Metaheuristic algorithm for timetabling problem

(Seminar III)

Camilo Rodríguez-Garzón
Juan Carlos Rivera

Universidad EAFIT
Mathematical Modeling Research Group

Medellín, Colombia

June 5, 2018

Updated: November 18, 2018

Overview

- Motivation
- Introduction
- Hard constraints
- Soft constraints
- Methodology
- Problem formulation
- Algorithmic solution
- Computational experiments
- Conclusions

Motivation:

- Currently EAFIT does not have an automated system that allows to schedule university timetabling.
- Each school or university has different characteristics that make the problem particular.
- This is a NP-Hard combinatorial optimization problem Abdelhalim & El Khayat (2016).
- The algorithm in developing can be applied to different topics that are related to timetabling scheduling.

Introduction:

Figure 1.1: Timetabling problem

Hard constraint events:

A factible solution meets the following set of hard constraints $HC_1 - HC_6$ presented below:

- HC_1: Two event can not be scheduled on the same day, period and room.
- HC_2: Assigned rooms must meet the characteristics required by the events.
- HC_3: The time load defined for each event must be satisfied.
- HC_4: Two lessons from the same event must be consecutive when scheduled for the same day, in case it is required by the event.

Hard constraint teachers:

- HC_5: A teacher can not be scheduled to more than one lesson in a given period.
- HC_6: A teacher can not be scheduled to a period in which she/he is unavailable.

Hard constraint students:

- HC_7: For students with perfect curriculum, at least six events for each course must be feasible.
- HC_8: No student can be assigned more than one event at the same time.
- HC_9: The number of students attending the event must be less than or equal to the capacity to the classroom.

Soft constraints:

In addition to the feasibility with respect to mandatory constraints, the requirements of $\mathrm{SC}_1 - \mathrm{SC}_4$ listed below should be satisfied as much as possible:

- SC_1: Avoid teachers' idle periods.
- SC_2: Students should not have a single course on a day.
- SC_3: Students should not have more than two consecutive courses.

Methodology:

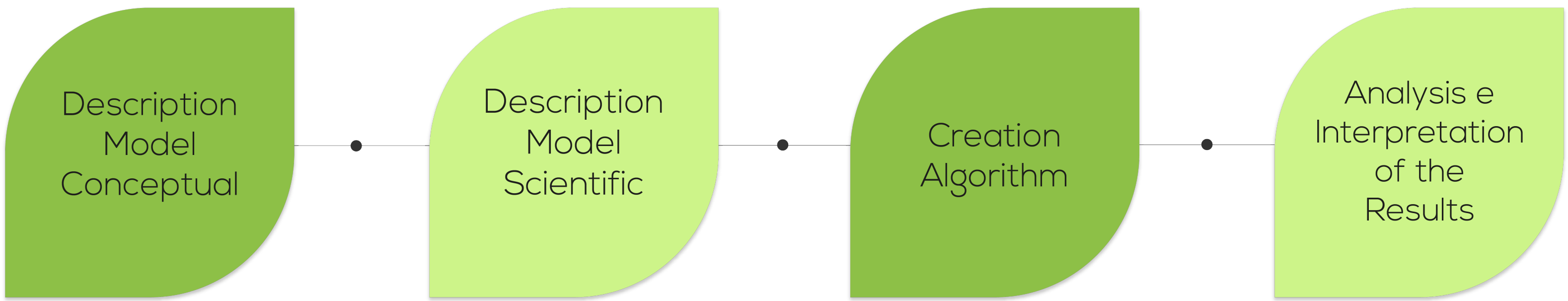

Figure 1.2: Timetabling problem methodology

Methodology:

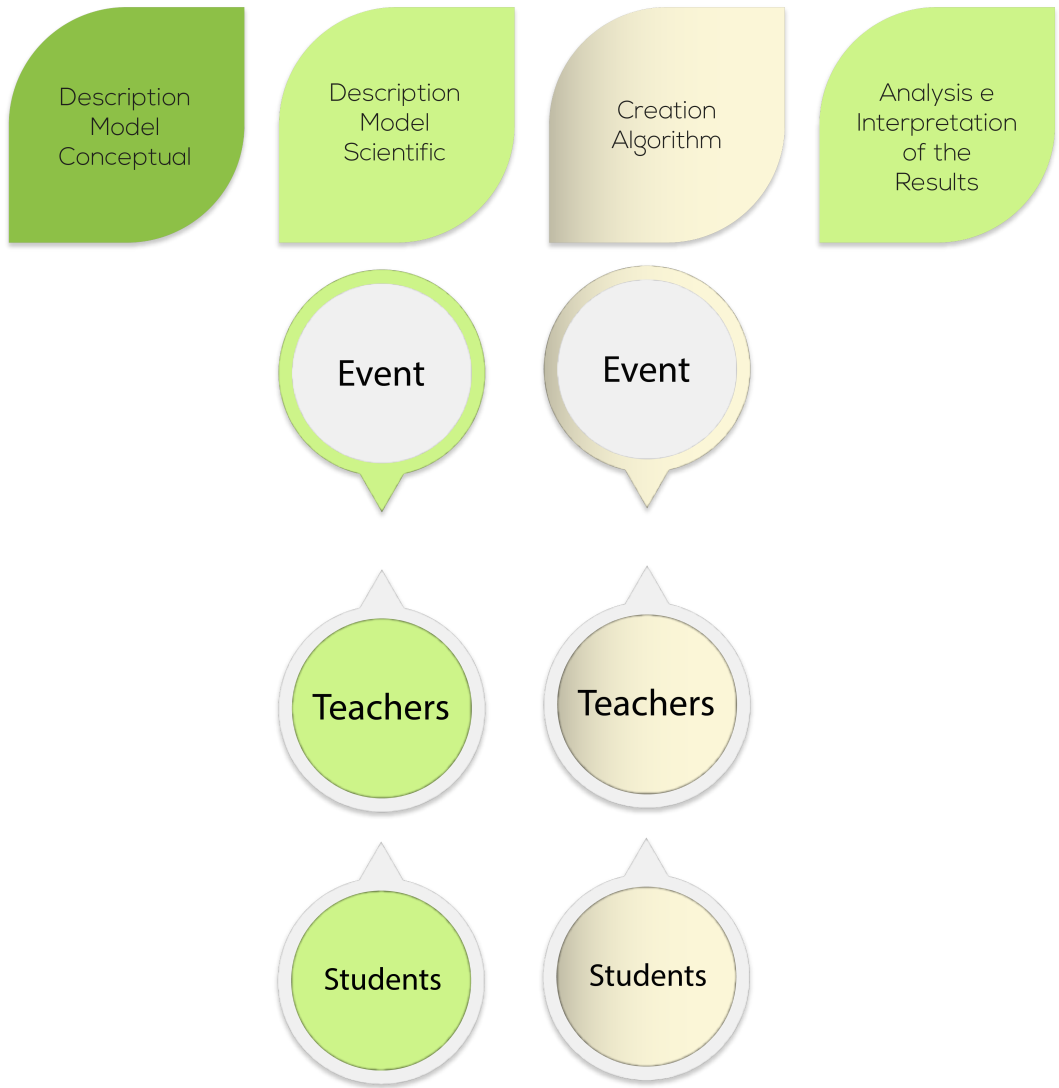

Figure 1.3: Timetabling problem methodology

Related work:

Techniques		Article name
Graph Coloring (GC)		An Introduction to TimeTabling de Werra (1985)
Integer/Linear Programming (IP/LP)		New integer linear programming approaches for course timetabling Boland *et al.* (2008) , Integer programming methods for large-scale practical classroom assignment problems Phillips *et al.* (2015)
Constraint Satisfaction Programming (CSP)		Timetable planning using the constraint-based reasoning Deris *et al.* (2000)
Multi-population	Genetic Algorithm (GA)	A Utilization-based Genetic Algorithm for Solving the University Timetabling Problem (UGA) Abdelhalim & El Khayat (2016)
	Ant Colony Optimization (ACO)	A MAX - MIN Ant System for the University Course Timetabling Problem Socha *et al.* (2002), A MAX-MIN Ant System for the University Course Timetabling Problem Brabazon *et al.* (2015)
	Artificial Bee Colony (ABC)	University course timetabling using hybridized artificial bee colony with hill climbing optimizer aro Bolaji *et al.* (2014)
	Memetic Algorithm (MA)	Using improved memetic algorithm and local search to solve university Course Timetabling problem (UCTP) Joudaki *et al.* (2011)
	Harmony Search Algorithm (HSA)	University course timetabling using a hybrid harmony search metaheuristic algorithm Al-Betar *et al.* (2012)
Single-population	Local Search (LS)	A fuzzy genetic algorithm with local search for university course timetabling Kohshori *et al.* (2011), Genetic algorithms with guided and local search strategies for university course timetabling Yang & Jat (2011)
	Variable Neighborhood Search (VNS)	An Investigation Of Variable Neighbourhood Search For University Course Timetabling Abdullah *et al.* (2005)
	Simulated Annealing (SA)	Solving the Course Scheduling Problem Using Simulated Annealing Aycan & Ayav (2009), A hybrid simulated annealing with Kempe Chain neighborhood for the university timetabling problem Tuga *et al.* (2007)
	Tabu Search (TS)	The effect of neighborhood structures on tabu search algorithm in solving course timetabling problem Aladag *et al.* (2009), Design and implementation of a course scheduling system using Tabu Search Alvarez-Valdes *et al.* (2002)
Novel Intelligent	Hybrid Algorithms (Hybrid Metaheuristic)	A new hybrid algorithm for university course timetabling problem using events based on groupings of students Badoni *et al.* (2014)
	Fuzzy method	Fuzzy genetic heuristic for university course timetable problem Chaudhuri & De (2010), A fuzzy solution based on Memetic algorithms for timetabling Golabpour *et al.* (2008)
	Clustering Algorithms	Applying a novel clustering technique based on FP- tree to University timetabling problem: A case study Shatnawi *et al.* (2010)
Multi-Agent Systems	Review papers Multi-Agent Systems	A multi-agent system for course timetabling Yanga & Paranjapea (2011), Implementation of class timetabling using multi agents Nandhini & Kanmani (2009)

Table 1.1: Brief literature about University Timetabling Problem

Problem formulation:

Symbols	Definition
Sets	
$s \in S$	Set of slot composed of days D and periods P
$r \in R$	Set of rooms
$e \in E$	Set of event composed of courses C and groups G
Parameters	
w_r	Unit cost of using the room r
f_e	frequency with which an event e is given in the week
dem_e	Intensity in hours with which an e event is imparted in a day
cap_s	Capacity in hours the slot s has
com_{er}	$\begin{cases} 1 & \text{if the event } e \text{ can be assigned to the room } r \\ 0 & \text{Otherwise.} \end{cases}$
ϕ_s	Subset of incompatible slots with slot s
Decision variables	
x_{esr}	$\begin{cases} 1 & \text{if the event } e \text{ is programmed to a time slot } s \text{ that is assigned to the room } r \\ 0 & \text{Otherwise.} \end{cases}$

Table 1.2: Notation used for modeling events at EAFIT University

Model:

The objective (1) is to minimize the use of artificial salons. The IP restrictions are described below:

$$\min \sum_{e\in E}\sum_{s\in S}\sum_{r\in R} w_r \cdot x_{esr} \tag{1}$$

s. t.

$$\sum_{e\in E} x_{esr} \leq 1, \quad \forall\, s \in S,\, r \in R \tag{2}$$

$$\sum_{r\in R}\sum_{s\in S} x_{esr} = f_e, \quad \forall\, e \in E \tag{3}$$

$$x_{esr} \leq com_{er}, \quad \forall\, e \in E,\, s \in S,\, r \in R \tag{4}$$

$$\sum_{e\in E}\sum_{s'\in \phi(s)} x_{es'r} \leq 1, \quad \forall\, r \in R,\, s \in S \tag{5}$$

$$M \cdot (x_{esr} - 1) \leq -\frac{dem_e}{cap_s} + 1, \quad \forall\, e \in E,\, s \in S,\, r \in R \tag{6}$$

Constraints:

- 2 - satisfies the restriction HC_1.
- 3 - satisfies the restriction HC_3.
- 4 - satisfies the restriction HC_2.
- 5 - only one event can be assigned to one of the slots of the subset of incompatible slots ϕ_s.
- 6 - satisfies the restriction HC_4.

