ASP-based Large Neighborhood Prioritized Search for Course Timetabling

Irumi Sugimori[1] Katsumi Inoue[2] Hidetomo Nabeshima[3]
Torsten Schaub[4] Takehide Soh[5] Naoyuki Tamura[5] Mutsunori Banbara[1]

[1]Nagoya University

[2]National Institute of Informatics

[3]University of Yamanashi

[4]Universität Potsdam

[5]Kobe University

LPNMR2024@Dallas
October 14th, 2024

Overview

- **Curriculum-based Course Timetabling** (CB-CTT) is one of the most studied educational timetabling problems. CB-CTT has been used in international timetabling competitions.
- **Large Neighborhood Prioritized Search** (LNPS; [Sugimori+,'24][1]) is a hybrid between systematic and stochastic local search for solving Combinatorial Optimization Problems (COPs).

Contributions

We present an approach to solve CB-CTT with LNPS based on ASP.

1. We develop domain-specific LNPS heuristics for CB-CTT solving.
2. The resulting *teaspoon-lnps* system demonstrates that LNPS can significantly enhance the performance of ASP for CB-CTT solving.

[1] Large Neighborhood Prioritized Search for Combinatorial Optimization with Answer Set Programming, KR2024

Timetabling

Timetabling is the task of assigning a set of entities (e.g., tasks, events, people) to the limited number of resources over time, subject to a given set of hard and soft constraints.

- The typical topics of this area include:

  Educational timetabling, Transport timetabling, Healthcare timetabling, Employee timetabling, Sports timetabling.
- International conferences and competitions have been held:
  - **PATAT**: International Series of Conferences on the Practice and Theory of Automated Timetabling ▸ Web
  - **ITC**: International Timetabling Competitions ▸ Web

Timetabling has received increasing attention from both researchers and practitioners.

Curriculum-based Course Timetabling (CB-CTT)

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

Hard constraints	Soft constraints	
H_1. Lectures	S_1. RoomCapacity	S_6. StudentMinMaxLoad
H_2. Conflicts	S_2. MinWorkingDays	S_7. TravelDistance
H_3. RoomOccupancy	S_3. IsolatedLectures	S_8. RoomSuitability
H_4. Availability	S_4. Windows	S_9. DoubleLectures
	S_5. RoomStability	

- Each lecture belonging to a **course** must take place in a **room** at a **period** on a **day**.
- The **hard constraints** must be strictly satisfied.
- The **soft constraints** are not necessarily satisfied but the sum of their violations should be minimal.

Formulations

- Soft constraints are different in each **formulation**.
- **UD2** was used in ITC-2007. **UD5** is the most difficult formulation.

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	-

teaspoon encoding [Banbara+,'13,'19]

teaspoon encoding is a collection of ASP encodings for CB-CTT solving.

Lessons learned from teaspoon encoding

A proper way of using different atoms depending on constraints can improve the efficiency of CB-CTT solving.

- The most direct encoding uses the predicate `assigned/4` only.
  - The atom `assigned(C,R,D,P)` represents that a lecture of a course `C` is assigned to a room `R` at a period `P` on a day `D`.
- The *teaspoon* encoding uses two different predicates `assigned/3` and `assigned/4` depending on constraints.
  - The atom `assigned(C,D,P)` drops the room information.

How does assigned/4 construct the solution?

The partial solution on the left forms the weekly timetables for the two curricula on the right.

partial answer set

```
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

Motivation

- ***teaspoon*** demonstrated that ASP can compete with state-of-the-art solving techniques on course timetabling, including
  - metaheuristic algorithms, integer programming, SAT/MaxSAT, etc.
- In fact, *teaspoon* managed to either improve or reproduce the best known bounds for 182 out of 305 combinations in total (all 61 instances in 5 formulations).

Motivation

- ***teaspoon*** demonstrated that ASP can compete with state-of-the-art solving techniques on course timetabling, including
  - metaheuristic algorithms, integer programming, SAT/MaxSAT, etc.
- In fact, *teaspoon* managed to either improve or reproduce the best known bounds for 182 out of 305 combinations in total (all 61 instances in 5 formulations).

Limitation of *teaspoon*

Especially for UD5, systematic search of ASP solvers quickly falls into saturated solutions for many instances.

Motivation

- ***teaspoon*** demonstrated that ASP can compete with state-of-the-art solving techniques on course timetabling, including
  - metaheuristic algorithms, integer programming, SAT/MaxSAT, etc.
- In fact, *teaspoon* managed to either improve or reproduce the best known bounds for 182 out of 305 combinations in total (all 61 instances in 5 formulations).

Limitation of *teaspoon*

Especially for UD5, systematic search of ASP solvers quickly falls into saturated solutions for many instances.

Idea of this research

We tackle this problem issue by taking advantage of **Large Neighborhood Prioritized Search** (LNPS).

Large Neighborhood Prioritized Search [Sugimori+,KR'24]

LNPS is a metaheuristic that starts with an initial solution and then iteratively tries to find better solutions by alternately destroying a current solution and reconstructing it with prioritized search.

- The prioritized search is a systematic search for which its branching heuristic can be configured or customized.
- The prioritized search can be implemented with heuristic-driven ASP solving (viz. `#heuristic` statement).

Algorithm 1 LNPS

Input: a feasible (current) solution x
Output: the best solution x^*

