Answer Set Programming as a Modeling Language for Course Timetabling

Mutsunori Banbara[1], Takehide Soh[1], Naoyuki Tamura[1], Katsumi Inoue[2], and Torsten Schaub[3]

[1]Kobe University
[2]National Institute of Informatics
[3]University of Potsdam

ICLP 2013
August 28th, 2013
@Armada Hotel, Istanbul

Timetabling

- **Timetabling** is a problem of assigning a set of entities (e.g., tasks, events, people) to the limited number of resources over time, subject to a set of pre-defined constraints.
- There are variants of timetabling with real world applications:
  - Sports timetabling
  - Employee timetabling
  - Transport timetabling
  - Educational timetabling
- International conference and competition are held:
  - **PATAT**: International Conference on the Practice and Theory of Automated Time tabling
  - **ITC**: International Timetabling Competition

Timetabling has received increasing attention from both researchers and practitioners.

Timetabling

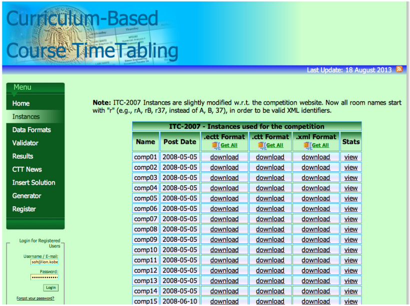
- **Timetabling** is a problem of assigning a set of entities (e.g., tasks, events, people) to the limited number of resources over time, subject to a set of pre-defined constraints.
- There are variants of timetabling with real world applications:
  - Sports timetabling
  - Employee timetabling
  - Transport timetabling
  - Educational timetabling
- International conference and competition are held:
  - **PATAT**: International Conference on the Practice and Theory of Automated Time tabling
  - **ITC**: International Timetabling Competition

Timetabling has received increasing attention from both researchers and practitioners.

We consider an educational timetabling called **Curriculum-Based Course TimeTabling (CB-CTT).**

Curriculum-Based Course TimeTabling (CB-CTT)

- While **CB-CTT** had been traditionally considered in Operations Research, it is nowadays tackled with various kinds of approaches ([Shaerf '99] and [Lewis '07] are good surveys).
- Benchmark instances and their best known bounds are well maintained in the CB-CTT web portal.



Name	Post Date	.ectt Format Get All	.ctt Format Get All	.xml Format Get All	Stats
comp01	2008-05-05	download	download	download	view
comp02	2008-05-05	download	download	download	view
comp03	2008-05-05	download	download	download	view
comp04	2008-05-05	download	download	download	view
comp05	2008-05-05	download	download	download	view
comp06	2008-05-05	download	download	download	view
comp07	2008-05-05	download	download	download	view
comp08	2008-05-05	download	download	download	view
comp09	2008-05-05	download	download	download	view
comp10	2008-05-05	download	download	download	view
comp11	2008-05-05	download	download	download	view
comp12	2008-05-05	download	download	download	view
comp13	2008-05-05	download	download	download	view
comp14	2008-05-05	download	download	download	view
comp15	2008-06-10	download	download	download	view

The CB-CTT web portal contains:

- Benchmark instances
  57 instance x 5 formulations
  (**285** combinations in total)
- Solution validator
- Best known bounds for each instance and who and what method obtain them.

http://tabu.diegm.uniud.it/ctt/

Methods and # of obtained Best Known Bounds

Methods	Authors	# of Bests
Tabu Search (110) Hybrid Methods (1)	A. Schaerf	111
Other	S. Abdullah & H. Turabieh	30
Tabu Search	Z. Lu & J. Hao	24
SAT-based	Barcelogic Team	19
Mathematical Programming	A. Phillips	18
Mathematical Programming	G. Lach	5
Local Search	T. Muller	5
Simulated Annealing	SaTT group	3
Very Large Neighborhood Search	A. Kiefer	2
Simulated Annealing	M. Muehlenthaler	1
Hybrid Methods	Khalid & Salwan	1

Many best known bounds are obtained by **Metaheuristics**, however, **Declarative Problem Solving** is missing.

Motivation

CB-CTT has the following characteristics:

- It is known to be difficult, since it contains both hard and soft constraints (those constraints contain cardinalities).
- It has several formulations for a single instance: soft constraints are different in each formulation.

ASP-based methods have the following features:

- It can declaratively and uniformly represent hard and soft constraints and is flexible enough to represent the formulations of CB-CTT.
- It also provides first-order predicate, cardinality, optimization.
- Modern ASP grounders and solvers are well developed, e.g., `gringo` and `clasp`.

We propose an ASP-based method to CB-CTT

Summary of Results (at Aug. 22, 2013)

Our ASP-based method provides **175 best known bounds**.

