Landa-Silva, Dario and Obit, Joe Henry (2011) Comparing hybrid constructive heuristics for university course timetabling. In: VII ALIO/EURO Workshop on Applied Combinatorial Optimization, May 4 - 6, 2011, Porto, Portugal.

Comparing Hybrid Constructive Heuristics for University Course Timetabling

Dario Landa-Silva * Joe Henry Obit †

* ASAP Research Group, School of Computer Science
University of Nottingham, United Kingdom
dario.landasilva@nottingham.ac.uk

† Labuan School of Informatics Science
University Malaysia Sabah, Malaysia
Research carried out while J. Obit was a PhD student in Nottingham.
joehenryobit@yahoo.com

ABSTRACT

This extended abstract outlines four hybrid heuristics to generate initial solutions to the University course timetabling problem. These hybrid approaches combine graph colouring heuristics and local search in different ways. Results of experiments using two benchmark datasets from the literature are presented. All the four hybrid initialisation heuristics described here are capable of generating feasible initial timetables for all the test problems considered in these experiments.

Keywords: Course timetabling, Hybrid heuristics, Event scheduling, Constructive heuristics

1. INTRODUCTION

We refer to the University course timetabling problem as described by Socha et al. [1] with: n events $E = \{e_1, e_2, \ldots, e_n\}$, k timeslots $T = \{t_1, t_2, \ldots, t_k\}$, m rooms $R = \{r_1, r_2, \ldots, r_m\}$ and a set S of students. Each room has a limited capacity and some additional features. Each event requires a room with certain features. Each student attends a number of events which is a subset of E. The problem is to assign the n events to the k timeslots and m rooms in such a way that all hard constraints are satisfied and the violation of soft constraints is minimised.

The *hard constraints* that must be satisfied for a timetable to be feasible are as follows. *HC1*: a student cannot attend two events simultaneously, i.e. events with students in common must be timetabled in different timeslots. *HC2*: only one event may be assigned per timeslot in each room. *HC3*: the room capacity must be equal to or greater than the number of students attending the event in each timeslot. *HC4*: the room assigned to an event must satisfy the features required by the event. The *soft constraints* that are desirable to satisfy in order to assess the quality of a timetable are as follows. *SC1*: students should not have only one event timetabled on a day. *SC2*: students should not attend more that two consecutive events on a day. *SC3*: students should not attend an event in the last timeslot of a day.

It has been shown in the literature that a *sequential heuristic* method can be very efficient for generating initial solutions [2, 3]. A sequential heuristic assigns events one by one, starting from the event which is considered the most difficult to timetable in some sense. The ‘difficulty’ of scheduling an event can be measured by different criteria (i.e. the number of other conflicting events or the number of students attending the event). However, a sequential heuristic alone does not guarantee that feasible solutions will be found even with the combination of more than one heuristic. For example, Abdullah et al. [4] proposed a method, based on a sequential heuristic, to construct initial timetables. However, their method failed to generate a feasible solution for the large instance of the Socha et al. problem instances [1].

We propose hybrid heuristics to create initial feasible timetables for the University course timetabling problem described above. We combine traditional graph colouring heuristics with various local search methods including a simple tabu search. In the experiments of this work we use the 11 benchmark data sets proposed by Socha et al. [1] and also the set of problem instances from the International Timetabling Competition (ITC) 2002 [5]. The proposed heuristics generate feasible timetables for all the instances in our experiments. However, these methods do not tackle the satisfaction of soft constraints. Then, we obtain feasible solutions that might still have relatively high number of soft constraint violations. The rationale for this is to allow flexibility for another algorithm, that seeks to improve the satisfaction of constraints, to start the search from the feasible timetables. This has proven to be beneficial in our related work helping the improving algorithm to achieve extremely good results [6, 7]. It is difficult to compare the results in this paper with the literature because most other works (e.g. [3]) incorporate the construction of initial timetables within the overall method to solve the problem, i.e. constructing initial solutions and improving them are combined into a single approach. The next section describes the proposed hybrid heuristics.

