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考察

山田 健太郎

番原研究室

卒業研究発表会

2020 **年** 2 **月** 20 **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，家庭や工場などから構成される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から構成．
  - 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．
- フロンティア法を用いた解法が提案されている [井上ほか '12]．
  - ERATO 湊離散構造処理系プロジェクトの成果．
  - 実用規模の配電網問題 (スイッチ数 468 個) の最適解を発見．

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，家庭や工場などから構成される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から構成．
  - 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．
- フロンティア法を用いた解法が提案されている [井上ほか '12]．
  - ERATO 湊離散構造処理系プロジェクトの成果．
  - 実用規模の配電網問題 (スイッチ数 468 個) の最適解を発見．

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．

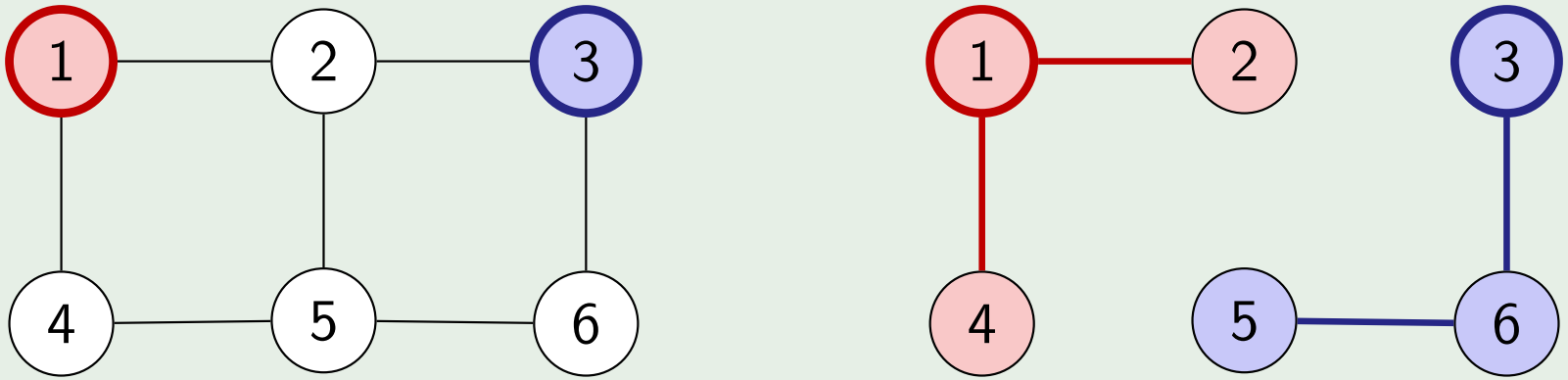
根付き全域森 (Spanning Rooted Forest)

根付き全域森　[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)**

根付き全域森の例



解集合プログラミング (Answer Set Programming; ASP)

- ASP **言語**は一階論理に基づく知識表現言語の一種である．
- ASP **システム**は論理プログラムから安定モデル意味論 [Gelfond and Lifschitz '88] に基づく解集合を計算するシステムである．
- 近年では SAT 技術を応用した高速 ASP システムが実現され，システム検証，プランニング，システム生物学など様々な分野への応用が拡大している．

ASP 技術を組合せ最適化問題に用いる利点

- ASP 言語の高い表現力を活かし，記号的な制約を簡潔に記述可能．
- ASPMT 技術を用いて，様々な背景理論ソルバーと連携可能．
- 充足不能コアを用いた効率的な最適値探索が利用可能．

