of the proposed approach we have also applied it to 12 datasets from the ITC2007 collection. For this purpose, our examination timetabling algorithm was somewhat adapted and extended.

When adapting our software for these problems we tried to make minimum alterations. However, the modification of some parts was unavoidable. Firstly, the procedures which check feasibility and evaluate a cost function were revised and enhanced in order to take into account the new constraints. Secondly, the initialisation procedure was also adapted. Similar to (Burke et al, 2004b), it employs the "Saturation Degree" sequencing heuristic, but the "degree" here is calculated as the number of available rooms in all available timeslots. In addition to this, special preferences are given to pairs of coincident exams, which are considered as a single exam and to pairs of consecutive exams, which are allocated in the required order. Thirdly, considerable modification was undertaken in respect of moves.

- A move which places an exam into a new timeslot is still in use but now we additionally randomly select its new room.
- When applying a Kempe chain procedure, we also have to select new rooms for all replaced exams. However, it was found that the random choice of a set of new rooms quite often leads to infeasibility. Therefore, we select new rooms deterministically using a simple bin packing heuristic. Namely, before replacing an exam we compare the number of remaining empty seats in all rooms. Correspondingly, the available room with the minimum number of empty seats is chosen for this exam.

In addition, two new types of moves were added.

- When a selected exam is coincident with another one, we have to move two coincident exams together into the same timeslot. New rooms for both exams are chosen randomly. We can assume that coincident exams share their conflicts (if one of them cannot be moved into some timeslot, then another one also cannot be moved there). Therefore, as coincident exams have the same degree, they have the same flexibility coefficient, which is used for the acceptance of this move.

- To reduce room-related penalties, our algorithm employs a simple "*room move*". Here a randomly chosen exam is reallocated into a different (randomly chosen) room without changing the timeslot. Being implemented in our software, these moves have appeared to be computationally inexpensive (the penalty evaluation is very fast). Correspondingly, we can add a relatively high number of the room moves into the search procedure without trade-off with other types of moves. However, this addition affects the ratios between different types of moves. Our new algorithm still selects a type of move randomly (as was the case with the previous variant), but now approximately 50% of all moves are assigned to be the room moves (this ratio was chosen after a series of preliminary tests). The remaining half of the moves are distributed with the ratio of 1/5 (similar to the process described in the previous section variant) between the swapping of two timeslots and the replacing of an exam into a different timeslot.

The introduction of the room moves can also arise a question about the effectiveness of the previously described Largest Degree based adaptation mechanism (AFDA/LD) in respect of these new moves. Obviously, when an exam's timeslot is invariable, the difficulty of a move is hardly dependent on the number of conflicting exams placed in other timeslots. Here, the exams which are most difficult to move are those that have the highest number of students. With this in mind, an alternative adaptive mechanism could be proposed. We suggest to use again Formula (3) for the calculation of the flexibility coefficient for each exam, but consider d_i as the number of students taking this exam and d_{max} as the maximum number of students among all exams. To distinguish this mechanism, we call it the Largest Number of Students (LNS) heuristic. In our experiments, we investigated both variants: (a) AFDA/LD for all (replacement and room) moves and (b) the combined approach (AFDA/LD+LNS), where k_f for room moves is based on the LNS while the replacement moves are still accepted using the LD mechanism.

Note, that all the amendments described in this section (i.e. room-related issues) are actually only used when the number of rooms is more than one. Therefore, they are omitted for dataset *Exam_4* as it has only one room.

4. An Experimental Analysis of FDA

4.1 Experiments with Toronto benchmark datasets

In a first phase, we have carried out experiments with benchmark problems from the Toronto collection. Our experimental software was developed in Delphi 7 and run on a PC Pentium 4 3.2 GHz with 2GB RAM under Windows XP SP3. We are presenting, in this section, four series of experiments.

To clearly understand the influence of the flexibility on the overall algorithm's performance, in the first series of experiments we have plotted cost progress diagrams. They were drawn by the same method as described in (Burke et al, 2004a). During the search procedure after every 1000 moves we mark a point, which indicates the current time and cost at the corresponding coordinates. To illustrate the situation, the consecutive points are connected with each other. Figure 4 presents an example (for the problem Kfu-s-93) of the comparison of two diagrams: the original Great Deluge with $k_f = 1$ and its flexible variant with $k_f = 0.05$. All other search parameters (including the initial solution and the shape of the lowering of *B*) were identical in both runs.

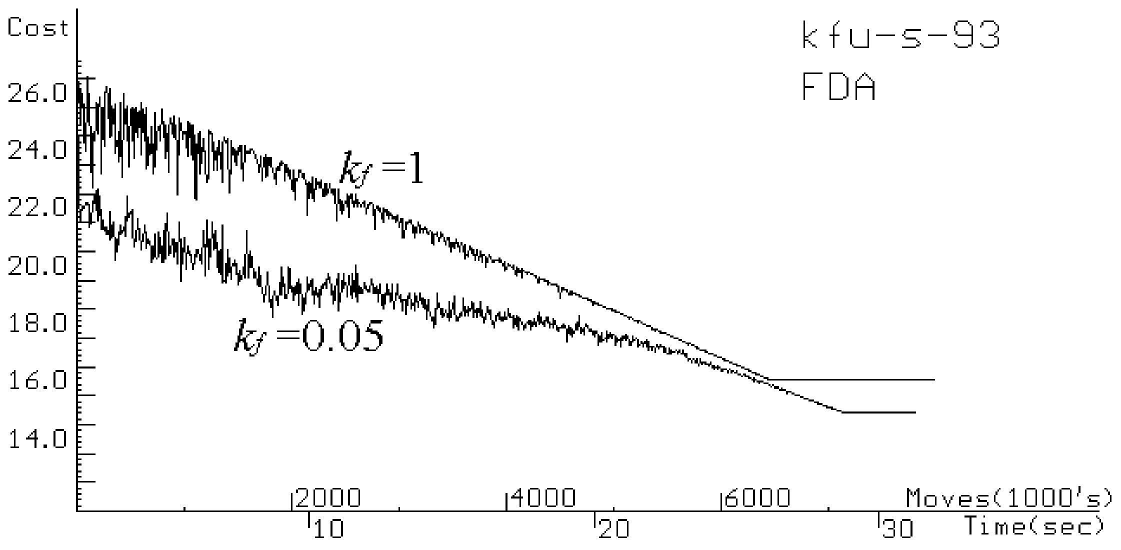

Figure 4: Cost progress diagrams for FDA with $k_f = 0.05$ and Great Deluge ($k_f = 1$)