Methods	Authors	# of Bests
ASP-based	M. Banbara	175
Tabu Search (110) Hybrid Methods (1)	A. Schaerf	111
Other	S. Abdullah & H. Turabieh	30
Tabu Search	Z. Lu & J. Hao	24
SAT-based	Barcelogic Team	19
Mathematical Programming	A. Phillips	18
Mathematical Programming	G. Lach	5
Local Search	T. Muller	5
Simulated Annealing	SaTT group	3
Very Large Neighborhood Search	A. Kiefer	2
Simulated Annealing	M. Muehlenthaler	1
Hybrid Methods	Khalid & Salwan	1

Definition of CB-CTT

CB-CTT is defined as the task of assigning all lectures of each course into a weekly timetable, subject to a given set of hard and soft constraints.

Hard constraints

H_1. Lectures
H_2. Conflicts
H_3. RoomOccupancy
H_4. Availability

Soft constraints

S_1. RoomCapacity
S_2. MinWorkingDays
S_3. IsolatedLectures
S_4. Windows
S_5. RoomStability
S_6. StudentMinMaxLoad
S_7. TravelDistance
S_8. RoomSuitability
S_9. DoubleLectures

- The **hard constraints** must be strictly satisfied.
- The **soft constraints** are not necessarily satisfied but the sum of violations should be desirably minimized.
- Soft constraints are different in each formulation.

Formulations

- Until now **5 formulations** of CB-CTT have been proposed.
- **UD1** is a basic formulation. **UD2** is a formulation used in ITC-2007. To capture more different scenarios, **UD3**, **UD4**, and **UD5** are proposed recently.

Constraint	UD1	UD2	UD3	UD4	UD5
H_1. Lectures	H	H	H	H	H
H_2. Conflicts	H	H	H	H	H
H_3. RoomOccupancy	H	H	H	H	H
H_4. Availability	H	H	H	H	H
S_1. RoomCapacity	1	1	1	1	1
S_2. MinWorkingDays	5	5	-	1	5
S_3. IsolatedLectures	1	2	-	-	1
S_4. Windows	-	-	4	1	2
S_5. RoomStability	-	1	-	-	-
S_6. StudentMinMaxLoad	-	-	2	1	2
S_7. TravelDistance	-	-	-	-	2
S_8. RoomSuitability	-	-	3	H	-
S_9. DoubleLectures	-	-	-	1	-

Input Example: toy.ectt

```
Name: Toy
Courses: 4
Rooms: 3
Days: 5
Periods_per_day: 4
Curricula: 2
Min_Max_Daily_Lectures: 2 3
UnavailabilityConstraints: 8
RoomConstraints: 3

COURSES:
SceCosC Ocra 3 3 30 1
ArcTec Indaco 3 2 42 0
TecCos Rosa 5 4 40 1
Geotec Scarlatti 5 4 18 1

ROOMS:
rA 32 1
rB 50 0
rC 40 0

CURRICULA:
Cur1 3 SceCosC ArcTec TecCos
Cur2 2 TecCos Geotec

UNAVAILABILITY_CONSTRAINTS:
TecCos 2 0
TecCos 2 1
TecCos 3 2
TecCos 3 3
ArcTec 4 0
ArcTec 4 1
ArcTec 4 2
ArcTec 4 3

ROOM_CONSTRAINTS:
SceCosC rA
Geotec rB
TecCos rC

END.
```

Input Example (ASP): toy.asp

```
name("Toy").
courses(4).
rooms(3).
days(5).
periods_per_day(4).
curricula(2).
min_max_daily_lectures(2,3).
unavailabilityconstraints(8).
roomconstraints(3).

course("SceCosC","Ocra",3,3,30,1).
course("ArcTec","Indaco",3,2,42,0).
course("TecCos","Rosa",5,4,40,1).
course("Geotec","Scarlatti",5,4,18,1).

room(rA,32,1).
room(rB,50,0).
room(rC,40,0).
```

```
curricula("Cur1","SceCosC").
curricula("Cur1","ArcTec").
curricula("Cur1","TecCos").
curricula("Cur2","TecCos").
curricula("Cur2","Geotec").

unavailability_constraint("TecCos",2,0).
unavailability_constraint("TecCos",2,1).
unavailability_constraint("TecCos",3,2).
unavailability_constraint("TecCos",3,3).
unavailability_constraint("ArcTec",4,0).
unavailability_constraint("ArcTec",4,1).
unavailability_constraint("ArcTec",4,2).
unavailability_constraint("ArcTec",4,3).

room_constraint("SceCosC",rA).
room_constraint("Geotec",rB).
room_constraint("TecCos",rC).
```

Entities of CB-CTT (Partial)

- **days**(5).
  There is 5 days, i.e., "Day 0" to "Day 4".
- **periods_per_day**(4).
  Each day has 4 periods, i.e., "Period 0" to "Period 3".
- **course**("TecCos", "Rosa", 5, 4, 40, 1).

