解集合プログラミングを用いた配電網問題の解法に関する研究

山田 健太郎

番原研究室

2021年度 番原研中間発表会
2021 年 12 月 3 日

配電網問題

求解困難な組合せ最適化問題の一種

- **配電網**とは，変電所と，一般家庭や工場を繋ぐ電力供給経路のネットワークである．
- 配電網の構成技術はスマートグリッドや，災害時の障害箇所の迂回構成などを支える重要な基盤技術として期待されている．
- **配電網問題**とは，
  - **トポロジ制約**と**電気制約**を満たしつつ，
  - 損失電力を最小にするスイッチの開閉状態を求めることが目的．
- これまで，メタヒューリスティクス等の解法が提案されている．
- 厳密解法として，フロンティア法を用いた解法が提案されている．
  - 実用規模の配電網問題 (**スイッチ数 468 個**) の最適解を求めることに成功 [井上ほか '12]．

トポロジ制約

トポロジ制約のみの配電網問題は，グラフと根と呼ばれる特別なノードから，**根付き全域森**を求める部分グラフ探索問題に帰着できる．

根付き全域森 (Spanning Rooted Forest) [川原・湊 '12]

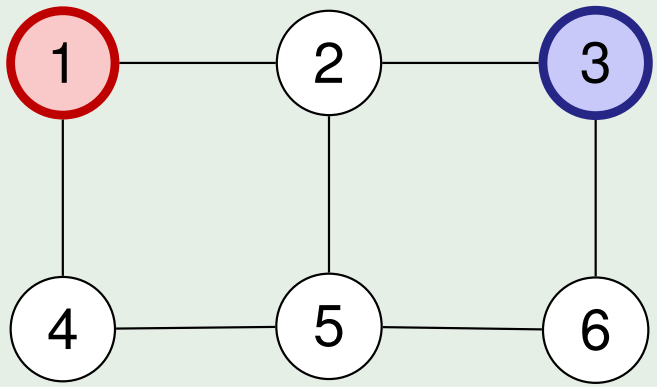
グラフ $G = (V, E)$ と，**根**と呼ばれる V 上のノードが与えられたとき，G 上の根付き全域森とは，以下の条件を満たす G の部分グラフ $G' = (V, E')$, $E' \subseteq E$ である．

1. G' はサイクルを持たない． (**非閉路制約**)
2. G' の各連結成分は，ちょうど 1 つの根を含む． (**根付き連結制約**)

- この部分グラフ探索問題を**根付き全域森問題**と呼ぶ．

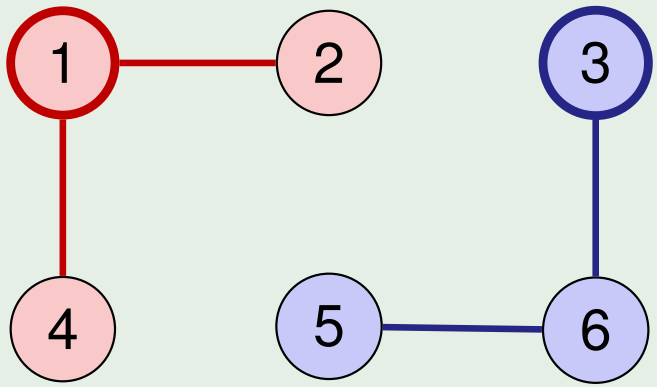
根付き全域森問題の例

入力例



⇒

解の例



- **配電網とグラフの対応**

配電網	配電区間	スイッチ	変電所
グラフ	ノード	辺	根

- **配電網問題のトポロジ制約**
  - 停電 (変電所と結ばれない区間)
  - 短絡 (供給経路上のループ，複数の変電所と結ばれる区間)

電気制約

- **電気制約**は，配電する**電流・電圧**の適正範囲を保証する制約．
- 電流が許容範囲を超えてしまうと，配電線が焼き切れてしまい，停電や事故などの危険につながってしまう．
- 電圧が許容範囲を下回ってしまうと，配電先で機械や電化製品が適切に動作できないなどの問題が発生してしまう．
- 配電システムによっては，その他にも様々な制約がある．

