# 解集合プログラミングに基づく系統的探索と確率的局所探索の統合的手法に関する研究

桑原 和也

番原研究室

2021 年度番原研中間発表会
2021 年 12 月 3 日

# 解集合プログラミング

**解集合プログラミング** (Answer Set Programming; ASP) は，論理プログラミングから派生した宣言的プログラミングパラダイム

- **ASP 言語**は，命題表現に限られる SAT を拡張し，述語表現を許した宣言的知識表現言語の一種である．
- **ASP ソルバー**は，安定モデル意味論 [Gelfond and Lifschitz '88] に基づく解集合を計算するプログラムである．
- 近年，SAT ソルバーを応用した高速 ASP ソルバーが実現され，AI 分野の諸問題への実用的応用が急速に拡大している．
- ごく最近では，時間割問題などの求解困難な**組合せ最適化問題**に対し，未解決問題の最適値決定を含む優れた結果を示している．

# 組合せ最適化問題に対して ASP を用いる利点と改善点

## 利点

- ASP 言語の**高い表現力**により，記号制約を簡潔に記述できる．
- 系統的探索 (分枝限定法) なので，**解の最適性を保証**できる．
- 探索ヒューリスティクスを簡単に試せる (#heuristic 文)．
- マルチショット ASP 解法を利用して，メタ戦略を実装できる．

## 改善点

- **大規模な問題**に対して，ASP の系統的探索は確率的局所探索と比べて，**解の精度が劣る**場合がある．

# カリキュラムベース・コース時間割問題 (CB-CTT)

- CB-CTT は代表的な教育時間割問題の一つである.
- 必ず満たすべき**ハード制約**と, できるだけ満たしたい**重み付きソフト制約**から構成される.
- 違反するソフト制約の重みの総和の最小化が目的となる.

## ASP の優れた点

- 系統的探索であることを活かして，最適値が未知であった 51 問の最適値決定に成功している [Banbara+ '19].

## 改善を要する点

- ソフト制約が多く含まれるような問題集では，確率的局所探索が, 多くの問題に対してより高精度な解を求めている.
- 既知の最良値との比の平均値は
  - ソフト制約が少ない問題集 (UD1) では +19.83%
  - ソフト制約が多い問題集 (UD5) では +100.17%

# 研究概要

## 研究目的

ASP 技術を用いて，系統的探索の長所である**最適性の保証**と確率的局所探索の長所である**計算時間相応の解精度**の両方を備えた統合的探索手法の実現を目指す.

- **方針**: 先行研究の**優先度付き巨大近傍探索 (LNPS)** [坡山ほか '18] をベースに，ASP に適した探索手法に関する研究開発を進める.

## 研究内容

1. 組合せ最適化ソルバー asprior の実装
   - LNPS を ASP ソルバー *clingo* 上に実装
2. カリキュラムベース・コース時間割問題を用いた評価実験

# LNPS: 優先度付き巨大近傍探索 [坡山・番原・田村 ’18]

組合せ最適化問題の最適値探索において，暫定解に含まれる変数の値割り当ての一部をランダムに選んで取り消し，他の値割り当てをなるべく維持したままで解を再探索する反復手法

# LNPS: 優先度付き巨大近傍探索 [坡山・番原・田村 '18]

組合せ最適化問題の最適値探索において，暫定解に含まれる変数の値割り当ての一部をランダムに選んで取り消し，他の値割り当てをなるべく維持したままで解を再探索する反復手法

## LNPS のアルゴリズム

1. 初期解を $x$ と置き，暫定解 $x^* := x$ とする．
2. 以下の destroy と re-search で $x$ から得られた解を $x^t$ と置く．
   - **destroy** は $x$ から値割当ての一部を**取り消し** $x'$ とする．
   - **re-search** は $x'$ の**値割当てをなるべく維持したまま再探索**する．
3. 受理条件を満たしていたら $x := x^t$ とする．
   - 例えば「$x^t$ が $x$ より改善された解なら」という更新条件を用いる．
4. $x^t$ が暫定解 $x^*$ より改善された解なら，$x^* := x^t$ とする．
5. 終了条件が満たされるまで，2〜4 を繰り返す．
   - 繰り返し回数や制限時間などを終了条件に用いる．
6. 暫定解 $x^*$ を返す．

