B-1. 計画研究B01班の報告

川原 純
（計画研究B01班代表，京都大学）

Combinatorial Reconfiguration

学術変革領域研究 (B) 2020年度～2022年度
組合せ遷移の展開に向けた計算機科学・工学・数学によるアプローチの融合

計画研究 B01班

工学アプローチによる組合せ遷移の展開：配電切替を足がかりとして汎用ソルバーへ

川原 純　（京都大学）
飯岡 大輔　（中部大学）
戸田 貴久　（電気通信大学）
宋 剛秀　（神戸大学）
鈴木 顕　（東北大学）
照山 順一　（兵庫県立大学）
中畑 裕　（奈良先端大）

B01班の目標

B01 班の大目標
「組合せ遷移に対する実装技術の構築と産業応用」

B01班の2年半の目標
(i) 配電制御システムを題材として
　　組合せ遷移の具体的な産業応用事例を提示
(ii) 組合せ遷移の汎用ソルバーを開発

メタ定理、高速化技法

アルゴリズム設計 (A01班)

数理手法、数学的性質

数学理論構築 (C01班)

B01班の目標

B01 班の大目標

「組合せ遷移に対する実装技術の構築と産業応用」

B01班の2年半の目標

(i) 配電制御システムを題材として
　　組合せ遷移の具体的な産業応用事例を提示

(ii) 組合せ遷移の汎用ソルバーを開発

メタ定理、高速化技法

アルゴリズム設計 (A01班)

数理手法、数学的性質

数学理論構築 (C01班)

B01班の目標

B01 班の大目標

「組合せ遷移に対する実装技術の構築と産業応用」

B01班の2年半の目標

(i) 配電制御システムを題材として

　　組合せ遷移の具体的な産業応用事例を提示

(ii) 組合せ遷移の汎用ソルバーを開発

メタ定理、高速化技法

アルゴリズム設計 (A01班)

数理手法、数学的性質

数学理論構築 (C01班)

B01班の目標

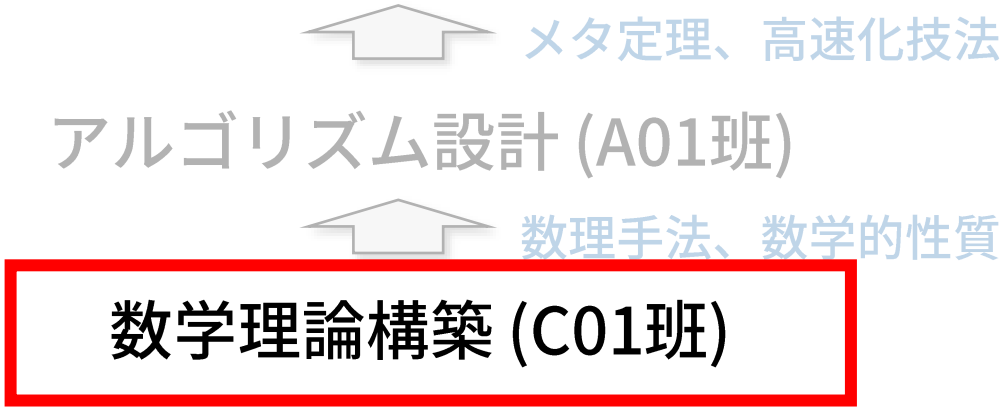
B01 班の大目標

「組合せ遷移に対する実装技術の構築と産業応用」

B01班の2年半の目標

(i) 配電制御システムを題材として
　　組合せ遷移の具体的な産業応用事例を提示

(ii) 組合せ遷移の汎用ソルバーを開発



B01班の目標

B01 班の大目標

「組合せ遷移に対する実装技術の構築と産業応用」

B01班の2年半の目標

(i) 配電制御システムを題材として
　　組合せ遷移の具体的な産業応用事例を提示

(ii) 組合せ遷移の汎用ソルバーを開発

B01班の目標

B01 班の大目標

「組合せ遷移に対する実装技術の構築と産業応用」

B01班の2年半の目標

(i) 配電制御システムを題材として

組合せ遷移の具体的な産業応用事例を提示

(ii) 組合せ遷移の汎用ソルバーを開発

メタ定理、高速化技法

アルゴリズム設計 (A01班)

数理手法、数学的性質

数学理論構築 (C01班)

Satisfiability, Nondeterministic constraint logic, Independent set, Clique, Vertex cover, Set cover, Dominating set, Hitting set, Integer programming, Vertex coloring, Edge coloring, L(2,1)-labeling, Circular coloring, Strong coloring, Graph homorphism, Feedback arc set, Feedback vertex set, Odd cycle transversal, Shortest path, Spanning tree, Induced forest, Induced tree, Induced bipartite subgraph, Cluster subgraph, Steiner tree, Matching, b-matching, Degree-constrained subgraph, Subset sum, Power supply, H-word, Constraint satisfaction

プロジェクト開始前の状況

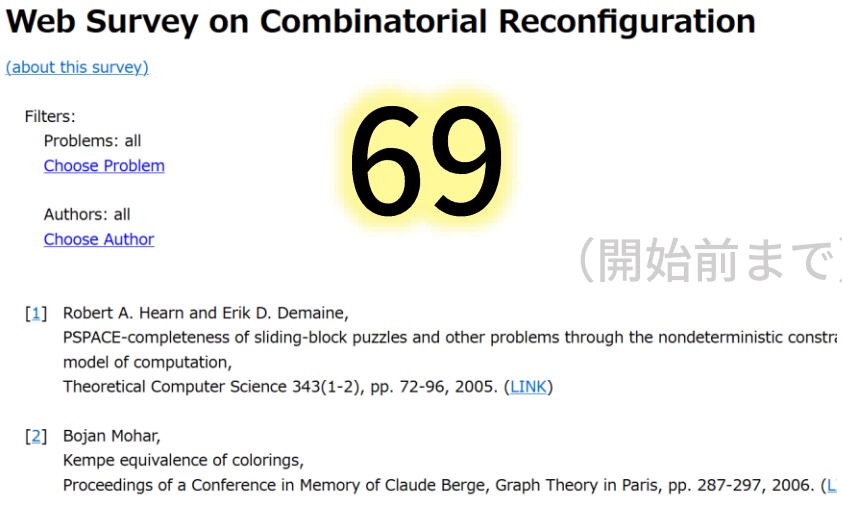
- 組合せ遷移の理論的研究

Satisfiability, Nondeterministic constraint logic, Independent set, Clique, Vertex cover, Set cover, Dominating set, Hitting set, Integer programming, Vertex coloring, Edge coloring, L(2,1)-labeling, Circular coloring, Strong coloring, Graph homorphism, Feedback arc set, Feedback vertex set, Odd cycle transversal, Shortest path, Spanning tree, Induced forest, Induced tree, Induced bipartite subgraph, Cluster subgraph, Steiner tree, Matching, b-matching, Degree-constrained subgraph, Subset sum, Power supply, H-word, Constraint satisfaction

プロジェクト開始前の状況

- 組合せ遷移の**理論**的研究

https://www.ecei.tohoku.ac.jp/alg/coresurvey/



Web Survey on Combinatorial Reconfiguration

(about this survey)