This example reveals an evident difference in the behaviour of these two algorithms and confirms the general properties of the FDA approach proposed in Section 2.1. During the original Great Deluge ($k_f = 1$), the current cost quite strictly follows the lowering of the level (its shape appears as an imaginary straight line, which bounds the upper part of the diagram). In contrast, the flexible search ($k_f = 0.05$) demonstrates an intermediate (between Great Deluge and Hill-Climbing) behaviour: its cost falls down rapidly at the beginning and, most of the time, it keeps some distance from the level. It can be seen that FDA decreases the cost more freely, where the uneven shape of the diagram might show that the algorithm somehow responds to the properties of an operation landscape. It could be proposed that this might be one of the reasons for the superior performance of FDA.

In addition, the above diagrams illustrate that at the final steps of the search, GDA and FDA perform in very similar ways. Both algorithms strictly follow the level and both converge exactly at the points where the level reaches the final cost. This observation confirms that the major practical advantages of the GDA are retained in FDA: both algorithms can operate with the same stopping condition and their time-predefinition can be carried out in the same way.

In the second series of experiments, we investigate the performance of the Flex-Deluge algorithm with fixed flexibility. Our goal was to illustrate the effectiveness of the entire idea of flexible acceptance (described in Section 2.1). In addition to this, we aimed to examine the influence of k_f on the overall performance of the search and, hopefully, to reveal a recommended value of the flexibility coefficient. For this purpose, we have collected average results over a number of runs with different k_f (fixed for all types of moves during a whole run), whilst keeping all other algorithmic parameters invariable for all runs. The following values of the flexibility coefficient were evaluated: 0 (equivalent to Hill-Climbing), 0.005, 0.01, 0.05, 0.1, 0.5 and 1 (equivalent to the Great Deluge). The algorithm was run 50 times with each value of k_f for each benchmark instance. Each run lasted 300 seconds. The average results are presented in Table 3, where the lowest values for each of the datasets are highlighted in bold. Note, that these are just the minimum average values. The actual best results achieved by our method are presented later (in Table 7).

Table 3: Average results for FDA (fixed flexibility) for 5 minute runs.

Dataset	k_f						
	0 (HC)	0.005	0.01	0.05	0.1	0.5	1 (GD)
Car-f-92	4.52	**4.21**	4.22	4.25	4.26	4.29	4.28
Car-s-91	5.49	5.19	**5.16**	5.19	5.23	5.21	5.25
Ear-f-83I	37.7	34.62	33.94	**33.82**	33.96	34.24	34.35
Ear-f-83IIc	39.92	36.83	36.36	**36.11**	36.28	36.52	36.75
Hec-s-92I	11.56	10.55	**10.41**	10.47	10.51	10.58	10.65
Hec-s-92II	11.49	10.53	**10.35**	10.38	10.42	10.51	10.44
Kfu-s-93	14.66	13.56	**13.44**	13.54	13.55	13.63	13.64
Lse-f-91	11.93	10.65	**10.47**	10.50	10.58	10.62	10.64
Pur-s-93	5.09	4.84	4.87	4.86	4.86	**4.83**	4.88
Rye-f-92	9.18	**8.44**	8.49	8.53	8.56	8.58	8.59
Sta-f-83I	157.37	**157.05**	157.06	157.06	157.07	157.10	157.11
Sta-f-83IIc	32.14	**30.72**	30.77	30.82	30.86	30.89	30.93
Tre-s-92	9.03	8.26	**8.17**	8.21	8.27	8.30	8.31
Uta-s-92I	3.69	**3.50**	**3.50**	3.54	3.53	3.54	3.56
Uta-s-92II	3.70	**3.48**	3.49	3.53	3.53	3.53	3.55
Ute-s-92	26.60	25.00	24.96	**24.93**	**24.93**	24.95	25.01
Yor-f-83I	39.14	37.15	**36.77**	36.91	37.12	37.33	37.57
Yor-f-83IIc	43.87	42.31	**41.43**	**41.43**	41.60	41.79	41.93

The presented results clearly show that the introduction of the flexibility coefficient really improves the performance of the search procedure. For all benchmark instances, the Flex-Deluge method with medium values of k_f was able to achieve better average results than both its extremes (hill-climbing and Great Deluge). As might be expected, the best particular value of k_f is different for different problems. However, it appears, from these experiments, that the best value of k_f is around 0.01.

The third series of experiments was undertaken to determine whether there is any benefit in varying k_f during the search. Here we evaluate the adaptive algorithm (AFDA/LD) presented in Section 3.1. To provide a clear comparison, all experimental conditions (algorithmic parameters, number of runs, execution time) were the same as in the previous experiments. However, now we have varied the *upper bound* of the flexibility interval k_f^{max} (through the same values as we used for k_f in the first set of experiments). We have excluded the case where $k_f^{max} = 0$ because there can be no variation here and, therefore, it is equivalent to the fixed flexibility case (already studied above). The results are presented in Table 4, where we have highlighted those results which are the best across the AFDA/LD and FDA experiments.