研究目的

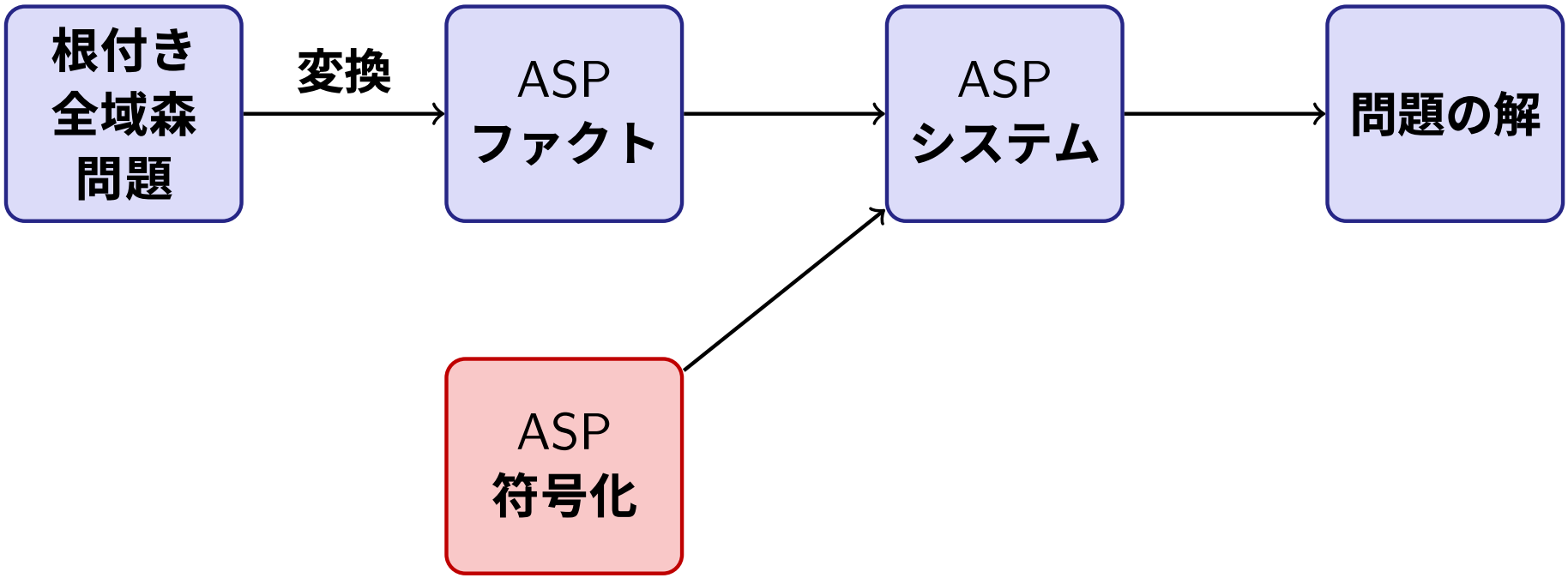
目的

ASP 技術を活用して，大規模な配電網問題を効率良く解くシステムの構築を目指す．

研究内容

1. 根付き全域森問題を解く，2 種類の ASP 符号化を考案した．
2. 実用規模の問題，及び，より大規模な問題を用いた評価実験．
3. 遷移問題へ拡張し，その符号化を考案した．
   - ある配電網の構成 (初期状態) から，目的とする配電網の構成 (目的状態) を得るための，スイッチの切替手順を求める問題．

ASP を用いた根付き全域森問題の解法



1. 与えられた問題インスタンスを ASP のファクト形式に変換する．
2. 変換した問題と，根付き全域森問題を解く ASP 符号化を結合し，ASP システムを用いて解を求める．