Course	Lecturer	# of Lectures	Min. Work Days	# of Students.
TecCos	Rosa	5	4	40

- **room**(rA,32,1).
  Room rA can seat up to 32 students.
- **curricula**("Cur1","TecCos"). **curricula**("Cur2","TecCos").
  Course "TecCos" belongs to both "Cur1" and "Cur2".
- **unavailability_constraint**("TecCos",2,0).
  Course "TecCos" cannot be held at "Period 0" on "Day 2".

An Optimal Solution of `toy.asp` with UD2

`assigned(C,R,D,P)` is used to express that a lecture of a course `C` is assigned to a room `R` at a period `P` on a day `D`.

```
assigned("TecCos",rB,0,1).
assigned("TecCos",rB,0,3).
assigned("TecCos",rB,1,3).
assigned("TecCos",rB,2,3).
assigned("TecCos",rB,4,3).
assigned("SceCosC",rB,3,0).
assigned("SceCosC",rB,2,2).
assigned("SceCosC",rB,4,2).
assigned("ArcTec",rB,3,1).
assigned("ArcTec",rB,0,2).
assigned("ArcTec",rB,1,2).
assigned("Geotec",rA,4,1).
assigned("Geotec",rA,0,2).
assigned("Geotec",rA,1,2).
assigned("Geotec",rA,2,2).
assigned("Geotec",rA,4,2).
```

Cur1

	Day0	Day1	Day2	Day3	Day4
0				SceCosC rB	
1	TecCos rB			ArcTec rB	
2	ArcTec rB	ArcTec rB	SceCosC rB		SceCosC rB
3	TecCos rB	TecCos rB	TecCos rB		TecCos rB

Cur2

	Day0	Day1	Day2	Day3	Day4
0					
1	TecCos rB				Geotec rA
2	Geotec rA	Geotec rA	Geotec rA		Geotec rA
3	TecCos rB	TecCos rB	TecCos rB		TecCos rB

Encoding of Hard Constraints H_1 and H_2

`assigned(C,D,P)` (room `R` is lacked) is additionally used.

H_1. Lectures

All lectures of each course must be scheduled, and they must be assigned to distinct timeslots (a pair of Period and Day).

```
N {assigned(C,D,P):d(D):ppd(P)} N :- course(C,_,N,_,_,_).
```

H_2. Conflicts

Lectures of courses in the same curriculum or taught by the same teacher must be all scheduled in distinct timeslots.

```
:- not {assigned(C,D,P):curricula(Cu,C)} 1, cu(Cu), d(D), ppd(P).
:- not {assigned(C,D,P):course(C,T,_,_,_,_)} 1, t(T), d(D), ppd(P).
```

Encoding of Hard Constraints H_3 and H_4

H_3. RoomOccupancy

Two lectures can not take place in the same room in the same timeslot.

```
1 {assigned(C,R,D,P):r(R)} 1 :- assigned(C,D,P).
:- not {assigned(C,R,D,P):c(C)} 1, r(R), d(D), ppd(P).
```

H_4. Availability

If the teacher of the course is not available to teach that course at a given timeslot, then no lecture of the course can be scheduled at that timeslot.

```
:- assigned(C,D,P), unavailability_constraint(C,D,P).
```

Encoding of Soft Constraints

We explain the case of UD1 consisting of S_1, S_2 and S_3.

```
penalty(S1,V,C) :- violation of S1 RoomCapacity
penalty(S2,V,C) :- violation of S2 MinWorkingDays
penalty(S3,V,C) :- violation of S3 IsolatedLectures
#minimize [ penalty(_,_,P) = P ].
```

- Each soft constraint S_1, S_2, S_3 is expressed by either one or two rules in which the head is the form of `penalty/3`.
- `penalty(S,V,C)` expresses that a constraint `S` is violated by a lecture assignment `V` and its penalty cost is `C`.
- Finally, the total cost is summed up and minimized at the line of `#minimize`.

Encoding of Soft Constraints S_3

S_3. IsolatedLectures

- Lectures belonging to the same curriculum **should be adjacent** to each other in consecutive timeslots.