Filters:
Problems: all
Choose Problem

Authors: all
Choose Author

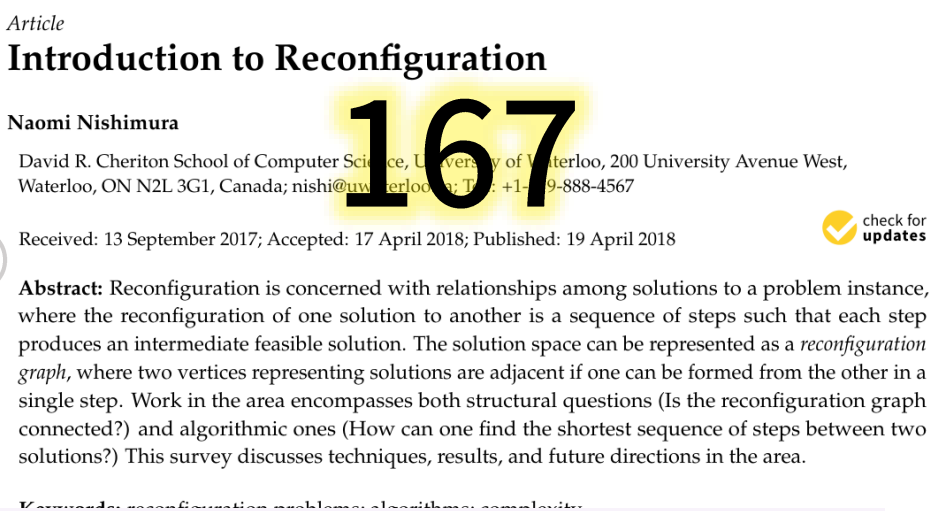
69

(開始前まで)

[1] Robert A. Hearn and Erik D. Demaine,
PSPACE-completeness of sliding-block puzzles and other problems through the nondeterministic constra
model of computation,
Theoretical Computer Science 343(1-2), pp. 72-96, 2005. (LINK)

[2] Bojan Mohar,
Kempe equivalence of colorings,
Proceedings of a Conference in Memory of Claude Berge, Graph Theory in Paris, pp. 287-297, 2006. (L

Nishimura, N.
Introduction to Reconfiguration.
Algorithms 2018, *11*, 52.



Article

Introduction to Reconfiguration

Naomi Nishimura

David R. Cheriton School of Computer Science, University of Waterloo, 200 University Avenue West, Waterloo, ON N2L 3G1, Canada; nishi@uwaterloo.ca; Tel.: +1-519-888-4567

167

Received: 13 September 2017; Accepted: 17 April 2018; Published: 19 April 2018

check for updates

Abstract: Reconfiguration is concerned with relationships among solutions to a problem instance, where the reconfiguration of one solution to another is a sequence of steps such that each step produces an intermediate feasible solution. The solution space can be represented as a *reconfiguration graph*, where two vertices representing solutions are adjacent if one can be formed from the other in a single step. Work in the area encompasses both structural questions (Is the reconfiguration graph connected?) and algorithmic ones (How can one find the shortest sequence of steps between two solutions?) This survey discusses techniques, results, and future directions in the area.

Satisfiability, Nondeterministic constraint logic, Independent set, Clique, Vertex cover, Set cover, Dominating set, Hitting set, Integer programming, Vertex coloring, Edge coloring, L(2,1)-labeling, Circular coloring, Strong coloring, Graph homorphism, Feedback arc set, Feedback vertex set, Odd cycle transversal, Shortest path, Spanning tree, Induced forest, Induced tree, Induced bipartite subgraph, Cluster subgraph, Steiner tree, Matching, b-matching, Degree-constrained subgraph, Subset sum, Power supply, H-word, Constraint satisfaction

プロジェクト開始前の状況

- 組合せ遷移の理論的研究

https://www.ecei.tohoku.ac.jp/alg/coresurvey/

Nishimura, N.
Introduction to Reconfiguration.
Algorithms 2018, *11*, 52.

個々の問題に対する理論研究は
ここ10年でかなり進んだ

Satisfiability, Nondeterministic constraint logic, Independent set, Clique, Vertex cover, Set cover, Dominating set, Hitting set, Integer programming, Vertex coloring, Edge coloring, L(2,1)-labeling, Circular coloring, Strong coloring, Graph homorphism, Feedback arc set, Feedback vertex set, Odd cycle transversal, Shortest path, Spanning tree, Induced forest, Induced tree, Induced bipartite subgraph, Cluster subgraph, Steiner tree, Matching, b-matching, Degree-constrained subgraph, Subset sum, Power supply, H-word, Constraint satisfaction