提案する ASP 符号化

- 根付き全域森問題に対する，2 種類の ASP 符号化を考案した．
- $|V|$ はグラフのノード数，$|R|$ は根ノード数をそれぞれ表す．

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

- 提案符号化 2 は，提案符号化 1 よりも基礎化後のルール数が少なくなるため，大規模な問題への有効性が期待できる．

実験概要

提案符号化の有効性を評価するために，以下の実験を行った．

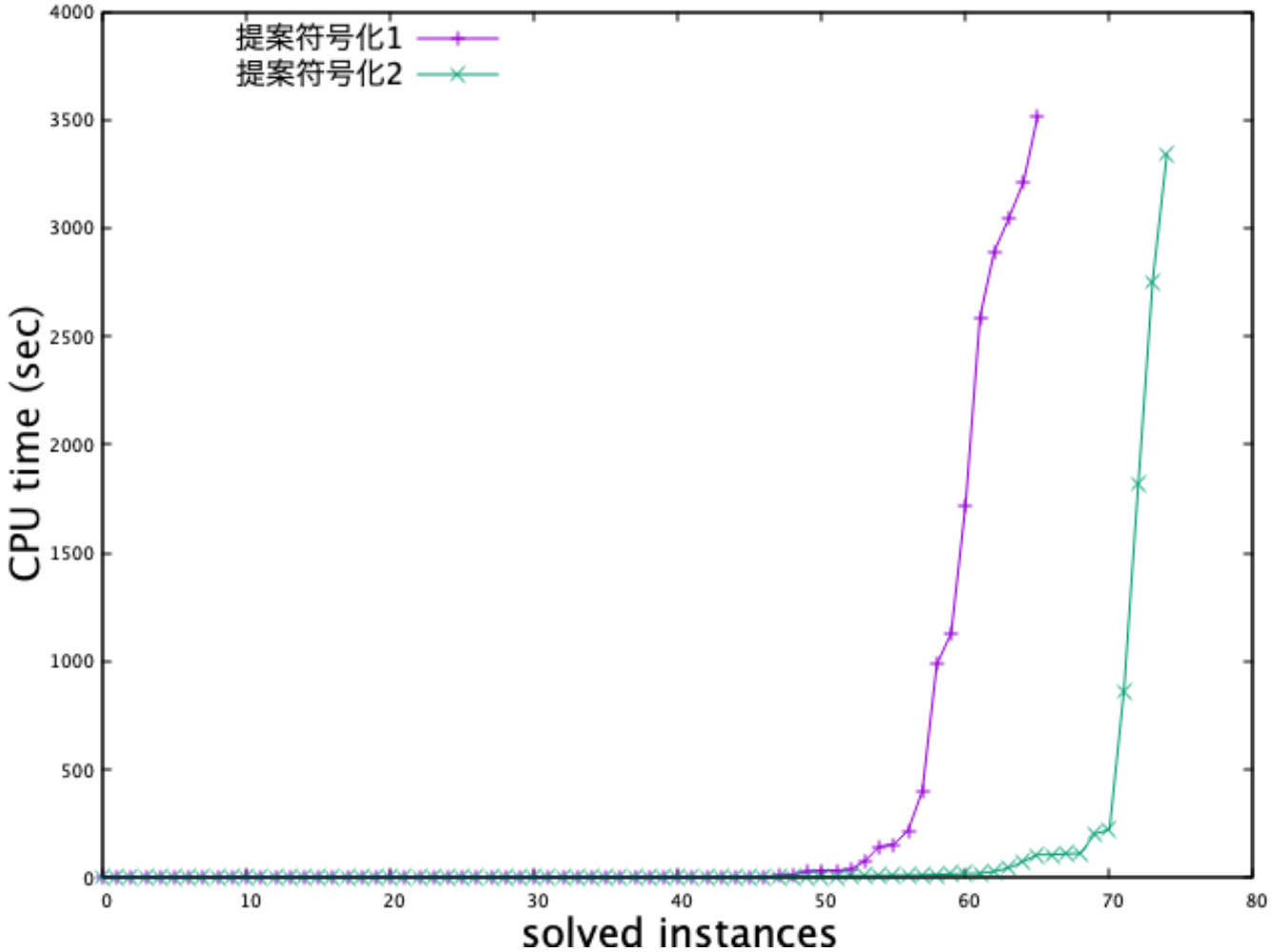
- **比較する** ASP **符号化**:
  - **提案符号化** 1
  - **提案符号化** 2
- **ベンチマーク問題**: **全** 85 **問**
  - DNET[†] **で公開されている配電網問題** 3 **問**
    (**トポロジ制約のみ，スイッチ数**: 16 **個，**36 **個，**468 **個**)
  - *Graph Coloring and its Generalizations* [‡] **で公開されている**
    グラフ彩色問題をベースに，独自に生成した 82 **問** [§]
    $(20 \leq$ (**辺の数**) $\leq 49,629)$
- 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 (1/2)：カクタスプロット