	Day0	Day0
0		TecCos rB
1	TecCos rB	
2	ArcTec rB	ArcTec rB
3	TecCos rB	TecCos rB
	Good :)	**Bad :(**

```
scheduled_curricula(Cu,D,P) :- assigned(C,D,P), curricula(Cu,C).
penalty("IsolatedLectures",isolated_lectures(Cu,D,P),1) : -
  scheduled_curricula(Cu,D,P),
  not scheduled_curricula(Cu,D,P-1),
  not scheduled_curricula(Cu,D,P+1).
```

Experiments on 57 Instances

Benchmark (from the CB-CTT web portal)

Name	# of Inst.	
ITC-2007 [Gaspero et al. '07]	21	comp*
DDS-2008 [Bonutti et al. '12]	7	DDS*
Erlangen	4	erlangen*
Test [Gaspero and Schaerf '03]	4	test*
EasyAcademy	12	EA*
Udine	9	Udine*

- Among them the instances of **Erlangen**, **EasyAcademy**, and **Udine** are very large, e.g., `erlangen2012_2.ectt` consists of 850 courses, 132 rooms, 850 curricula, 7,780 unavailability constraints, and 45,603 room constraints.
- Time Limit: 3 hours
  except erlangen* (24 hours), EA03 and EA07 (12 hours)

of obtained Best Known Bounds

We succeeded either in updating the best known bounds or reaching the same bounds for 175 combinations (65% of 285).

Benchmark	# Inst.	UD1	UD2	UD3	UD4	UD5	Total
comp*	21	9	3	14	7	1	34
DDS*	7	6	5	2	2	1	16
erlangen*	4	4	0	4	4	4	16
test*	4	1	0	2	1	0	4
EA*	12	12	12	12	12	12	60
Udine*	9	9	9	9	9	9	45
Total	57	41	29	43	35	27	175

46 combinations are newly closed (optimality is proved).

Conclusion

We showed an ASP-based approach to CB-CTT problem.

- We succeeded either in updating the previously known bounds or reaching the same bounds for many combinations of problem instances and formulations.
- This approach has the following advantages:
  1. ASP is expressive enough to specify a wide variety of soft constraints and objective functions.
  2. Our encoding is extensible enough for capturing new constraints, since the constraints can be compactly expressed by ASP, and all we have to do is adding rules.
  3. It is capable to switch constraints between hard and soft.
  4. It is flexible enough to deal with different formulations.
- TimeTabling can be a **killer application** of ASP and solving other TimeTabling problems is an interesting future work.
- All source code is available from `http://kaminari.istc.kobe-u.ac.jp/resource/ctt/cttasp-0.8.tgz`.

Supplemental Slides

How about Sugar?

- We have tried to solve the CB-CTT problem by using our SAT-based constraint solver **Sugar** since 2008, but have not obtained good results.
- Especially, Sugar is not scalable for very large instances due to expensive SAT encoding of cardinality constraints.
- In contrast, the ASP solver clasp deals with weighted-cardinality constraints without encoding and showed good performance in this problem.
- In addition, the direct symbolic processing is an advantage of ASP.

Other Statistics of Experiments

Details of Experimental Results

Benchmark	status	UD1	UD2	UD3	UD4	UD5
comp*	updated	1	0	9	5	0
(21 instances)	same	8	3	5	2	1
	optimal	0	0	8	4	0
DDS*	updated	1	0	0	2	0
(7 instances)	same	5	5	2	0	1
	optimal	1	0	0	1	0
erlangen*	updated	4	0	4	4	4
(4 instances)	same	0	0	0	0	0
	optimal	0	0	0	0	0
test*	updated	0	0	0	0	0
(4 instances)	same	1	0	2	1	0
	optimal	0	0	1	1	0

Comparison to the Best Known Bounds

Table shows a comparison to the previously known best bounds.
updated: #bounds that our method updated
same: #bounds that our method reached

Benchmark	status	UD1	UD2	UD3	UD4	UD5
comp*	updated	1	0	9	5	0
(21 instances)	same	8	3	5	2	1
DDS*	updated	1	0	0	2	0
(7 instances)	same	5	5	2	0	1
erlangen*	updated	4	0	4	4	4
(4 instances)	same	0	0	0	0	0
test*	updated	0	0	0	0	0
(4 instances)	same	1	0	2	1	0

We succeeded either in updating the previous best bounds or reaching the same bounds for 70 combinations (39% in total).

New Results

- Very recently two new benchmark sets have been added to the web portal for CB-CTT (no best bounds are registered).
  - `EasyAcademy` consisting of 12 instances denoted by `EA*` from various Italian universities
  - `Udine` consisting of 9 real-world instances denoted by `Udine*` from the school of Engineering of University of Udine.
- We solved 21 instances as optimization problems with five different formulations UD1–UD5 (105 combinations in total).

Benchmark	status	UD1	UD2	UD3	UD4	UD5
EA*	optimal	12	10	0	0	0
Udine*	optimal	7	7	0	0	0

Our encoding was able to give upper bounds for 105 combinations. In the case of UD1 and UD2, it succeeded in finding new optimal solutions for 34 combinations (80% in the tested 42 combinations).

