Perturbation based variable neighbourhood search in heuristic space for examination timetabling problem

Samad Ahmadi[1,2], Rossano Barone[2,1], Peter Cheng[2], Edmund Burke[1], Peter Cowling[3] and Barry McCollum[4]

[1] ASAP research group, School of Computer Science & IT, University of Nottingham, Jubilee Campus, Nottingham NG8 1BB,UK,
sza@cs.nott.ac.uk

[2] CREDIT research group, School of Psychology, University of Nottingham, Nottingham NG7 2RD, UK,
rb@psychology.nott.ac.uk, peter.cheng@nottingham.ac.uk

[3] MOSAIC research group, Department of Computing, University of Bradford, Bradford BD7 1DP, UK,
Peter.Cowling@scm.brad.ac.uk

[4] School of Computer Science, Queen's University Belfast, Belfast, BT7 1NN, UK
b.mccollum@qub.ac.uk

Abstract. Efficient constructive heuristics have been designed for most of the difficult combinatorial optimisation problems. Performance of these heuristics may vary from one instance to another and hence it is beneficial to be able to select an instance of the heuristic which is best suited to a given instance of the problem. Both of the heuristic search algorithm and hyper-heuristic approaches try to provide methodologies for selecting a heuristic from a set of given heuristics. In this paper we propose a perturbation based algorithm to search an infinite space of parameterised heuristics. This algorithm aims to find the best heuristic in a set of given heuristics and hence its success or failure are measured in terms of the quality of the solution generated by the selected heuristic compared to the other heuristics in the heuristic space. This approach was applied to a highly constrained instance of examination timetabling problem. Computational results show the ability of the perturbation based algorithm in finding good combinations of heuristics and their parameters with reasonable computational effort.

1 Introduction

Efficient constructive heuristics have been designed for most of the difficult combinatorial optimisation problems. Performance of these heuristics may vary from one instance to another and hence it is beneficial to be able to select an instance of the heuristic which is best suited to a given instance of the problem. Both of the hyper-heuristic approach and the heuristic search algorithms try to provide methodologies

for selecting a heuristic from a set of given heuristics. In this paper we propose a perturbation based algorithm to search an infinite space of parameterised heuristics. This algorithm aims to find the best heuristic in the set of given heuristics and hence its success or failure are measured in terms of the quality of the solution generated by the selected heuristic compared to the other heuristics in the heuristic space. The underlying application for the heuristic algorithms is the examination timetabling problem.
Our approach in searching the space of heuristics by the use of perturbation is based on our experience with the human approach to selecting heuristics. These experiments have been conducted using the HuSSH (human selection of scheduling heuristics) interface which has been designed in conjunction with the STARK (Semantically Transparent Approach to Representing Knowledge) visual interface for examination timetabling (Ahmadi, Barone et al. 2002; Cowling, Ahmadi et al. 2002; McCollum, Ahmadi et al. 2002; P.Cheng, R.Barone et al. 2002; Ahmadi and Osman 2003; Cheng, Barone et al. 2003).
This paper is organized in following order: in the next section, a brief overview of the examination timetabling problem is presented. Section (3) outlines the design of flexible constructive heuristics based on weighted decision functions. We introduce a set of constructive heuristics for examination timetabling problem in section (4). Our perturbation based Heuristic search algorithm and its implementation for examination timetabling problem are discussed in Section (5) and computational results are presented in section (6). Conclusions and directions for further research are presented in section (7).

2 The Examination timetabling problem

The Examination timetabling problem is a difficult combinatorial optimisation problem that needs to be solved several times a year in schools and universities. The problem demands that a given number of exams need to be scheduled to a limited number of periods and rooms in such a way that there are no clashes (*i.e.* no student will have more than one exam at a time) and resource constraints (available space and time) are not violated. Institutions differ in the types of constraints they use and their level of severity. Constraints which really must not be violated are called hard constraints. This includes clashes, examination ordering constraints, resource constraints, restriction of exams to the specific time windows or venues and preference constraints. Soft constraints are mostly related to preferences and their violations can be tolerated to some extent but need to be minimized. The objective of examination timetabling problem is to find a schedule which satisfies all the hard constraints (or minimizes their violations) and minimises the total violations of the soft constraints.
This problem has been tackled with different heuristic, optimisation and metaheuristic algorithms. The underlying model of the problem considered by most of the algorithms is the graph coloring problem. Burke, Bykov et al. (2001) used a multicriteria approach to solve the problem in several phases. Thompson and Dowsland (1998) used simulated annealing with Kempe chain neighbourhoods to preserve the feasibility

of the solutions in terms of first order clashes during the local moves. Ross, Hart et al. (1998) analysed the behaviour of genetic algorithm on different instances of the problem. Arani and Lotfi (1989) investigated a three phase process using the quadratic assignment problem, the set covering problem and the traveling salesman problem in different phases respectively. Further references and a general survey of the problem are in (Carter, Laporte et al. 1997).
In next section we describe a weighted decision function to provide a flexible approach to accommodate the varieties of the constraints.