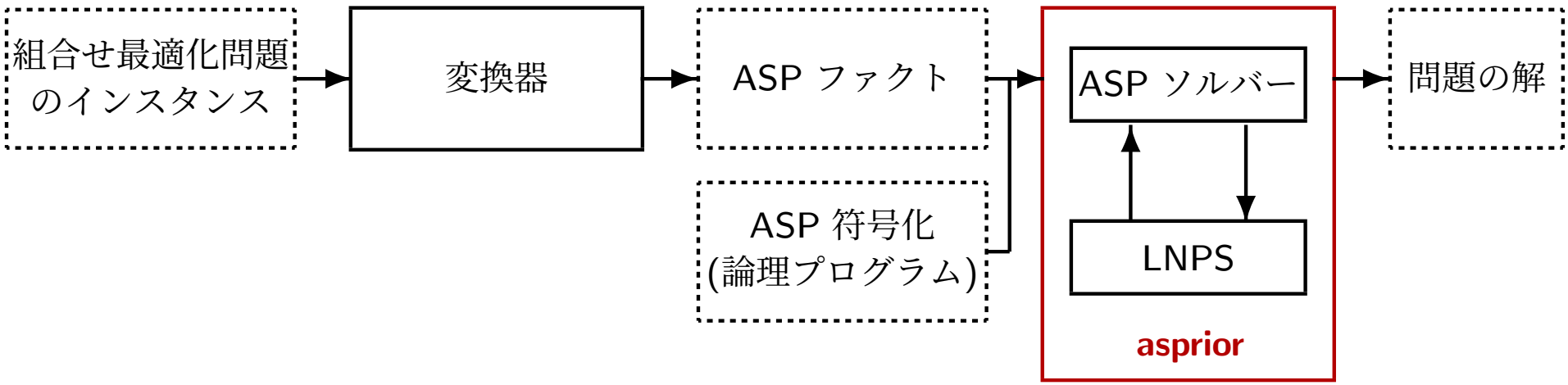
# LNPS アルゴリズム (擬似コード)

**Algorithm** Large Neighborhood Prioritized Search

1: input: a feasible solution $x$
2: $x^* := x$;
3: **repeat**
4: $\quad x^t := \mathit{re\text{-}search}(\mathit{destroy}(x))$;
5: $\quad$ **if** $\mathit{accept}(x^t, x)$ **then**
6: $\quad\quad x := x^t$;
7: $\quad$ **end if**
8: $\quad$ **if** $c(x^t) < c(x^*)$ **then**
9: $\quad\quad x^* := x^t$;
10: $\quad$ **end if**
11: **until** stop criterion is met
12: **return** $x^*$

# 組合せ最適化ソルバー asprior の実装

ASP ソルバー *clingo* の python インターフェースを用いて LNPS を実装




- **探索ヒューリスティクス (#heuristic 文)**：LNPS の値割当てをなるべく維持したままの再探索 (re-search) を，*clingo* の系統的探索で自然に実装している．
- **マルチショット ASP 解法**：LNPS の反復処理を，ASP システムを複数回起動することなく，1 回だけ起動し動的にアトムを追加・削除しながら繰返し解くことで実装している．

# 実験概要

提案する LNPS アルゴリズムの有効性を評価するために実験を行った.

- **CB-CTT ベンチマーク問題**
  - 国際時間割競技会 ITC-2007 の問題 (全 21 問)
  - ソフト制約が最も多い問題集 (UD5) を使用
- **比較した手法**
  - 既存手法: ASP ソルバー *clingo*
  - 提案手法: LNPS を *clingo* 上に実装
- **LNPS のパラメータ**
  - 初期解探索の打切り: $\#\text{conflict} \geq 2{,}500{,}000 \vee \#\text{restart} \geq 5{,}000$
  - re-search の打切り: $\#$ conflict $\geq 30{,}000$
- **CB-CTT の ASP 符号化**: *teaspoon* 符号化 [Banbara+'19]
- **ASP システム**: *clingo-5.4.0*
- **制限時間**: 1 時間/問
- **実験環境**: Mac OS (CPU : Intel Core i7 3.2GHz, メモリー : 64GB)