Table 4: Average results for AFDA/LD for 5 minute runs.

Dataset	k_f^{max}					
	0.005	0.01	0.05	0.1	0.5	1
Car-f-92	4.37	4.25	4.23	4.25	4.27	4.27
Car-s-91	5.36	5.27	5.19	5.22	5.23	5.24
Ear-f-83I	36.26	34.97	**33.69**	33.84	34.09	34.23
Ear-f-83IIc	38.37	37.25	**35.82**	36.13	36.46	36.58
Hec-s-92I	10.98	10.52	**10.36**	10.43	10.54	10.56
Hec-s-92II	10.94	10.44	**10.29**	10.37	10.46	10.51
Kfu-s-93	14.15	13.79	**13.43**	13.48	13.58	13.61
Lse-f-91	11.38	11.01	**10.41**	10.48	10.52	10.56
Pur-s-93	5.04	4.92	**4.82**	4.87	4.90	4.87
Rye-f-92	8.70	8.51	8.45	8.50	8.55	8.56
Sta-f-83I	157.09	**157.05**	157.07	157.08	157.09	157.09
Sta-f-83IIc	30.78	30.78	30.87	30.88	30.96	30.98
Tre-s-92	8.64	8.43	**8.16**	8.22	8.27	8.30
Uta-s-92I	3.62	**3.50**	3.52	3.57	3.54	3.54
Uta-s-92II	3.58	3.51	3.49	3.51	3.53	3.53
Ute-s-92	25.41	25.12	24.90	**24.88**	24.97	25.02
Yor-f-83I	38.01	37.31	**36.65**	36.94	37.24	37.35
Yor-f-83IIc	43.18	42.30	**41.33**	41.43	41.66	41.72

The results in this table reveal the following properties:

- The dependence of the performance of the self-adaptive technique on the value of k_f^{max} is similar to that of the fixed flexibility method on k_f. Namely, the best value of k_f^{max} is placed somewhere between two extremes. Also, the best value is different for different problems.

- The algorithm is definitely ineffective with very small k_f^{max}. This property is predictable, because the algorithm should have *sufficient room* to vary the flexibility.

- With relatively larger values of k_f^{max} AFDA/LD generally provides better results than the fixed flexibility FDA with the same value of k_f. The exception is the Sta-f-83IIc problem. Here, the results of the self-adaptive method are steadily worse. It could be that the biasing tendencies in this problem instance have a somewhat positive effect. For example, they might bias the search towards a promising region of the search space.

- The overall effectiveness of the self-adaptive method (versus the fixed flexibility case) is highly instance-dependent. In our experiments, AFDA/LD produced the best overall results for 13 problems whilst FDA had the best results for 7 problems. However, a more attentive observation reveals that for 9 problems the difference in the best results is very small (or absent) and could be taken to be insignificant. Thus, we can say that for 8 problems the adaptation provides a sensible improvement; for 9 problems it has little effect and for one problem the performance is definitely worse. For these benchmarks, this justifies the general benefit of the proposed self-adaptive mechanism: in most cases it either provides an improvement or is neutral.

In summary, it can be argued that the majority of the best results in both experiments (10 out of 18) were produced by the AFDA/LD method with k_f^{max}=0.05. Therefore, we can conclude this as the most effective variant and have chosen it for our fourth series of experiments.

The fourth series of experiments was aimed at imitating a real-world examination timetabling process in order to determine how the proposed AFDA/LD method might be employed in practice. Here, we did not employ a high number of short runs because this has been shown to be an ineffective strategy for the Great Deluge algorithm (Burke et al, 2004a). Instead, we ran the algorithm for quite a long time (several hours) and produced just five runs for each benchmark instance. Note, as mentioned earlier, that it is perfectly acceptable in most real world situations to run an examination timetabling process for a few hours. This could represent an overnight (or even a weekend) run. The exam timetable, in most situations, is generated several months before it is required and the length of the time needed to build it is not at all critical. The execution times (in hours) and the final penalties from our experiments are given in Table 5. In this table, the runs are formally sorted from the "first" to the "fifth" by their penalty. All algorithmic parameters (except the execution time) were the same as in our second experiment. The execution time was assigned to 5 hours for all five runs for all problems. As the run time predefinition is imprecise in the Great Deluge algorithm (see Burke et al, 2004a), the real run times deviated slightly from this.

Table 5: Execution times and penalties for "Long Time" experiments on AFDA/LD with k_f^{max}=0.05