```
x* ← x
while stop criterion is not met do
    x^t ← prioritized-search(destroy(x))
    if accept(x^t, x) then
        x ← x^t
    end if
    if c(x^t) < c(x*) then
        x* ← x^t
    end if
end while
return x*
```

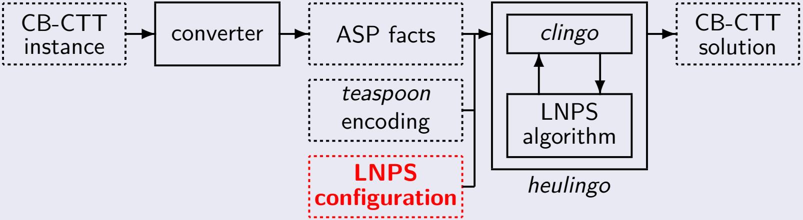
heulingo: an ASP-based implementation of LNPS

- **Variability**: *heulingo* provides a flexible search without strongly depending on the destroy operators compared to traditional LNS [Shaw,'98], since the undestroyed part is not fixed.
- **Optimality**: *heulingo* can guarantee the optimality of solutions.
- **Domain heuristics**: *heulingo* allows for easy incorporation of domain-specific heuristics in a declarative way.
- **Efficiency**: *heulingo* has demonstrated that LNPS can significantly enhance the solving performance of ASP on hard optimization problems, such as traveling salesperson, social golpher, sudoku generation, weighted strategic company and so on.

We focus on the development of domain-specific LNPS heuristics for efficient CB-CTT solving.

teaspoon-lnps approach (proposal)

We develop an approach to solve CB-CTT with ASP-based LNPS.



1. The resulting *teaspoon-lnps* solver accepts a CB-CTT instance and converts it into ASP facts.
2. In turn, these facts are combined with *teaspoon* encoding and an LNPS configuration, which are afterward solved by *heulingo*.

- LNPS configurations define the behavior of the LNPS heuristic, especially for the destroy and prioritized-search operators.

LNPS configurations

We develop five LNPS configurations for CB-CTT solving.

- **Random** is the simplest domain-independent heuristic that randomly destroys a current solution.
- **Day-Period** and **Day-Room** are domain-specific heuristics inspired by the traditional LNS heuristic for CB-CTT solving [Kiefer+,'17].
- Swap-Room and DP-Swap-Room are novel domain-specific configurations for CB-CTT solving, taking advantage of prioritized search.

Random N

Random N randomly destroys N% of a current solution and keeps the undestroyed part as much as possible in each iteration of LNPS.

```
#program config.
_lnps_project(assigned,4).
_lnps_destroy(assigned,4,15,p(n)).
_lnps_prioritize(assigned,4,1,true).
```

- The atom `_lnps_project(assigned,4)` means that the atoms of `assigned/4` belonging to an answer set are subject to LNPS.
- The atom `_lnps_destroy(assigned,4,15,p(n))` means that `n`% of a current solution characterized by `assigned/4` are destroyed.
  - The 3rd argument $15 = (1111)_2$ represents that all possible tuples (`C`,`R`,`D`,`P`) of `assigned(C,R,D,P)` are subject to destruction.

Random N

Random N randomly destroys N% of a current solution and keeps the undestroyed part as much as possible in each iteration of LNPS.

```
#program config.
_lnps_project(assigned,4).
_lnps_destroy(assigned,4,15,p(n)).
_lnps_prioritize(assigned,4,1,true).
```

- The atom `_lnps_prioritize(assigned,4,1,true)` means that the undestroyed part is kept as much as possible.
- Technically, this atom corresponds to *clingo*'s heuristic statement

  `#heuristic assigned(C,R,D,P):`*Body*`. [1,true]`.

Day-Period

Day-Period randomly selects a single day-period pair. In turn, it destroys parts of a current solution including the selected pair and keeps the undestroyed part as much as possible.

```
1 #program config.
2 _lnps_project(assigned,4).
3 _lnps_destroy(assigned,4,3,n(1)).
4 _lnps_prioritize(assigned,4,1,true).
```

- The difference from Random *N* is the fact in Line 3.
- The fact means that parts of a current solution characterized by `assigned/4` containing a randomly selected **day-period pair** are destroyed.
  - The 3rd argument $3 = (0011)_2$ represents that all possible pairs (`D`,`P`) of `assigned(C,R,D,P)` are subject to destruction.

Swap-Room N

- Swap-Room N is similar to Random N, but tries to keep course-day-period assignments as much as possible.
- This is designed to find better solutions by swapping rooms in the current solution.

