University Scheduling with Greedy Heuristics

Maria Albert Gimeno
Adrien Haegel

Optimization using Metaheuristics

May 11, 2015

Introduction

- Greedy methods: human way of solving a problem
- 4 greedy methods are implemented

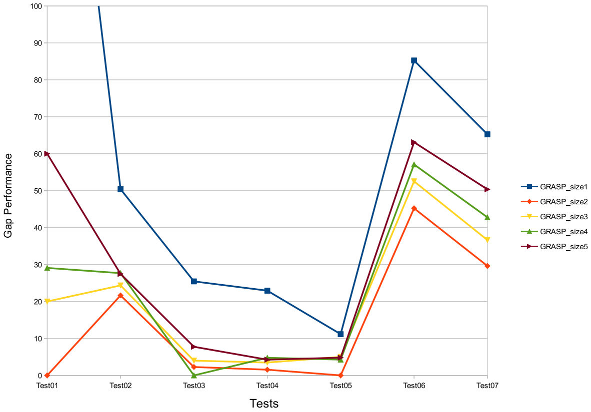
GRASP: Algorithm

```
min = +infinity
while time < timelimit do
    Current Solution ← new empty solution
    while lectures can still be scheduled do
        Construct N best choices for scheduling an unscheduled lecture in an empty Timeslot and Room
        Schedule randomly one of these best choices
        Update current solution value with delta evaluations functions
    end while
    if current solution value < min then
        min ← current solution value
        Best solution ← current solution
    end if
end while
```

GRASP: Solution construction algorithm

for each unscheduled lecture **do**
    for each available time slot **do**
        Calculate time slot delta adding cost Δt
        for each available room **do**
            Calculate room delta adding cost Δr
            if $\Delta t + \Delta r < 0$ **then**
                if $\Delta t + \Delta r < \max_N$ best solutions value(N) **then**
                    Remove solution whose value is the maximum of the best solutions
                    Store current solution in best solutions
                end if
            end if
        end for
    end for
end for

GRASP: Tuning of the window size



GRASP: Tuning of the window size

Window size N	Average gap	Deviation
$N = 1$	68.11	0
$N = 2$	14.33	10.87
$N = 3$	20.85	12.96
$N = 4$	23.66	10.94
$N = 5$	31.10	11.99

Lecture-GRASP : Introduction

- Reduce the time needed to search through all possible solutions, by only adding one lecture at a time.
- The question is: how to order the lectures we choose?
- It is possible to construct a problem related ordering : Get rid of the lectures with many constraints first!

Lecture-GRASP : Pseudo-code

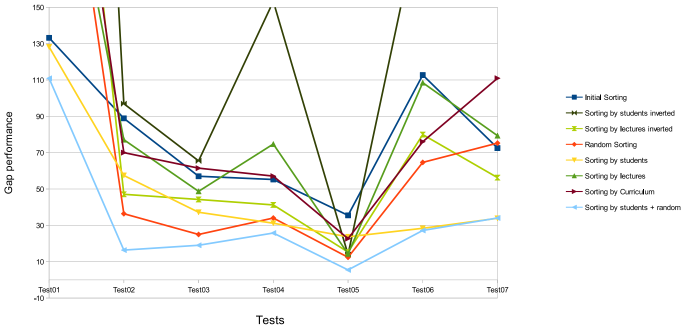
```
min = +infinity
while time < timelimit do
    Current Solution ← new empty solution
    Sort lectures
    for each lecture to be scheduled do
        Construct N best choices for scheduling this lecture in an empty Timeslot and Room
        Current Solution ← Schedule randomly one of these best choices
        Update current solution value with delta evaluations functions
    end for
    if current solution value < min then
        min ← current solution value
        Best solution ← current solution
    end if
end while
```

Lecture-GRASP : Sorting methods

Different sorting methods:

- Initial Sorting
- Random Sorting
- Sorting by students (+ reversed)
- Sorting by lectures (+ reversed)
- Sorting by curriculum
- Sorting by students + random

Lecture-GRASP : Tuning of the sorting methods



Lecture-GRASP: Tuning of window size

Sorting method	Average gap	Deviation
Initial Sorting	79.30	7.91
Random Sorting	80.71	11.55
Sorting by students	48.57	6.86
Sorting by students inverted	238.27	7.22
Sorting by lectures	98.96	9.35
Sorting by lectures inverted	101.36	9.20
Sorting by Curriculum	107.28	13.16
Sorting by students + random	34.11	8.03