# CB-CTT に対する destroy 演算子

## 既存研究 [Kiefer ほか'16] を参考に実装

1. **Random $N$ (R-$N$)**
2. **Day-Period (DP)**
3. **Day-Room (DR)**

## 新しく考案した destroy 演算子

4. **Swap-Room $N$ (SR-$N$)**
   - Random $N$ と同様に，暫定解から変数の値割当ての $N$ %をランダムに選んで取り消す．ただし，科目に対する曜日と時限の割当てはできるだけ維持する．
5. **DP-Swap-Room $N$ (DPSR-$N$)**
   - Day-Period と同様に，曜日 D と時限 P をランダムに $N$ 組選び，暫定解から D 曜 P 限に関する変数の値割当てをすべて取り消す．ただし，科目に対する曜日と時限の割当てはできるだけ維持する．

# 実験結果: 得られた最適値と最良値

| 問題名 | 既存手法 ASP | 提案手法 LNPS | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | R-0,3,5 | DP | DR | SR-5 | SR-10 | DPSR-1 | DPSR-2 |
| comp01 | 129 | **11** | 13 | **11** | **11** | **11** | **11** | **11** |
| comp02 | 331 | 239 | **172** | 242 | 201 | 245 | 292 | 227 |
| comp03 | 302 | 154 | 149 | 173 | **146** | 157 | 149 | 154 |
| comp04 | ***49** | ***49** | ***49** | ***49** | ***49** | ***49** | ***49** | ***49** |
| comp05 | 1940 | **797** | 926 | 1116 | 891 | 861 | 1038 | 957 |
| comp06 | 822 | 135 | 140 | 204 | 118 | **106** | 150 | 160 |
| comp07 | 924 | 131 | 118 | 129 | **72** | 77 | 76 | 74 |
| comp08 | ***55** | ***55** | ***55** | ***55** | ***55** | ***55** | ***55** | ***55** |
| comp09 | 254 | 154 | 149 | 146 | 145 | 151 | 149 | **139** |
| comp10 | 822 | 177 | 167 | 133 | **97** | 109 | 108 | 102 |
| comp11 | ***0** | ***0** | ***0** | ***0** | ***0** | ***0** | ***0** | ***0** |
| comp12 | 1246 | 728 | 705 | **694** | 756 | 809 | 769 | 730 |
| comp13 | 301 | 163 | 168 | 165 | 151 | 155 | 155 | **149** |
| comp14 | ***67** | 145 | 179 | 165 | 104 | 83 | 128 | 139 |
| comp15 | 607 | 213 | 234 | 238 | 215 | 224 | **211** | 236 |
| comp16 | 944 | 156 | 180 | 162 | 223 | 158 | **146** | 165 |
| comp17 | 412 | 226 | 230 | 234 | **199** | 208 | 202 | 235 |
| comp18 | 471 | 168 | 152 | 158 | 144 | 149 | 153 | **136** |
| comp19 | 890 | 192 | 187 | 219 | 176 | **163** | 198 | 174 |
| comp20 | 1386 | 274 | 280 | 305 | 293 | **265** | 284 | 312 |
| comp21 | 310 | 202 | 222 | 235 | 192 | **178** | 181 | 193 |
| #最適値・最良値 | 4 | 5 | 4 | 5 | **8** | **8** | 6 | 7 |

- 提案手法は，既存の ASP より，多くの問題でより良い解を生成した.
- 提案手法の中では，SR ，次いで DPSR が良い性能を示した.