Algorithmic solution:

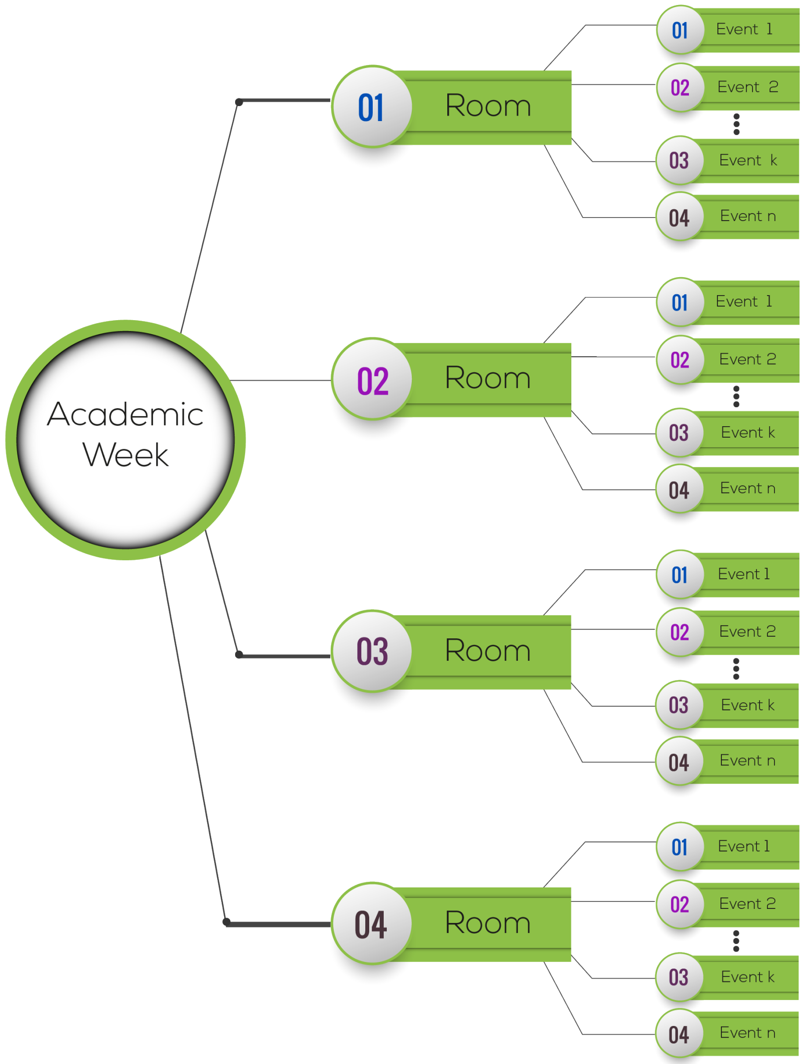

Figure 1.4: Chain Responsability to Timetabling problem

Algorithmic solution:

```
procedure GRASP METAHEURÍSTIC
    f* ← α
    input ← ReadInput();
    if validate(input) then
        for i ≤ i_max do
            x ← GreedyRandomized();
            x ← LocalSearch(x);
            if f(x) < f* then
                f* ← f(x);
                x* ← x;
            end if
        end for
    end if
    WriteOutput(x*);
end procedure
```

Computational experiments:

According to (aro Bolaji *et al.* , 2014; McCollum, 2007) the algorithm will be tested using the following instances:

Instance	small	medium	large
Events	100	400	400
Room	5	5	10
Day	5	5	5
Slot	8	8	8

Table 1.3: Set of test instances at EAFIT University

Computational experiments:

Room	Type Room	Capacity	Event	Group	Max Student	Day	Start Time	End Time
35401	Aula Normal	50	METODOLOGÍA DEL APRENDIZAJE	024	26	T	7.0	9.0
35401	Aula Normal	50	MODELACIÓN Y SIMULACIÓN I	001	30	T	9.0	12.0
35401	Aula Normal	50	METODOLOGÍA DEL APRENDIZAJE	024	26	W	14.0	15.0
35401	Aula Normal	50	MATEMÁTICAS I	002	40	TH	18.0	21.0
35401	Aula Normal	50	MATEMÁTICAS 1	010	37	F	6.0	9.0
35403	Aula Normal	36	MATEMÁTICAS III (ECONOMÍA)	003	32	F	6.0	9.0
35501	Aula Normal	50	BIOLOGÍA MOLECULAR	101	30	F	6.0	9.0
35501	Aula Normal	50	MATEMÁTICAS 1	001	37	S	12.0	15.0
38107	Aula Normal	25	FUNDAMENTOS DE FISICOQUÍMICA	156	20	T	15.0	17.0
38107	Aula Normal	25	FUNDAMENTOS DE FISICOQUÍMICA	156	20	S	9.0	12.0
38201	Aula Normal	50	BIOÉTICA	001	30	M	6.0	8.0
38201	Aula Normal	50	METODOLOGÍA DEL APRENDIZAJE	022	26	W	14.0	15.0
38201	Aula Normal	50	ESTADÍSTICA 2 (ECONOMÍA)	001	26	W	9.0	12.0
38201	Aula Normal	50	BIOÉTICA	001	30	F	11.0	12.0

Table 1.4: Timetabling Events at EAFIT University

Conclusions:

- The first stage of the problem was successfully modeled, which corresponds to the assignment of events.
- An algorithm was proposed for the scheduling of events at the EAFIT University.
- An algorithm was proposed for the scheduling of teacher at the EAFIT University.

Bibliography

Abdelhalim, Esraa A., & El Khayat, Ghada A. 2016. A Utilization-based Genetic Algorithm for Solving the University Timetabling Problem (UGA). *Alexandria Engineering Journal*, **55**(2), 1395–1409. 3, 11

Abdullah, Salwani, Abdullah, Salwani, Burke, Edmund K., & Mccollum, Barry. 2005. An Investigation Of Variable Neighbourhood Search For University Course Timetabling. *IN THE 2 ND MULTIDISCIPLINARY INTERNATIONAL CONFERENCE ON SCHEDULING: THEORY AND APPLICATIONS (MISTA*, 413—-427. 11

Al-Betar, Mohammed Azmi, Khader, Ahamad Tajudin, & Zaman, Munir. 2012. University course timetabling using a hybrid harmony search metaheuristic algorithm. *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, **42**(5), 664–681. 11

Aladag, Cagdas Hakan, Hocaoglu, Gulsum, & Basaran, Murat Alper. 2009. The effect of neighborhood structures on tabu search algorithm in solving course timetabling problem. *Expert Systems with Applications*, **36**(10), 12349–12356. 11

Alvarez-Valdes, Ramon, Crespo, Enric, & Tamarit, Jose M. 2002. Design and implementation of a course scheduling system using Tabu Search. *European Journal of Operational Research*, **137**(3), 512–523. 11

aro Bolaji, Asaju La, Khader, Ahamad Tajudin, Al-Betar, Mohammed Azmi, & Awadallah, Mohammed A. 2014. University course timetabling using hybridized artificial bee colony with hill climbing optimizer. *Journal of Computational Science*, **5**(5), 809–818. 11, 17

Aycan, E., & Ayav, T. 2009. Solving the Course Scheduling Problem Using Simulated Annealing. *2009 IEEE International Advance Computing Conference*, 6–7. 11

Badoni, Rakesh P., Gupta, D.K., & Mishra, Pallavi. 2014. A new hybrid algorithm for university course timetabling problem using events based on groupings of students. *Computers & Industrial Engineering*, **78**(dec), 12–25. 11

Boland, Natashia, Hughes, Barry D., Merlot, L. T G, & Stuckey, Peter J. 2008. New integer linear programming approaches for course timetabling. *Computers and Operations Research*, **35**(7), 2209–2233. 11

Brabazon, Anthony, O'Neill, Michael, & McGarraghy, Seán. 2015. A MAX-MIN Ant System for the University Course Timetabling Problem. *Pages 141–170 of: Natural Computing Series*, vol. 28. 11