Dataset	First run		Second run		Third run		Fourth run		Fifth run	
	Time (h)	Penalty	Time (h)	Penalty	Time (h)	Penalty	Time (h)	Penalty	Time (h)	Penalty
Car-f-92	5.1	3.67	4.8	3.69	5.4	3.70	5.2	3.72	5.3	3.73
Car-s-91	5.2	4.32	5.5	4.34	5.2	4.34	5.4	4.35	5.2	4.37
Ear-f-83I	5.4	32.62	5.0	32.64	4.9	32.65	4.7	32.68	5.3	32.73
Ear-f-83IIc	5.6	34.59	5.7	34.60	5.1	34.80	5.4	34.87	5.7	34.91
Hec-s-92I	5.3	10.06	5.1	10.08	5.0	10.11	4.9	10.14	4.9	10.23
Hec-s-92II	5.0	10.03	4.8	10.06	4.7	10.06	5.2	10.09	5.1	10.11
Kfu-s-93	5.1	12.80	5.6	12.81	5.3	12.82	5.1	12.91	5.1	12.93
Lse-f-91	4.9	9.78	5.1	9.80	5.2	9.81	5.6	9.87	5.3	9.94
Pur-s-93	5.7	3.88	5.2	3.90	5.6	3.92	5.7	3.93	5.5	3.93
Rye-f-92	5.2	7.91	5.7	7.92	5.2	7.94	5.1	7.96	5.4	7.96
Sta-f-83I	4.6	157.03	4.8	157.03	5.1	157.03	4.9	157.06	5.0	157.06
Sta-f-83IIc	4.7	30.59	5.0	30.61	4.9	30.62	4.8	30.62	4.8	30.65
Tre-s-92	5.6	7.64	5.2	7.68	5.0	7.69	5.9	7.69	5.9	7.71
Uta-s-92I	5.3	2.98	5.5	3.01	5.4	3.01	5.3	3.02	5.1	3.03
Uta-s-92II	7.2	2.92	7.1	2.94	7.1	2.96	7.1	3.00	7.0	3.02
Ute-s-92	4.9	24.78	4.9	24.80	5.1	24.83	5.3	24.84	4.7	24.85
Yor-f-83I	4.7	34.71	5.4	34.72	5.5	34.76	5.0	34.81	5.1	34.93
Yor-f-83IIc	5.2	40.06	4.8	40.10	5.1	40.13	5.5	40.18	5.2	40.26

Although, the computational cost is expensive (but appropriate for this problem), we have carried out 5 separate runs to demonstrate that the difference in cost function values between our five runs is relatively small (a few percent) and this demonstrates a certain level of stability (at least on these instances).

4.2 Experiments with the ITC2007 datasets

In the second experimental phase, we have conducted a series of tests with different adaptive (AFDA/LD and AFDA/LD+LNS) and fixed-flexibility variants on the twelve ITC2007 datasets. The number of runs in each test and the processing time of each run were changed in order to comply with the competition rules. Now, we collect our average and best results during 10 runs of each variant. The maximum processing time was calculated using a special benchmarking utility provided by the ITC2007 organisers. For our computer, the limit on running time was set to be 481 seconds. When running our algorithm for this time (or less) we are confident that we do not run it for longer than any competitor's technique.

During the tests with these alternative datasets, all our algorithms showed a similar behaviour to that illustrated for the Toronto collection. All variants of FDA outperformed the original GDA (and of course the HC). The best value of k_f was again quite problem dependent, but generally, the adaptive versions were able to achieve better results than the fixed-flexibility ones. In addition, it was found that AFDA/LD+LNS more frequently outperforms AFDA/LD than vice versa. The approximate best value of k_f^{max} (where the adaptive variants achieved the average best performance) was around 0.002. In Table 6, we present an example of average results produced by the fixed and adaptive variants with this value of the flexibility and the results of the original GDA. The best average results are shown in bold. For comparison, the best results produced by AFDA/LD+LNS in 10 runs and its average run time are also given in this table.

Table 6: The results of GDA, FDA with k_f=0.002 and AFDA with k_f^{max} = 0.002

Dataset	GDA (average)	FDA (average)	AFDA/LD (average)	AFDA/LD+LNS		Average time (s)
				Average	best	
Exam_1	4162.6	3927.8	3819.8	**3792.5**	3691	449
Exam_2	458.2	397.6	395.5	**393.1**	385	476
Exam_3	8240.5	7690.7	7670.6	**7611.8**	7359	458
Exam_4	13848.7	12368.7	**12062.9**	12100.4	11329	462
Exam_5	2681.5	2529.3	2545.7	**2512.9**	2482	461
Exam_6	25607.5	25432.5	**25380.5**	25491.5	25265	432
Exam_7	4143.3	3935.5	3827.4	**3755.1**	3608	464
Exam_8	7463.8	7048.4	6989.7	**6949.9**	6818	455
Exam_9	993.4	936.0	944.5	**930.0**	902	419
Exam_10	13430.8	12987.4	**12950.8**	12975.7	12900	423
Exam_11	25307.8	23809.6	**23447.7**	23931.7	22875	438
Exam_12	5371.4	5211.9	5214.9	**5176.3**	5107	437

The example in this table illustrates the typical (and expected) performances of our algorithms. The best performance is shown by the adaptive variants (AFDA/LD+LNS is slightly better than AFDA/LD); the results of the fixed variant are generally slightly worse and the original GDA performs significantly worse.

4.3 Comparison with published results

The comparison of our results with the published ones is presented in Table 7 (Toronto collection) and Table 8 (ITC2007 collection). Our AFDA/LD results are compared in Table 7 using the terminology and naming conversions presented in (Qu et al, 2009). Note that we have excluded from the comparison the result of Casey & Thomson (2003) for Sta-f-83IIc because they have reported this result as scheduled into 13 timeslots which is definitely impossible as the chromatic number of its graph is 35. Indeed they use a version of the dataset which is incorrect (called Sta-f-83II in Qu et al, 2009). There is a converted version of this dataset called Sta-f-83IIc and we have presented the first results on that dataset. Also, note that our entry for Uta-s-92II is the only one because it appears that our result (see Table 7) is the first to be published on this particular instance (see Qu et al, 2009).