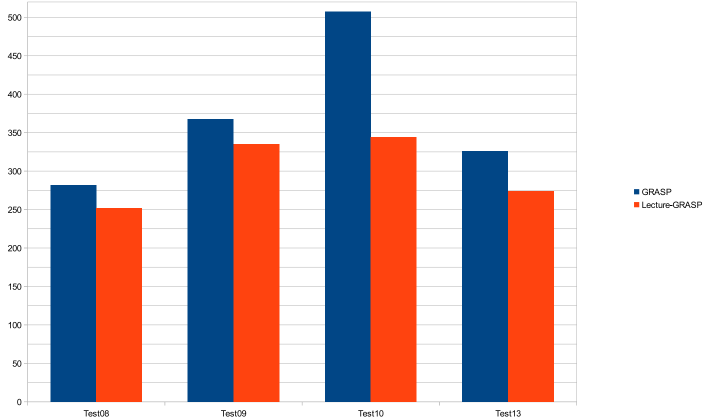
- students + random benefits from the two best methods we had

Lecture-GRASP: Tuning of window size

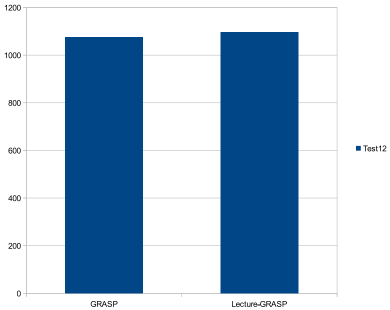
Window size	Average gap	Deviation
1	3.42	9.00
2	11.70	12.68
3	18.88	8.78
4	23.65	10.21
5	34.11	7.29
6	64.95	6.86

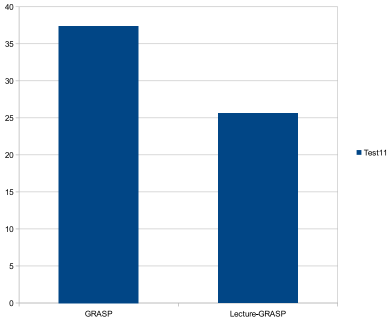
- The most greedy selection method is the most efficient.
- The randomness is already in the sorting method.

Lecture-GRASP : Mean value of solution



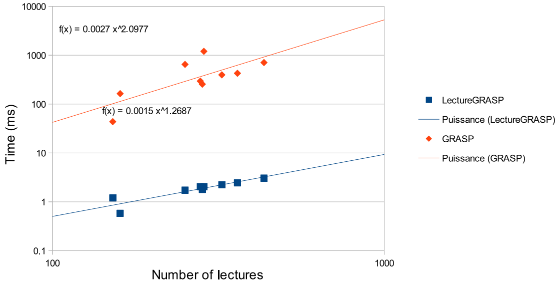
Lecture-GRASP : Mean value of solution





Lecture-GRASP : Computational time analysis

- Lecture-Grasp is better than GRASP, mostly since the computational time for one iteration is much lower



Lecture-GRASP : Conclusions

- However, there is no local search, it takes a lot of time to construct a solution...
- ... that is going to be discarded right afterwards
- We need to perform local searches!
- We have not been doing this for nothing: we now have construction methods
- It is easy to implement LNS with that.

LNS: Introduction

- Take advantage of the existing solution
- Perform a local search vs. constructing a new solution
- Construct a new solution based on the old one with **DESTROY** and **REPAIR** functions
- The questions are:
  - What to destroy?
  - How to repair?

LNS: Introduction

- *DESTROY COURSE*
- *DESTROY ROOM*
- *DESTROY TIME SLOT*
- *DESTROY CURRICULUM*
- *DESTROY RANDOM*

- *REPAIR GRASP* with window length of 1
- *REPAIR GRASP* with window length of 2
- *REPAIR LECTURE-GRASP* with window length of 1

LNS: Algorithm

```
min = +infinity
Current Solution ← feasible solution
while time < timelimit do
    Destroy current solution partially
    Repair destroyed solution
    Update current solution value with delta evaluations functions
    if current solution value < min then
        min ← current solution value
        Best solution ← current solution
    end if
end while
```