3 Weighted decision functions: an adaptive model for dealing with hard and soft constraints in the design of heuristics

Conflicting objectives and the changing set of constraints in different institutions makes the examination timetabling problem challenging for the design of a generic algorithms. To deal with hard constraints efficiently, one may algorithmically prevent their violation. In this method hard and soft constraints need to be specified at the design time of the algorithm which makes the algorithm inflexible towards changes in the sets of hard and soft constraints and prevents the algorithm from accepting infeasible solutions that may be necessary for bypassing the disconnectedness of the solution space.

An alternative approach is to use a decision function that is the weighted sum of the violations of all the constraints. This provides the flexibility of switching the soft and hard constraints for different instances and institutions. The drawback of this approach is that it is highly parametrised. We introduce a perturbation based heuristic search algorithm to explore the space of heuristics for suitable heuristics and their corresponding weights. Note that the decision function is a greedy selection function which could dynamically change its weights, where the objective function will use constant weights. For our scheduling algorithm we incorporate violations of all the different constraints (hard and soft) in a weighted decision function. The potential cost of scheduling an exam *e* to a period *p* is defined as weighted sum of following values:

1. **Clash and consecutive constraint violations:** if exam *e* has $n_1,\ldots,n_k$ students in common with exams $e_1,\ldots,e_k$, respectively, its potential clash constraint violation penalty value in period *p* is defined as:

$$k\sum_{i=1}^{k} n_i \qquad (1)$$

This will give weight to the number of students involved in each clash and not only to the number of clashing exams. At the same time, the above formula stresses the importance of having smaller number of clashing exams in a period. Note that a large number of small clashes may need more effort to resolve the violations later. As an example note that above formula assigns penalty of *9* to the case of 3 exams in period *p,* each having clashes of size 1 with exam *e* compared to the case of having only one exam in this period with total number of students in common of 3 which will be assigned penalty of *3*. The same measure is defined for consecutive exam violations.

2. **Order constraint violations:** if an exam *e* is required to be scheduled at the same time (coincidental) with exams $e_1,\ldots,e_k$ and *k'* of these exams are scheduled in period *p*, then the coincidental penalty associated with this period is *k-k'*, *i.e.* number of coincidental exams of *e* which are scheduled in a period other than *p.* This rewards assignment of exam *e* to periods with a higher number of its coincidental exams in them. For penalty values of followed exams and excluded exam violations similar definitions are used.
3. **Resource constraints:** Resources (periods, rooms) with duration/capacity shorter than the duration/size of the exam are discouraged by penalizing them according to the amount by which they differ.
4. **Exams Restrictions to specific time windows or venues:** For exams with pre-specified periods, a penalty is associated their assignment to other periods. For pre-specified rooms a similar method is used.

In next section, we propose several constructive heuristics for the examination timetabling problem based on a weighted decision function, a ranking method of heuristic selection and several basic heuristics derived from the graph theoretical models.

4 Constructive heuristics for examination timetabling

Sequential heuristics and clustering heuristics are the two major categories of constructive heuristics for examination timetabling. Sequential constructive heuristics for examination timetabling problem can be divided into three main heuristic components: exam selection heuristics, period selection heuristics (scheduling heuristics) and room selection heuristics. We propose the following heuristics for exam, period and room selection:

4.1 Exam selection heuristics

1. **Intersections heuristics:** The unscheduled exams with highest number of students in common with other scheduled and unscheduled exams will be scheduled next. This is similar to largest degree first heuristic for the graph colouring problem with the following difference: for an exam *e*, we define its intersection value as the number of intersecting exams multiplied by the total number of students in common with other exams. For example, if exam e has intersections of 2, 4 and 1 students with three other exams (say n=3 and $s=2+4+1=7$), the intersection value of *e* is defined as: $3\times7=21$. Note that this will give higher priority to exams with larger number of intersecting exams (see 3.1). To provide an ordering for all the exams in ascending order, the actual value used in our procedure is:

$$\frac{1}{sn+1} \qquad (2)$$

2. **Restrictions heuristics:** Some of exams are restricted to be held at pre-specified periods or venues. Dealing with such exams in later stages of scheduling process may create problems due to usage of their corresponding rooms and periods for other exams. In this heuristic we prioritise restricted exams to be scheduled first. The restrictions number is calculated as: $p \times r$ where p is the number of available periods and r is the number of available rooms.
3. **Available Periods:** This heuristic dynamically finds the number of available periods where an exam can be scheduled without penalty for each exam *e* each time an examination is added. Availability of a period is checked against clashes, room and period pre-assignments, duration of exam and size constraints. This is a generalisation of least saturation degree for graph colouring problem where we consider constraints other than clashes for checking the availability of each period. At each iteration, the exam with the minimum number of available periods is selected to be scheduled.
4. **Available resources:** In the previous heuristic, the number of available rooms in each period is not considered. Note that, for two exams with an equal number of available periods, the exam with a smaller number of available rooms should be prioritised. In this algorithm if for an exam *e, m* periods and *n* rooms are available, the priority value of *e* is defined as:

$$n + \frac{n}{t}\mathrm{m} \qquad (3)$$

where t is the total number of periods available. This adds the total number of available rooms to the average number of available rooms per period multiplied by the number of periods available. The $\frac{n}{t}$ factor is a normalisation factor for the number of periods for changing it to the scale of number of rooms.

5. **Combined heuristic using heuristic ranking:** Some of the above heuristics may use measures which have limited range of effectiveness. For example, the restriction heuristic will only differentiate between exams with rooms and periods restrictions and the other exams will be considered to have the same priority. Using other heuristics to break the ties when a heuristic fails to differentiate between two exams is a reasonable option. The combination heuristic uses an ordering of different measures of importance for an exam such as size of its order constraints, number of clashes, number of available rooms and periods. For dealing with more complex constraints such as coincidental exams (which needs a group of exams to be aggregated as one exam) an extra priority value is added to the other related exams after a member of their group is scheduled. This reduces the possibility of other exams occupying the same period before members of the coincidental exams group. The scheduling heuristic will use the potential cost value (section 3.2) to locate suitable periods for these exams when considering its related exams.
6. **Random selection of exams:** A random heuristic is included in the pool of exam selection heuristics to examine the ability of our algorithm in learning to escape from heuristics of low quality and to provide a mechanism for diversification.

3.2 Period selection heuristics

After selecting an exam for scheduling, a period needs to be selected. The main approach for this process is to select a period with minimal violations of constraints for the exam. This mimics the human schedulers criteria in selecting periods for scheduling based on significance of their violations of constraints. (Burke and Newall 1999) use penalty values including soft intersection violation values and another measure which will encourage scheduling exams with intersecting exams in common to the same period. Our approach uses a more general penalty function which contains violations of all the constraints. This will find a period in which the least combination of clashes, consecutive exams, order constraint violations, size violations, duration excess and pre-specified rooms and periods violations occur.

1. **Penalty based scheduling heuristic:** in this algorithm for a given exam *e*, the potential penalty of assigning *e* to that period is calculated and the period with the minimum penalty is selected. Note that, the penalty function is a weighted sum of violations of all the constraints and hence this selection will minimise the violations of constraints. This single criterion in dealing with all the different constraints is the central idea when designing generic and flexible constructive heuristics to deal with different constraints and their different levels of importance.
2. **Random selection of period**: similar to 4.1.6., but for periods.

3.3 Room selection heuristics

After an exam is selected, different periods are examined for the availability of rooms in the order induced by the period heuristic. For each period, a list of permitted rooms for the exam (room restriction constraint) are ordered based on following heuristics:

1. **Best fit heuristic:** this heuristic will find the smallest room with enough remaining capacity for the exam;
2. **Largest-first heuristics:** in this heuristic priority is given to fill the largest spaces available (sports halls, large lecture theatres). This policy will be reasonable in the context of optimising usage of large spaces to minimse number of venues and invigilators engaged;
3. **Random selection of rooms:** similar to 4.1.6., but for rooms.

If there are no rooms available in a period without penalty, the periods are examined in increasing order of penalty value. If no single room can accommodate the exam, the exam is split into more than one room in the period with lowest penalty. It will be desireable to assign some penalty for splitting of an exam to be able to decide on the right balance between the penalty of period and penalty of splitting. To minimise the number of splits, the splits are delayed until all the available periods are examined. This may degrade the quality of the solution in terms of the objective function, as no cost is associated with splitting an exam. This approach manages to accommodate all the exams with room restrictions in our real problem instance and hence no post-optimisation of space allocation needs to be performed.

Above heuristics define a set:

$$H = \{ \ (h_1, O, h_2, h_3, w_1, ..., w_9) \, | h_i \in \mathbb{N}, i = 1, 2, 3, 1 \le h_1 \le 6, 1 \le h_2 \le 2, 1 \le h_3 \le 3, O \text{ is a permutation of 5 digits out of 9 digits } 00012345, \mathrm{w}_j \in [0,1000], j = 1, ...9 \ \}$$