ASP Representation of `toy.ectt`

toy.ectt and its ASP Representation

toy.ectt

```
Name: Toy
Courses: 4
Rooms: 3
Days: 5
Periods_per_day: 4
Curricula: 2
Min_Max_Daily_Lectures: 2 3
UnavailabilityConstraints: 8
RoomConstraints: 3
```

ASP

```
name("Toy").
courses(4).
rooms(3).
days(5).
periods_per_day(4).
curricula(2).
min_max_daily_lectures(2,3).
unavailabilityconstraints(8).
roomconstraints(3).
```

- The first nine facts express the scalar values of each entity.
- This instance named `Toy` consists of
  - 4 courses,
  - 3 rooms,
  - 2 curricula,
  - 8 unavailability constraints, and
  - 3 room constraints.
- The weekly timetable consists of 5 days and 4 periods per day, where they start from 0.

`toy.ectt` and its ASP Representation

toy.ectt

```
COURSES:
SceCosC Ocra 3 3 30 1
ArcTec Indaco 3 2 42 0
TecCos Rosa 5 4 40 1
Geotec Scarlatti 5 4 18 1
```

ASP

```
course("SceCosC","Ocra",3,3,30,1).
course("ArcTec","Indaco",3,2,42,0).
course("TecCos","Rosa",5,4,40,1).
course("Geotec","Scarlatti",5,4,18,1).
```

- The fact `course`(C, T, N, MWD, M, DL) expresses that a course C taught by a teacher T has N lectures, which must be spread into MWD days.
- The number of students that attend the course C is M.
- The course C requires double lectures if $DL = 1$.

toy.ectt and its ASP Representation

toy.ectt

```
ROOMS:
rA 32 1
rB 50 0
rC 40 0

CURRICULA:
Cur1 3 SceCosC ArcTec TecCos
Cur2 2 TecCos Geotec
```

ASP

```
room(rA,32,1).
room(rB,50,0).
room(rC,40,0).

curricula("Cur1","SceCosC").
curricula("Cur1","ArcTec").
curricula("Cur1","TecCos").
curricula("Cur2","TecCos").
curricula("Cur2","Geotec").
```

- The fact `room(`R`,`CAP`,`BLD`)` expresses that a room R located in a building BLD has a seating capacity of CAP.
- The fact `curricula(`CUR`, `C`)` expresses that a curriculum CUR includes a course C.

toy.ectt and its ASP Representation

toy.ectt

```
UNAVAILABILITY_CONSTRAINTS:
TecCos 2 0
TecCos 2 1
TecCos 3 2
TecCos 3 3
ArcTec 4 0
ArcTec 4 1
ArcTec 4 2
ArcTec 4 3

ROOM_CONSTRAINTS:
SceCosC rA
Geotec rB
TecCos rC
```

ASP

```
unavailability_constraint("TecCos",2,0).
unavailability_constraint("TecCos",2,1).
unavailability_constraint("TecCos",3,2).
unavailability_constraint("TecCos",3,3).
unavailability_constraint("ArcTec",4,0).
unavailability_constraint("ArcTec",4,1).
unavailability_constraint("ArcTec",4,2).
unavailability_constraint("ArcTec",4,3).

room_constraint("SceCosC",rA).
room_constraint("Geotec",rB).
room_constraint("TecCos",rC).
```

- The fact `unavailability_constraint(`C`,`D`,`P`)`, which is used to specify H_4, expresses that a course C is not available at a period P on a day D.
- The fact `room_constraint(`C`,`R`)`, which is used to specify S_8, expresses that a room R is not suitable for a course C.

ASP Output

```
assigned("SceCosC",rB,3,0). assigned("SceCosC",rB,2,2).
assigned("SceCosC",rB,4,2). assigned("ArcTec",rB,3,1).
assigned("ArcTec",rB,0,2).  assigned("ArcTec",rB,1,2).
assigned("TecCos",rB,0,1).  assigned("TecCos",rB,0,3).
assigned("TecCos",rB,1,3).  assigned("TecCos",rB,2,3).
assigned("TecCos",rB,4,3).  assigned("Geotec",rA,4,1).
assigned("Geotec",rA,0,2).  assigned("Geotec",rA,1,2).
assigned("Geotec",rA,2,2).  assigned("Geotec",rA,4,2).
```

- The above shows an optimal solution with zero cost of the tiny instance `toy.ectt` with the UD2 formulation.
- In this solution, all three lectures of the course `SceCosC` are assigned to the room `rB` at
  - the third period (2) on Wednesday (2),
  - the first period (0) on Thursday (3), and
  - the third period (2) on Friday (4).