```
1 #program config.
2 _lnps_project(assigned,4).
3 _lnps_destroy(assigned,4,15,p(n)).
4 _lnps_prioritize(assigned,4,1,true).
5 _lnps_project(assigned,3).
6 _lnps_destroy(assigned,3,7,p(0)).
7 _lnps_prioritize(assigned,3,1,true).
```

- The difference from Random N is the addition of Lines 5–7.
- The additional facts mean that the course-day-period assignments represented by `assigned/3` are kept as much as possible.

DP-Swap-Room N

- DP-Swap-Room N is similar to Swap-Room N, but randomly selects N day-period pairs and destroys all atoms including any selected pair.
- This is designed to find better solutions by swapping rooms at N day-period pairs in the current solution.

```
1  #program config.
2  _lnps_project(assigned,4).
3  _lnps_destroy(assigned,4,3,n(n)).
4  _lnps_prioritize(assigned,4,1,true).
5  _lnps_project(assigned,3).
6  _lnps_destroy(assigned,3,7,p(0)).
7  _lnps_prioritize(assigned,3,1,true).
```

- The difference from Swap-Room N is the fact in Line 3.
- The fact means that parts of a current solution containing any one of randomly selected `n` day-period pairs are destroyed.

Experiments

We carry out experiments to evaluate our *teaspoon-lnps* approach.

- We use all 61 instances of standard CB-CTT benchmark set within the most difficult **UD5 formulation**.
- ASP encoding: *teaspoon* encoding [Banbara+,'13,'19]
- We compare
  - *clingo*-5.6.2 with the two best options [Banbara+,'19].
  - ***teaspoon-lnps*** with the five configurations:
    1. Random N with 3 different percentages $N \in \{2, 4, 6\}$
    2. Day-Period
    3. Day-Room
    4. Swap-Room N with 3 different percentages $N \in \{8, 10, 12\}$
    5. DP-Swap-Room N with 3 different cardinalities $N \in \{1, 2, 3\}$
- Time limit: 1 hour for each

Comparison of LNPS configurations on ITC-2007

Instance	*clingo*		*teaspoon-lnps*										
	bb	usc	Random N			Day-Period	Day-Room	Swap-Room N			DP-Swap-Room N		
			2%	4%	6%			8%	10%	12%	1	2	3
comp01	115	283	13	**11**	13	**11**	**11**	**11**	**11**	**11**	**11**	**11**	**11**
comp02	989	331	269	249	196	187	244	**178**	230	218	221	213	216
comp03	791	302	207	207	189	189	194	172	175	194	**165**	168	174
comp04	66	* **49**	* **49**	* **49**	* **49**	* **49**	* **49**	* **49**	* **49**	* **49**	* **49**	* **49**	* **49**
comp05	2662	1940	929	964	1011	**753**	863	1016	803	839	1040	936	978
comp06	777	1025	229	218	221	184	195	185	174	207	166	**156**	213
comp07	924	1149	196	197	342	163	172	**144**	243	251	203	233	181
comp08	332	* **55**	* **55**	* **55**	* **55**	* **55**	* **55**	* **55**	* **55**	* **55**	* **55**	* **55**	* **55**
comp09	563	254	183	175	174	165	161	173	169	160	168	179	**156**
comp10	821	1229	196	179	195	170	153	**127**	160	147	147	180	167
comp11	287	* **0**	* **0**	* **0**	* **0**	* **0**	* **0**	* **0**	* **0**	* **0**	* **0**	* **0**	* **0**
comp12	2626	1246	656	649	657	664	677	637	624	642	638	644	**622**
comp13	652	301	196	187	178	181	196	165	161	**160**	164	168	166
comp14	746	* **67**	158	165	168	169	196	122	119	111	130	113	117
comp15	820	607	206	212	211	**194**	226	218	223	228	223	224	204
comp16	944	1090	191	199	244	209	217	**154**	199	177	173	200	173
comp17	979	412	216	218	239	234	238	193	**174**	207	194	178	214
comp18	503	340	158	153	158	152	158	146	149	144	148	151	**143**
comp19	890	919	186	198	209	191	205	182	182	189	194	188	**173**
comp20	3304	1386	362	391	996	403	361	400	**359**	402	393	451	544
comp21	891	310	255	268	281	238	247	193	197	179	215	**166**	183
#best bounds	0	4	3	4	3	6	4	**8**	6	5	5	6	**8**

- Swap-Room $N = 8$ and DP-Swap-Room $N = 3$ found the most number of best solutions.

Comparison with other approaches 1/3

Instance	Best known bound (♯)	*clingo* Obtained bound	*clingo* Rate to ♯ (%)	*teaspoon-lnps* Obtained bound	*teaspoon-lnps* Rate to ♯ (%)
comp01	11	115	+945	11	**0**
comp02	130	331	+154	178	**+36**
comp03	142	302	+112	165	**+16**
comp04	49	49	**0**	49	**0**
comp05	570	1940	+240	753	**+32**
comp06	85	777	+814	156	**+83**
comp07	42	924	+2100	144	**+242**
comp08	55	55	**0**	55	**0**
comp09	150	254	+69	156	**+4**
comp10	72	821	+1040	127	**+76**
comp11	0	0	**0**	0	**0**
comp12	483	1246	+157	622	**+28**
comp13	147	301	+104	160	**+8**
comp14	67	67	**0**	111	+65
comp15	176	607	+244	194	**+10**
comp16	96	944	+883	154	**+60**
comp17	155	412	+165	174	**+12**
comp18	137	340	+148	143	**+4**
comp19	125	890	+612	173	**+38**
comp20	124	1386	+1017	359	**+189**
comp21	151	310	+105	166	**+9**

- The previous best known bounds (♯) have been obtained by metaheuristic algorithms, ILP, SAT/MaxSAT, etc.
- We calculate $\frac{\text{obtained bound} - \sharp}{\sharp}$ as the rate of distance to the best known bounds.

Comparison with other approaches 2/3

Instance	Best known bound (♯)	*clingo*		*teaspoon-lnps*	
		Obtained bound	Rate to ♯ (%)	Obtained bound	Rate to ♯ (%)
DDS1	1831	6536	+256	3557	**+94**
DDS2	64	398	+521	70	**+9**
DDS3	22	22	**0**	22	**0**
DDS4	96	3123	+3153	1732	**+1704**
DDS5	76	76	**0**	76	**0**
DDS6	96	849	+784	167	**+73**
DDS7	52	645	+1140	56	**+7**
EA01	196	807	+311	207	**+5**
EA02	128	990	+673	113	**-11**
EA03	90	2555	+2738	660	**+633**
EA04	18	52	**+188**	311	+1627
EA05	14	19	+35	14	**0**
EA06	99	543	+448	152	**+53**
EA07	205	2831	+1280	1142	**+457**
EA08	40	276	+590	40	**0**
EA09	48	48	**0**	48	**0**