ASP を用いて実装する上での課題

- 電流や電圧が影響し合う実数ドメイン上の制約である．
- 虚数を含むインピーダンスの計算など，純粋な ASP のみで扱うには難しいと考えられる．

研究目的

目的

- ASP 技術を活用して，大規模な配電網問題，及びその解の遷移問題を効率良く解くシステムを構築する．
- 実用規模のベンチマーク問題を用いてシステムを評価し，ASP の特長を活かした配電網の構成技術の利点・実用性を明らかにする．

研究内容

1. **根付き全域森問題を解く ASP 符号化**
   - これまでに ASP 符号化 3 種を考案．(ASP のルール 6 行程度)
   - 実用規模の問題及び，より大規模な問題を用いての性能評価．
2. **配電網問題を解く ASP 符号化**
   - 電流に関する電気制約の ASP 符号化を考案．
3. **配電網問題の解の遷移問題への拡張**
   - ベンチマーク問題のための解空間グラフの作成．

提案する ASP 符号化

符号化	根付き連結制約	非閉路制約
基本 符号化	at-least-one 制約と at-most-one 制約	各連結成分の 辺数の制約
改良 符号化 1	**ASP の個数制約**	各連結成分の 辺数の制約
改良 符号化 2	**ASP の個数制約**	**ノードの入次数の制約**

- 根付き連結制約に ASP の個数制約を用いることで，基礎化後のルール数が少なくなるため，大規模な問題に対して有効である．
- 改良符号化 2 は，与えられる無向グラフを双方向グラフに拡張することで，各根，各ノードにおける入次数の制約を導入した．