Table 7: Comparison of AFDA/LD results with published ones for Toronto datasets

		Car-f-92	Car-s-91	Ear-f-83I	Ear-f-83IIc	Hec-s-92I	Hec-s-92II	Kfu-s-93	Lse-f-91	Pur-s-93
Abdullah et al (2014)		4.00	4.62	33.14	-	10.43	-	13.59	10.75	-
Asmuni et al (2009)		4.54	5.29	37.02	-	11.78	-	15.80	12.09	-
Burke et al (2004a)		4.2	4.8	35.4	-	10.8	-	13.7	10.4	4.8
Burke et al (2010)		3.9	4.6	32.8	-	10.0		13.0	10.0	-
Caramia et al (2008)		6.0	6.6	**29.3**	-	**9.2**	-	13.8	**9.6**	
Carter et al (1996)		6.2	7.1	36.4	-	10.8	-	14.0	10.5	3.9
Casey & Thompson (2003)		4.4	5.4	-	34.8	-	10.8	14.1	14.7	-
Di Gaspero & Schaerf (2001)		5.2	6.2	45.7	-	12.4	-	18.0	15.5	-
Merlot et al (2003)		4.3	5.1	35.1	-	10.6	-	13.5	10.5	-
Petrovic et al (2007)		3.93	4.50	33.75	-	10.83	-	13.82	10.63	-
Pillay (2009)		4.1	4.8	34.97	-	10.99	-	13.89	10.6	-
AFDA/LD	Best	**3.67**	**4.32**	32.62	**34.59**	10.06	**10.03**	**12.80**	9.78	**3.88**
	Average	3.70	4.34	32.66	34.75	10.12	10.07	12.85	9.84	3.91

Table 7: (continued)

		Rye-f-92	Sta-f-83I	Sta-f-83IIc	Tre-s-92	Uta-s-92I	Uta-s-92II	Ute-s-92	Yor-f-83I	Yor-f-83IIc
Abdullah et al (2014)		9.17	157.06	-	8.00	3.27	-	25.16	35.58	-
Asmuni et al (2009)		10.38	160.42	-	8.67	3.57	-	28.07	39.80	-
Burke et al (2004a)		8.9	159.1	-	8.3	3.4	-	25.7	36.7	-
Burke et al (2010)		-	**156.9**	-	7.9	3.2	-	24.8	34.9	-
Caramia et al (2008)		**6.8**	158.2	-	9.4	3.5	**-**	**24.3**	36.2	-
Carter et al (1996)		7.3	161.5	-	9.6	3.5	-	25.8	41.7	-
Casey and Thompson (2003)		-	-	-	8.7	-	-	25.4	-	**37.5**
Di Gaspero and Schaerf (2001)		-	160.8	-	10.0	4.2	-	29.0	41.0	-
Merlot et al (2003)		8.4	157.3	-	8.4	3.5	-	25.1	37.4	-
Petrovic et al (2007)		8.53	165.27	-	7.92	3.14	-	25.33	36.35	-
Pillay (2009)		9.08	157.22	-	8.26	3.24	-	26.23	38.38	-
AFDA/LD	Best	7.91	157.03	30.59	**7.64**	**2.98**	2.92	24.78	**34.71**	40.06
	Average	7.94	157.04	30.62	7.68	3.01	2.97	24.82	34.79	40.15

Table 7 shows that for 9 out of 16 problems, our best results (highlighted in bold) have a penalty that improves upon the previously published ones. As the variation in the resulting penalty is relatively small across our different runs, our average results for 8 problems are also better than the previously published ones. This provides evidence that this method is especially effective for relatively large problems (which was also highlighted in (Burke et al, 2004a)) for the original exam timetabling Great Deluge algorithm). However, for small and very small problems it is probably advisable to employ other methods.

Finally, in Table 8 we compare the best results, produced by AFDA/LD+LNS (given in the last column) with the ones available in the literature and on the web. The second column of Table 8 represents the best results, achieved during ITC2007 and presented on its web site. For 10 datasets, the best results were provided by the 1st place method (Muller, 2008); for *Exam_10* dataset - by the 4th place method and for *Exam_12* dataset – by the 3rd place method. In the third column of Table 8, we give eight pre-competition results of the 1st place approach (Muller, 2008), which were achieved during 100 runs. The results, presented in the fourth and the fifth columns were published in (McCollum et al, 2009). There, the authors presented the results of

50 runs of Muller's algorithm (the fourth column) and the results of their own technique (the fifth column). We have also included in the comparison (the sixth column) the results of (Gogos et al 2010), most of which are the best up to date results. However, they were produced using multi-processor computing without compliance with the ITC2007 rules. These results are generated without a hardware or time limit. The overall best results for each dataset in Table 8 are highlighted in bold.

Table 8: Best AFDA/LD+LNS and published results for ITC2007 datasets

Dataset	ITC2007 web best	Muller (2008)	Muller 51 runs	McColum et al (2009)	Gogos et al (2010)	AFDA/ LD+LNS
Exam_1	4370	4356	4370	4633	4128	**3691**
Exam_2	400	390	385	405	**380**	385
Exam_3	10049	9568	9378	9064	7769	**7359**
Exam_4	18141	16591	15368	15663	13103	**11329**
Exam_5	2988	2941	2988	3042	2513	**2482**
Exam_6	26585	25775	26365	25880	25330	**25265**
Exam_7	4213	4088	4138	4037	**3537**	3608
Exam_8	7742	7565	7516	7461	7087	**6818**
Exam_9	1030	-	1014	1071	913	**902**
Exam_10	14778	-	14555	14374	13053	**12900**
Exam_11	34129	-	31425	29180	24369	**22875**
Exam_12	5264	-	5357	5693	**5095**	5107

The presented comparison once again confirms the strong performance and the generality of the proposed method: 9 of the best AFDA/LD+LNS results are better than the previous best ones. To further highlight this, it is possible to compare the published results with our average ones (from Table 6). It can be seen that, for many instances, the average results produced by different versions of the flexible approach are still better than the best results of the previous methods (AFDA/LD is better for 7 instances, AFDA/LD+LNS and FDA - for 6 instances). Meanwhile, the performance of our original GDA is not so strong (it has no average results better than the best published ones). Taking into account the high level of popularity of GDA among the exam timetabling research community (most of the results in Table 8 were produced by GDA-based techniques), this observation suggests the high practical benefit of our study. GDA-practitioners might improve their results with minimum effort, simply by employing the flexibility coefficient introduced in this paper.