2. GENERATING INITIAL TIMETABLES

In order to develop effective algorithms for tackling hard constraints in the subject problem, we combine techniques such as graph colouring, local search and tabu search. We found that the search components incorporated in the hybrid methods are interdependent on their ability to produce a feasible timetable. In other words, when one of these components is disabled or removed, the remaining components are not able to produce feasible solutions in particular for medium and large instances. Therefore, the hybrids described next are effective tailored mechanisms to generate feasible timetables for the subject problem.

2.1. Largest Degree, Local Search and Tabu Search (IH1)

We adopted the heuristic proposed by Chiarandini et al. [8] and added the Largest Degree heuristic to Step one as described next. Largest Degree refers to the event with the largest number of conflicting events (events that have at least one student in common).

Step one - Largest Degree Heuristic. In each iteration, the unscheduled event with the Largest Degree is assigned to a timeslot selected at random without respecting conflicts between events. Once all events have been assigned into a timeslot, the maximum matching algorithm for bipartite graph is used to assign each event to a room. At the end of this step, there is no guarantee for the timetable to be feasible. Then, steps one and two below are executed iteratively until a feasible solution is constructed.

Step two - Local Search. We employ two neighbourhood moves in this step. Move one (M1) selects one event at random and assigns it to a feasible pair timeslot-room also chosen at random. Move two (M2) selects two events at random and swaps their timeslots and rooms while ensuring feasibility is maintained. That is, neighbourhood moves M1 and M2 seek to improve the timetable generated in Step one. A move is only accepted if it improves the satisfaction of hard constraints (because the moves seek feasibility). This step terminates if no move produces a better (closer to feasibility) solution for 10 iterations.

Step three - Tabu Search. We apply a simple tabu search using a slight variation of move M1 above. Here, M1 only selects an event that violates hard constraints. The motivation is that the algorithm should now only target events that violate hard constraints instead of randomly rescheduling other events like in Step two. The tabu list contains events that were assigned less than tl iterations before calculated as $tl = rand(10) + \delta \times n_c$, where $0 \leq rand(10) \leq 10$, n_c is the number of events involved in hard constraint violations in the current timetable, and $\delta = 0.6$. The usual aspiration criterion is applied to override tabu status, i.e. accept the move when a best known solution is found. This step terminates if no move produces a better (closer to feasibility) solution for *ts* iterations.

2.2. Saturation Degree, Local Search and Tabu Search (IH2)

This heuristic uses Saturation Degree, which refers to the number of resources (timeslots and rooms) still available to timetable a given event without conflicts in the current partial solution. In the previous heuristic IH1 the assignment of events in Step one is done without checking conflicts. The difference in heuristic IH2 is that we first check conflicts between the unassigned event and those events already assigned to the selected timeslot. If there are timeslots with no-conflicting events already assigned (saturation degree of the event to assign is greater than zero), the event is assigned to a feasible timeslot selected at random. If there are no such timeslots (saturation degree of the event to assign is zero), the events already assigned to the timeslot are ejected and put in a list of events to re-schedule. The heuristic then attempts to re-assign these ejected events into conflict free timeslots if possible. Otherwise, these ejected events are put into random timeslot-room, even if conflicts arise, then later the local and tabu search of Step two and Step three as described above, will deal with these ejected events and the remaining conflicting assignments. In essence, in addition to using Saturation Degree instead of Largest Degree, this second heuristic IH2 tries to fix some conflicts in the timetable before starting Steps two and three.