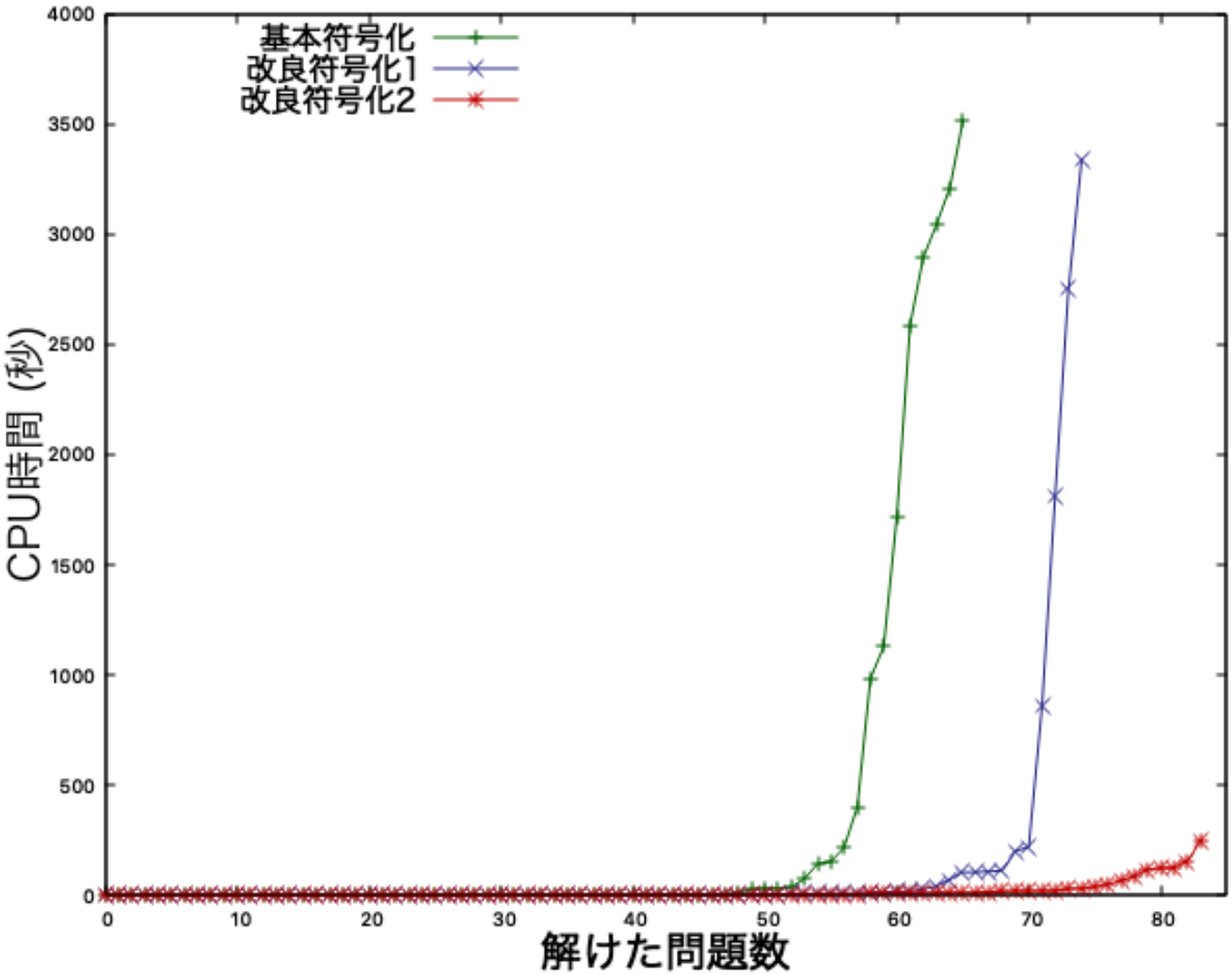
実験概要

考案した符号化の性能を評価するために，以下の実験を行った．

- **比較する ASP 符号化:**
  - 基本符号化
  - 改良符号化 1
  - 改良符号化 2
- **ベンチマーク問題:** 全 85 問
  - DNET[†] で公開されている配電網問題 3 問
    (トポロジ制約のみ，スイッチ数: 16 個，36 個，**468 個**)
  - *Graph Coloring and its Generalizations* [‡] で公開されている
    グラフ彩色問題をベースに，独自に生成した 82 問 [§]
    ($20 \leq$ 辺数 $\leq 49{,}629$)
- **ASP システム:** *clingo-5.4.0* + *trendy*
- **制限時間:** 3600 秒/問
- **実験環境:** Mac mini，3.2GHz Intel Core i7，64GB メモリ