配電制御（開閉器切替）

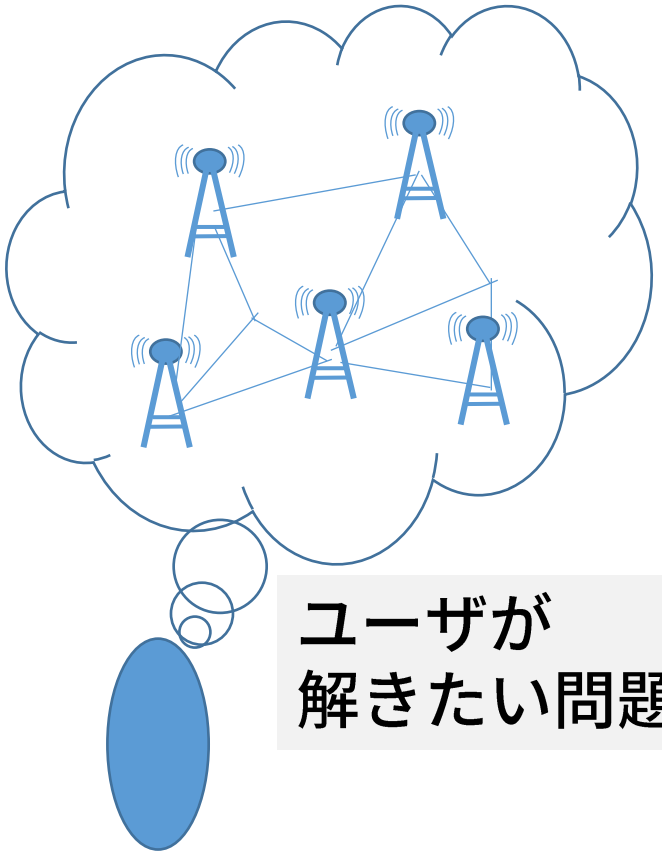
プロジェクト開始前の状況

- 組合せ遷移の観点からの適用事例研究

（ただし他の観点からの研究で、組合せ遷移とみなせるものはある）

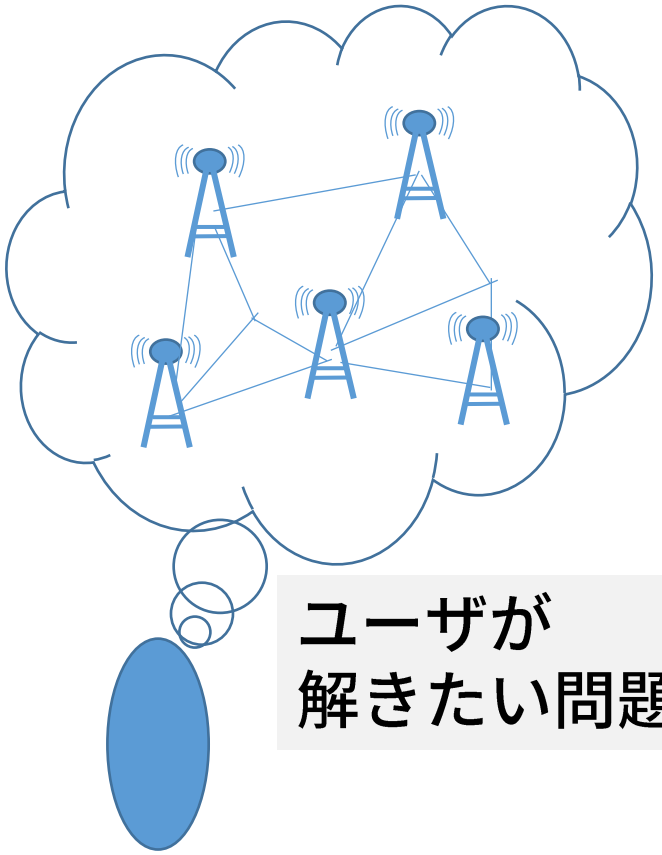
配電制御（開閉器切替）

プロジェクト開始前の状況

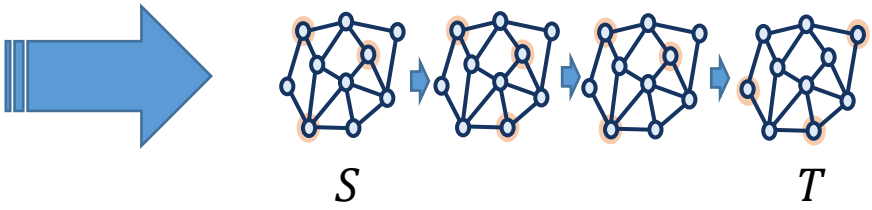


ユーザが
解きたい問題

プロジェクト開始前の状況

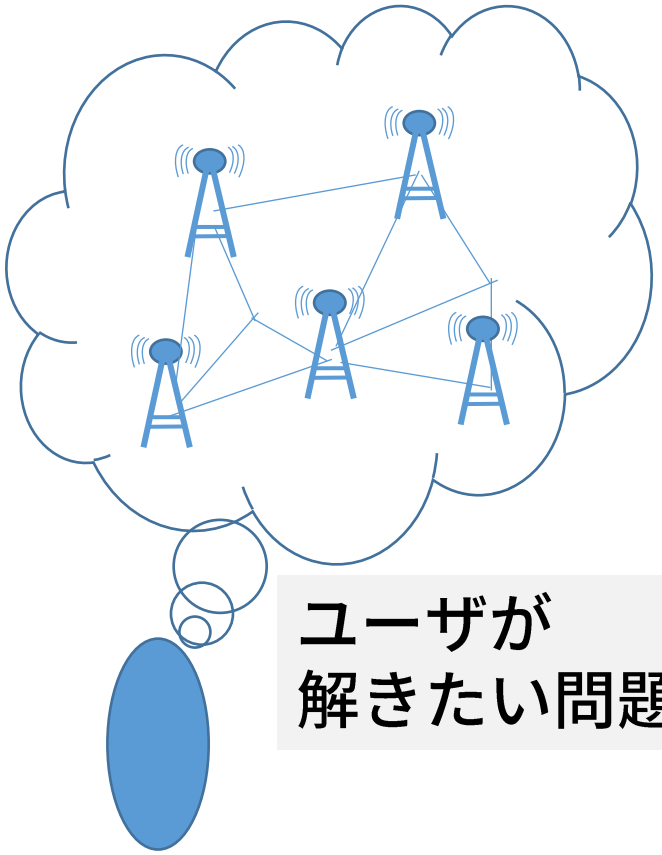


ユーザが
解きたい問題

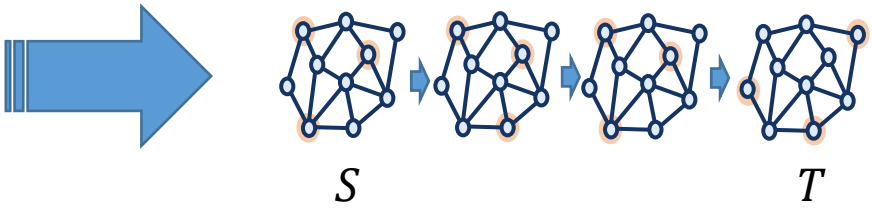


組合せ遷移としてモデル化

プロジェクト開始前の状況



ユーザが
解きたい問題

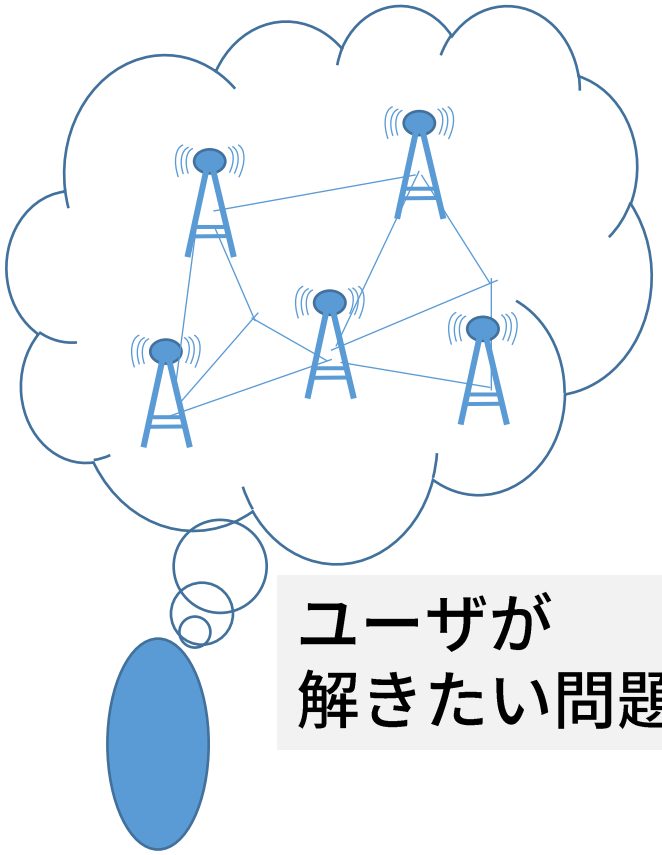


組合せ遷移としてモデル化

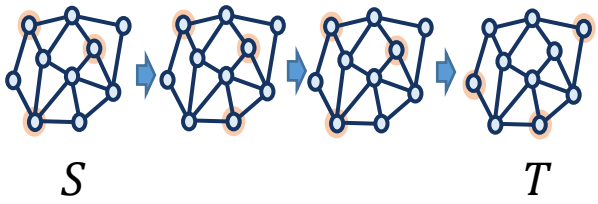
```
boolean
x(1), x(2), x(3), x(4)

solution space
not(x(1) & x(2)) & not(x(1) & x(3)) &
not(x(2) & x(4)) & not(x(2) & x(4))

start space
x(1) = 1 x(2) = 0 x(3) = 0 x(4) = 1

goal space
x(2) & (x(1) + x(3) + x(4) == 1)

reconf rule
x(1) = 1, x(2) = 0 -> x(1) = 0, x(2) = 1
x(1) = 0, x(2) = 1 -> x(1) = 1, x(2) = 0
```

ソルバーが理解できる
記述言語

プロジェクト開始前の状況



ユーザが
解きたい問題



組合せ遷移としてモデル化

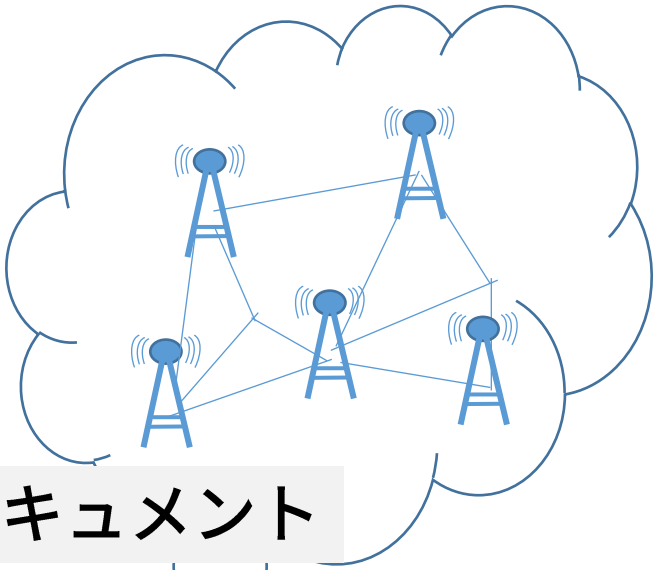
```
boolean
x(1), x(2), x(3), x(4)

solution space
not(x(1) & x(2)) & not(x(1) & x(3)) &
not(x(2) & x(4)) & not(x(2) & x(4))

start space
x(1) = 1 x(2) = 0 x(3) = 0 x(4) = 1

goal space
x(2) & (x(1) + x(3) + x(4) == 1)

reconf rule
x(1) = 1, x(2) = 0 -> x(1) = 0, x(2) = 1
x(1) = 0, x(2) = 1 -> x(1) = 1, x(2) = 0
```