of heuristics for examination timetabling, where h_1, h_2, h_3 are heuristics for exam, period and room selections, respectively. If $h_1=6$, then using permutation O a combination of different exam selection heuristics are used. If $h_2=1$, then period selection heuristic is based on weights $w_1,..., w_9$, which are used to define the level of severity of different constraint violations in the decision function of the period selection (scheduling) heuristic. For this class of heuristics a parameter α will control the amount of perturbation to be increased if no improvements were found at the early stages of the search. Excluding the variations of the heuristics based on weights or α, the number of different combinations of heuristics is 90750. This approach of using combinations and weighted decision functions introduces a very large space of heuristics for examination timetabling. Hence, it is necessary to develop a search algorithm for finding the best heuristic for a given instance, because the performance of these heuristics may vary from one instance to another.
In the next section we propose two neighbourhood structures to be searched by efficient algorithms similar to the variable neighbourhood search approach. This method is inspired by human method's of selection and manipulation of heuristics in our experiments with the HuSSH and the STARK systems.

5 Heuristic Search Algorithm

In designing a local search algorithm, instead of using the concept of moves to define neighbourhoods, one may use problem specific heuristics to define neighbourhood of constructive solutions. Storer, Wu et al. (1992)) used this method to define heuristic space neighbourhoods for the Job Shop Scheduling problem by utilizing a parametrised family of heuristics defined as the weighted linear combination of dispatching rules. This method of encoding the solutions by means of heuristics is also called "indirect encoding" of the solutions (Terashima-Marin, Ross et al. 1999) and provides an easy way for creating different neighbourhood structures by means of using constructive heuristics. This method works at a higher level than normal neighbourhood structures by manipulating the methods of generating solutions (heuristics), rather than working directly on the solutions.
For an instance **p** of the examination timetabling problem, a heuristic algorithm **h** can be considered as a mapping of **p** into its associated solution **s** in the solution space S, i.e. **s=h(p).** Through solving **p**, a set of heuristics H will be mapped into a subset S_H of S. Using this sets one can define corresponding neighbourhoods in the heuristic space and the solution space.

Definition: for a given parametric heuristic **h**, a neighbourhood of **h** is defined by the means of perturbing its parameters with a perturbation algorithm. The neighbourhood **h** induces a neighbourhood for the solution **s=h(p)** in the solution space. In other words, we define the perturbation based neighbourhood of **s** as all the solutions **s**' produced by **h**', where **h**' is derived from **h** by a perturbation of its parameters.

Example: in our heuristic search algorithm, the concept of perturbation is used as perturbation of parameters (weights) and also as changing the heuristic components of ***h***. For example if:

$$h = (6, 501030024, 1, 1, 800, 100, 300, 300, 300, 200, 200, 100, 500)$$

then some of the perturbations of this heuristic are:

- Changing the permutation S:

$$h = (6, 150300240, 1, 1, 800, 100, 300, 300, 300, 200, 200, 100, 500)$$

- Changing type of the heuristic:

$$h = (1, -, 1, 1, 800, 100, 300, 300, 300, 200, 200, 100, 500)$$

note that, as $h_1 \neq 6$, the permutation *O* does not need to be considered.

- Changing the weights:

$$h = (6, 501030024, 1, 1, 801, 98, 296, 302, 301.9, 198.75, 199, 100, 499)$$

Different perturbation algorithms generate different neighbourhoods. As, there is a clear difference between nature of different dimensions of H, namely heuristic types and the weights, we define our perturbation algorithms to perturb either the heuristic types (and orders) or the weights. This defines two different types of neighbourhoods in subsets of H with smaller dimensions. We use an approach similar to the variable neighbourhood search approach to switch between two neighbourhoods during the search process.

The first type of neighbourhood of a heuristic $h = (h_1, O, h_2, h_3, w_1, ..., w_9)$, $N_1(h)$, keeps the heuristic types the same as h and perturbs the weights randomly by $\alpha\%$. The value of α is initialised by $0.1\% + \frac{l_1}{l_2}$, where l_1 is the number of iterations from last improvement and l_2 is the total number of iterations. This value is reset to 0.1% after an improving heuristic is found.

The second type of neighbourhood of the heuristic $h = (h_1, O, h_2, h_3, w_1, ..., w_9)$, $N_2(h)$,is defined as follows:

$$N_2(h) = \left\{ h' \,\middle|\, h' = (h_1', O', h_2', h_3', w_1, ..., w_9) \right\}$$

where $h_i', i = 1, 2, 3$,are selected randomly from the appropriate ranges and $O' = Next\_Permutation(O)$. Note that weights of h are the same as weights of h' . The function *Next_Permutation* from C++ standard template library(STL) derives the next permutation by minor modifications to the current permutation.

We use a descent local search in a variable neighbourhood search framework to explore the above neighbourhoods. Our algorithm starts by sampling the heuristic space by generating 50 random combinations of heuristics. The best heuristic out of 50 iterations, namely h_B , is selected as the initial solution of the search algorithm. In the first phase of the search, $N_2(h)$ is searched for r_2 iterations. The value of α in this neighbourhood icreases gradually to include larger layers in the neighbourhood. If an improvement is found, the weight of h_B is reset to the associated weight of the improving heuristic and $N_2(h_B)$ for the new heuristic is searched. If no improvements are found in $N_2(h_B)$, then $N_1(h_B)$ is searched .