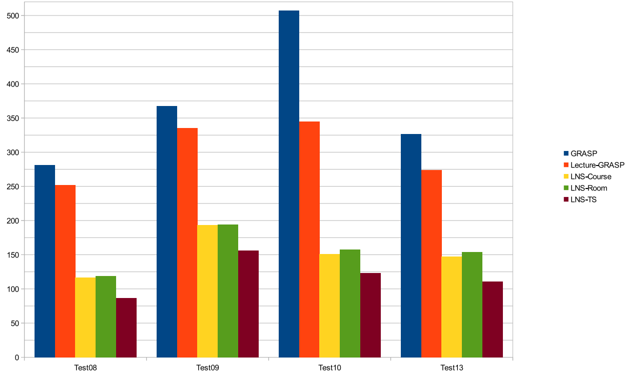
LNS: Tuning of destroying and repairing functions

		Destroy Function				
		Course	Curriculum	Room	Time slot	Random
Destroy size	1	(107.1, 32.0)	(117.6, 7.9)	(54.6, 7.0)	(84.0, 26.1)	(62.4, 7.8)
	2	(107.2, 21.2)	(116.9, 7.0)	(54.8, 7.4)	(89.9, 27.8)	(60.9, 6.7)
	3	(46.4, 6.4)	(140.8, 8.6)	(81.9, 6.7)	(24.8, 6.3)	
	4	(51.7, 5.9)			(35.9, 7.5)	
	5	(57.2, 6.5)			(45.6, 9.0)	
Destroy size	1	(112.6, 25.8)	(166.3, 9.8)	(88.5, 7.1)	(28.4, 13.2)	(94.4, 5.6)
	2	(107.5, 32.7)	(160.0, 9.9)	(90.7, 6.1)	(37.2, 37.9)	(97.1, 5.9)
	3	(63.7, 7.9)	(168.6, 7.7)	(109.4, 7.4)	(63.9, 10.3)	
Destroy size	1	(103.8, 10.3)	(145.8, 8.1)	(78.7, 6.6)	(33.3, 13.5)	(86.8, 5.8)
	2	(101.6, 12.2)	(146.1, 6.9)	(78.8, 7.2)	(30.9, 17.4)	(85.1, 7.1)
	3	(92.3, 10.3)	(161.5, 9.5)	(96.8, 6.9)	(53.9, 10.2)	
	4	(90.7, 5.9)			(59.6, 10.9)	
	5	(89.5, 7.1)			(67.6, 10.1)	

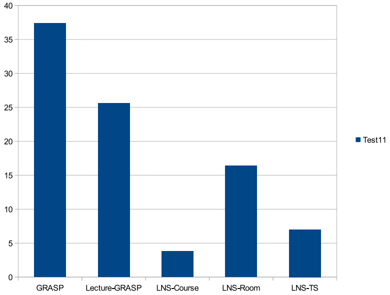
LNS: Comments

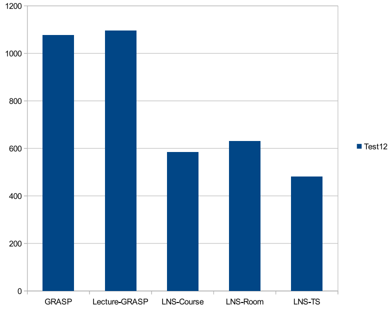
- DESTROY TIME SLOT gives in general good results
- Other destroying functions give good results for certain repairing functions and window lengths
- The random method does not give very good results

LNS: Mean value solution

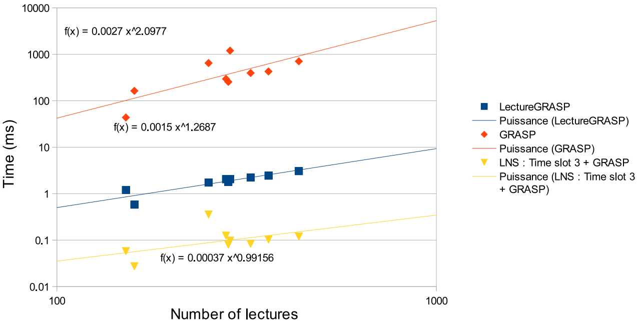


LNS: Mean value solution





LNS: Computational time analysis



LNS: Conclusions

- Abrupt gap in solution compared to GRASP and Lecture-GRASP
- We can just choose one destroy and repairing method
- Random has not shown a good performance

ALNS : Introduction

Problems:

- Find a way to combine efficiently all the different LNS methods we implemented earlier.
- Some methods are better than others on specific datasets.
- Just using the differents methods randomly is not more efficient.

Solution: Use a reward system to incite the algorithm to use the efficient methods $\rightarrow$ ALNS.