ソルバーが理解できる
記述言語

```
./program sample1.col --sptree --stfile=sample1.dat --st
Input graph file parsed. # of vertices = 5, # of edges = 6
s 1 2 4 5
t 1 3 4 6
Solution space ZDD construction start
Solution space ZDD construction end
Start searching a reconfiguration sequence from s to t
Step 1 time = 0.000083, si
Step 2 time = 0.000072, si
t found
```

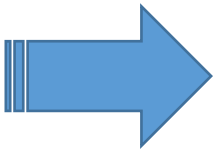
ソルバーエンジン

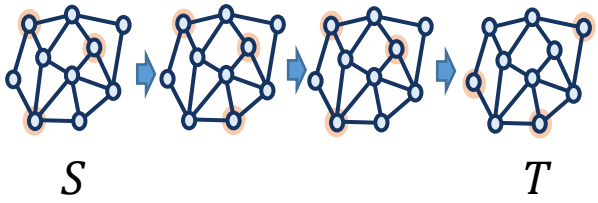
プロジェクト開始前の状況



ドキュメント

ユーザが
解きたい問題





組合せ遷移としてモデル化

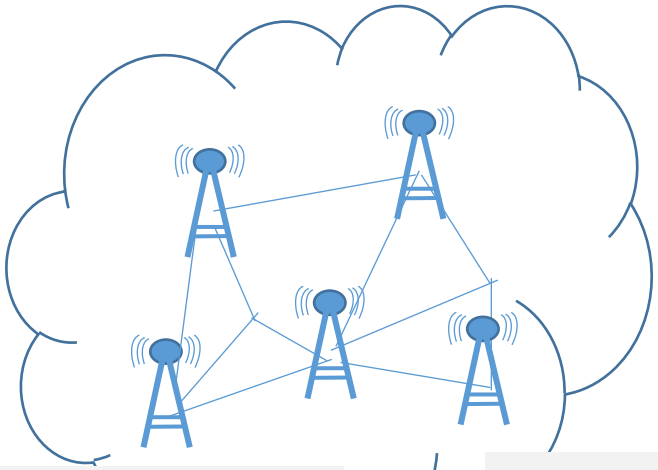
```
boolean
x(1), x(2), x(3), x(4)

solution space
not(x(1) & x(2)) & not(x(1) & x(3)) &
not(x(2) & x(4)) & not(x(2) & x(4))

start space
x(1) = 1 x(2) = 0 x(3) = 0 x(4) = 1

goal space
x(2) & (x(1) + x(3) + x(4) == 1)

reconf rule
x(1) = 1, x(2) = 0 -> x(1) = 0, x(2) = 1
x(1) = 0, x(2) = 1 -> x(1) = 1, x(2) = 0
```

ソルバーが理解できる
記述言語

```
./program sample1.col --sptree --stfile=sample1.dat --st
Input graph file parsed. # of vertices = 5, # of edges = 6
s 1 2 4 5
t 1 3 4 6
Solution space ZDD construction start
Solution space ZDD construction end
Start searching a reconfiguration sequence from s to t
Step 1 time = 0.000083, si
Step 2 time = 0.000072, si
t found
```

ソルバーエンジン

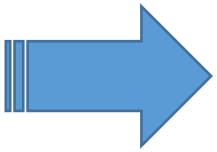
プロジェクト開始前の状況

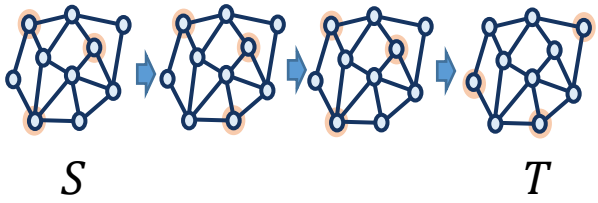


ドキュメント

コミュニティ

ユーザが
解きたい問題





組合せ遷移としてモデル化

```
boolean
x(1), x(2), x(3), x(4)

solution space
not(x(1) & x(2)) & not(x(1) & x(3)) &
not(x(2) & x(4)) & not(x(2) & x(4))

start space
x(1) = 1 x(2) = 0 x(3) = 0 x(4) = 1

goal space
x(2) & (x(1) + x(3) + x(4) == 1)

reconf rule
x(1) = 1, x(2) = 0 -> x(1) = 0, x(2) = 1
x(1) = 0, x(2) = 1 -> x(1) = 1, x(2) = 0
```

ソルバーが理解できる
記述言語

```
./program sample1.col --sptree --stfile=sample1.dat --st
Input graph file parsed. # of vertices = 5, # of edges = 6
s 1 2 4 5
t 1 3 4 6
Solution space ZDD construction start
Solution space ZDD construction end
Start searching a reconfiguration sequence from s to t
Step 1 time = 0.000083, si
Step 2 time = 0.000072, si
t found
```

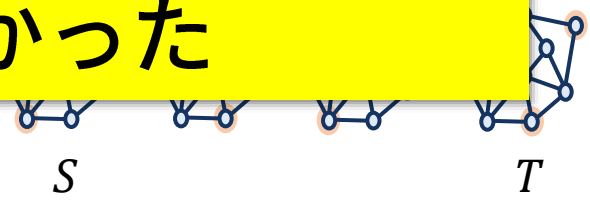
ソルバーエンジン

プ

ユーザが解きたい問題を
組合せ遷移としてモデル化して
解く環境は無かった

ドキュメント

コミュニティ



組合せ遷移としてモデル化

ユーザが
解きたい問題

```
boolean
x(1), x(2), x(3), x(4)

solution space
not(x(1) & x(2)) & not(x(1) & x(3)) &
not(x(2) & x(4)) & not(x(2) & x(4))

start space
x(1) = 1 x(2) = 0 x(3) = 0 x(4) = 1

goal space
x(2) & (x(1) + x(3) + x(4) == 1)

reconf rule
x(1) = 1, x(2) = 0 -> x(1) = 0, x(2) = 1
x(1) = 0, x(2) = 1 -> x(1) = 1, x(2) = 0
```

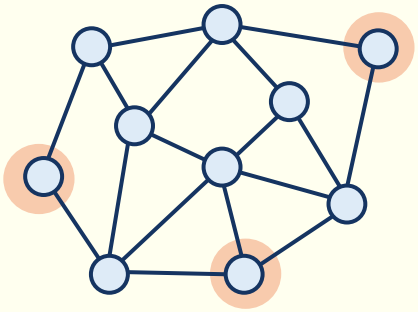
```
./program sample1.col --sptree --stfile=sample1.dat --st
Input graph file parsed. # of vertices = 5, # of edges = 6
s 1 2 4 5
t 1 3 4 6
Solution space ZDD construction start
Solution space ZDD construction end
Start searching a reconfiguration sequence from s to t
Step 1 time = 0.000083, si
Step 2 time = 0.000072, si
t found
```

