Deterministic Optimization of Examination Timetables

Siti Khatijah Nor Abdul Rahim
University of Nottingham (Malaysia Campus)
Prof Andrzej Bargiela
Dr Rong Qu
(University of Nottingham, UK Campus)

Outline

- Introduction
- Overview of the Proposed Optimization Method
- Optimization Methods
- Computational Results & Discussions
- Conclusions & Future Work
- References

2

Introduction

- Timetabling problem exists in numerous areas: educational, transportation, sports, etc
- Educational timetabling; most widely studied
  - School timetabling, university course timetabling, examination timetabling, etc
- Exam timetabling signifies a challenging computational problem between exams (many-to-many relationship between students and exams)
- In the exam timetabling problems, the general objective is to generate feasible schedules which satisfy basic constraints

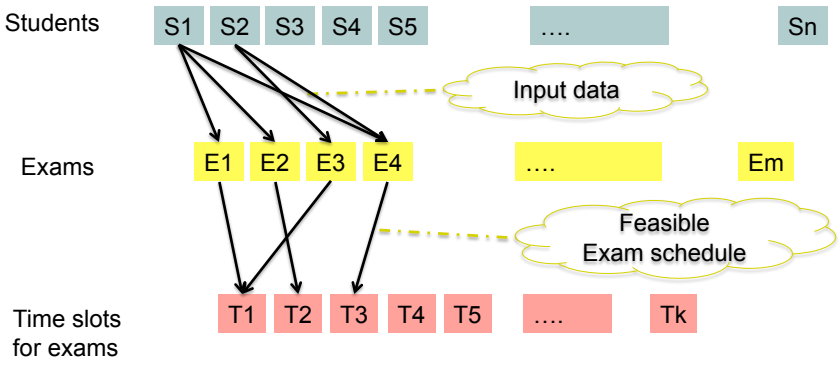
3

Introduction (contd.)

- Two types of constraints:
  - Hard constraints
    - Must be satisfied at all times
      - Principal hard constraint: two exams with a common student cannot be scheduled in the same timeslot
      - Another hard constraint: room capacity must be enough
  - Soft constraints
    - Not critical but beneficial
      - Eg: spreading out exams taken by individual students

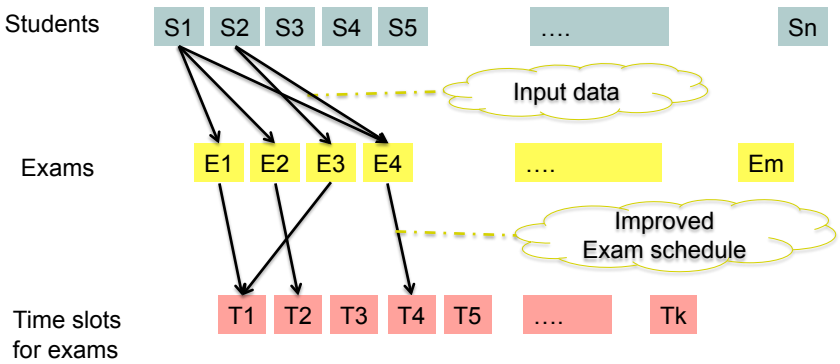
4

Introduction (contd.)



5

Introduction (contd.)



6

Introduction (contd.)

- Published methods for finding examination schedules include:
  - Evolution Strategies,
  - Constraint Logic Programming,
  - Ant Colony,
  - …
  - Other heuristics/metaheuristics

(*The optimisation process involves direct exploration of the large search space and it frequently depends on random choices;*

The optimisation is complex because it involves simultaneous consideration of hard and soft constraints)

7

Proposed Optimization Method

- We separate the consideration of hard and soft constraints
- **Stage 1 (hard constraints)** : We find a feasible solution using a standard graph-coloring approach
- **Stage 2 (soft constraints):** We optimise the schedule by two deterministic processes:
  - Re-ordering the exam-slots (greedy minimisation of cost)
  - Re-assigning of exams to alternative feasible slots (greedy minimisation of cost)

(iterating the two processes above until no improvement is found)