EA10	93	444	+377	362	**+289**
EA11	40	211	+427	40	**0**
EA12	27	234	+766	27	**0**

- *teaspoon-lnps* succeeds in finding an **improved bound** of EA02.

Comparison with other approaches 3/3

Instance	Best known bound (♯)	*clingo* Obtained bound	*clingo* Rate to ♯ (%)	*teaspoon-lnps* Obtained bound	*teaspoon-lnps* Rate to ♯ (%)
erlangen2011_2	12353	17035	+37	15480	**+25**
erlangen2012_1	28236	25061	**-11**	25763	**-8**
erlangen2012_2	37103	34360	**-7**	35220	**-5**
erlangen2013_1	28997	28302	**-2**	25339	**-12**
erlangen2013_2	30533	29140	**-4**	28870	**-5**
erlangen2014_1	28655	24510	**-14**	23926	**-16**
test1	232	607	+161	232	**0**
test2	20	20	**0**	20	**0**
test3	68	117	+72	76	**+11**
test4	166	260	+56	144	**-13**
toy	0	0	**0**	0	**0**
Udine1	138	953	+590	226	**+63**
Udine2	81	402	+396	86	**+6**
Udine3	37	333	+800	71	**+91**
Udine4	106	106	**0**	106	**0**
Udine5	47	518	+1002	46	**-2**
Udine6	36	285	+691	38	**+5**
Udine7	64	131	+104	64	**0**
Udine8	88	606	+588	107	**+21**
Udine9	56	163	+191	67	**+19**
UUMCAS_A131	19699	28688	+45	25698	**+30**
Average rate to ♯			+447		**+99**

- *teaspoon-lnps* succeeds in finding **improved bounds** of 5 erlangen instances, test4, and Udine5.
- *teaspoon-lnps* was able to significantly reduce the rate to $+99\%$ on average compared to $+447\%$ of *clingo*.

Related Work

- Besides LNPS [Sugimori+,'24], the use of adaptive LNS in ASP optimization has been explored [Eiter+,AAAI'22,KR'22,AIJ'24].
- ASP has been so far successfully applied in a wide variety of timetabling problems, such as
  - nurse scheduling problem [Alviano+,'22],
  - rehabilitation scheduling problem [Cardellini+,'24],
  - chemotherapy treatment scheduling problem [Dodaro+,'21],
  - personalized course schedule planning [Kahraman+,'19], and
  - shift design problem [Abseher+,'16].

By our results, it would be interesting to explore effective LNPS configurations for these problems.

Conclusion

We presented an approach to solve **CB-CTT** with **LNPS** based on ASP.

- *teaspoon-lnps* was able to reduce the ratio of previously best known bounds from $+447\%$ of *clingo* to **$+99\%$**.
- We succeeded in finding improved bounds of **8** instances.

All source code is available from:
https://github.com/banbaralab/lpnmr2024 Web

Future work

1. Extending our approach to the most recent ITC-2019
2. Developing effective LNPS configurations for other timetabling problems, such as nurse scheduling problem

Appendix

Discussion

☹ In general, it is a time-consuming task to find the best parameter of LNPS configurations.

- The parameter settings were obtained in our preliminary experiments.
- We first measured the distance (i.e., the percentage of change) between two consecutive intermediate solutions obtained by *teaspoon* encoding with *clingo*.
- We then tested, for each configuration, some percentages (or cardinalities) of destruction less than the distance of *clingo*, taking the variability of LNPS into account.

We plan to extend our approach for adaptive LNPS, which selects in each iteration a potentially more effective destroy operator.

Large Neighborhood Search (LNS; [Shaw,'98])

LNS is an SLS-based metaheuristic that starts with an initial solution and then iteratively tries to find better solutions by alternately **destroying** and **repairing** a current solution.

Pros and Cons

- ☺ Since the **repair** operators can be implemented with systematic solvers, LNS has been shown to be highly compatible with ASP [Eiter+,AAAI-22,KR-22], as well as MIP [Fischetti+,'03; Danna+,'05] and CP [Shaw,'98; Dekker+,'18; Björdal+,'19,'20].
- ☹ However, LNS strongly depends on the **destroy** operators since the undestroyed part is fixed.
- ☹ Moreover, in general, LNS cannot guarantee the optimality of solutions.

The main differences of LNPS from LNS

LNS

destroyed

fixed

1. The undestroyed part is fixed.
2. The percentage of destruction should be sufficiently large such that a neighborhood includes better solutions, and be sufficiently small such that the solver finds one of them.
3. The optimality cannot be guaranteed in general.

LNPS

destroyed

not fixed (varying)

1. The undestroyed part is not fixed (varying) and can be prioritized.
2. Due to this variability, the percentage of destruction can be smaller.
3. The optimality can be guaranteed by appropriately designing a stop criterion of prioritized search.

Predicates for specifying LNPS configurations

LNPS configurations define the behavior of the LNPS heuristic, especially for the destroy and prioritized-search operators.

heulingo provides 3 special predicates for specifying LNPS configurations in the `config` subprogram.

- `_lnps_project/2`: Used to define what subset of the atoms included in an answer set is subject to LNPS.
  - The atoms via `_lnps_project/2` are called **projected atoms**.
- `_lnps_destroy/4`: Used to define what part of the projected atoms is destroyed and by what percentage (or how many).
- `_lnps_prioritize/4`: Used to define how the projected atoms in the undestroyed part are prioritized.

Day-Room

Day-Room randomly selects a single day-room pair. In turn, it destroys parts of a current solution including the selected pair and keeps the undestroyed part as much as possible.