The intuition behind this search strategy is to try to optimise the weights associated with the current combination of h_B .If better weights are found, algorithm tries to optimise the combination of heuristics with these weights and the process continues until a total number of iterations has reached.

Note that the quality of the solutions in S_H , the set of all the solutions generated with heuristics in *H,* is dependant on the quality of the heuristics defined in *H*: one may include more exhaustive heuristics (such as Simulated annrealing, tabu search, etc.) in *H* to ensure the existance of higher quality of the solutions in S_H . Using a high quality constructive heuristic **h** with a small amount of perturbation, the quality of the resulting neighbourhood is expected to be high. The high quality of the neighbourhood will ensure that even with a simple local search the algorithm may produce good results(Ahmadi 1998; Ahmadi and Osman 2003).

Hence, the aim of the search process is to find near optimal sequences of heuristics in S_H (solutions generated by set of heuristics H) rather than in S (complete set of solutions). This approach also proposes a search method based on perturbation of parameters to the common problem of fine-tuning the parameters of computational procedures.

6 Computational results

To show the ability of the algorithm in finding the approximate range of the weights of the constraints and the sequences of the heuristics for solving the given instance effi ciently, the algorithm was executed several times with different random starts. We used a real data set from the Nottingham University in year 1994 due to its rich set of constraints and the heuristics described in section (4) in our experiment. This data set

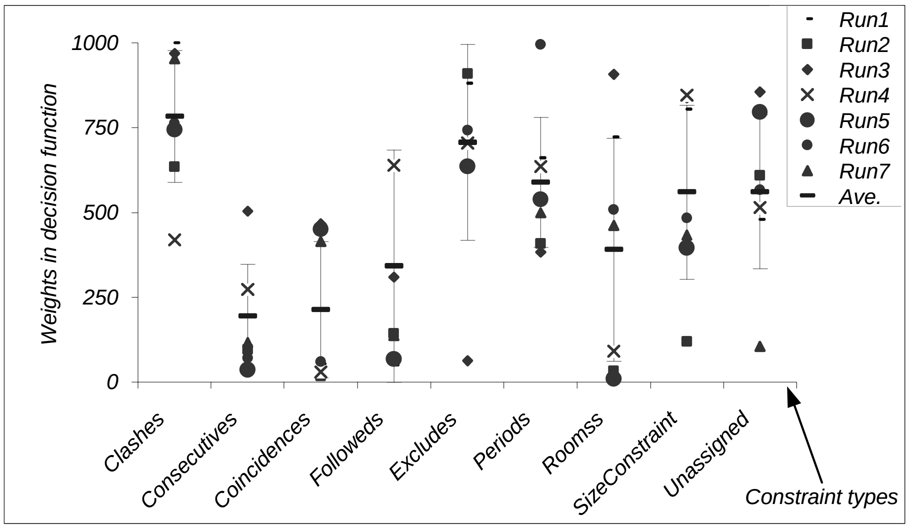

Figure 1: Best weights of constraints in the period selection decision function in 7 independent random runs

includes 7896 students, 800 exams, 33998 enrollments and all the 4 classes of different constraints described in section 3.
To increase the statistical significance of the experiment, several random start experiments were conducted.

Run No.	BST50R	AVF50R	BST	BST-HEURS	ITER	tag	α
5	4075	108734.1	2568	5-135402-0-0	337	1	0.015
3	11678	116307.5	2710	5-500100324-0-2	300	1	0.007
6	4650	156326.58	3712	5-513004002-0-1	90	1	0.039
1	8354	197558.0	4400	5-351402-0-0	415	1	0.009
4	7315	115338.74	4774	5-150000234-0-1	368	1	0.005
2	11487	137457.7	6008	2-X-0-2	423	1	0.027
7	13782	107028.8	6060	1-X-0-1	500	2	0.039

Table 1: Results of different random start runs

Rows of the Table 1 are sorted in the order of the runs where the best solutions were found. The first column shows the run number. The second and the third columns show the total penalty values of the best solution found (BST50R) and the average solutions (AVF50R) of the first 50 randomly generated solutions, respectively. In the next two columns, the best results of the heuristic space search improvement in every run are reported. This includes the best objective function value (BST), the best combinations of heuristics (BST-HEURS) and the iteration number (ITER) which the best solution was created. The last two columns present the type of the neighbourhood (tag) and the amount of perturbation (**α**), for the best solution. As the results demonstrate, the algorithm finds "combination heuristics" for exam selection as the best heuristic in this class in 5 out of 7 of the runs. The results also suggest the combinations of of heuristics 1 and 5 to be the best combination of the exam selection heuristics in all the 5 cases. This suggests that most of the random runs find similar patterns of heuristics as the best heuristics for the Nottingham data set. This result is quite significant due to the large space of the heuristics and the small number of total iterations, 500.