Note that when presenting our best results for all instances (in both collections), special care has been taken with respect to their correctness. These solutions are available at http://www.yuribykov.com/flex-deluge/ to enable the scientific community to examine them.

5. Future Research Directions

The strong performance of the presented algorithm on the international benchmarks shows how effective it is as an automated examination timetabling procedure. However, we see the major contribution of this paper in a more general context. Indeed, the demonstration of the general effectiveness of the suggested method motivates a wider range of different experiments on different problems as the subject of future research. A programme of experiments of this order is beyond the scope of this paper but we will briefly discuss some of the key issues.

Although the Flex-Deluge algorithm was initially developed as an extension of the Great Deluge method, it could be formally positioned as a more general algorithm, where both Great Deluge and Hill-Climbing are just two extreme cases of the general technique. Our experiments with two collections of exam timetabling problems have shown that the flexibility coefficient can significantly affect the performance of the Flex-Deluge method. For all our problem instances, the algorithm with *middling* levels of flexibility performed better than both its extremes. A similar kind of behaviour could be seen in other problems, i.e. the tuning of fixed flexibility could help to achieve better final results. Note, that the Flex-Deluge algorithm can be applied to any problem, which is solvable by either of its extremes. Thus, this paper could be seen as a certain template for the investigation of the suggested technique for a wide range of search problems.

The presented *activity-based* self-adaptive heuristic is aimed at reducing negative biasing tendencies during a search, i.e. to smooth the difference in *activity* between different variables. We believe that a significant number of search and optimisation problems have variables with different levels of activity. Therefore, we propose that it is well worth exploring whether this heuristic would also be effective for other problems. Of course, it depends on the particular problem being studied. Furthermore, it could be beneficial to explore further possible adaptation strategies.

In this study, we have suggested two adaptive mechanisms (AFDA/LD and AFDA/LD+LNS), which can be viewed as examples of how well-known sequencing heuristics (*Largest Degree First* and *Largest Number of Students First*) can be incorporated into a metaheuristic search procedure. It could be beneficial to also investigate the performance of other heuristic-based self-adaptive strategies, for example: *Largest Weighted Degree First* or *Smallest Degree Last* or even to investigate a dynamic recalculation of the flexibility by some analogue of the *Saturation Degree* heuristic.

It was observed that during a typical Flex-Deluge search process, the properties of different moves were particularly varied across quite a high range. The ranking of such moves could be the focus of a research programme to investigate adaptive acceptance. The adaptation could be based on any of the aspects, that could influence the search procedure, including the properties of a target problem such as variables, their domains, constraints, neighbourhoods, operational landscapes, the particular properties of current and/or candidate solutions, a list of previous solutions and any information collected during the search. Such information might be acceptance ratios, best achieved solutions and search trajectories. The development of such heuristics could be viewed as a highly promising direction for future studies. In this context, the proposed Flex-Deluge algorithm has the potential to be thought of as a general-purpose tool (which provides the opportunity for heuristic regulation within the search procedure).

6. Conclusions

This paper presents an extension of the Great Deluge algorithm, where the acceptance condition is flexible. It explores two ideas for improving the overall performance of the search technique. The first one is to make *uphill* moves more slow and the second one is to adjust the algorithm in response to the properties of a particular problem. Moreover, the proposed algorithm retains all the advantages of the original Great Deluge method, i.e. a low number of input parameters and the ability to predefine the amount of required computational time.

Two variants were introduced: a fixed Flex-Deluge approach and an adaptive methodology. Both proposed variants were applied to the university exam timetabling problem and both were able to outperform the original Great Deluge algorithm. The fixed-flexibility variant is very simple for implementation and enables the improvement of the performance of the Great Deluge technique with minimum effort.

In contrast, the adaptive variant requires more attention to the properties of a problem (and that of a search procedure) and can be implemented in different ways (we have presented two examples of possible adaptive mechanisms). However, it was shown that the employment of an appropriate self-adaptation technique can further improve the performance of the Flex-Deluge algorithm over its fixed variant.

We compared the performance of the presented method to that of previous techniques (published in the literature and on the web) using two collections of exam timetabling instances. The results produced by our algorithm are better than the best previously published ones for 9 out of 16 instances in the first collection and for 11 out of 12 instances in the second collection. The corresponding average results for 19 problems (over both collections) are also better than any of the published results, which demonstrates the robustness of the methodology.

Acknowledgements

The work described in this paper was carried out under a grant (GR/S70197/01) awarded by the UK Engineering and Physical Sciences Research Council (EPSRC).

References

Abdul Rahman, S., A. Bargiela, E. K. Burke, E. Ozcan, B. McCollum, P. McMullan. 2014. Adaptive linear combination of heuristic ordering in constructing examination timetables. *European Journal of Operational Research* **232** 287-297.

Abdullah, S., S. Ahmadi, E. K. Burke, M. Dror. 2007. Investigating Ahuja-Orlin’s large neighbourhood search for examination timetabling. *OR Spectrum* **29** 351-372.

Abdullah, S., M. Alzaqebah. 2014. An adaptive artificial bee colony and late-acceptance hill-climbing algorithm for examination timetabling. *Journal of Scheduling* **17** 249-262.

Al-Betar, M. A., A. T. Khader, I. Abu Doush. 2014. Memetic techniques for examination timetabling. *Annals of Operations Research* **218** 23-50.

Asmuni, H., E. K. Burke, J. Garibaldi, B. McCollum, A. Parkes. 2009. An investigation of fuzzy multiple heuristic orderings in the construction of university examination timetables. *Computers and Operations Research* **36** 981-1001.