```
1 #program config.
2 _lnps_project(assigned,4).
3 _lnps_destroy(assigned,4,6,n(1)).
4 _lnps_prioritize(assigned,4,1,true).
```

- The difference from Random N is the atom `_lnps_destroy(assigned,4,6,n(1))` in Line 3.
- The atom means that parts of a current solution characterized by `assigned/4` containing a randomly selected **day-room pair** are destroyed.
- The 3rd argument $6 = (0110)_2$ represents that all possible pairs (`R`,`D`) of `assigned(C,R,D,P)` are subject to destruction.

Comparison of LNPS configurations 1/3

Instance	*clingo*		*teaspoon-lnps*										
	bb	usc	Random N			Day-Period	Day-Room	Swap-Room N			DP-Swap-Room N		
			2%	4%	6%			8%	10%	12%	1	2	3
comp01	115	283	13	**11**	13	**11**	**11**	**11**	**11**	**11**	**11**	**11**	**11**
comp02	989	331	269	249	196	187	244	**178**	230	218	221	213	216
comp03	791	302	207	207	189	189	194	172	175	194	**165**	168	174
comp04	66	***49**	***49**	***49**	***49**	***49**	***49**	***49**	***49**	***49**	***49**	***49**	***49**
comp05	2662	1940	929	964	1011	**753**	863	1016	803	839	1040	936	978
comp06	777	1025	229	218	221	184	195	185	174	207	166	**156**	213
comp07	924	1149	196	197	342	163	172	**144**	243	251	203	233	181
comp08	332	***55**	***55**	***55**	***55**	***55**	***55**	***55**	***55**	***55**	***55**	***55**	***55**
comp09	563	254	183	175	174	165	161	173	169	160	168	179	**156**
comp10	821	1229	196	179	195	170	153	**127**	160	147	147	180	167
comp11	287	***0**	***0**	***0**	***0**	***0**	***0**	***0**	***0**	***0**	***0**	***0**	***0**
comp12	2626	1246	656	649	657	664	677	637	624	642	638	644	**622**
comp13	652	301	196	187	178	181	196	165	161	**160**	164	168	166
comp14	746	***67**	158	165	168	169	196	122	119	111	130	113	117
comp15	820	607	206	212	211	**194**	226	218	223	228	223	224	204
comp16	944	1090	191	199	244	209	217	**154**	199	177	173	200	173
comp17	979	412	216	218	239	234	238	193	**174**	207	194	178	214
comp18	503	340	158	153	158	152	158	146	149	144	148	151	**143**
comp19	890	919	186	198	209	191	205	182	182	189	194	188	**173**
comp20	3304	1386	362	391	996	403	361	400	**359**	402	393	451	544
comp21	891	310	255	268	281	238	247	193	197	179	215	**166**	183

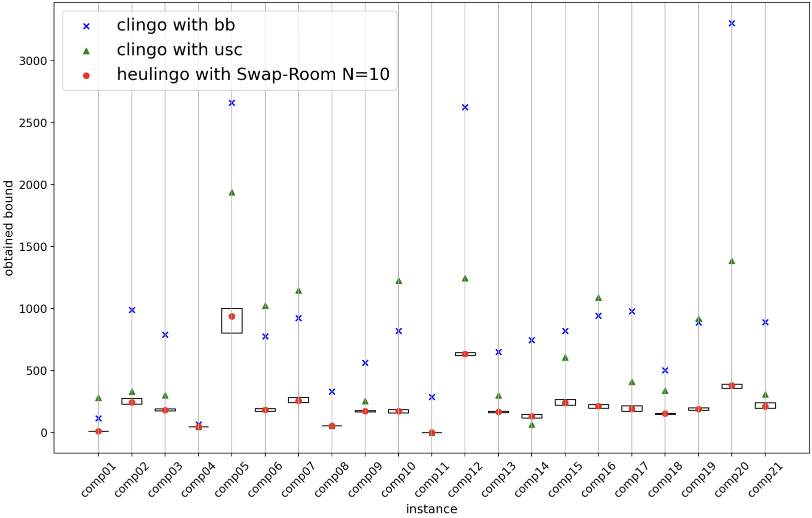
Comparison of LNPS configurations 2/3

Instance	*clingo*		*teaspoon-lnps*										
	bb	usc	Random N			Day-Period	Day-Room	Swap-Room N			DP-Swap-Room N		