A second observation is the value of the weights of the constraints in period selection heuristics. There are an infinite number of combinations of weights in the [0,1000]. Due to the complex relations between changes of weights and their effect on other constraints, finding the right values (or ranges of values) for the weights is a complex decision. Figure 1, shows the weights of the constraints found for the best solution of all the randomly started runs. In this figure each column shows the weights of one of the constraints in different runs. For example weights of clash constraints are in the first column of the graph and their values are 999.768, 633.522, 968.234, 419.51, 744.362, 763.224, 952.837 and 852.015. We also report the average weights and average plus and minus standard deviation of the weights (vertical line). Results of our 7 randomly started runs, shows that the weights associated with best solutions found in all the random runs are quite homogenous and algorithm manages to suggest specific ranges for the weights of different constraints. Note that this approach will find

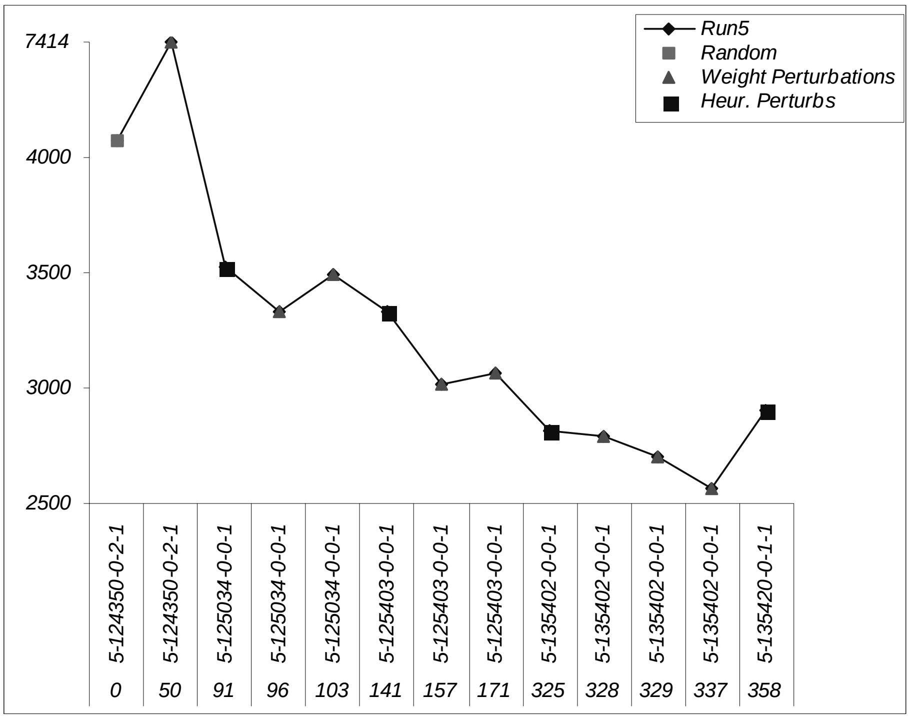

Figure 2: Best solutions in each neighbourhood explored by the search algorithm for the 5th run

the weights suitable for solving each given instance of the problem and hence could be viewed as a valuable tool for finding instance-specific information. The averages (Ave), which can be used to define the order of importance and severity of the constraints, are very similar to the weights defined experimentally for Nottingham 94 data set in PATAT 2002 competition and our other experiments.

Figure 2 summarises the search procedure for the best run (Run 5) by showing the quality of the best solutions in the explored neighbourhoods. On the horizontal axis the combination of heuristics used to generate the solution and its iteration number are reported. Neighborhoods are generated by fixing the heuristics and perturbing the weights of h_B and also by fixing the weights of h_B and perturbing its heuristics. The process of alternating between two different types of neighbourhoods, demonstrates our Variable Neighbourhood Seach procedure in perturbation based neighbourhoods.

Success of 5th run in relatively small number of iterations is mainly due to its good starting solution, 5-000124350-0-2. The search procedure manages to refine the structure of the heuristics and their associated weights of constraints even further. For examples of cases, starting from more different starting points see Figure 3. Quality of all the visited solutions during the search procedure for run 5 are reported in Figure 4.