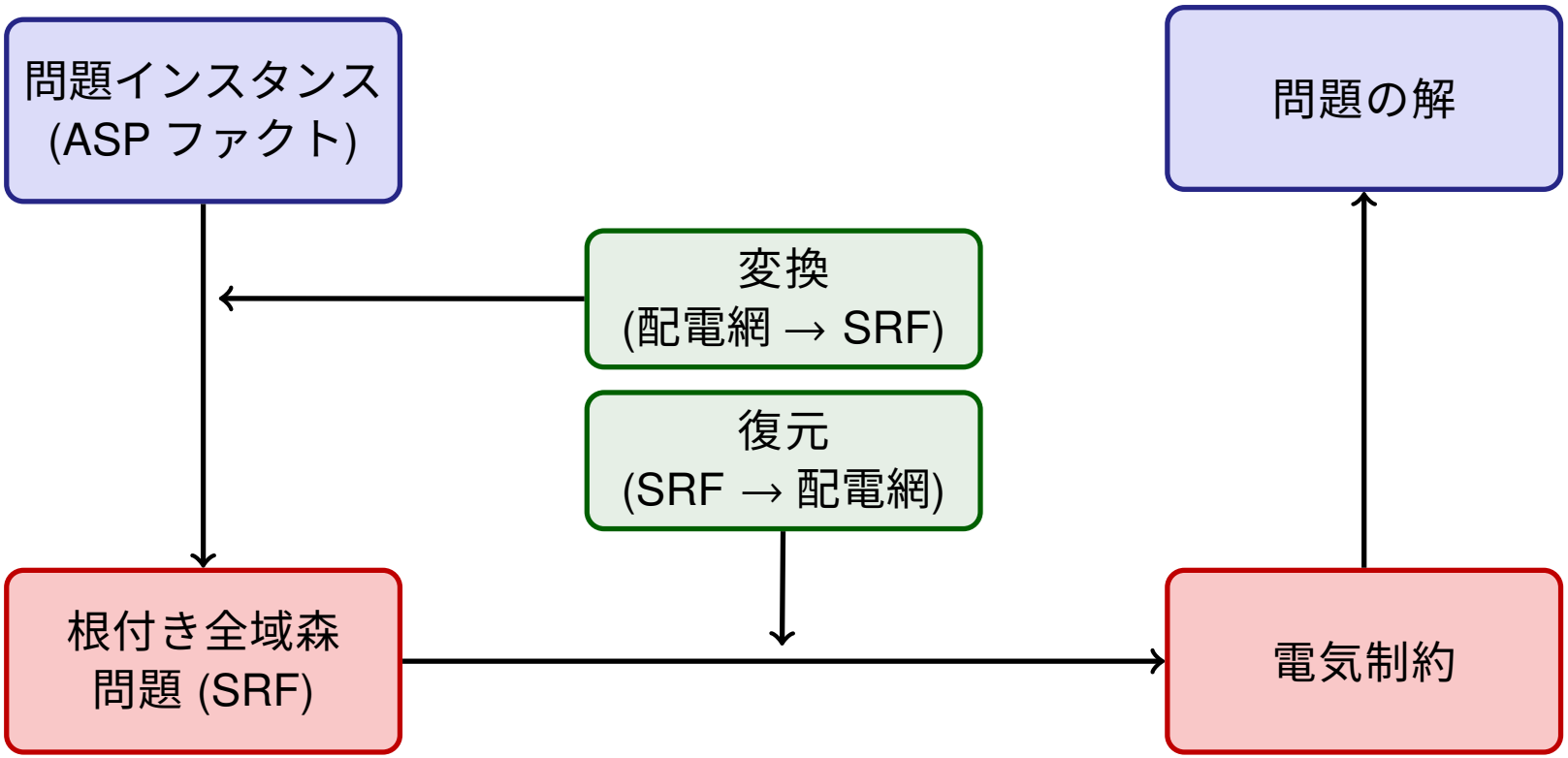
[†] https://github.com/takemaru/dnet
[‡] https://mat.tepper.cmu.edu/COLOR04/
[§] 各問題に対し，全ノードのうち 1/5 個をランダムに根として与えた．

実験結果：カクタスプロット



- 改良符号化２は，他の符号化と比較して，より多くの問題を高速に解いている．

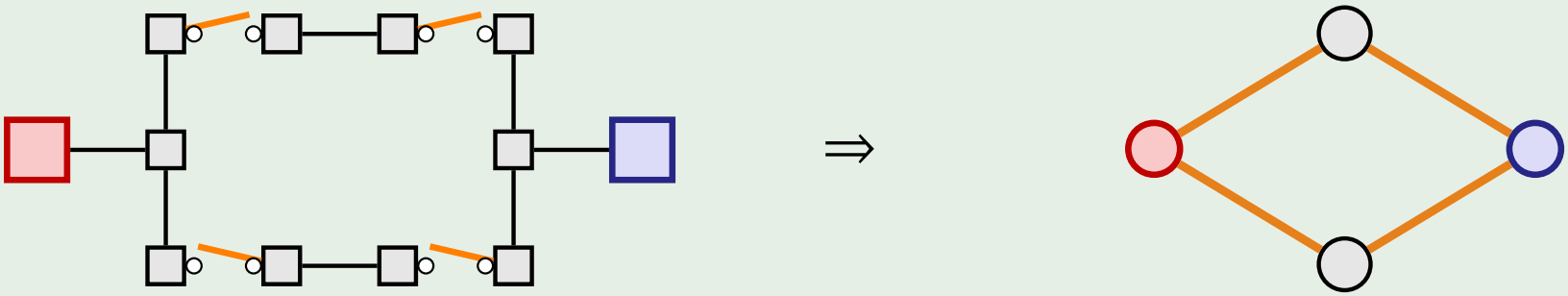
ASP を用いた配電網問題の解法



- 与えられた配電網問題を一度，根付き全域森問題に変換して解く．
- 得られた解から配電網を復元 (構成) し，電気制約を適用して問題の解を得る．
- 新たに，**変換**，**復元**，**電流に関する制約**の ASP 符号化を考案した．