Bardadym, V. 1996. Computer-aided school and university timetabling: The new wave. *Practice and Theory of Automated Timetabling, Springer Lecture Notes in Computer Science* **1153** 22-45.

Burke, E. K., J. Newall. 2003. Enhancing timetable solutions with local search methods. *Practice and Theory of Automated Timetabling IV, Springer Lecture Notes in Computer Science* **2740** 344-354.

Burke, E. K., J. Newall. 2004. Solving examination timetabling problems through adaptation of heuristic orderings. *Annals of Operations Research* **129** 107-134.

Burke, E. K., S. Petrovic. 2002. Recent research directions in automated timetabling. *European Journal of Operational Research* **140** 266-280.

Burke, E. K., Y. Bykov, J. Newall, S. Petrovic. 2003. A time-predefined approach to course timetabling. *Yugoslav Journal of Operations Research (YUJOR)* **13** 139-151.

Burke, E. K., Y. Bykov, J. Newall, S. Petrovic. 2004a. A time-predefined local search approach to exam timetabling problems. *IIE Transactions* **36** 509-528.

Burke, E. K., Bykov, Y., 2006. Solving exam timetabling problems with the flex-deluge algorithm (abstract). *PATAT 2006 Proceedings of the 6th International Conference on the Practice and Theory of Automated Timetabling*, ISBN 80-210-3726-1, 370-372.

Burke, E. K., D. Elliman, P. Ford, R. Weare. 1996. Examination timetabling in British universities: a survey. *Practice and Theory of Automated Timetabling, Springer Lecture Notes in Computer Science* **1153** 76-90.

Burke, E. K., J. Kingston, K. Jackson, R. Weare. 1997. Automated university timetabling: The state of the art. *The Computer Journal* **40** 565-571.

Burke, E. K., J. Kingston, D. De Werra. 2004b. 5,6: Application to Timetabling. In Gross, J., Yellen, J. (eds): The Handbook of Graph Theory. Chapman Hall/CRC Press, 445-474.

Burke, E. K., S. Petrovic, R. Qu. 2006. Case Based Heuristic Selection for Timetabling Problems, *Journal of Scheduling* **9** 115-132.

Burke, E. K., B. McColumn, A. Meisels, S. Petrovic, R. Qu. 2007. A graph-based hyper-heuristic for educational timetabling problems. *European Journal of Operational Research* **176** 177-192.

Burke, E. K., A. Eckersley, B. McCollum, S. Petrovic, R. Qu. 2010. Hybrid variable neighbourhood approaches to university exam timetabling. *European Journal of Operational Research* **206** 46-53.

Bykov Y. 2003. Time-predefined and trajectory based search: single and multiobjective approaches to exam timetabling. Ph.D. thesis, The University of Nottingham, UK.

Caramia, M., P. Dell'Olmo, G. Italiano. 2008. Novel local-search based approaches to university examination timetabling. *INFORMS Journal on Computing* **20** 86-99.

Carter, M.W. 1986. A survey of practical applications of examination timetabling algorithms. *Operations Research* **34** 193-201.

Carter, M.W., G. Laporte. 1996. Recent developments in practical examination timetabling. *Practice and Theory of Automated Timetabling, Springer Lecture Notes in Computer Science* **1153** 3-21.

Carter, M.W., G. Laporte, S. Lee. 1996. Examination timetabling: algorithmic strategies and applications. *Journal of the Operational Research Society* **47** 373-383.

Casey, S., J. Thompson. 2003. GRASPing the examination scheduling problem. *Practice and Theory of Automated Timetabling IV, Springer Lecture Notes in Computer Science* **2740** 232-246.

Cheong, C.Y., K.C. Tan, B. Veeravalli, 2009. A multi-objective evolutionary algorithm for examination timetabling. *Journal of Scheduling* **12** 121-146.

Corne, D., P. Ross. 1996. Peckish initialisation strategies for evolutionary timetabling. *Practice and Theory of Automated Timetabling, Springer Lecture Notes in Computer Science* **1153** 227-240.

de Werra, D. 1985. An introduction to timetabling. *European Journal of Operational Research* **19** 151-162.

David, P. 1998. A constraint-based approach for examination timetabling using local repair techniques. *Practice and Theory of Automated Timetabling II, Springer Lecture Notes in Computer Science* **1408** 169-186.

Di Gaspero, L., A. Schaerf. 2001. Tabu search techniques for examination timetabling. *Practice and Theory of Automated Timetabling III, Springer Lecture Notes in Computer Science* **2079** 104-117.

Di Gaspero, L. 2002. Recolour, shake and kick: a recipe for the examination timetabling problem (abstract). *PATAT 2002 Proceedings of the 4th International Conference on the Practice and Theory of Automated Timetabling*, ISBN 90-806096-1-7, 404-407.

Dueck, G. 1993. New optimisation heuristics. The great deluge algorithm and record-to-record travel. *Journal of Computational Physics* **104** 86-92.

Erben, W. 2001. A grouping genetic algorithm for graph colouring and exam timetabling. *Practice and Theory of Automated Timetabling III, Springer Lecture Notes in Computer Science* **2079** 132-156.

Fong, C. W., H. Asmuni, B. McCollum, P. McMullan, S. Omatu. 2014. A new imperialist swarm-based optimization algorithm for university timetabling problems. *Information Sciences* **283** 1-21.

Gogos, C., P. Alefragis, E. Housos. 2008. A multi-staged algorithmic process for the solution of the examination timetabling problem. *PATAT 2008 Proceedings of the 7th International Conference on the Practice and Theory of Automated Timetabling*, Montreal, Canada, August 2008.

Gogos, C., G. Goulas, P. Alefragis, V. Kolonias, E. Housos. 2010. Distributed Scatter Search for the examination timetabling problem. *PATAT 2010 Proceedings of the 8th International Conference on the Practice and Theory of Automated Timetabling*, Belfast, UK, August 2010.