			2%	4%	6%			8%	10%	12%	1	2	3
DDS1	6536	7544	4119	4063	4251	3923	4118	3805	3837	3746	4367	3823	**3557**
DDS2	398	433	76	76	75	76	77	73	72	74	72	**70**	72
DDS3	**22**	391	26	24	**22**	24	**22**	**22**	**22**	**22**	**22**	**22**	**22**
DDS4	10484	3123	3102	3129	3168	2975	2005	1768	1864	1902	**1732**	1821	1752
DDS5	538	***76**	***76**	***76**	***76**	***76**	***76**	***76**	***76**	***76**	***76**	***76**	***76**
DDS6	849	964	208	210	199	175	**167**	217	214	222	192	258	265
DDS7	645	1485	107	305	579	122	104	**56**	64	58	66	64	57
EA01	887	807	263	240	238	232	250	218	**207**	208	218	216	212
EA02	990	1215	134	148	131	140	145	114	**113**	130	138	140	129
EA03	8182	2555	1231	1509	1704	996	**660**	799	757	816	875	878	852
EA04	**52**	1873	1083	1607	1591	1018	666	413	807	778	340	353	311
EA05	19	513	22	20	20	23	23	**14**	**14**	**14**	**14**	**14**	**14**
EA06	543	1027	221	222	258	189	165	173	**152**	177	169	159	168
EA07	10103	2831	1633	2432	2758	1621	1179	1166	1440	1665	1175	1143	**1142**
EA08	276	991	123	717	868	150	77	**40**	**40**	**40**	42	**40**	**40**
EA09	**48**	865	67	54	54	55	59	**48**	**48**	**48**	**48**	**48**	**48**
EA10	1711	444	367	1272	1609	**362**	390	379	417	436	439	417	433
EA11	211	771	51	45	55	45	52	44	**40**	41	41	**40**	44
EA12	234	518	57	40	59	44	49	**27**	**27**	**27**	30	**27**	**27**

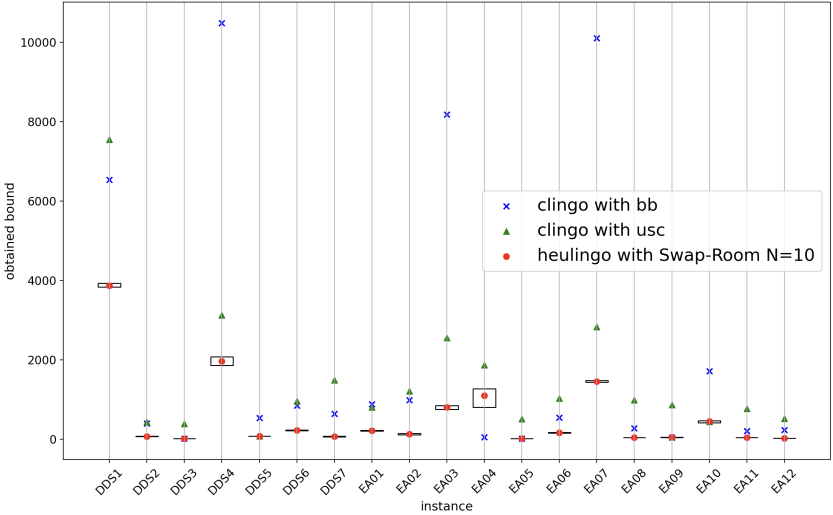
Comparison of LNPS configurations 3/3

Instance	*clingo*		*teaspoon-lnps*										
	bb	usc	Random N			Day-Period	Day-Room	Swap-Room N			DP-Swap-Room N		
			2%	4%	6%			8%	10%	12%	1	2	3
erlangen2011_2	17035	17263	15627	16009	15828	15901	15556	15743	15819	15587	15608	**15480**	15572
erlangen2012_1	32541	**25061**	25763	25763	25763	25763	25763	25763	25763	25763	25763	25763	25763
erlangen2012_2	38180	**34360**	35220	35220	35220	35220	35220	35220	35220	35220	35220	35220	35220
erlangen2013_1	29459	28302	**25339**	**25339**	**25339**	**25339**	**25339**	**25339**	**25339**	**25339**	**25339**	**25339**	**25339**
erlangen2013_2	34568	29140	**28870**	**28870**	**28870**	**28870**	**28870**	**28870**	**28870**	**28870**	**28870**	**28870**	**28870**
erlangen2014_1	30249	24510	**23926**	**23926**	**23926**	**23926**	**23926**	**23926**	**23926**	**23926**	**23926**	**23926**	**23926**
test1	607	751	233	**232**	**232**	**232**	**232**	**232**	**232**	**232**	**232**	**232**	237