ソルバーエンジン

ソルバーが理解できる
記述言語

組合せ遷移問題の特徴

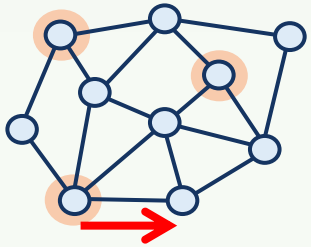
遷移対象



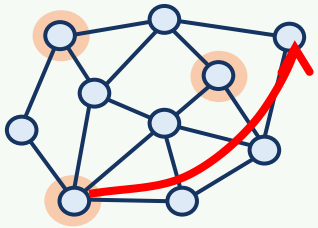
独立集合
支配集合
頂点被覆
ヒッティングセット
…

×

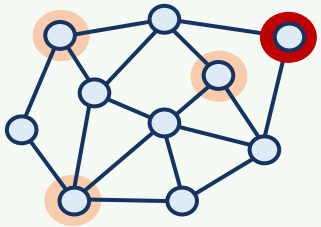
遷移規則



トークンスライド



トークンジャンプ



トークン追加削除

（×対象グラフ）

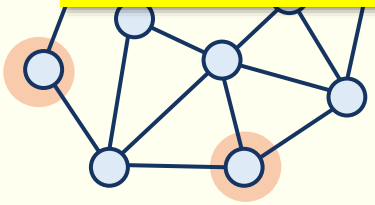
組合せにより
様々な変種が存在

組合せ遷移問題の特徴

遷移対象

遷移規則

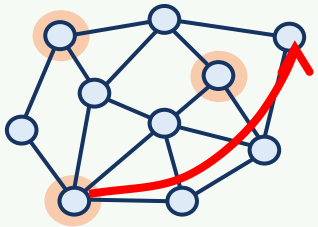
個々に、ソルバーやアルゴリズムを作成するのは効率が悪い



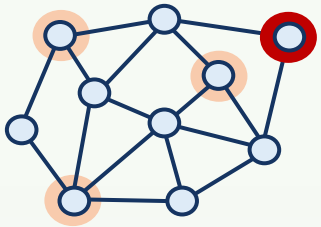
独立集合
支配集合
頂点被覆
ヒッティングセット
…

×

トークンスライド



トークンジャンプ



トークン追加削除

（×対象グラフ）

組合せにより
様々な変種が存在

B01班の研究成果全体図

様々な適用事例への展開

プログラミング競技会
CoRe Challenge の開催

配電網の多段融通 (B-3)

ソフトウェア公開 (B-2.1)

ソルバーフレームワーク

B01班の研究成果全体図

様々な適用事例への展開

プログラミング競技会
CoRe Challenge の開催

配電網の多段融通 (B-3)

ソフトウェア公開 (B-2.1)

ソルバーフレームワーク

- ZDDベースソルバー (B-2.2) 川原
- BMCベースソルバー (B-2.3) 戸田
- SATベースソルバー (B-2.4) 宋
- 計算量解析（2022/9 発表）照山・中畑

B01班の研究成果全体図

様々な適用事例への展開

プログラミング競技会 CoRe Challenge の開催

配電網の多段融通 (B-3)

ソフトウェア公開 (B-2.1)

- GUI ビュアー
- コマンドラインソルバー
- Python インターフェイス（仮公開）
- グラフ一階論理式の　　戸田
  ブール符号化ライブラリ（公開予定）
- ポートフォリオ型実行基盤（公開予定）

ソルバーフレームワーク

- ZDDベースソルバー (B-2.2) 川原
- BMCベースソルバー (B-2.3) 戸田
- SATベースソルバー (B-2.4) 宋
- 計算量解析（2022/9 発表）照山・中畑

B01班の研究成果全体図

様々な適用事例への展開

プログラミング競技会 CoRe Challenge の開催

配電網の多段融通 (B-3)

- 組合せ遷移問題として定式化
- ZDDアルゴリズム
- 再停電を防ぐ工夫
- 6段融通の例を発見

飯岡・鈴木
- 伊藤(A01)
- 明電舎
- 川原

ソフトウェア公開 (B-2.1)

- GUI ビュアー
- コマンドラインソルバー
- Python インターフェイス（仮公開）
- グラフ一階論理式の　戸田
  ブール符号化ライブラリ（公開予定）
- ポートフォリオ型実行基盤（公開予定）

ソルバーフレームワーク

- ZDDベースソルバー (B-2.2) 川原
- BMCベースソルバー (B-2.3) 戸田
- SATベースソルバー (B-2.4) 宋
- 計算量解析（2022/9 発表）照山・中畑

B01班の研究成果全体図

様々な適用事例への展開

プログラミング競技会 CoRe Challenge の開催

- ベンチマークデータ整備
- アルゴリズム比較基盤構築
- コミュニティ醸成

宋・伊藤(A01)
・岡本(C01)

配電網の多段融通 (B-3)

- 組合せ遷移問題として定式化
- ZDDアルゴリズム
- 再停電を防ぐ工夫
- 6段融通の例を発見

飯岡・鈴木
・伊藤(A01)
・明電舎
・川原

ソフトウェア公開 (B-2.1)

- GUI ビュアー
- コマンドラインソルバー
- Python インターフェイス（仮公開）
- グラフ一階論理式の　戸田
  ブール符号化ライブラリ（公開予定）
- ポートフォリオ型実行基盤（公開予定）

ソルバーフレームワーク

- ZDDベースソルバー (B-2.2) 川原
- BMCベースソルバー (B-2.3) 戸田
- SATベースソルバー (B-2.4) 宋
- 計算量解析（2022/9 発表）照山・中畑

B01班の研究成果全体図: 展開

様々な適用事例への展開

経路のリルート
（最短路遷移）

指導学生のテーマ

国際会議投稿済

B01班の研究成果全体図: 展開

様々な適用事例への展開

経路のリルート（最短路遷移）

指導学生のテーマ

国際会議投稿済

変更の少ない選挙区割遷移（グラフ分割遷移）

文教大 堀田先生との共同研究

学生シンポジウムで発表

B01班の研究成果全体図: 展開

様々な適用事例への展開

経路のリルート（最短路遷移）

指導学生のテーマ

国際会議投稿済

変更の少ない選挙区割遷移（グラフ分割遷移）

文教大 堀田先生との共同研究

学生シンポジウムで発表

荷物の詰め替え（ナップサック遷移）

信州大 藤原先生との共同研究

学生シンポジウムで発表

B01班の研究成果全体図: 展開

様々な適用事例への展開

経路のリルート（最短路遷移）

指導学生のテーマ

国際会議投稿済

変更の少ない選挙区割遷移（グラフ分割遷移）

文教大 堀田先生との共同研究

学生シンポジウムで発表

荷物の詰め替え（ナップサック遷移）

信州大 藤原先生との共同研究

学生シンポジウムで発表

ボールソートパズルのソルバー

JAIST 上原先生
岩手大 山中先生
AFSA メンバー
本 PJ メンバーとの共同研究

B01班の研究成果全体図: 展開

様々な適用事例への展開

経路のリルート（最短路遷移）

指導学生のテーマ

国際会議投稿済

変更の少ない選挙区割遷移（グラフ分割遷移）

文教大 堀田先生との共同研究

学生シンポジウムで発表

荷物の詰め替え（ナップサック遷移）

信州大 藤原先生との共同研究

学生シンポジウムで発表

ボールソートパズルのソルバー

JAIST 上原先生
岩手大 山中先生
AFSA メンバー
本 PJ メンバーとの共同研究

光ファイバーの経路再設定（辺素パス遷移）

指導学生のテーマ

共同研究準備

B01班の研究成果全体図

様々な適用事例への展開

- 経路のリルート
- 選挙区割の変更
- 荷物の詰め替え
- ボールソートパズル
- 光ファイバー経路

プログラミング競技会 CoRe Challenge の開催

- ベンチマークデータ整備
- アルゴリズム比較基盤構築
- コミュニティ醸成

宋・伊藤(A01)
・岡本(C01)

配電網の多段融通 (B-3)

- 組合せ遷移問題として定式化
- ZDDアルゴリズム
- 再停電を防ぐ工夫
- 6段融通の例を発見

飯岡・鈴木
・伊藤(A01)
・明電舎
・川原

ソフトウェア公開 (B-2.1)

- GUI ビュアー
- コマンドラインソルバー
- Python インターフェイス（仮公開）
- グラフ一階論理式の　戸田
  ブール符号化ライブラリ（公開予定）
- ポートフォリオ型実行基盤（公開予定）