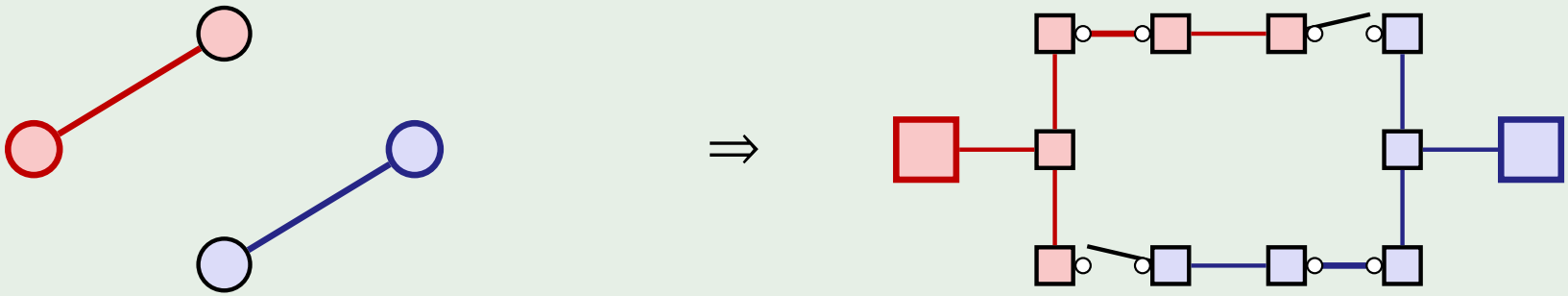
変換・復元

変換 (配電網 → 根付き全域森問題)



- スイッチで区切られる区間ごとにノードとして扱う．
- ASP のルール 5 行程度で簡潔に表現可能．

復元 (根付き全域森問題 → 配電網)