- 提案符号化 2 は，提案符号化 1 と比較して，より多くの問題を高速に解いている．

実験結果 (2/2)：解けた問題数による比較

辺の範囲	問題数	提案符号化 1	提案符号化 2
1 ～ 1,000	30	30	30
1,001 ～ 4,000	20	20	20
4,001 ～ 7,000	11	9	10
7,001 ～ 10,000	8	4	6
10,001 ～ 20,000	9	2	5
20,001 ～ 30,000	2	1	2
30,001 ～ 40,000	1	0	0
40,001 ～ 50,000	4	0	2
計	85	66	75

- **大規模な問題に対する提案符号化 2 の有効性が確認できた．**
- **提案符号化 2 は，辺の数が 40,000 を超える問題も解いた．**

まとめと今後の課題

まとめ

1. **根付き全域森問題を解く，2 種類の ASP 符号化を考案した．**
   - ASP 言語の表現力を用いて，根付き全域森の制約を簡潔に表現できることが確認できた．(ASP のルールで 5 つ程度)
2. **実用規模の問題，及び，より大規模な問題を用いた評価実験．**
   - 提案符号化 2 は，辺の数が 40,000 を超えるような問題も解くことができ，大規模な問題に対する有効性が確認できた．
3. 遷移問題へ拡張し，その符号化を考案した． (本発表では省略)

今後の課題

- 電気制約への対応．
  - ASPMT 技術を用いた実装．
- 遷移問題へ拡張した符号化の評価実験．

- 補足用 -

補足：スマートグリッド

- スマートグリッドとは，電力の供給側，需要側において双方向のやり取りを可能にする次世代の賢い電力網である．
- 従来と違い，通信技術の発達により，使用状況などをリアルタイムに把握することが可能となった．
- その時に応じた最適な配電網を構成し，制御するといったことが考えられている．
  - 電力需要の変化による，配電ロスの少ない構成．
  - 自然エネルギーによる発電量の変動を補う構成．
- ASP 言語の表現力や拡張性が，こうした条件の追加に活用できる可能性がある．

補足：電気制約

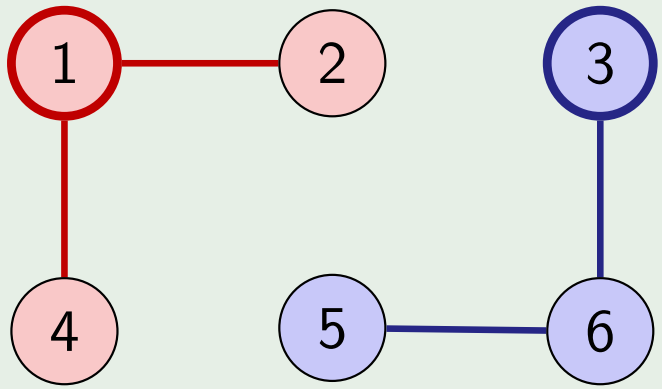
- **電気制約**は，送電する電流・電圧の適正範囲を保証する制約．
  - 供給経路の各区間で許容電流を超えない．
  - 電気抵抗による電圧降下が許容範囲を超えない．
  - etc.
- 電流と電圧が影響し合う**実数ドメイン上の制約**によって表される．
- 実数ドメイン上の制約は，純粋な ASP のみで扱うのは**困難**．
  - 緩和問題として，変電所から供給できる家庭の数に上限をつける．
  - ASPMT 技術により，ASP で得られた解について，背景理論ソルバーと連携して実数ドメイン上の制約を調べる．