Jaddi, N. S., S. Abdullah. 2014. Nonlinear Great Deluge Algorithm for rough set attribute reduction. *Journal of Information Science and Engineering* **29** 49 62.

Lewis, R. 2008. A survey of metaheuristic-based techniques for university timetabling problems. *OR Spectrum* **30** 167-190.

McCollum, B. 2007. A perspective on bridging the gap between theory and practice in University timetabling. *Practice and Theory of Automated Timetabling VI, Springer Lecture Notes in Computer Science* **3867** 3-23.

McCollum, B., A. Schaerf, B. Paechter, P.J. McMullan, R. Lewis, A.J. Parkes, L. Di Gaspero, E.K. Burke, R. Qu. 2010. Setting the research agenda in automated timetabling: The second international timetabling competition. *INFORMS Journal on Computing* **22** 120-130.

McCollum, B., P.J. McMullan, A. J. Parkes, E.K. Burke, S Abdullah. 2009. An Extended Great Deluge Approach to the Examination Timetabling Problem. *MISTA09. Proceedings of the 4th Multidisciplinary International Conference on Scheduling: Theory and Applications*, Dublin, Ireland, August 2009 424-434.

McMullan, P. 2007. An extended implementation of the great deluge algorithm for course timetabling. *Springer Lecture Notes in Comput. Sci*. 4487 538-545.

Merlot, L., N. Boland, B. Hughes, P. Stuckey. 2003. A hybrid algorithm for the examination timetabling problem. *Practice and Theory of Automated Timetabling IV, Springer Lecture Notes in Computer Science* **2740** 207-231.

Meyers, C., J.B. Orlin. 2007. Very large scale neighbourhood search techniques in timetabling problems. *Practice and Theory of Automated Timetabling VI, Springer Lecture Notes in Computer Science* **3867** 24-39.

Muller, T. 2008. ITC2007 solver description: a hybrid approach. *PATAT 2008 Proceedings of the 7th International Conference on the Practice and Theory of Automated Timetabling,* Montreal, Canada, August 2008.

Obit, J. H., D. Landa-Silva, D. Ouelhadj, M. Sevaux. 2009. Non-linear great deluge with learning mechanism for solving the course timetabling problem. In *Proceedings of MIC 2009: the VIII Metaheuristic International Conf.*, Hamburg, Germany, July 2009, id-1-10.

Ozcan, E., M. Birben, Y. Bykov, E.K. Burke. 2009. Examination timetabling using late acceptance hyper-heuristics. *CEC 2009 Proceedings of the 2009 IEEE Congress on Evolutionary Computation*, Trondheim, Norway, May 2009, 997-1004.

Ozcan, E., M. Misir, G. Ochoa, E.K. Burke. 2010. A reinforcement learning - great deluge hyper-heuristic for examination timetabling. International Journal of Applied Metaheuristic Computing **1** 39-59.

Paquete, L.F., C.M. Fonseca. 2001. A study of examination timetabling with multiobjective evolutionary algorithms. *4th Metaheuristics International Conference (MIC 2001)*, 149-154.

Petrovic, S., Y. Bykov. 2003. A multiobjective optimisation technique for exam timetabling based on trajectories. *Practice and Theory of Automated Timetabling IV, Springer Lecture Notes in Computer Science* **2740** 179-192.

Petrovic, S., E.K. Burke. 2004. University timetabling, Ch. 45. J.Y-T. Leung, ed. *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*. CRC Press, FL, USA.

Petrovic, S., Y. Yang, M. Dror. 2007. Case-based selection of initialisation heuristics for metaheuristic examination timetabling. *Expert Systems with Applications* **33** 772-785.

Pillay, N. 2009. The revised developmental approach to the uncapacitated examination timetabling problem. *2009 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists,* Vanderbijlpark, South Africa, October 2009, 187-192.

Pillay, N. 2014. A review of hyper-heuristics for educational timetabling. *Annals of Operations Research* doi:10.1007/s10479-014-1688-1, Published online 09 August 2014.

Qu, R., E.K. Burke, B. McCollum, L.T.G. Merlot, S.Y. Lee. 2009. A survey of search methodologies and automated system development for examination timetabling. *Journal of Scheduling* **12** 55-89.

Schaerf, A. 1999. A survey of automated timetabling. *Artificial Intelligence Review* **13** 87-127.

Sin, E.S., N.S.M. Kham. 2012. Hyper heuristic based on great deluge and its variants for exam timetabling problem. *International Journal of Artificial Intelligence and Applications* **3** 149-162.

Thompson, J.M., K.A. Dowsland. 1996a. General cooling schedules for s simulated annealing based timetabling system. *Practice and Theory of Automated Timetabling, Springer Lecture Notes in Computer Science* **1153** 345-363.

Thompson, J.M., K.A. Dowsland. 1996b. Variants of simulated annealing for the examination timetabling problem. *Annals of Operations Research* **63** 105-128.

Turabeih, H., S. Abdullah. 2011. A hybrid fish swarm optimisation algorithm for solving examination timetabling problem. 2011. *Learning and Intelligent Optimization, Springer Lecture Notes in Computer Science* **6683** 539-551.

Ulker, O., E. Ozcan, E.E. Korkmaz. 2007. Linear linkage encoding in grouping problems: applications on graph coloring and timetabling. *Practice and Theory of Automated Timetabling VI, Springer Lecture Notes in Computer Science* **3867** 347-363.

White, G.M., B.S. Xie. 2001. Examination timetables and tabu search with longer term memory. *Practice and Theory of Automated Timetabling III, Springer Lecture Notes in Computer Science* **2079** 85-103.