- 根付き全域森に含まれる辺が，閉じたスイッチに対応する．
- ASP のルール 2 行で簡潔に表現可能．

電気制約の ASP 符号化

- 電気制約のうち，**電流制約**の ASP 符号化を考案した．

電流制約

各変電所 R に対して，以下の条件を満たす．

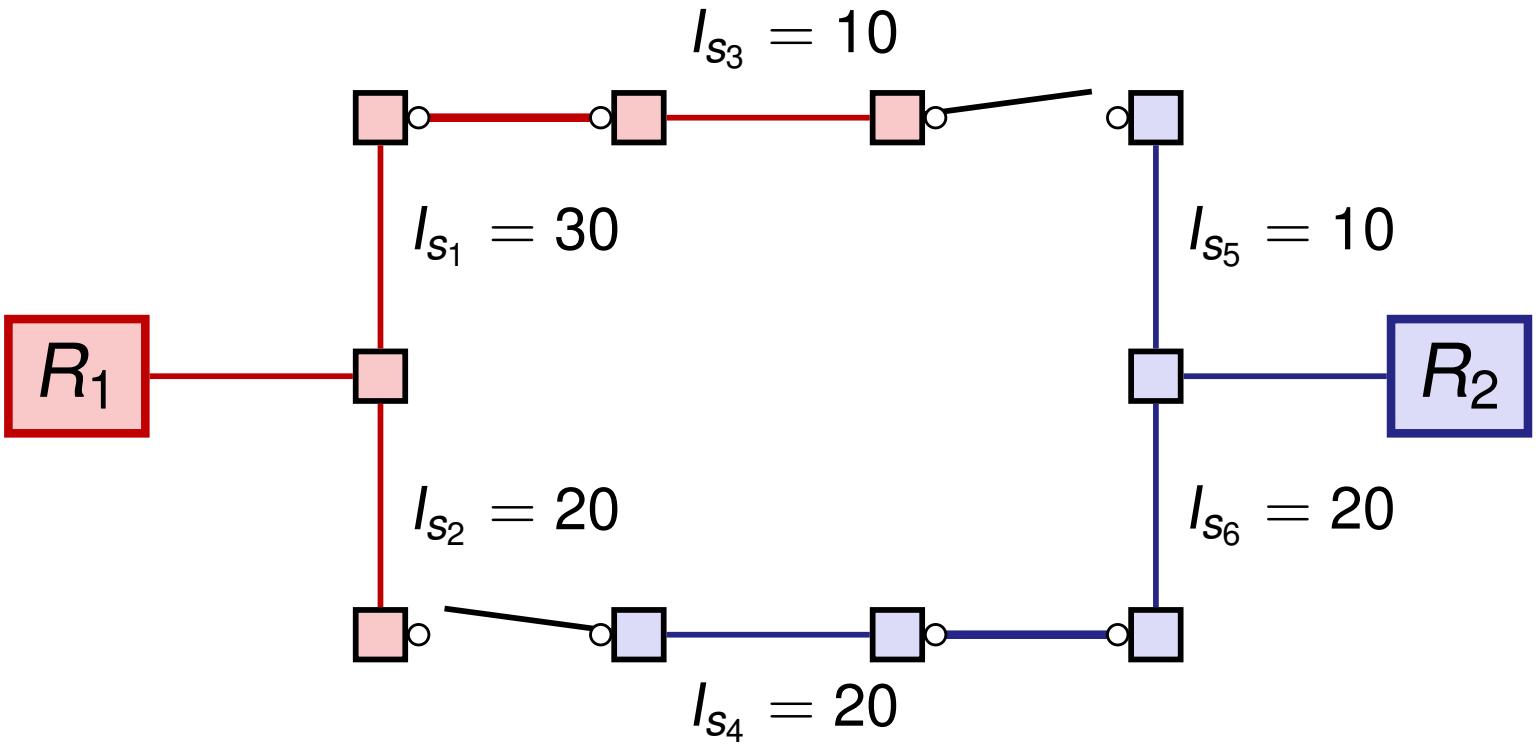
$$\sum_{s \in R_{sup}} I_s \leq I_{max}$$

R_{sup}	変電所 R から配電を受ける区間の集合
I_s	区間 s における電流値
I_{max}	電流の上限値

- ASP のルール 7 行程度で簡潔に表現可能．

電流制約の例

- $I_{max} = 60\ _{(A)}$ とする



- $R1_{sup} = \{s1, s2, s3\}.$ $\quad I_{s_1} + I_{s_2} + I_{s_3} = 60\ (= I_{max})$
- $R2_{sup} = \{s4, s5, s6\}.$ $\quad I_{s_4} + I_{s_5} + I_{s_6} = 50\ (< I_{max})$