Details of Soft Constraints

Soft Constraints

Soft constraints are divided into two types:

- ones with **constant cost** (S_3 and S_7–S_9)
- ones with **calculated cost** (S_1–S_2 and S_4–S_6)

- S_1. **RoomCapacity**
  - For each lecture, the number of students that attend the course must be less than or equal the number of seats of all the rooms that host its lectures.
  - The penalty points, reflecting the number of students above the capacity, are imposed on each violation.
- S_2. **MinWorkingDays**
  - The lectures of each course must be spread into a given minimum number of days.
  - The penalty points, reflecting the number of days below the minimum, are imposed on each violation.

Soft Constraints

- S_3. **IsolatedLectures**
  - Lectures belonging to a curriculum should be adjacent to each other in consecutive timeslots.
  - For a given curriculum we account for a violation every time there is one lecture not adjacent to any other lecture within the same day.
  - Each isolated lecture in a curriculum counts as 1 violation.
- S_4. **Windows**
  - Lectures belonging to a curriculum should not have time windows (periods without teaching) between them.
  - For a given curriculum we account for a violation every time there is one window between two lectures within the same day.
  - The penalty points, reflecting the length in periods of time window, are imposed on each violation.

Soft Constraints

- S_5. **RoomStability**
  - All lectures of a course should be given in the same room.
  - The penalty points, reflecting the number of distinct rooms but the first, are imposed on each violation.
- S_6. **StudentMinMaxLoad**
  - For each curriculum the number of daily lectures should be within a given range.
  - The penalty points, reflecting the number of lectures below the minimum or above the maximum, are imposed on each violation.

Soft Constraints

- S_7. **TravelDistance**
  - Students should have the time to move from one building to another one between two lectures.
  - For a given curriculum we account for a violation every time there is an **instantaneous move**:
    - two lectures in rooms located in different building in two adjacent periods within the same day.
  - Each instantaneous move in a curriculum counts as 1 violation.
- S_8. **RoomSuitability**
  - Some rooms may be not suitable for a given course because of the absence of necessary equipment.
  - Each lecture of a course in an unsuitable room counts as 1 violation.

Soft Constraints

- S_9. **DoubleLectures**
  - Some courses require that lectures in the same day are grouped together (**double lectures**).
  - For a course that requires grouped lectures, every time there is more than one lecture in one day, a lecture non-grouped to another is not allowed.
  - Two lectures are grouped if they are adjacent and in the same room.
  - Each non-grouped lecture counts as 1 violation.

Details of Encoding Hard Constraints

Direct Encoding

- The most direct modeling would be using a quaternary predicate `assigned/4`.
- The predicate `assigned(`C`,`R`,`D`,`P`)` is intended to express that a lecture of a course C is assigned to a room R at a period P on a day D.

Direct Encoding of H_1

```
% H1. Lectures
N {assigned(C,R,D,P):r(R):d(D):ppd(P)} N :- course(C,_,N,_,_,_).
:- not {assigned(C,R,D,P):r(R)} 1, c(C), d(D), ppd(P).
```

- It uses special constructs called **cardinality expressions** of the form $\ell\{a_1, \ldots, a_k\}u$ where each a_i is an atom and ℓ and u are non-negative integers denoting the lower bound and the upper bound of the cardinality expression.
- For H_1, the first rule, for every course `C` having `N` lectures, generates a solution candidate at first and then constrains that there are exactly `N` lectures such that `assigned(C,R,D,P)` holds.
- The second rule constrains that, for every course `C`, day `D`, and period `P`, there is at most one room `R` such that `assigned(C,R,D,P)` holds.

Direct Encoding of H_2

```
% H2. Conflicts
:- not {assigned(C,R,D,P):r(R):course(C,T,_,_,_,_)} 1,
   t(T), d(D), ppd(P).
:- not {assigned(C,R,D,P):r(R):curricula(Cu,C)} 1,
   cu(Cu), d(D), ppd(P).
```

- For H_2, the third rule constrains that, for every teacher `T`, day `D`, and period `P`, there is at most one course `C` taught by `T` such that `assigned(C,R,D,P)` holds.
- The fourth rule constrains that, for every curriculum `Cu`, day `D`, and period `P`, there is at most one course `C` that belongs to `Cu` such that `assigned(C,R,D,P)` holds.

Direct Encoding of H_3 and H_4

```
% H3. RoomOccupancy
:- not {assigned(C,R,D,P):c(C)} 1, r(R), d(D), ppd(P).
```

- For H_3, the fifth rule constrains that, for every room `R`, day `D`, and period `P`, there is at most one course `C` such that `assigned(C,R,D,P)` holds.

```
% H4. Availability
:- assigned(C,R,D,P), r(R), unavailability_constraint(C,D,P).
```

- For H_4, the sixth rule constrains that, for every room `R`, a course `C` is not assigned to a room `R` at a period `P` on a day `D`, if `unavailability_constraint(C,D,P)` holds.