ALNS : Algorithm

min = +infinity
Current Solution $\leftarrow$ feasible solution
while time $<$ timelimit **do**
    Choose a destroy method from a probability array DestroyProba
    Destroy current solution partially
    Choose a repair method from a probability array RepairProba
    Repair destroyed solution
    if new solution value $<$ Selection-Threshold $\cdot$ old solution value **then**
        Update current solution value with delta evaluations functions
        if current solution value $<$ min **then**
            min $\leftarrow$ current solution value
            Best solution $\leftarrow$ current solution
            Update DestroyProba and RepairProba with reward ω_1
        else if current solution value $<$ old solution value **then**
            Update DestroyProba and RepairProba with reward ω_2
        else (current solution value $>$ old solution value)
            Update DestroyProba and RepairProba with reward ω_3
        end if
    else
        Do NOT update the solution
        Update DestroyProba and RepairProba with reward ω_4
    end if
end while

ALNS : Tuning

- A lot of parameters! Choice of the methods in ALNS, ω_1, ω_2, ω_3, ω_4, λ, selection-threshold.
- “Hill-Climbing” on the tuning, and some assumptions.

ALNS : Selection of the methods

We tested different sets of methods:

- ALNS-Full : All the LNS methods
- ALNS-select : Only the most efficient LNS methods
- ALNS-combined : The destroy and repair methods are combined

	ALNS-Full	ALNS-select	ALNS-combined
Average Gap	45.34	21.46	24.47
Deviation	11.78	12.55	10.77

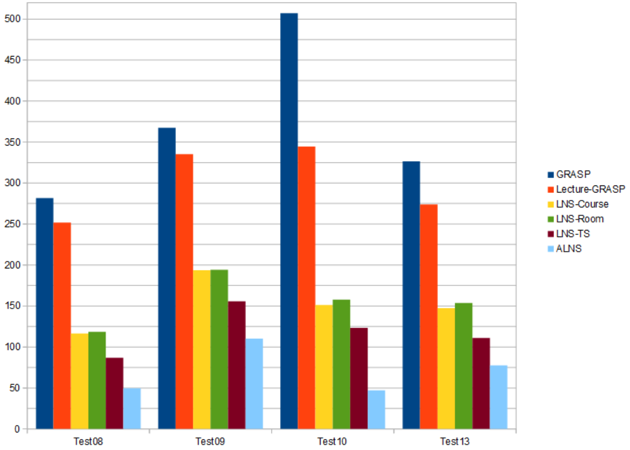
ALNS : Tuning the rewards and the damping factor

λ	10^{-2}	$5 \cdot 10^{-3}$	$2.5 \cdot 10^{-3}$	10^{-3}	$5 \cdot 10^{-4}$	10^{-4}
S = 10	x	34.3	x	34.0	37.9	36.1
S = 50	x	30.6	x	30.7	32.2	32.1
S=100	x	28.1	x	30.4	28.7	32.0
S=250	26.2	28.9	30.1	x	x	x
S=500	28.1	27.1	25.4	27.0	30.4	34.7
S=750	26.5	29.5	30.3	x	x	x
S=1000	x	30.3	x	39.4	30.3	40.3
S=10000	x	x	x	79.2	x	79.2

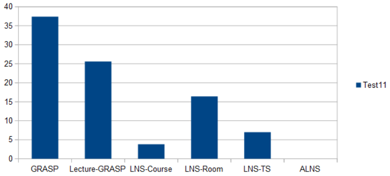
ALNS : Tuning the selection threshold

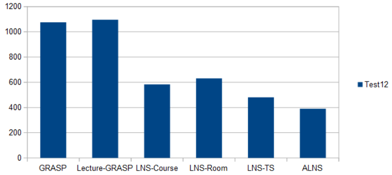
Selection Threshold	Average gap	Average deviation
1	112.8	18.5
1.01	43.1	7.7
1.015	39.0	9.5
1.02	28.6	6.6
1.025	27.8	15.7
1.03	32.9	17.5
1.04	41.3	14.5
1.07	67.6	19.2
1.1	91.6	25.8
1.2	125.4	23.6
1.3	118.3	21.3
1.4	115.8	8.1

ALNS: Mean value solution



ALNS: Mean value solution





Conclusions

	Average gap	Deviation
GRASP	482.6	16.8
Lecture-GRASP	372.9	8.2
LNS-Course	132.0	6.4
LNS-Room	140.5	5.7
LNS-TS	83.1	4.4
ALNS	7.7	7.4

- Good perfomance due to a fast delta evaluation
- This was a construction process, and ALNS benefits from the optimization of the previous algorithms

Thank you