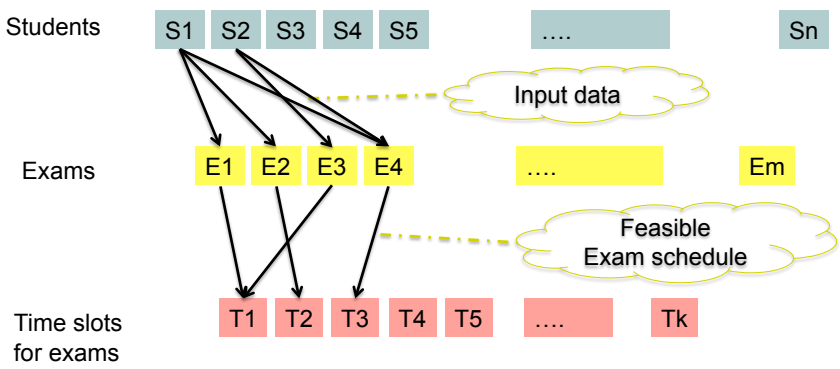
8

Proposed Optimization Method

- The potential advantage of the proposed method derives from :
  - Processing of aggregated (granulated) data entities (re-ordering of exam slots); and
  - Performing re-assignment of exams using the optimised ordering of time slots above.

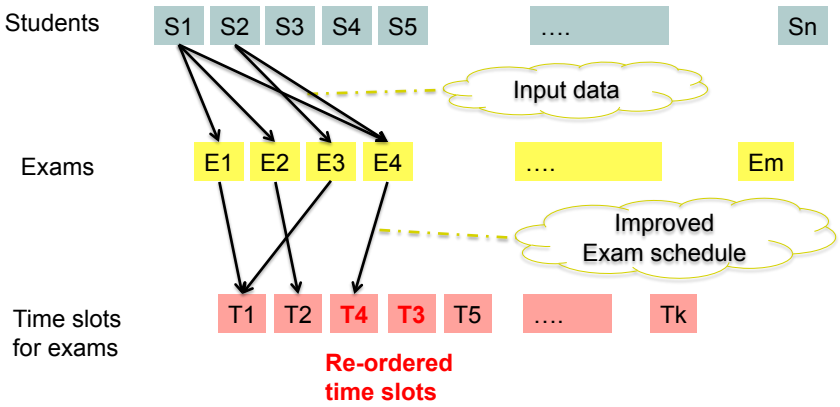
9

Re-ordering of time slots



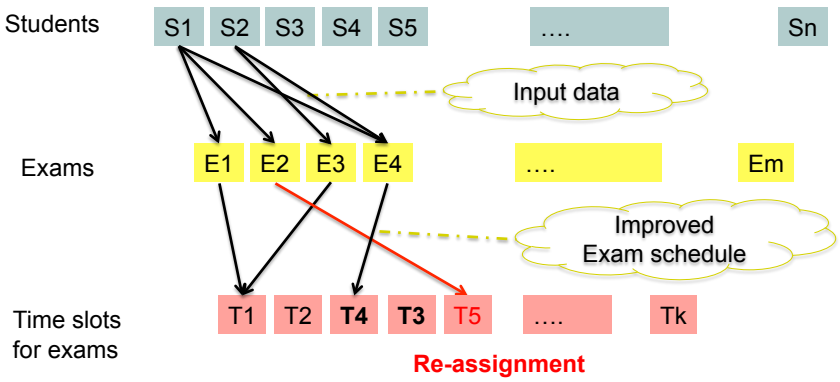
10

Re-ordering of time slots



11

Re-assigning of exams



12

The cost function

- The cost of the timetable is measured using the cost function originally proposed by Carter et al [5]:
- $$\frac{1}{T}\sum_{i=1}^{N}\sum_{j=i+1}^{N} s_{ij} w_{|pj - pi|}$$
- where
  - *N* is the number of exam
  - *sij* is the number of students enrolled in both exam *i* and *j*,
  - *pj* is the time slot where exam *j* is scheduled,
  - *pi* is the time slot where exam *i* is scheduled and
  - *T* is the total number of students.
- Based on this cost function, a student taking two exams that are | *pj* - *pi* | slots apart, where | *pj* - *pi* | ={1, 2, 3, 4, 5 }, leads to a cost of 16, 8, 4, 2, and 1, respectively.