# 他のアプローチとの比較

| 問題名 | 既知の最良値 (♯) | 既存手法 ASP | | 提案手法 LNPS | |
|---|---|---|---|---|---|
| | | 最良値 | ♯ との比 (%) | 最良値 | ♯ との比 (%) |
| comp01 | 11 | 129 | +1,072 | 11 | 0 |
| comp02 | 130 | 331 | +154 | 172 | +32 |
| comp03 | 142 | 302 | +112 | 146 | +2 |
| comp04 | 49 | 49 | 0 | 49 | 0 |
| comp05 | 570 | 1,940 | +240 | 797 | +39 |
| comp06 | 85 | 822 | +867 | 106 | +24 |
| comp07 | 42 | 924 | +2,100 | 72 | +71 |
| comp08 | 55 | 55 | 0 | 55 | 0 |
| comp09 | 150 | 254 | +69 | 139 | **-7** |
| comp10 | 72 | 822 | +1,041 | 97 | +34 |
| comp11 | 0 | 0 | 0 | 0 | 0 |
| comp12 | 483 | 1,246 | +157 | 694 | +43 |
| comp13 | 147 | 301 | +104 | 149 | +1 |
| comp14 | 67 | 67 | 0 | 83 | +23 |
| comp15 | 176 | 607 | +244 | 211 | +19 |
| comp16 | 96 | 944 | +883 | 146 | +34 |
| comp17 | 155 | 412 | +165 | 199 | +28 |
| comp18 | 137 | 471 | +243 | 136 | **-1** |
| comp19 | 125 | 890 | +612 | 163 | +30 |
| comp20 | 124 | 1,386 | +1,017 | 265 | +113 |
| comp21 | 151 | 310 | +105 | 178 | +17 |
| ♯ との比の平均 | | | +437 | | +23 |

- 既知の最良値との比を，+437% から +23% に**大幅に改善**
- comp09,18 について，**既知の最良値を更新**することに成功

# 他の問題集での比較

- CB-CTT ベンチマーク問題
  - **ITC-2007 以後に追加された問題 (全 40 問)**
  - ソフト制約が最も多い問題集 (UD5) を使用

**既知の最良値との比の平均 (%)**

| 既存手法 ASP [Banbara+ '19] | 提案手法 LNPS |
|---|---|
| +121 | +80 |

- 提案手法は，既存手法と比較して既知の最良値との比を改善
- EA11,test4 について，**既知の最良値を更新**することに成功

# まとめと今後の課題

1. **組合せ最適化ソルバー asprior の実装**
   - ASP ソルバー *clingo* のマルチショット ASP 解法と#heuristic 文を用いて LNPS を簡潔に実装
2. **カリキュラムベース・コース時間割問題を用いた評価実験**
   - ソフト制約が多い問題集 (UD5) に対して，既知の最良値との比を，+437% から +23% に**大幅に改善**
   - 61 問中 4 問について，**既知の最良値を更新**することに成功

## 今後の課題

- **他の組合せ最適化問題への適用**
  - **巡回セールスマン問題**, グラフ彩色問題
- 他の時間割問題への適用
- アダプティブ LNPS への拡張

# LNS (Large Neighborhood Search)

## LNS のアルゴリズム [Pisinger 2010]

1. 初期解を $x$ と置き，最良解 $x^* := x$ とする．
2. 以下の destroy と repair で $x$ から得られた解を $x^t$ と置く．
   - destroy は $x$ から値割当ての一部を取り消し $x'$ とする．
   - repair は $x'$ の値割当てを変化させずに解を再構築する．
3. 更新条件を満たしていたら $x := x^t$ とする．
   - 例えば「$x^t$ が $x$ より改善された解なら」という更新条件を用いる．
4. $x^t$ が最良解 $x^*$ より改善された解なら，$x^* := x^t$ とする．
5. 終了条件が満たされるまで，2〜4 を繰り返す．
   - 例えば繰り返し回数や制限時間などを終了条件に用いる．
6. 最良解 $x^*$ を返す．

- 再構築した解 $x_t$ では，取り消された変数に対してのみ再割当てが行われ，$x'$ の値割当ては変化しない．
- VRP (Vehicle Routing Problem) など，比較的独立した複数の部分問題に分割できる場合に良い性能を示すことが報告されている．

# 時間割問題

求解困難な組合せ最適化問題の一種である.