test2	***20**	***20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**
test3	192	117	104	103	98	99	105	82	**76**	84	77	86	84
test4	705	260	179	194	181	**144**	170	147	155	170	165	153	157
toy	***0**	***0**	***0**	***0**	***0**	***0**	***0**	***0**	***0**	***0**	***0**	***0**	***0**
Udine1	1079	953	268	269	518	293	297	247	237	241	**226**	258	240
Udine2	505	402	100	95	95	90	98	88	96	87	**86**	92	99
Udine3	333	385	110	82	97	87	106	85	99	80	**71**	99	83
Udine4	317	***106**	***106**	***106**	***106**	***106**	***106**	***106**	***106**	***106**	***106**	***106**	***106**
Udine5	518	695	67	62	81	84	68	49	55	**46**	**46**	53	52
Udine6	285	628	40	46	51	51	53	**38**	40	**38**	39	**38**	40
Udine7	131	733	75	71	76	76	70	69	**64**	**64**	67	**64**	**64**
Udine8	606	941	126	134	112	118	126	120	**107**	118	112	134	120
Udine9	610	163	74	73	68	78	77	68	**67**	69	73	72	78
UUMCAS_A131	37274	28688	27670	28419	28350	27113	27125	28258	28474	28575	**25698**	26637	27405
#best bounds	5	10	10	12	12	16	15	23	**27**	21	22	24	23

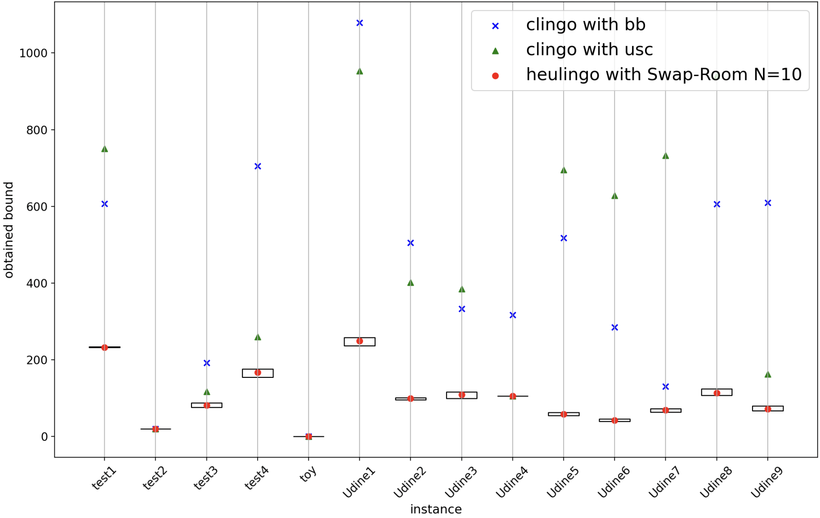
Comparison of *teaspoon-lnps* with *clingo* 1/4



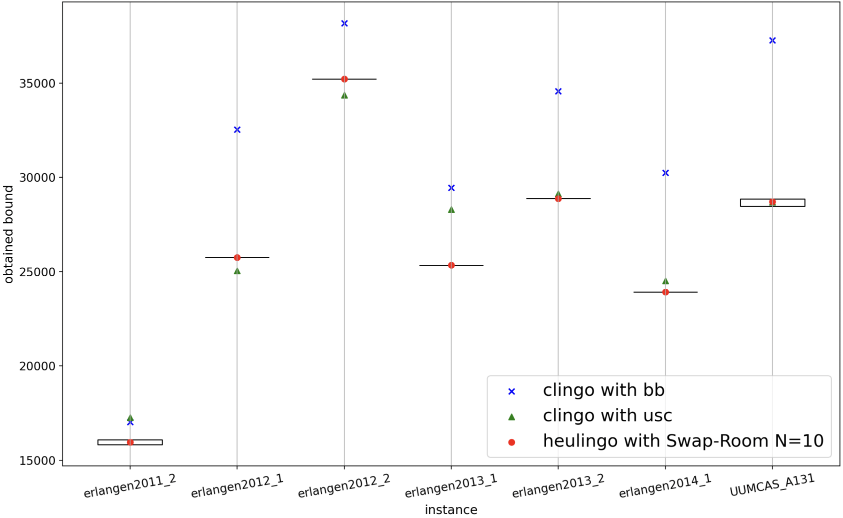
Comparison of *teaspoon-lnps* with *clingo* 2/4



Comparison of *teaspoon-lnps* with *clingo* 3/4



Comparison of *teaspoon-lnps* with *clingo* 4/4



Comparison of LNPS and LNS 1/3

Instance	LNS			LNPS		
	Random N			Random N		
	12%	14%	16%	2%	4%	6%
comp01	25	25	17	13	**11**	13
comp02	310	284	272	269	249	**196**
comp03	208	222	214	207	207	**189**
comp04	* **49**	* **49**	* **49**	* **49**	* **49**	* **49**
comp05	1208	1052	1036	**929**	964	1011
comp06	265	254	233	229	**218**	221
comp07	254	253	213	**196**	197	342
comp08	* **55**	* **55**	* **55**	* **55**	* **55**	* **55**
comp09	201	187	176	183	175	**174**
comp10	265	241	242	196	**179**	195
comp11	* **0**	* **0**	* **0**	* **0**	* **0**	* **0**
comp12	766	836	960	656	**649**	657
comp13	190	213	192	196	187	**178**
comp14	202	203	227	**158**	165	168
comp15	254	254	253	**206**	212	211
comp16	248	220	252	**191**	199	244
comp17	273	276	247	**216**	218	239
comp18	164	161	174	158	**153**	158
comp19	231	222	233	**186**	198	209
comp20	417	446	461	**362**	391	996
comp21	313	300	284	**255**	268	281

Comparison of LNPS and LNS 2/3

Instance	LNS			LNPS		
	Random N			Random N		
	12%	14%	16%	2%	4%	6%
DDS1	3411	3432	**3329**	4119	4063	4251
DDS2	84	94	88	76	76	**75**