2.3. Largest Degree, Saturation Degree, Local Search and Tabu Search (IH3)

This heuristic incorporates both Largest Degree and Saturation Degree. The difference with heuristic IH2 is that in Step one, events are first sorted based on Largest Degree. After that, we choose the unassigned event with the Largest Degree and calculate its Saturation Degree. Then, Step one of this heuristic IH3 proceeds as in heuristic IH2, but when attempting to re-assign the ejected events, only those ejected events with Saturation Degree greater than zero (still available timeslots and room) are assigned to any feasible timeslot-room. All ejected events with Saturation Degree zero are moved from the re-schedule list to the list of unscheduled events. After each re-assigning, we re-calculate the Saturation Degree for all ejected events in the re-schedule list. This process in Step one continues and if after some given computation time there are still events in the unscheduled list, these events are then assigned to random timeslot-room without respecting conflicts. Steps one and two as described above follow implementing the local and tabu search respectively. In essence, compared to heuristic IH2, this heuristic IH3 combines Saturation Degree and Largest Degree in Step one trying to re-scheduled ejected events with less resources first. Algorithm 1 shows the pseudo-code for the hybrid heuristic IH3, which in a sense, is the most elaborate one among methods IH1, IH2 and IH3.

2.4. Constraint Relaxation Approach (IH4)

In this fourth heuristic approach, we introduce extra *dummy* timeslots to place events with zero Saturation Degree and in this way enforce the no-conflicts constraint by relaxing the availability of timeslots. The number of extra dummy timeslots needed is determined by the size of the problem instance. This heuristic works as follows. First, we sort the events using Largest Degree. The event with the Largest Degree is chosen to be scheduled first. If the event has zero Saturation Degree, the event is assigned randomly to one of the extra dummy timeslots. Once the algorithm assigns all events in the valid timeslots plus the extra *dummy* timeslots without conflicts, we then perform great deluge search [6] using moves M1 and M2 to reduce the number of timeslots down to 45 valid timeslots if necessary. In this local search, only the 45 valid timeslots are considered, so no events are allowed to move into any of the extra dummy timeslots. This hybrid heuristic is much slower that the other three methods above, mainly due to the great deluge search. Algorithm 2 shows the pseudo-code for the hybrid heuristic IH4, which in a sense, is the most different among all methods described here.

3. RESULTS AND DISCUSSION

The proposed hybrid heuristic initialisation methods were applied to the Socha et al. [1] instances and also to the ITC 2002 instances [5]. We did not impose time limit as a stopping condition, each algorithm stops when it finds a feasible solution.

All methods successfully generate initial solution for small instances in just few seconds. The medium and large Socha et al. instances are more difficult as well as all ITC 2002 instances. However, the proposed methods generated feasible solutions for all instances demonstrating that the hybridisation compensates weakness in one component with strengths in another one in order to produce feasible solutions in reasonable computation times.

Table 1 and Table 2 compare the performance of each method on the Socha et al. and the ITC 2002 instances respectively. The first column in each table indicates the problem instance. The next four columns give the best objective function value (soft constraints violation) obtained by each heuristic. The last column in each table indicates the best computation time in seconds and the corresponding heuristic.

The results show that none of the heuristics clearly outperforms the others in terms of the objective function value (soft constraints violation) obtained. Each of the four heuristics outperforms the other three in some of the problem instances. With respect to computation time we can see in Table 1 that for the Socha et al. problems, the heuristic that achieved the best objective value was almost never the fastest one (except in problem instance M2). However, for the ITC 2002 problems, we see in Table 2 that in several cases the heuristic producing the best objective value was also the

Algorithm 1: Initialisation Heuristic 3 (IH3)

```
Input: List of Unscheduled events E;
Sort E by non-increasing Largest Degree (LD);
while (E is not empty) do
    Choose event e from E with LD (random tie-break);
    Calculate SD for event e;
    if (SD = 0) then
        Select a timeslot t at random;
        Move events scheduled (if any) in timeslot t that
        conflict with event e (if any) to the Reschedule list;
        Assign event e to timeslot t;
        for (each event in Reschedule list with SD > 0) do
            Select feasible timeslot t for event e at
            random;
            Re-calculate SD for all events in Reschedule
            list;
        end
        Move all events with SD = 0 that remain in
        Re-schedule list to the Unscheduled list E;
    end
    else
        Select a feasible timeslot t at random for event e;
    end
    if (Unscheduled list E is not empty and time has
    elapsed) then
        One by one, place events from the Unscheduled
        list into any random selected timeslot without
        respecting the conflict between the events;
    end
end
S = current solution;
loop = 0;
while (S not feasible ) do
    if (loop < 10) then
        if ( coinflip()) then
            S* = M1(S); // apply M1 to S
        end
        else
            S* = M2(S); // apply M2 to S
        end
        if ( f(S*) ≤ f(s)) then
            S ← S* // accept new solution;
        end
    end
    else
        EHC = set of events that violate hard constraints;
        e = randomly selected from EHC;
        S* = M1(S, e); // perform one Tabu Search
        iteration with move M1 using event e;
        if ( f(S*) < f(S) then
            S ← S*; // accept new solution
        end
        if (loop >= ts ) then
            loop = 0;
        end
    end
    loop++;
end
Output: S feasible solution (timetable);
```