13

Granulation of problem data for Stage 2 optimisation

- Granulated problem data:
  - *Exam conflict matrix* - a square matrix of dimension equal to the number of exams. Entries in this matrix at position (*i*,*j*) represent the number of students causing conflict between exams *i* and *j*.
  - *Spread matrix* is a square matrix of dimension S, where *S* is a number of slots. Entries in the spread matrix at position (p,q) represent the number of students who take both an exam from slot *p* and slot *q*.
  - *Exam-slot conflict matrix* indicating at position (r,s) if exam "r" is in conflict with slot "s" (i.e. there is one or more exams in slot "s" which conflict with exam "r")

14

Stage 2 optimisation example

- An example of a spread matrix

0	1044	1108	918	948	708	628	47	222	7	1
1044	0	1349	1119	1302	593	753	118	322	9	2
1108	1349	0	1282	1198	575	786	166	342	9	3
918	1119	1282	0	921	518	656	95	181	33	4
948	1302	1198	921	0	684	733	159	194	33	5
708	593	575	518	684	0	546	92	23	45	6
628	753	786	656	733	546	0	79	140	43	7
47	118	166	95	159	92	79	0	35	12	8
222	322	342	181	194	23	140	35	0	25	9
7	9	9	33	33	45	43	12	25	0	10
1	2	3	4	5	6	7	8	9	10	0

- The cost function assigns a weight
  - “16” to exams that 1 slot apart (ie blue cells in the spread matrix (1,2),(2,3),(3,4),etc)
  - “8” to exams that are 2 slots apart (ie green cells in the spread matrix (1,3),(2,4),(3,5), etc),
  - And so on…

15

Stage 2 optimisation example

- Since the total number of students is 2749,

0	1044	1108	918	948	708	628	47	222	7	1
1044	0	1349	1119	1302	593	753	118	322	9	2
1108	1349	0	1282	1198	575	786	166	342	9	3
918	1119	1282	0	921	518	656	95	181	33	4
948	1302	1198	921	0	684	733	159	194	33	5
708	593	575	518	684	0	546	92	23	45	6
628	753	786	656	733	546	0	79	140	43	7
47	118	166	95	159	92	79	0	35	12	8
222	322	342	181	194	23	140	35	0	25	9
7	9	9	33	33	45	43	12	25	0	10
1	2	3	4	5	6	7	8	9	10	0

- The cost function evaluates to:

[[(1104 + 1349 + 1282 + 921 + 684 + 546 + 79 + 35 + 25) * 16] +
[(1108 + 1119 + 1198 + 518 + 733 + 92 + 140 + 12)* 8] +
[(918 + 1302 + 575 + 656 + 159 + 23 + 43) * 4] +
[(948 + 593 + 786 + 95 + 194 + 45) * 2] +
[(708 + 753 + 166 + 181 + 33) * 1]] / 2749
= 56.99

16

Stage 2 optimisation example

- A single run of the permutations of slots:

Before

0	1044	1108	918	948	708	628	47	222	7	1
1044	0	1349	1119	1302	593	753	118	322	9	2
1108	1349	0	1282	1198	575	786	166	342	9	3
918	1119	1282	0	921	518	656	95	181	33	4
948	1302	1198	921	0	684	733	159	194	33	5
708	593	575	518	684	0	546	92	23	45	6
628	753	786	656	733	546	0	79	140	43	7
47	118	166	95	159	92	79	0	35	12	8
222	322	342	181	194	23	140	35	0	25	9
7	9	9	33	33	45	43	12	25	0	10
1	2	3	4	5	6	7	8	9	10	0

After

0	575	342	1198	9	1282	166	1108	786	1349	3
575	0	23	684	45	518	92	708	546	593	6
342	23	0	194	25	181	35	222	140	322	9
1198	684	194	0	33	921	159	948	733	1302	5
9	45	25	33	0	33	12	7	43	9	10
1282	518	181	921	33	0	95	918	656	1119	4
166	92	35	159	12	95	0	47	79	118	8
1108	708	222	948	7	918	47	0	628	1044	1
786	546	140	733	43	656	79	628	0	753	7
1349	593	322	1302	9	1119	118	1044	753	0	2
3	6	9	5	10	4	8	1	7	2	0

Large entries on the first minor diagonal are replaced with much smaller values

17

Stage 2 optimisation example

- The new cost after permutations of slots:

0	575	342	1198	9	1282	166	1108	786	1349	3
575	0	23	684	45	518	92	708	546	593	6
342	23	0	194	25	181	35	222	140	322	9
1198	684	194	0	33	921	159	948	733	1302	5
9	45	25	33	0	33	12	7	43	9	10
1282	518	181	921	33	0	95	918	656	1119	4
166	92	35	159	12	95	0	47	79	118	8
1108	708	222	948	7	918	47	0	628	1044	1
786	546	140	733	43	656	79	628	0	753	7
1349	593	322	1302	9	1119	118	1044	753	0	2
3	6	9	5	10	4	8	1	7	2	0

[[(575 + 23 + 194 + 33 + 33 + 95 + 47 + 628 + 753) * 16] +
[(342 + 684 + 25 + 921 + 12 + 918 + 79 + 1044)* 8] +
[(1198 + 45 + 181 + 159 + 7 + 656 + 118) * 4] +
[(9 + 518 + 35 + 948 + 43 + 1119) * 2] +
[(1282 + 92 + 222 + 733 + 9) * 1]] / 2749
= 31.81

18

Stage 2 optimisation

Reassignments of Exams Between Slots

- Aided by the observation that the most beneficial reassignment moves an exam from the slot that has high conflict counts on the first minor diagonal of the *spread matrix* to the feasible slot that is on higher minor diagonals of the *spread matrix*
  - 2 methods:
    - Single reassignments / Group reassignments

19

Stage 2 optimisation

- **Single Reassignments**
  - For each exam, the algorithm looks for a slot that can accept the exam (a slot which has no conflicts when the exam is moved from the initial slot to the new slot)
  - Carter Cost is calculated as if it will be moved to the new slot
  - If cost reduction can be achieved, the exam will be reassigned to the new slot
  - Slow, but reliable convergence

20

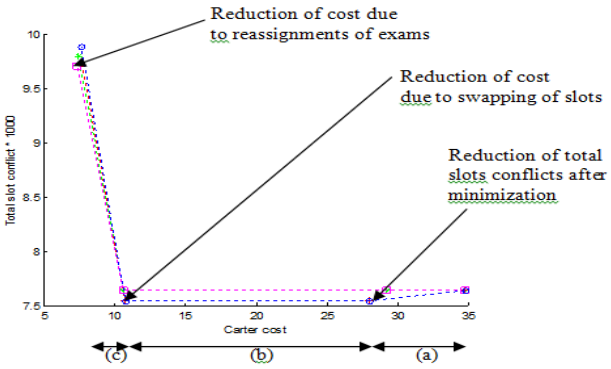
Stage 2 optimisation

- **Group Reassignments**
  - The algorithm identifies group of exams which could be moved to another slot which could reduce the Carter cost
  - The generated possible moves is being evaluated against a history of two steps behind
    - The purpose is to eliminate cyclic moves (infinite swapping)
  - Faster but prone to cyclic swaps

21

Stage 2 optimisation

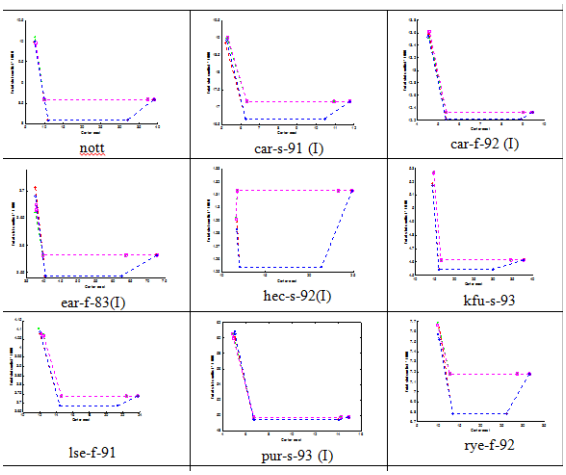
- Deterministic pattern of cost reduction



22

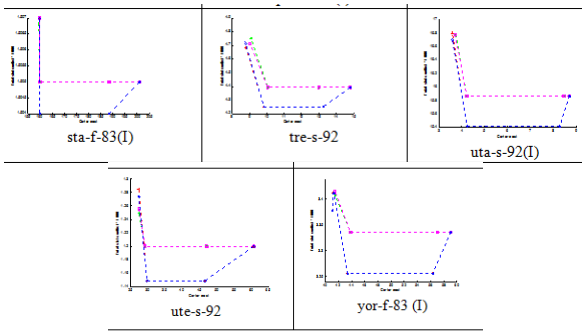
Results (Toronto benchmarks)

- Using the data gathered from the experiments, we have plotted graphs for the **Cost** versus **Total Slot Conflicts**:



23

Results (Toronto benchmarks)



24

Results & Discussions (contd.)