DDS3	24	28	**22**	26	24	**22**
DDS4	**1711**	2078	2765	3102	3129	3168
DDS5	***76**	***76**	***76**	***76**	***76**	***76**
DDS6	237	209	200	208	210	**199**
DDS7	129	129	137	**107**	305	579
EA01	258	254	264	263	240	**238**
EA02	203	192	175	134	148	**131**
EA03	550	**511**	512	1231	1509	1704
EA04	82	**66**	74	1083	1607	1591
EA05	25	23	22	22	**20**	**20**
EA06	238	265	266	**221**	222	258
EA07	**1232**	1447	1899	1633	2432	2758
EA08	54	**44**	48	123	717	868
EA09	73	65	68	67	**54**	**54**
EA10	566	499	538	**367**	1272	1609
EA11	109	84	61	51	**45**	55
EA12	77	76	70	57	**40**	59

Comparison of LNPS and LNS 3/3

Instance	LNS			LNPS		
	Random N			Random N		
	12%	14%	16%	2%	4%	6%
erlangen2011_2	**15250**	15357	15294	15627	16009	15828
erlangen2012_1	**25763**	**25763**	**25763**	**25763**	**25763**	**25763**
erlangen2012_2	**35220**	**35220**	**35220**	**35220**	**35220**	**35220**
erlangen2013_1	**25339**	**25339**	**25339**	**25339**	**25339**	**25339**
erlangen2013_2	**28870**	**28870**	**28870**	**28870**	**28870**	**28870**
erlangen2014_1	**23926**	**23926**	**23926**	**23926**	**23926**	**23926**
test1	284	273	289	233	**232**	**232**
test2	31	29	25	**20**	**20**	**20**
test3	119	115	110	104	103	**98**
test4	269	273	268	**179**	194	181
toy	***0**	***0**	***0**	***0**	***0**	***0**
Udine1	300	288	**260**	268	269	518
Udine2	105	108	123	100	**95**	**95**
Udine3	136	127	108	110	**82**	97
Udine4	***106**	***106**	***106**	***106**	***106**	***106**
Udine5	81	82	**60**	67	62	81
Udine6	50	55	50	**40**	46	51
Udine7	90	78	79	75	**71**	76
Udine8	153	155	154	126	134	**112**
Udine9	94	92	87	74	73	**68**
UUMCAS_A131	**23426**	23574	23561	27670	28419	28350
#best bounds	15	14	15	26	25	**28**

teaspoon encodings of hard constraints 1/2

H_1. Lectures

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

```
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) : 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).
```

teaspoon encodings of hard constraints 2/2

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).
```

Soft Constraints 1/5

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 2/5

- 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 3/5

- 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 4/5

- 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 5/5

- 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.