Algorithm 2: Initialisation Heuristic 4 (IH4)

```
Input: List of Unscheduled events E;
Generate dummy timeslots according to problem instance;
Sort events in E by non-increasing Largest Degree (LD);
while (Unscheduled list E is not empty) do
    Choose event e from E with the LD (random tie-break);
    Calculate SD for event e;
    if (SD = 0) then
        Select dummy timeslot at random for event e;
    end
    else
        Chose any feasible timeslot for event e;
        Update the new solution;
    end
end
S = current solution;
Calculate initial cost function f(S);
Initial water level B = f(S);
ΔB = 0.01;
while (dummy timeslots are not empty) do
    if ( coinflip()) then
        S* = M1(S); // apply M1 to S
    end
    else
        S* = M2(S); // apply M2 to S
    end
    if (f(s*) ≤ f(s)) or (f(s*)(≤ B)) then
        S ← S*; // accept new solution
    end
    B = B − ΔB; // lower the water level
    if (B - f(S) ≤ 1) then
        B = B + 5; // increase the water level
    end
end
Output: S feasible solution (timetable);
```

fastest. As indicated above, the hybrid initialisation heuristic (IH4) that uses *dummy* timeslots to deal with conflicts and then great deluge as the local search to bring the solution to feasibility, is never the fastest approach. However, this heuristic IH4 was capable of producing the best solutions for two of the Socha et al. instances and six of the ITC 2002 instances.

In our preliminary experiments, we implemented a sequential heuristic (see [2, 3]) but were able to generate feasible timetables only for the small instances of the Socha et al. dataset (in fact, these small instances are considered to be easy). Even after considerably extending the computation time, the sequential heuristic was not able to generate feasible solutions for the medium and large Socha et al. instances or the ITC 2002 datasets.

4. CONCLUSIONS

Many approaches have been proposed in the literature to tackle the University course timetabling problem. In this extended abstract we have outlined four variants of hybrid heuristics designed to generate initial feasible solutions to this problem. These hybrid approaches combine traditional graph colouring heuristics, like Largest Degree and Saturation Degree, with different types of local search. The four hybrid variants were tested using two sets of benchmark problem instances, the Socha et al. [1] and the International Timetabling Competition 2002 [5] datasets.

All the hybrid initialisation heuristics described here were capable of producing feasible timetables for all the problem instances.

Problem	IH1	IH2	IH3	IH4	Min Time
S1	**173**	198	207	200	0.077 (IH2)
S2	211	217	**189**	208	0.078 (IH2)
S3	**176**	190	188	209	0.062 (IH2)
S4	250	**174**	203	192	0.047 (IH1)
S5	229	238	226	**217**	0.078 (IH2)
M1	817	**772**	802	774	5.531 (IH3)
M2	793	**782**	784	802	6.342 (IH2)
M3	**795**	867	828	817	6.64 (IH3)
M4	**735**	785	811	795	5.828 (IH2)
M5	773	771	784	**769**	16.670 (IH1)
L	**1340**	1345	1686	1670	300.0 (IH1)

Table 1: Results obtained with each hybrid initialisation heuristic (IH1 to IH4) on the 11 Socha et al. problem instances, best results indicated in bold.