- 時間割に関する国際会議 PATAT および**国際的な時間割競技会**が開催され，時間割ソルバーの性能向上に貢献している.
  - 教育時間割
    - **カリキュラムベース・コース時間割 (CB-CTT)**
    - ポストエンロールメント・コース時間割
    - 試験時間割
  - 輸送時間割
  - 従業員時間割
  - スポーツ時間割
- 本発表では，最も研究が盛んな CB-CTT を対象とする.
- CB-CTT は，以下のように定式化される.
  - 必ず満たすべき**ハード制約**と, できるだけ満たしたい**重み付きソフト制約**から構成される.
  - 違反するソフト制約の重み (ペナルティ) の総和の最小化が目的.

# 制約と問題集 (カリキュラムベース・コース時間割)

| 制約 | 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 | - |

# CB-CTT に対する既存 ASP の結果 [Banbara+ '19]

| 問題 | UD1 | | UD2 | | UD3 | | UD4 | | UD5 | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 既知のベスト | ASP | 既知のベスト | ASP | 既知のベスト | ASP | 既知のベスト | ASP | 既知のベスト | ASP |
| comp01 | 4 = | 4 | 5 = | 5 | 8 = | 8 | 6 | 9 | 11 | 19 |
| comp02 | 12 $=^*$ | 12 | 24 $=^*$ | 24 | 12 $=^*$ | 12 | 26 | 55 | 130 | 231 |
| comp03 | 38 | 53 | 64 | 109 | 25 | 47 | 362 | 405 | 142 | 204 |
| comp04 | 18 = | 18 | 35 = | 35 | 2 $=^*$ | 2 | 13 $=^*$ | 13 | 59 $>^*$ | 49 |
| comp05 | 219 | 504 | 284 | 624 | 264 | 556 | 260 | 459 | 570 | 1081 |
| comp06 | 14 $=^*$ | 14 | 27 = | 27 | 8 $=^*$ | 8 | 15 $>^*$ | 9 | 85 | 88 |
| comp07 | 3 = | 3 | 6 = | 6 | 0 = | 0 | 3 $=^*$ | 3 | 42 | 256 |
| comp08 | 19 = | 19 | 37 = | 37 | 2 $=^*$ | 2 | 15 $=^*$ | 15 | 62 $>^*$ | 55 |
| comp09 | 54 | 63 | 96 | 169 | 8 $=^*$ | 8 | 38 | 50 | 150 | 196 |
| comp10 | 2 = | 2 | 4 = | 4 | 0 = | 0 | 3 $=^*$ | 3 | 72 | 73 |
| comp11 | 0 = | 0 | 0 = | 0 | 0 = | 0 | 0 = | 0 | 0 = | 0 |
| comp12 | 239 | 343 | 294 | 456 | 51 | 114 | 99 | 388 | 483 | 1135 |
| comp13 | 32 $>^*$ | 31 | 59 = | 59 | 22 | 50 | 41 | 111 | 148 > | 147 |
| comp14 | 27 $=^*$ | 27 | 51 = | 51 | 0 = | 0 | 16 $>^*$ | 14 | 95 > * | 67 |
| comp15 | 38 | 53 | 62 | 109 | 16 | 22 | 30 | 68 | 176 | 254 |
| comp16 | 11 $=^*$ | 11 | 18 = | 18 | 4 $=^*$ | 4 | 7 $=^*$ | 7 | 96 | 438 |
| comp17 | 30 $=^*$ | 30 | 56 = | 56 | 12 $=^*$ | 12 | 26 $>^*$ | 21 | 155 | 352 |
| comp18 | 34 | 48 | 61 | 81 | 0 = | 0 | 27 | 46 | 137 | 228 |
| comp19 | 32 $>^*$ | 29 | 57 = | 57 | 24 | 32 | 32 | 82 | 125 | 283 |
| comp20 | 2 = | 2 | 4 = | 4 | 0 = | 0 | 9 $>^*$ | 3 | 124 | 704 |
| comp21 | 43 | 94 | 74 | 124 | 6 = | 6 | 36 | 76 | 151 | 166 |