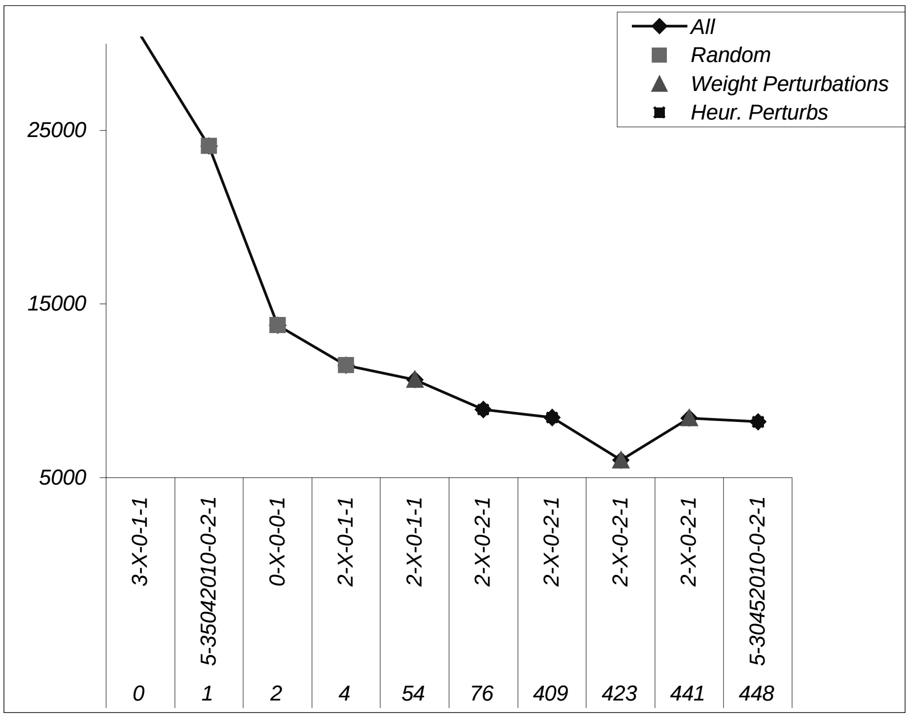

Figure 3: Best solutions in each neighbourhood explored by the search algorithm for another run

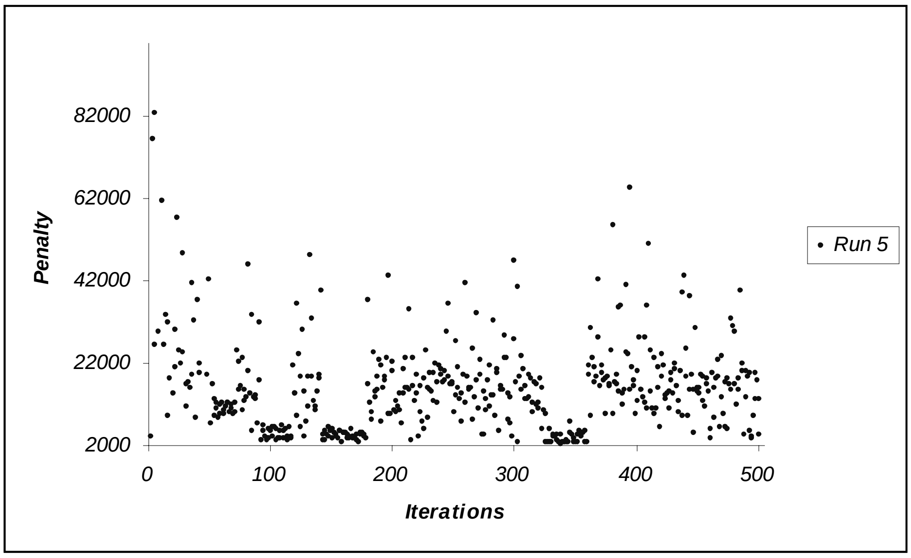

Figure 4: quality of all the visited solutions during the search procedure for run 5

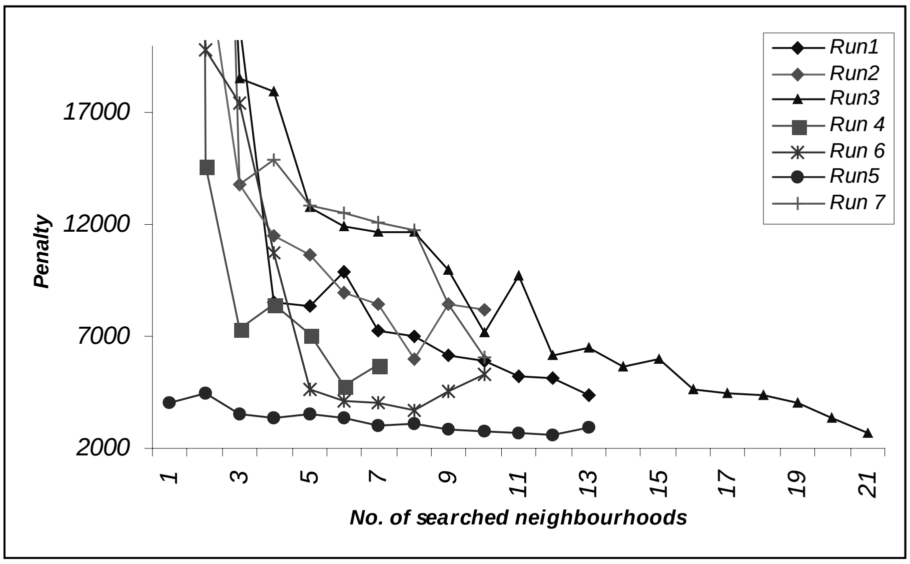

Figure 5: Summary of the search procedure in terms of number of visited neighbourhoods in each run