Linked Encoding

- It is obvious that we do not always have to take account of the room information to specify the hard constraints except H_3.

```
% H1. Lectures
N {assigned(C,D,P):d(D):ppd(P)} N :- course(C,_,N,_,_,_).
% H2. Conflicts
:- not {assigned(C,D,P):course(C,T,_,_,_,_)} 1, t(T), d(D), ppd(P).
:- not {assigned(C,D,P):curricula(Cu,C)} 1, cu(Cu), d(D), ppd(P).
% H3. RoomOccupancy
1 {assigned(C,R,D,P):r(R)} 1 :- assigned(C,D,P).
:- not {assigned(C,R,D,P):c(C)} 1, r(R), d(D), ppd(P).
% H4. Availability
:- assigned(C,D,P), unavailability_constraint(C,D,P).
```

- The difference from the direct encoding is that we use a ternary predicate `assigned/3` in addition to `assigned/4`.
- The predicate `assigned`(C, D, P) is intended to express that a lecture of a course C is assigned to a period P on a day D.

Linked Encoding

- The hard constraints except H_3 are expressed by the first three and sixth rules that are slightly modified to adjust the predicate `assigned/3` by just deleting `r(R)` from the corresponding rules of the direct encoding.
- For H_3, the fourth rule first generates a solution candidate and then constrains that there is exactly one room `R` such that `assigned(C,R,D,P)` holds if `assigned(C,D,P)` holds.
- That is, the predicate `assigned/3` is linked to `assigned/4` in this rule.

Linked Encoding

- The fifth rule is the same as one of the direct encoding.
- In the linked encoding, the constraints can be expressed more concisely by using different predicates for each, than the direct encoding.
- In addition, the following rule constrains that, for a given number of rooms `N`, and for every day `D` and period `P`, there are at most `N` lectures such that `assigned(C,D,P)` holds.

```
:- not { assigned(C,D,P) : c(C) } N, d(D), ppd(P), rooms(N).
```

Comparison between Direct and Linked Encoding

Comparison between Direct and Linked Encoding

- To evaluate the efficiency of our proposed encodings, we carry out experiments on four different benchmark sets:
  - ITC-2007 [Gaspero et al. 2007] consisting of 21 instances denoted by comp*,
  - DDS-2008 [Bonutti et al. 2012] of 7 instances by DDS*,
  - Erlangen of 4 instances by erlangen*, and
  - Test [Gaspero and Schaerf 2003] of 4 instances by test*.
- We solve the instances as decision problems by taking only the hard constraints into account.
- We use the grounder gringo 3.0.4 and the solver clasp 2.1.1 on Mac OS X with 1.8 GHz Intel Core i7 and 4 GB memory.

Comparison between Direct and Linked Encoding

- Results of Direct Encoding (Average)

Benchmark	CPU	#Choices	#Conf.	#Restarts
comp*	3.0	109870	40965	10.7
DDS*	27.6	4189584	183231	395.0
erlangen*	59.6	2620135	18391	0.8
test*	2.1	104655	58312	18.5

- Results of Linked Encoding (Average)

Benchmark	CPU	#Choices	#Conf.	#Restarts
comp*	1.2	14585	8924	1.0
DDS*	3.5	371239	9512	1.4
erlangen*	27.9	113006	1	0.0
test*	0.4	6359	4035	0.3

- The **linked encoding** is 3 times faster, 12, 9, and 95 times smaller on the number of choices, conflicts, and restarts respectively than the direct encoding on the average.
- From these observations, we decide to adopt the **linked encoding** as a basis for expressing the soft constraints.

Direct and Linked Encoding (Summary)

Encoding of Constraints

Auxiliary Rules

```
c(C)          :- course(C,_,_,_,_,_).
t(T)          :- course(_,T,_,_,_,_).
r(R)          :- room(R,_,_).
cu(Cu)        :- curricula(Cu,_).
d(0..D-1)     :- days(D).
ppd(0..P-1)   :- periods_per_day(P).
```

- `course(C,T,N,MWD,M,DL)`: a course `C` taught by a teacher `T` has `N` lectures, which must be spread into `MWD` days.
- `room(R,CAP,BLD)`: a room `R` located in a building `BLD` has a seating capacity of `CAP`.
- `curricula(CUR,C)`: a curriculum `CUR` includes a course `C`.
- First four rules generate `c(C)`, `t(T)`, `r(R)`, and `cu(Cu)`.
- Last two rules generate `d(0..D-1)` and `ppd(0..P-1)` express the days and the periods per day.

Encoding of Hard Constraints

H_1. Lectures

All lectures of each course must be scheduled, and they must be assigned to distinct timeslots.