# #heuristic 文 (1/3)

## コード例

```
{a ; b}.
:- a, b.
#heuristic a. [1, true]
#heuristic b. [2, true]
```

- 1 行目のルールは，選択子を使ってアトム $a$ と $b$ を導入している．
- 2 行目のルールは，$a$ と $b$ が同時に成り立たないことを表している．
  - このプログラム例の解集合は $\{\},\{a\},\{b\}$ の 3 つである．
- 3 行目の #heuristic 文は，優先度 1 で $a$ に真を割当てることを表している．
- 同様に，4 行目は優先度 2 で $b$ に真を割当てることを表している．

# #heuristic 文 (2/3)

## #heuristic 文を無効にした実行例

```
% clingo heu.lp -n 0
clingo version 5.4.0
Reading from heu.lp
Solving...
Answer: 1

Answer: 2
b
Answer: 3
a
SATISFIABLE
```

# #heuristic 文 (3/3)

## #heuristic 文を有効にした実行例

```
% clingo heu.lp -n 0 --heu=domain
clingo version 5.4.0
Reading from heu.lp
Solving...
Answer: 1
b
Answer: 2
a
Answer: 3

SATISFIABLE
```

- 優先度の高いアトムから順に，値割当が行われていることがわかる．

# #heuristic 文を用いた LNPS の re-search の実装

```
#heuristic p(t_1,...,t_n) : heuristic(p(t_1,...,t_n),W,T). [W,true]
```

- アトム $p(t_1, \ldots, t_n)$ は暫定解を表す.
- 補助アトム heuristic($p(t_1, \ldots, t_n)$,W,T) は，各反復 T において，$p(t_1, \ldots, t_n)$ に優先度 W で真を割当ることを意味する.
- インクリメンタル ASP 解法を用いて，補助アトム heuristic($p(t_1, \ldots, t_n)$,W,T) を，動的に追加・削除することで，$p(t_1, \ldots, t_n)$ への値割り当てをなるべく維持したままで解を再探索を実現している.

# 教室に関するソフト制約

- **day-period**
  - ランダムに曜日 ($D$) と時限 ($P$) を選び，暫定解から $D$ 曜 $P$ 限の値割当てをすべて取り消す.
  - 教室に関するソフト制約のペナルティを減らすことが狙い.

1. RoomCapacity
   - 各科目について, 受講者数が使用する教室の収容可能人数を超えてはいけない. 違反した場合, 超過人数に応じたペナルティが課される.
2. RoomStability
   - 同一科目のすべての講義は, 同一教室で開講される. 違反した場合, 異なる教室数 (最初の教室は除く) に応じたペナルティが課される.
3. TravelDistance
   - 各課程について, 同一曜日に異なる建物の教室で開講される連続した講義があると違反となり, 建物間の移動毎にペナルティが課される.
4. RoomSuitability
   - 各科目の講義は, 開講不可能な教室で開講されることはない. 違反した講義毎にペナルティが課される.

# 時限に関するソフト制約

- **day-room**
  - ランダムに曜日 ($D$) と教室 ($R$) を選び，暫定解から $D$ 曜日の $R$ 教室の値割当てをすべて取り消す.
  - 時限に関するソフト制約のペナルティを減らすことが狙い.

1. IsolatedLectures
   - 同一課程に属する講義は, 連続した時限に開講される. 同一曜日に同一課程に属する他のどの講義とも隣接していない (孤立した) 講義がある場合に違反となり, 孤立した講義毎にペナルティが課される.
2. Windows
   - 同一課程に属する講義は, 空き時限なしで開講される. 同一曜日に同一課程に属する 2 つの講義の間に空き時限がある場合に違反となり, 空き時限の長さに応じたペナルティが課される.
3. DoubleLectures
   - 連続講義の形態をとる科目は, 同一曜日に複数の講義がある場合, それらは連続した時限に同一教室で開講される. 違反した講義毎にペナルティが課される.