- Group reassignments (12.6% improvement) outperformed single reassignments (12.5% improvements) in 9 datasets from a total of 16 datasets
  - Nature of group reassignments: escapes more easily from local optima
- Single reassignment is prone to the convergence to local optima

25

Results & Discussions (contd.)

- Comparison of results with other constructive methods in literature:

Dataset	[5]	[2]	[3]	[9]	[14]	[4]	[15]	[16]	Ours
Nott									
car-s-91 (I)	7.1	**4.97**	5.45	5.29	5.08	5.03	5.17	5.12	5.19
car-f-92 (I)	6.2	4.32	4.5	4.54	4.38	**4.22**	4.74	4.41	4.49
ear-f-83(I)	36.4	36.16	36.15	37.02	38.44	**36.06**	40.91	36.91	37.57
hec-s-92(I)	**10.8**	11.61	11.38	11.78	11.61	11.71	12.26	11.31	11.47
kfu-s-93	**14**	15.02	14.74	15.8	14.67	16.02	15.85	14.75	14.36
lse-f-91	**10.5**	10.96	10.85	12.09	11.69	11.15	12.58	11.41	11.9
pur-s-93 (I)	**3.9**	-	-	-	-	-	5.87	5.87	4.88
rye-f-92	**7.3**	-	-	10.38	9.49	9.42	10.11	9.61	9.8
sta-f-83(I)	161.5	161.9	**157.21**	160.4	157.72	158.86	158.12	157.52	158.25
tre-s-92	9.6	8.38	8.79	8.67	8.78	**8.37**	9.3	8.76	8.74
uta-s-92(I)	3.5	**3.36**	3.55	3.57	3.55	3.37	3.65	3.54	3.59
ute-s-92	**25.8**	27.41	26.68	28.07	26.63	27.99	27.71	26.25	27.37
yor-f-83 (I)	41.7	40.88	42.2	39.8	40.45	**39.53**	43.98	39.67	41.1

26

Results & Discussions (contd.)

- Percentage difference between the results reported in [5], [15], [16]

Dataset	[5]		[15]		[16]		Ours		Best Constructive Cost
	Cost	%	Cost	%	Cost	%	Cost	%	
car-s-91 (I)	7.10	42.86	5.17	4.02	5.12	3.02	5.19	4.43	4.97
car-f-92 (I)	6.20	46.92	4.74	12.32	4.41	4.50	4.49	6.40	4.22
ear-f-83(I)	36.40	0.94	40.91	13.45	36.91	2.36	37.57	4.19	36.06
hec-s-92(I)	10.80	0.00	12.26	13.52	11.31	4.72	11.47	6.20	10.80
kfu-s-93	14.00	0.00	15.85	13.21	14.75	5.36	14.36	2.57	14.00
lse-f-91	10.50	0.00	12.58	19.81	11.41	8.67	11.90	13.33	10.50
pur-s-93 (I)	3.90	0.00	5.87	50.51	5.87	50.51	4.88	25.13	3.90
rye-f-92	7.30	0.00	10.11	38.49	9.61	31.64	9.80	34.25	7.30
sta-f-83(I)	161.50	2.73	158.12	0.58	157.52	0.20	158.25	0.66	157.21
tre-s-92	9.60	14.70	9.30	11.11	8.76	4.66	8.74	4.42	8.37
uta-s-92(I)	3.50	4.17	3.65	8.63	3.54	5.36	3.59	6.85	3.36
ute-s-92	25.80	0.00	27.71	7.40	26.25	1.74	27.37	6.09	25.80
yor-f-83 (I)	41.70	5.49	43.98	11.26	39.67	0.35	41.10	3.97	39.53
Average Percentage Difference To Best Constructive Cost(%)	**9.06**		**15.72**		**9.47**		**9.11**		**Consistent performance**

Conclusion

- The proposed optimization method is a simple yet very competitive to improve the cost of benchmark objective function in timetabling research
- The method is consistently competitive (*by contrast to other methods that deliver very good results on some benchmarks and rather poor results on others*)
- The method is deterministic (reproducible results)

28