ソルバーフレームワーク

- ZDDベースソルバー (B-2.2) 川原
- BMCベースソルバー (B-2.3) 戸田
- SATベースソルバー (B-2.4) 宋
- 計算量解析（2022/9 発表）照山・中畑

B-2. 実装技術基盤：組合せ遷移ソルバーとその技術

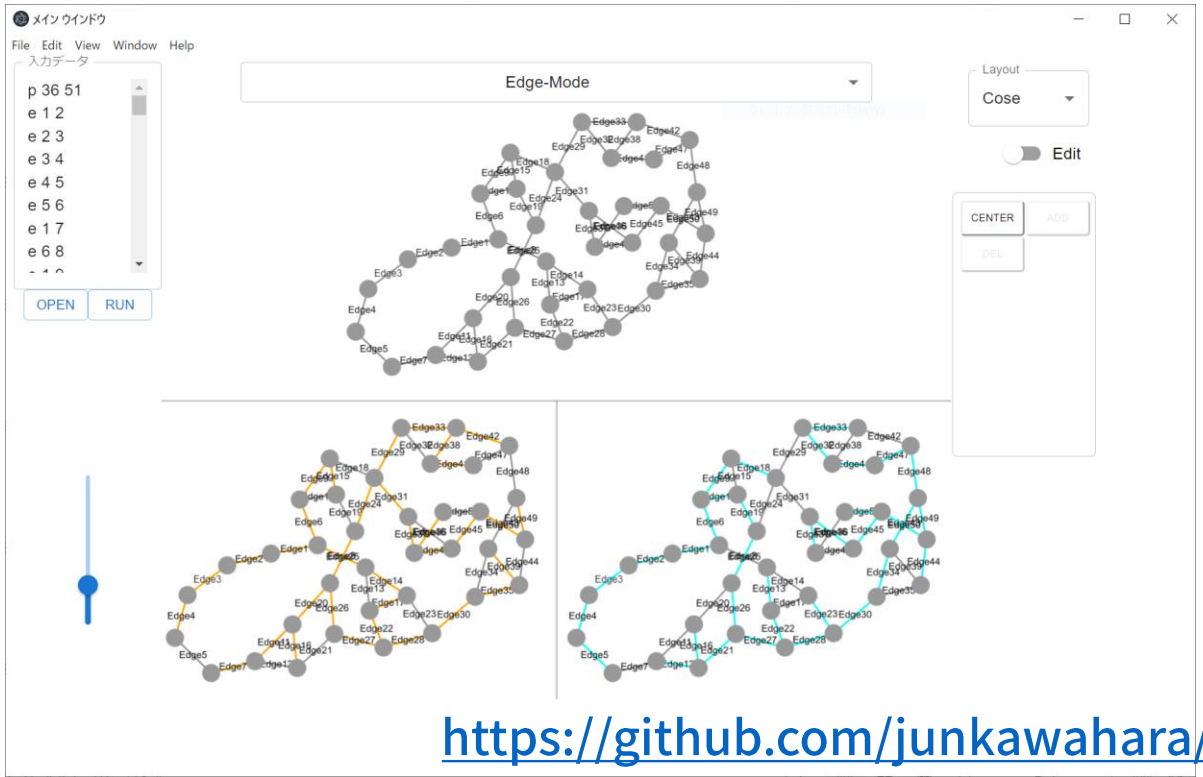
B-2.1. GUI 実演

川原 純
（京都大学）

Combinatorial Reconfiguration

GUI ビュアー

- CoReViewer
  - 組合せ遷移ソルバーの GUI フロントエンド
  - Windows / Mac / Linux 対応
  - グラフの頂点変数タイプ（独立集合遷移など）
    辺変数タイプ（全域木遷移やマッチング遷移など）対応



ソルバーエンジン部分は Core Challenge 参加の任意のプログラムに差し替え可

https://github.com/junkawahara/CoReViewer

GUI ビュアーに同梱のソルバー

- ポートフォリオ型実行基盤
  - 複数のソルバーをマルチスレッドで動かす
  - 最も速く解を出力したソルバーの解を採用
  - 公開（CoReViewer に同梱）予定

reconfillion

- graphillion + combinatorial reconfiguration
- graphillion で扱うのが可能な部分グラフ集合を、遷移させることが可能
- 今のところ、graphillion と混ぜて使うことはできない（graphillion を勝手に拡張した単独のライブラリ）
  - 将来的に対応予定

https://github.com/junkawahara/reconfillion-kari
このページは仮のページです。本公開のときは
URL が変更されます。

メイン開発者: 山崎 宏紀 氏（元京都大学）

reconfillion の使い方

台グラフ（universe）を設定

```
>>> vertices = [1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> edges = [(1, 2), (1, 4), (2, 3), (2, 5),
>>>     (3, 6), (4, 5), (4, 7), (5, 6),
>>>     (5, 8), (6, 9), (7, 8), (8, 9)]
>>> setset.set_universe(vertices)
```

すべての独立集合

```
>>> iss = reconf.get_independent_setset(vertices, edges)
```

reconfillion の使い方

開始、目標独立集合

```
>>> s = {2, 4, 6}
>>> t = {1, 6, 8}
```

遷移列を 1 つ求める

```
>>> seq = reconf.get_reconf_seq(s, t, iss)
```

遷移列を print

```
>>> for x in seq:
>>>     print(x)
```

reconfillion の使い方

遷移列を 1 つ求める（token-addition-removal モデル）

```
>>> seq = reconf.get_reconf_seq(s, t, iss, method='tar', k=2)
```

reconfillion の使い方

マニュアルより引用

- get_reconf_seq_ts(s, t, search_space, edges)
  - s から t への search_space 内での遷移列を求める
  - edges で指定した要素間の遷移のみ可能 (token sliding)
  - s, t は setset
  - returns 遷移列 (list of sets)

- get_longest_seq(s, search_space, method, k)
  - s から search_space 内での最長 の遷移列を求める
  - s は setset or GraphSet
  - method, k の動作は get_reconf_seq と同じ
  - returns 遷移列 (list of sets)