Encoding of H_1. Lectures

```
N {assigned(C,R,D,P):r(R):d(D):ppd(P)} N :- course(C,_,N,_,_,_).
:- not {assigned(C,R,D,P):r(R)} 1, c(C), d(D), ppd(P).
```

A predicate `assigned(C,R,D,P)` is used for encoding, which expresses that a lecture of a course `C` is assigned to a room `R` at a period `P` on a day `D`.

1. For every course `C` having `N` lectures, generates a solution candidate at first and then constrains that there are exactly `N` lectures such that `assigned(C,R,D,P)` holds.
2. For every course `C`, day `D`, and period `P`, there is at most one room `R` such that `assigned(C,R,D,P)` holds.

Encoding of Hard Constraints

H_3. RoomOccupancy

Two lectures can not take place in the same room in the same timeslot.

Encoding of H_3. RoomOccupancy

```
:- not {assigned(C,R,D,P):c(C)} 1, r(R), d(D), ppd(P).
```

- For every room R, day D, and period P, there is at most one course C such that assigned(C,R,D,P) holds.

Improving Encoding of Hard Constraints

- By introducing a ternary predicate `assigned(C,D,P)` in addition to `assigned(C,R,D,P)`, we do not have to take account of the room information for H_1, H_2, and H_4.
- A predicate `assigned(C,D,P)` expresses that a lecture of a course `C` is assigned to a period `P` on a day `D`.

Encoding of H_1 and H_3 with `assigned(C,D,P)`

```
% H1. Lectures
N {assigned(C,D,P):d(D):ppd(P)} N :- course(C,_,N,_,_,_).
% H3. RoomOccupancy
1 {assigned(C,R,D,P):r(R)} 1 :- assigned(C,D,P).
:- not {assigned(C,R,D,P):c(C)} 1, r(R), d(D), ppd(P).
```

- H_1, H_2, H_4 can be encoded as almost same as the one using `assigned(C,R,D,P)`.
- For H_3, the second rule generates a solution candidate at first and then constrains that there is exactly one room `R` such that `assigned(C,R,D,P)` holds if `assigned(C,D,P)` holds.

Experiments on only Hard Constraints

Comparing Two Encodings (on 1.8 GHz CPU, 4GB Mem.)

- One using `assigned(C,R,D,P)` (Direct Encoding)
- One additionally using `assigned(C,D,P)` (Linked Encoding)

(Grounder & Solver) gringo 3.0.4, clasp 2.1.1

Benchmark (considering only Hard Constraints)

- `ITC-2007` [Gaspero et al. 2007] (21 instances) `comp*`
- `DDS-2008` [Bonutti et al. 2012] (7 instances) `DDS*`
- `Erlangen` (4 instances) `erlangen*`
- `Test` [Gaspero and Schaerf 2003] (4 instances) **test***

The linked encoding is 3 times faster (CPU time), 12 times (#choices), 9 times (#conflicts), and 95 times (#restarts) smaller than the direct encoding on the average.

Encoding of Soft Constraints

We present an ASP encoding of the soft constraints based on the linked encoding.

- A predicate `penalty(`S_i`,V,C)` expresses that a constraint S_i is violated by `V` and its penalty cost is `C`.
- Each constraint S_i is expressed by either one or two rules in which the head is the form of `penalty(`S_i`,V,C)`.
- A violation `V` and its penalty cost `C` are detected and calculated respectively in the body.
- The constants denoted by `penalty_of_*` indicate the weights associated with each soft constraint defined.

Encoding of Soft Constraints S_1

S_1. RoomCapacity

- For each lecture, the number of students that attend the course must be less than or equal the number of seats of all the rooms that host its lectures.
- The penalty points, reflecting the number of students above the capacity, are imposed on each violation.

Encoding of S1. RoomCapacity

```
penalty("RoomCapacity",assigned(C,R,D,P),
                  (N-Cap)*penalty_of_room_capacity) :-
  assigned(C,R,D,P), course(C,_,_,_,N,_), room(R,Cap,_), N > Cap.
```

Encoding of Soft Constraints S_2

S_2. MinWorkingDays

- The lectures of each course must be spread into a given minimum number of days.
- The penalty points, reflecting the number of days below the minimum, are imposed on each violation.

```
working_day(C,D) :- assigned(C,D,P).
penalty("MinWorkingDays",course(C,MWD,N),
        (MWD-N)*penalty_of_min_working_days) :-
        course(C,_,_,MWD,_,_), N = [ working_day(C,_) ], N < MWD.
```

Full Encoding

- The objective of the CB-CTT problem is to find a feasible solution of minimal penalty costs.
- The **objective function** of the problem is expressed by the following only one rule.

```
#minimize [ penalty(_,_,P) = P ].
```

- Full linked encoding consists of the following rules.
  1. Auxiliary rules
  2. Hard constraints H_1–H_4
  3. Soft constraints S_1–S_9
     (differ by which formulation we consider)
  4. Objective function