補足：ASP システム

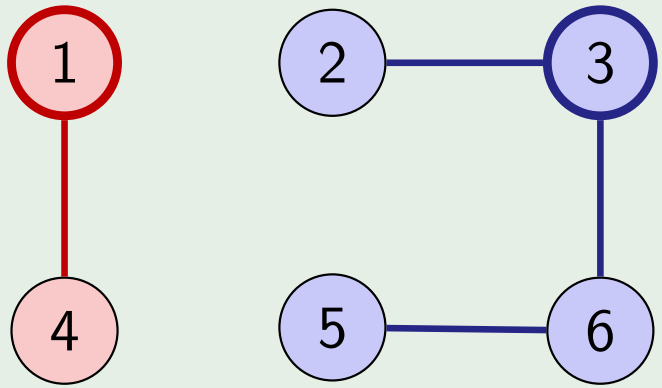
1. 一階 ASP プログラムを基礎化ソルバーによって，命題 ASP プログラムに基礎化する．
2. 命題 ASP プログラムについて，SAT 技術を応用した解集合ソルバーが解集合を探索する．

補足：遷移問題

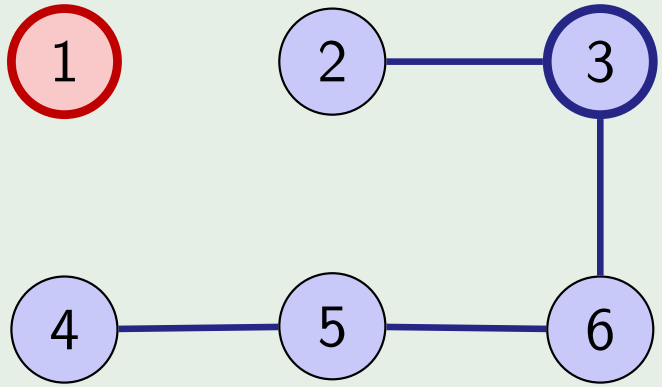
根付き全域森の制約を満たしたまま，各遷移時に変化できる辺の数は 2 つ以下として，遷移を求める．



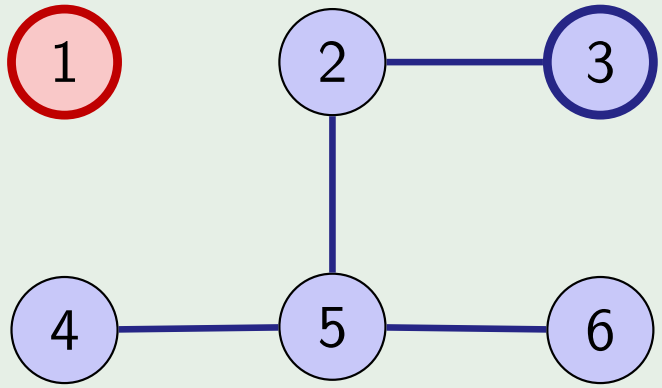
$t = 0$ (**初期状態**)



$t = 1$



$t = 2$



$t = 3$ (**目的状態**)

補足：提案符号化 1 の ASP プログラム

```
1  %% choose edge
2  { inForest(X,Y) } :- edge(X,Y).
3
4  %% generate reached
5  reached(R,R) :- root(R).
6  reached(X,R) :- reached(Y,R), inForest(Y,X).
7  reached(X,R) :- reached(Y,R), inForest(X,Y).
8
9  %% non-cycle constraint
10 :- root(R),
11    not 1 #sum{ 1,X:reached(X,R) ; -1,X,Y:inForest(X,Y),reached(X,R),reached(Y,R)} 1.
12
13 %% rooted connection constraint
14 :- node(X),  not reached(X,_).
15 :- reached(X,R1), reached(X,R2), R1 < R2.
```

補足：提案符号化 2 の ASP プログラム

```
1  %% choose edge
2  { inForest(X,Y) } :- edge(X,Y).
3
4  %% generate reached
5  reached(R,R) :- root(R).
6  reached(X,R) :- reached(Y,R), inForest(Y,X).
7  reached(X,R) :- reached(Y,R), inForest(X,Y).
8
9  %% non-cycle constraint
10 :- root(R),
11    not 1 #sum{ 1,X:reached(X,R) ; -1,X,Y:inForest(X,Y),reached(X,R),reached(Y,R)} 1.
12
13 %% rooted connection constraint
14 :- node(X), not 1 { reached(X,R) } 1.
```