科研費 学術変革領域研究(B) 2020-2022年度
組合せ遷移の展開に向けた計算機科学・工学・数学によるアプローチの融合

B-2. 実装技術基盤：組合せ遷移ソルバーとその技術

B-2.2. ZDDベースソルバー

川原 純
（京都大学）

2023年2月21日(火) 最終報告会

Combinatorial Reconfiguration

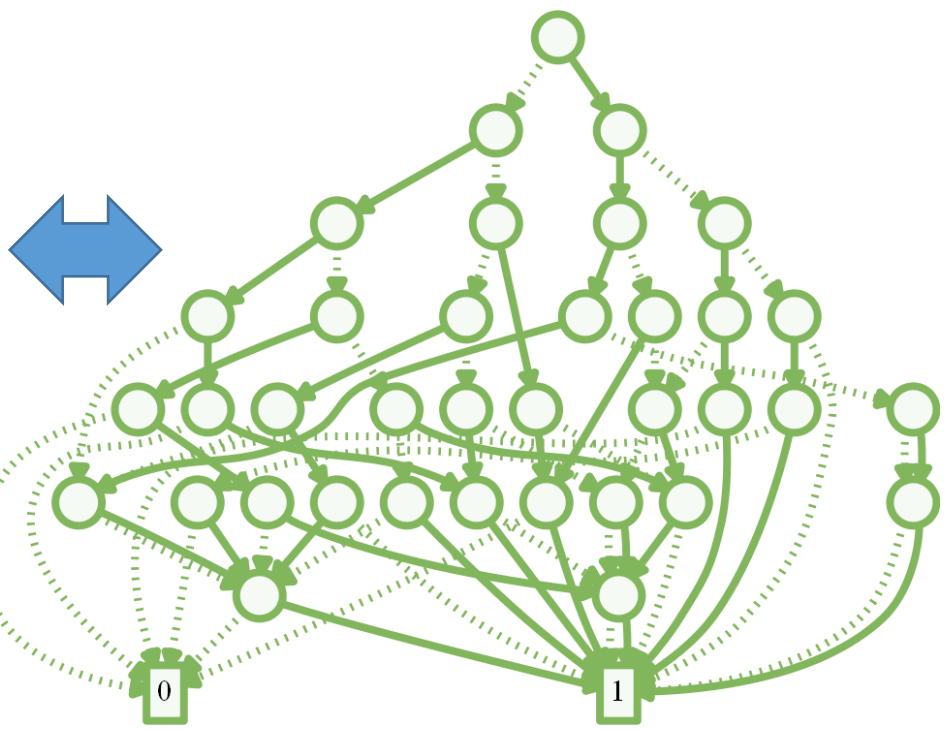
ZDDフレームワーク

- ZDD (Zero-suppressed decision diagram) [Minato 1993]
  集合族を圧縮して保持

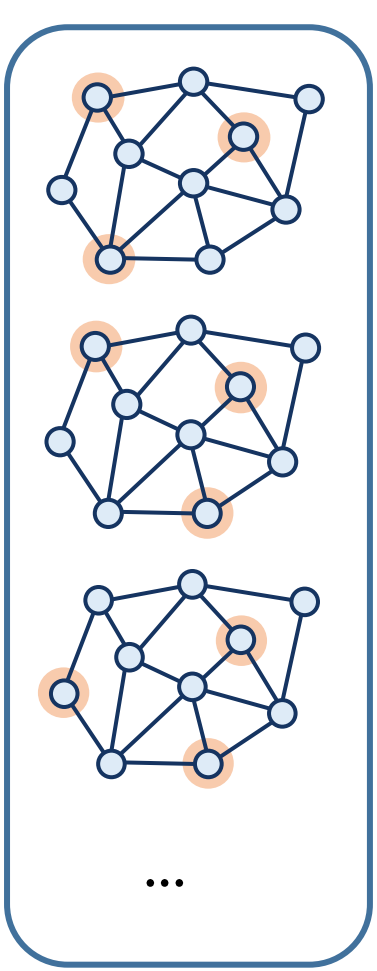
集合族

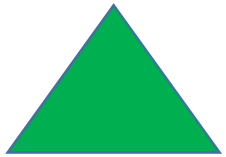
{2, 3, 5}, {1, 2, 3, 4}, {1, 3}, {3, 6}, {2, 5, 6, 7},
{1, 2, 6, 7}, {1, 6, 7}, {1, 2, 5, 7}, {2, 3, 6},
{2, 5, 6, 7}, {1, 2, 4, 5, 6, 7}, {1, 4}, {1, 5, 6},
{1, 2, 3, 5, 7}, {1, 2, 3, 6}, {1, 2}, {1, 6, 7},
{1, 2, 4, 7}, {2, 5, 6, 7}, {1, 3, 4, 5, 6}, {1, 3},
{5, 6, 7}, {1, 4, 5, 6, 7}, {3, 6, 7}, {3, 4, 7}, {1},
{2}, {6, 7}, {1, 2, 5}, {7}, {2, 5, 7}, {2, 6},
{1, 5, 7}, {3, 5, 7}, {1, 2, 6, 7}, {2, 3, 5, 6, 7},
{2, 5}, {2, 3, 4, 6}, {}, {2, 3}, {1, 6}, {1, 2, 4},
{2, 3, 5, 7}, {2, 3, 6, 7}, {3, 5, 6, 7}, {1, 5, 6},
{3}, {2, 6, 7}, {3, 4}, {2, 4, 6, 7}, {1, 2, 3, 4},
{2, 3, 5}, {1, 2, 3, 6, 7}, {1, 2, 3, 4, 6}, {5, 7},
{5}, {2, 5, 6, 7}, {1, 3, 4, 6}, {1, 2, 5, 6},
{2, 3, 4, 5, 6}, {3, 4, 5, 6}, {3, 4, 7}, {1, 5, 7},
{3, 4, 5, 7}