Problem	IH1	IH2	IH3	IH4	Min Time
Com01	805	**786**	805	805	1.93 (IH3)
Com02	**731**	776	**731**	778	1.36 (IH3)
Com03	**760**	812	**760**	777	1.14 (IH2)
Com04	1201	**1178**	1201	1236	4.46 (IH2)
Com05	1246	1243	1246	**1135**	2.11 (IH3)
Com06	1206	1219	1206	**1133**	1.33 (IH3)
Com07	1391	1388	1391	**1265**	2.10 (IH3)
Com08	1001	**968**	1001	1006	1.81 (IH2)
Com09	**841**	859	**841**	843	1.46 (IH1)
Com10	**786**	816	**786**	799	4.64 (IH3)
Com11	852	877	852	**839**	1.05 (IH1)
Com12	814	831	814	**788**	2.21 (IH2)
Com13	**1008**	1010	**1008**	1009	2.26 (IH1)
Com14	1040	**1032**	1040	1355	3.71 (IH2)
Com15	1165	1162	1165	**1161**	1.56 (IH3)
Com16	**887**	911	**887**	888	1.09 (IH3)
Com17	1227	**1032**	1227	1199	1.13 (IH2)
Com18	793	**724**	793	763	1.29 (IH3)
Com19	**1184**	1212	**1184**	1209	3.22 (IH3)
Com20	**1137**	1161	**1137**	1205	0.08 (IH3)

Table 2: Results obtained with each hybrid initialisation heuristic (IH1 to IH4) on the 20 ITC 2002 problem instances, best results indicated in bold.

None of the approaches showed to be clearly better that the others. For a given instance, the heuristic producing the best quality initial timetable is often not the fastest among the four approaches. However, for all the problem instances there is at least one hybrid heuristic capable of generating a feasible timetable in very short time, from less than a second to few seconds depending of the problem instance. The exception is the largest Socha et al. instance which is still regarded in the literature as a very challenging problem. Having some methods capable of generating feasible solutions for the University course timetabling problem is important because the effort of more elaborate methods can then be focused on tackling the violation of soft constraints in order to improve the timetable quality.

In a following more detailed description on this research, we intend to present a statistical comparison between the proposed initialisation heuristics, compare these approaches against other procedures to generate feasible solutions to the University course timetabling problem and analyse the effect of each component in the four hybrid heuristics.

5. REFERENCES

[1] K. Socha, J. Knowles, and M. Samples, "A max-min ant system for the university course timetabling problem," in *Ant Algorithms: Proceedings of the Third International Workshop (ANTS 2002), LNCS 2463*. Springer, 2002, pp. 1–13.

[2] E. Burke, B. A. McCollum, A. Meisels, S. Petrovic, and Q. Rong, "A graph based hyper-heuristic for educational timetabling problems," *European Journal of Operational Research*, vol. 176, pp. 177–192, 2007.

[3] P. Kostuch, "The university course timetabling problem with a three-phase approach," in *The Practice and Theory of Automated Timetabling V, LNCS 3616*. Springer, 2005, pp. 109–125.

[4] S. Abdullah, E. Burke, and B. McCollum, *Using a Randomised Iterative Improvement Algorithm with Composite Neighborhood Structures for University Course Timetabling*. Springer, 2007, pp. 153–172.

[5] B. Paechter, L. M. Gambardella, and O. Rossi-Doria. (2002) International timetabling competition 2002. Metaheuristics Network. [Online]. Available: http://www.idsia.ch/Files/ttcomp2002/

[6] J. H. Obit and D. Landa-Silva, "Computational study of non-linear great deluge for university course timetabling," in *Intelligent Systems - From Theory to Practice, Studies in Computational Intelligence, Vol. 299*, V. Sgurev, M. Hadjiski, and J. Kacprzyk, Eds. Springer-Verlag, 2010, pp. 309–328.

[7] J. H. Obit, "Developing novel meta-heuristic, hyper-heuristic and cooperative search for course timetabling problems," Ph.D. dissertation, 2010.

[8] M. Chiarandini, M. Birattari, K. Socha, and O. Rossi-Doria, "An effective hybrid algorithm for university course timetabling," *Journal of Scheduling*, vol. 9, pp. 403–432, 2006.