実験

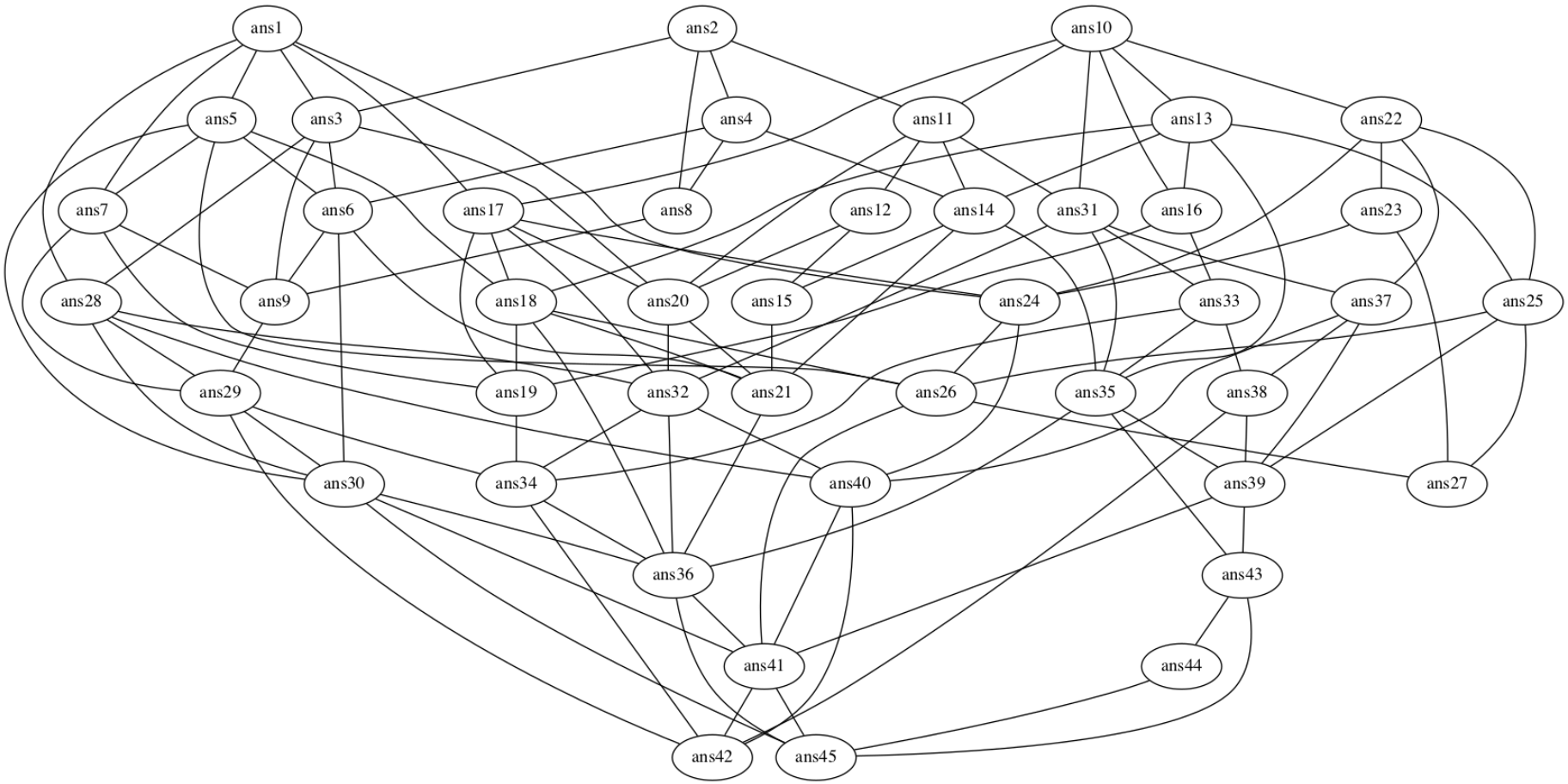
- 解の遷移問題への拡張のために行った予備実験の結果を示す．
- 根付き全域森問題の符号化には，改良符号化 2 を用いた．

問題名	変電所	区間	スイッチ	I_{max}	CPU 時間	総数
test	3	15	16	200	0.008s	UNSAT
test	3	15	16	250	0.026s	45
test	3	15	16	300	0.060s	114
test	3	15	16	400	0.009s	279
baran32	1	32	36	300	0.013s	UNSAT
baran32	1	32	36	350	0.116s	51127
fukui-tepco	72	432	468	300	TO†	358 億+

†制限時間：432000s = 5 日間

実験結果

- **test** ($l_{max} = 250$) の解空間グラフ † (解の総数：45 個)



†あるスイッチ 2 個を切り替えて遷移可能な解を隣接関係にした．

まとめ

1. **根付き全域森問題を解く ASP 符号化**
   - 非閉路制約にノードの入次数の制約を用いることで，リスタート数を大きく削減することができ，大規模な問題への有効性を示した．
2. **配電網問題を解く ASP 符号化**
   - 配電網問題と根付き全域森問題を対応付ける ASP 符号化を考案し，ASP のルール 7 行程度で簡潔に記述できることを確認した．
   - 電流制約の ASP 符号化を考案し，ASP のルール 7 行程度で簡潔に記述できることを確認した．
3. **配電網問題の解の遷移問題への拡張**
   - 各ベンチマークについて解を列挙し，解空間グラフを作成した．