# day-period による destroy

- **day-period**
  - ランダムに曜日 ($D$) と時限 ($P$) を選び，暫定解から $D$ 曜 $P$ 限の値割当てをすべて取り消す.
  - 教室に関するソフト制約のペナルティを減らすことが狙い.

| | 1 時限目 | … |
|---|---|---|
| ⋮ | | |
| 水曜日 | **教室 A：アルゴリズム**<br>**教室 B：コンパイラ**<br>**教室 C：プログラミング** | |
| ⋮ | | |

収容人数：
教室 A　30 人
教室 B　50 人
教室 C　70 人

受講者数：
アルゴリズム 60 人
コンパイラ 40 人
プログラミング 30 人

- destroy で曜日として水曜日, 時限として 1 限を選ぶとする.
- 同一曜日, 同一時限の講義間での教室の交換を促すことで, 教室に関するソフト制約違反の改善を狙う.
  - アルゴリズムとプログラミングの教室を交換することで, 収容人数オーバーを改善することができる.

# dp-swap-room $N$ による destroy

- **dp-swap-room $N$**
  - 曜日 D と時限 P をランダムに $N$ 組選び，暫定解から D 曜 P 限に関する変数の値割当てをすべて取り消す．ただし，科目に対する曜日と時限の割当てはできるだけ維持する．

## アトム

- **assigned($C, R, D, P$).**
  - 講義 $C$ が曜日 $D$ の時限 $P$ に教室 $R$ で開講されることを表す．
- **assigned($C, D, P$).**
  - 講義 $C$ が曜日 $D$ の時限 $P$ に開講されることを表す．

- assigned($C, D, P$) を 0%取り消し (100%再利用)，
  assigned($C, R, D, P$) を一部取り消す．
  - assigned($C, R, D, P$) が取り消されても，assigned($C, D, P$) が再利用によって優先的に割り当てられるため，該当する講義を同じ曜日・時限に優先的に割り当てながら，教室についてのみの変更を促す．

# day-room による destroy

- **day-room**
  - ランダムに曜日 ($D$) と教室 ($R$) を選び，暫定解から $D$ 曜日の $R$ 教室の値割当てをすべて取り消す.
  - 時限に関するソフト制約のペナルティを減らすことが狙い.

| | 1 時限目 | 2 時限目 | 3 時限目 |
|---|---|---|---|
| ⋮ | | | |
| 水曜日 | **教室 A：アルゴリズム**<br>教室 B：コンパイラ<br>教室 C：プログラミング | 教室 A：<br>教室 B：数値解析<br>教室 C：データベース | **教室 A：オートマトン**<br>教室 B：パターン認識<br>教室 C： |
| ⋮ | | | |

- destroy で曜日として水曜日, 教室として教室 A を選ぶとする.
- 同一曜日, 同一教室の講義間での時限の交換を促すことで, 時限に関するソフト制約違反の改善を狙う.
  - アルゴリズムとオートマトンが同一課程に属する場合, オートマトンを 2 限に移すことで, 特定のソフト制約違反を改善することができる.

# swap-room $N$ による destroy

- **swap-room $N$**
  - 暫定解からランダムに $N$%の値割り当てを選び，曜日と時限はそのままで，割り当てられている教室の情報を取り消す.

## アトム

- **assigned($C, R, D, P$).**
  - 講義 $C$ が曜日 $D$ の時限 $P$ に教室 $R$ で開講されることを表す.
- **assigned($C, D, P$).**
  - 講義 $C$ が曜日 $D$ の時限 $P$ に開講されることを表す.

- assigned($C, D, P$) を 0%取り消し (100%再利用),
  assigned($C, R, D, P$) を $N$%取り消す.
  - assigned($C, R, D, P$) が取り消されても，assigned($C, D, P$) が再利用によって優先的に割り当てられるため，該当する講義を同じ曜日・時限に優先的に割り当てながら，教室についてのみの変更を促す.

# 追加実験

- CB-CTT ベンチマーク問題
  - 国際時間割競技会 ITC-2007 の問題 (全 21 問)
  - **ソフト制約: UD1, UD2, UD3, UD4**

## 既知の最良値との比の平均 (%)

| | 既存手法 ASP[Banbara+'19] | 提案手法 LNPS |
|---|---|---|
| UD1 | +20 | +18 |
| UD2 | +24 | +11 |
| UD3 | +25 | +13 |
| UD4 | +58 | +19 |

- 提案手法は，UD5 よりもソフト制約が少ない UD1〜UD4 でも，既存手法と比較して既知の最良値との比を改善