Figure 5 summarizes quality of the best solutions in visited neighbourhoods for all the runs. As the graph shows, different runs take different number of neighbourhoods to find their best solution. In general the algorithm shows robustness in terms of searching the space of heuristics efficiently with different random starts. This may provide a good tool to be used in the context of the selection of heuristics for solving different instances of a problem and also in finding the values of parameters in experimental design.

7 Further research

In this paper we presented an approach for searching the space of heuristics for examination timetabling problem with perturbation based neighbourhood structures. The algorithm is promising in terms of its capacity to find good combinations of heuristics and their associated weights in a small number of iterations. This may provide a good tool to be used in the context of selection of heuristics for solving different instances of a problem and also in finding the values of parameters in experimental design. Further work will investigate application of this procedure on other datasets from the examination timetabling literature and also on selecting local search heuristics. Similarities between this approach and human strategies of discovery will be investigated.

References

1. Ahmadi, S. (1998). Metaheuristics for the Capacitated Clustering Problem. Operational Research. Canterbury, University of Kent at Canterbury**:** 223.
2. Ahmadi, S., R. Barone, et al. (2002). Integrating human abilities and automated systems for timetabling: A competition using STARK and HuSSH representations at the PATAT 2002 conference". 4th international conference on the practice and theory of automated timetabling (PATAT), KaHo St.-Lieven, Gent Gent, Department of Industrial Engineering, Belgium.
3. Ahmadi, S. and I. H. Osman (2003). "Density based problem space search for capacitated clustering problem." special issue of the Annals of Operations Research on Metaheuristics.
4. Arani, T. and V. Lotfi (1989). "A 3-Phased Approach to Final Exam Scheduling." Iie Transactions **21**(1): 86-96.
5. Burke, E. K., Y. Bykov, et al. (2001). "A Multicriteria Approach to Examination Timetabling." Lecture Notes in Computer Science **2079**: 118--??
6. Burke, E. K. and J. P. Newall (1999). "A multistage evolutionary algorithm for the timetable problem." Ieee Transactions on Evolutionary Computation **3**(1): 63-74.
7. Carter, M. W., G. Laporte, et al. (1997). "Examination timetabling: Algorithmic strategies and applications (vol 47, pg 373, 1996)." Journal of the Operational Research Society **48**(2): 225.
8. Cheng, P. C.-H., R. Barone, et al. (2003). Integrating human abilities with the power of automated scheduling systems: Representational epistemological interface design. AAAI Spring Symposium on Human Interaction with Autonomous Systems in Complex Environments, Palo Alto, CA, AAAI Press.

9. Cowling, P. I., S. Ahmadi, et al. (2002). Combining Human and Machine Intelligence to Produce Effective Examination Timetables. The 4th Asia-Pacific Conference on Simulated Evolution And Learning (SEAL2002), Singapore.
10. McCollum, B., S. Ahmadi, et al. (2002). A review of existing interfaces of automated examination and lecture scheduling systems. The 4th international conference on the practice and theory of automated timetabling (PATAT), KaHo St.-Lieven, Gent Gent, Department of Industrial Engineering, Belgium.
11. P.Cheng, R.Barone, et al. (2002). Opening the information bottleneck in complex scheduling problems with a novel representation: STARK diagrams, Springer-Verlag, Heidelberg.
12. Ross, P., E. Hart, et al. (1998). "Some observations about GA-based exam timetabling." Practice and Theory of Automated Timetabling Ii **1408**: 115-129.
13. Storer, R. H., S. D. Wu, et al. (1992). "New Search Spaces for Sequencing Problems with Application to Job Shop Scheduling." Management Science **38**(10): 1495-1509.
14. Terashima-Marin, H., P. Ross, et al. (1999). Evolution of constraint satisfaction strategies in examination timetabling. Gecco-99: Proceedings of the Genetic and Evolutionary Computation Conference. San Francisco, MORGAN KAUFMANN PUB INC**:** 635-642.
15. Thompson, J. M. and K. A. Dowsland (1998). "A robust simulated annealing based examination timetabling system." Computers & Operations Research **25**(7-8): 637-648.