今後の課題

- 電流制約を含む ASP 符号化の解の遷移問題への拡張及び評価．
- 解空間グラフをもとに遷移問題ベンチマーク問題の作成．

- 補足用 -

根付き連結制約

基本符号化

```
:- node(X),  not reached(X,_).
:- reached(X,R1), reached(X,R2), R1 < R2.
```

改良符号化 1・2

```
:- node(X), not 1 { reached(X,R) } 1.
```

- 各ノード X について，ちょうど 1 つの根からのみ到達可能であることを意味する．

基礎化後のルール数

- グラフのノード数を $|V|$，根ノードの数を $|R|$ とする．

符号化	根付き連結制約の表現方法	基礎化後のルール数
基本符号化	at-least-one 制約と at-most-one 制約を用いて表現	$\|V\|\left(1+\binom{\|R\|}{2}\right)$
改良符号化 1·2	ASP の個数制約を用いて表現	$\|V\|$

非閉路制約

基本符号化・改良符号化 1 (辺数の制約)

```
:- root(R),
   not 1 #sum{ 1,X:reached(X,R) ;
              -1,X,Y:inForest(X,Y),reached(X,R),reached(Y,R)
             } 1.
```

改良符号化 2 (ノードの入次数の制約)

```
:- root(R), inForest(_,R).
:- node(X), not root(X), not 1 { inForest(_,X) } 1.
```

- 各連結成分が**木の性質**を満たすことにより，サイクルを持たないことを保証する．

補足：根付き全域森 改良符号化２

```
 1 %% spanning rooted forest
 2
 3 %% choose edge
 4 { inForest(X,Y); inForest(Y,X) } 1 :- edge(X,Y).
 5
 6 %% tree constraint
 7 :- root(R), inForest(_,R).
 8 :- node(X), not root(X), not 1 { inForest(_,X) } 1.
 9
10 %% connectivity constraint
11 :- node(X), not 1 { reached(X,R) } 1.
12
13 %% generate reached
14 reached(R,R) :- root(R).
15 reached(X,R) :- reached(Y,R), inForest(Y,X).
16 % reached(X,R) :- reached(Y,R), inForest(X,Y).
```

補足：変換の ASP 符号化

```
 1 %%% Pre-processing 2
 2
 3 swt_node(X) :- dnet_node(X,S), switch(S).
 4 jct_node(X) :- dnet_node(X,_), not swt_node(X).
 5
 6 section(S,X) :- dnet_node(X,S), not switch(S).
 7 section(S)   :- section(S,_).
 8
 9 jct_section(S) :- section(S,X), jct_node(X).
10 swt_section(S) :- section(S,X), swt_node(X).
11
12 switch(SW,S,T) :- section(S,X), section(T,Y), S!=T,
13                      dnet_node(X,SW), dnet_node(Y,SW), switch(SW).
14
15 %% spanning rooted tree
16 node(X,  S) :- jct_section(S), dnet_node(X,S), jct_node(X).
17 node(Min,S) :- swt_section(S), not jct_section(S),
18                 Min = #min { X : dnet_node(X,S) }.
19
20 root(X) :- node(X,S), substation(S).
21 node(X) :- node(X,_).
22
23 edge(X,Y) :- node(X,S), node(Y,T), X<Y, switch(SW,S,T).
```

補足：復元・電流制約の ASP 符号化

```
1  %%% electrical constraints
2
3  connected(SW,S,T) :- inForest(X,Y), node(X,S), node(Y,T),
4                       switch(SW,S,T).
5
6  closed_switch(SW) :- connected(SW,_,_).
7
8  entrance_section(S,X) :- substation(S), section(S,X).
9  entrance_section(S,X) :- jct_node(X), section(S,X),
10                          connected(_,_,S).
11
12 downstream(S,T) :- connected(_,S,T).
13 downstream(S,T) :- entrance_section(S,X), section(T,X), S!=T.
14
15 suppliable(R,R) :- substation(R).
16 suppliable(T,R) :- downstream(S,T), suppliable(S,R).
17
18 :- substation(R),
19    not #sum{ I,S:suppliable(S,R), load(S,I) } max_current.
```