Chaudhuri, Arindam, & De, Kajal. 2010. Fuzzy genetic heuristic for university course timetable problem. *International Journal of Advances in Soft Computing and its Applications*, **2**(1), 100–123. 11

de Werra, D. 1985. An introduction to timetabling. *European Journal of Operational Research*, **19**(2), 151–162. 11

Deris, Safaai, Omatu, Sigeru, & Ohta, Hiroshi. 2000. Timetable planning using the constraint-based reasoning. *Computers and Operations Research*, **27**(9), 819–840. 11

Golabpour, Amin, Farahi, Ahmad, Beigi, Hossein, Shirazi, Hossein Mozdorani, & Kootiani, Ahmad Zadali Mohammad. 2008. A fuzzy solution based on Memetic algorithms for timetabling. *Pages 108–110 of: Proceedings - 2008 International Conference on MultiMedia and Information Technology, MMIT 2008*. 11

Joudaki, M., Imani, M., & Mazhari, N. 2011. Using improved memetic algorithm and local search to solve university Course Timetabling problem (UCTP). *In: Proceedings of the 2011 International Conference on Artificial Intelligence, ICAI 2011*, vol. 2. 11

Kohshori, M S, Abadeh, M S, & Sajedi, H. 2011. A fuzzy genetic algorithm with local search for university course timetabling. *Pages 250–254 of: Data Mining and Intelligent Information Technology Applications (ICMiA), 2011 3rd International Conference on*. 11

McCollum, B. 2007. *International Timetabling Competition*. `http://www.cs.qub.ac.uk/itc2007/`. [Online; accessed 23-September-2017]. 17

Nandhini, M., & Kanmani, S. 2009. Implementation of class timetabling using multi agents. *In: 2009 International Conference on Intelligent Agent and Multi-Agent Systems, IAMA 2009*. 11

Phillips, Antony E., Waterer, Hamish, Ehrgott, Matthias, & Ryan, David M. 2015. Integer programming methods for large-scale practical classroom assignment problems. *Computers & Operations Research*, **53**, 42–53. 11

Shatnawi, Safwan, Al-Rababah, Khaleel, & Bani-Ismail, Basel. 2010. Applying a novel clustering technique based on FP- tree to University timetabling problem: A case study. *Pages 314–319 of: Proceedings, ICCES'2010 - 2010 International Conference on Computer Engineering and Systems*. 11

Socha, Krzysztof, Knowles, Joshua, & Sampels, Michael. 2002. A MAX - MIN Ant System for the University Course Timetabling Problem. *Ant algorithms*, 1–13. 11

Tuga, Mauritsius, Berretta, Regina, & Mendes, Alexandre. 2007. A hybrid simulated annealing with Kempe Chain neighborhood for the university timetabling problem. *Pages 400–405 of: Proceedings - 6th IEEE/ACIS International Conference on Computer and Information Science, ICIS 2007; 1st IEEE/ACIS International Workshop on e-Activity, IWEA 2007*. 11

Yang, Shengxiang, & Jat, Sadaf Naseem. 2011. Genetic algorithms with guided and local search strategies for university course timetabling. *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, **41**(1), 93–106. 11

Yanga, Yan, & Paranjapea, Raman. 2011. A multi-agent system for course timetabling. *Intelligent Decision Technologies*, **5**(2), 113–131. 11

(Bonus Slides)

Solution construction flow:

procedure GreedyRandomized
    teacher $\leftarrow$ **SetOutData(GetInData(***event***));**
    event $\leftarrow$ **SetNext(***teacher***);**
    room $\leftarrow$ **SetOutData(GetInData(***event***));**
    room $\leftarrow$ **SetNext(***event***);**
    slot $\leftarrow$ **SetSlotFree(GetInData(***slot***));**
    slot $\leftarrow$ **SetOutData(GetInData(***room***));**
    slot $\leftarrow$ **SetNext(***room***);**
    processor(*slot***);**
end procedure

Processor:

procedure PROCESSOR
    MakeObject($inData$**);**
    if Next() $\neq null$ **then**
        Next().processor()
    end if
end procedure