ZDD



ZDD フレームワーク全体図

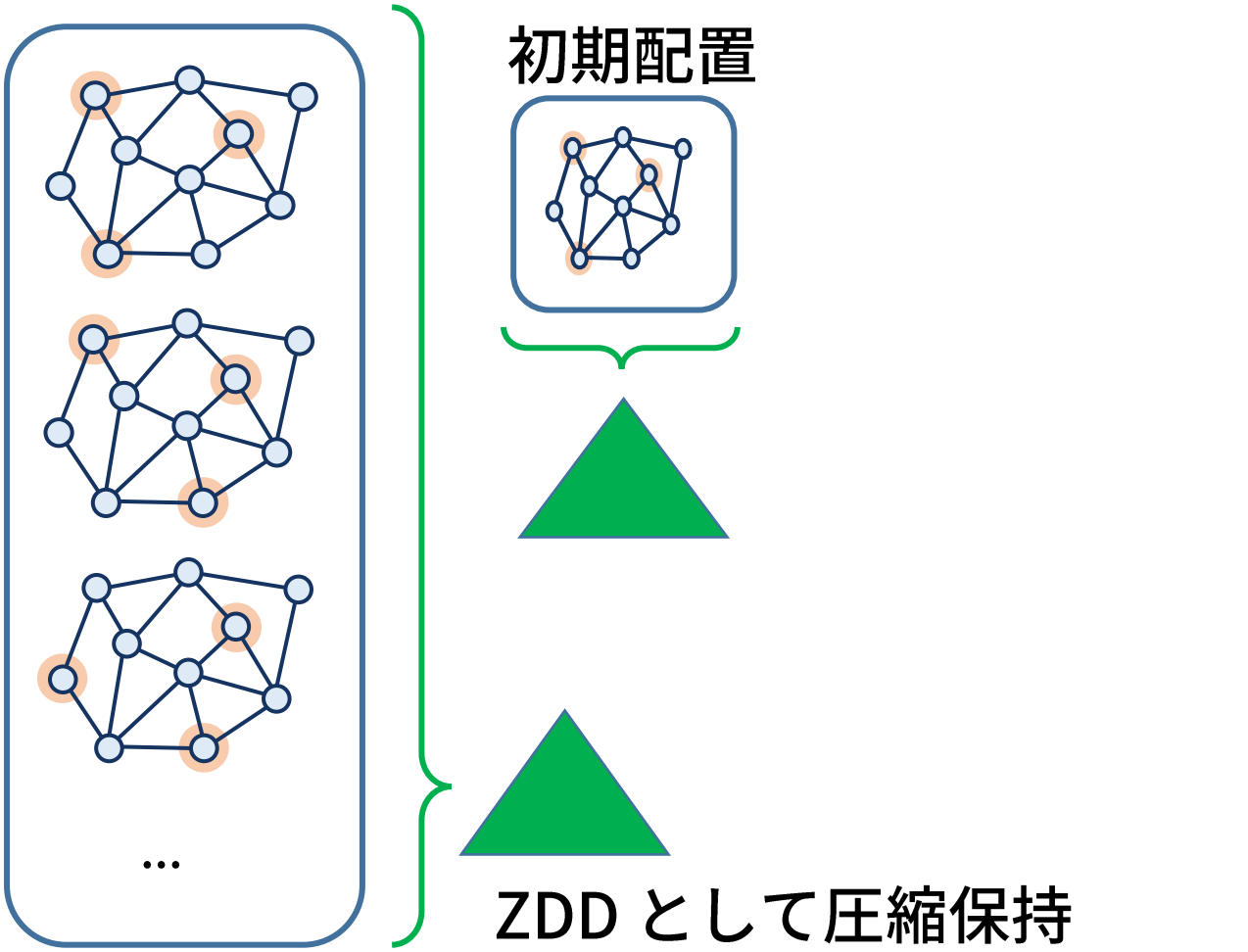




ZDD として圧縮保持
（既存アルゴリズム）

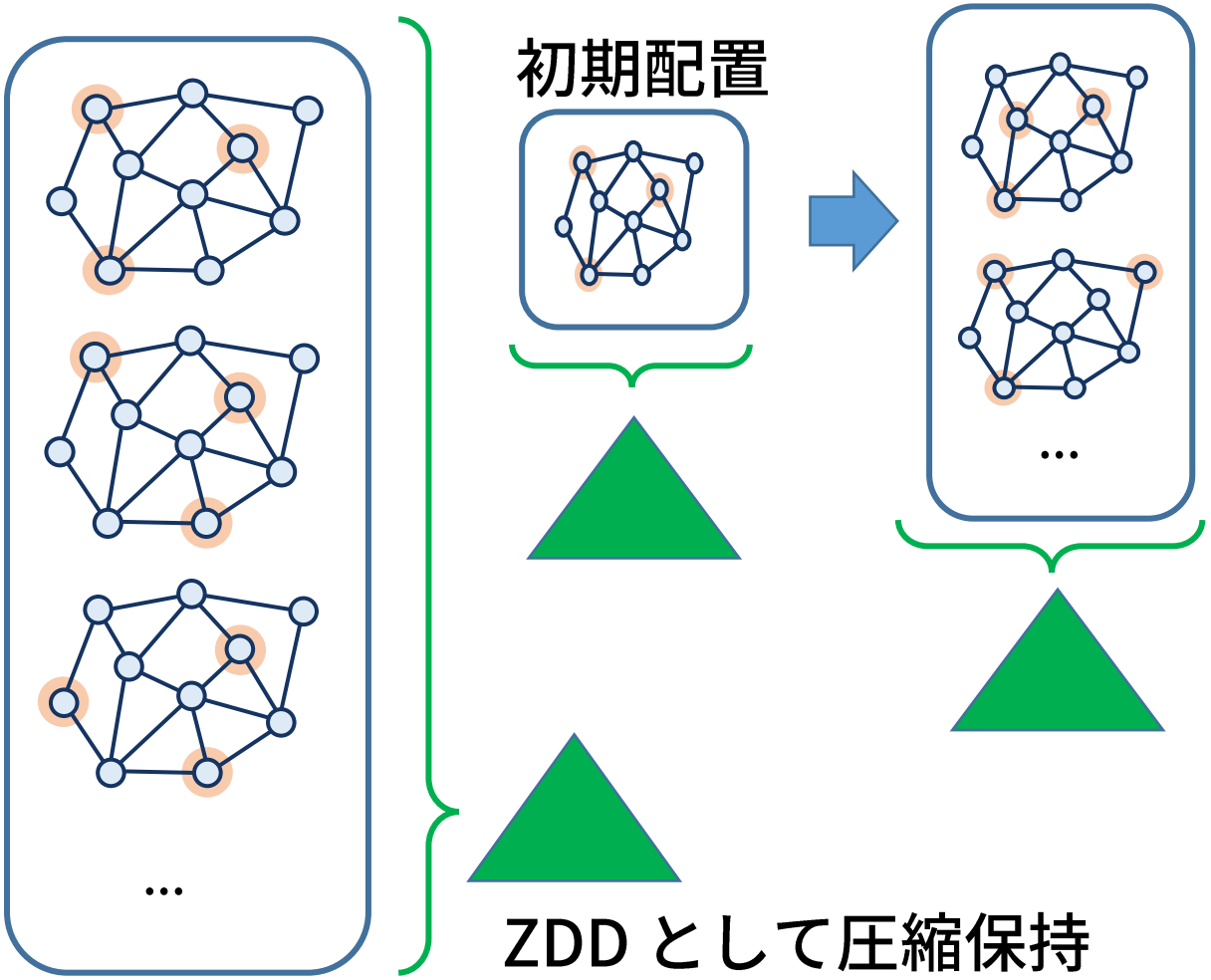
解空間（可能な全配置）

ZDD フレームワーク全体図



ZDD フレームワーク全体図

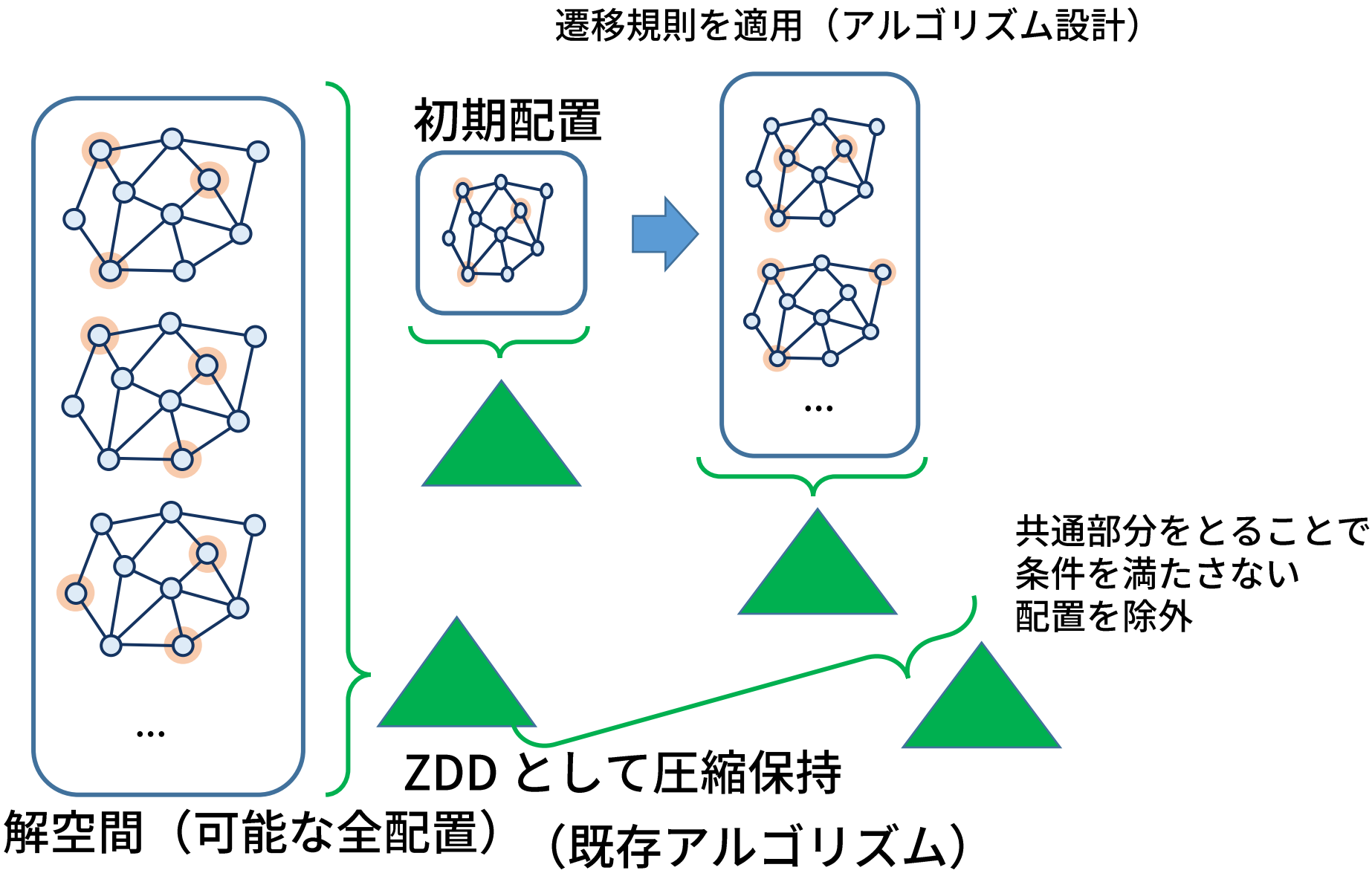
遷移規則を適用（アルゴリズム設計）



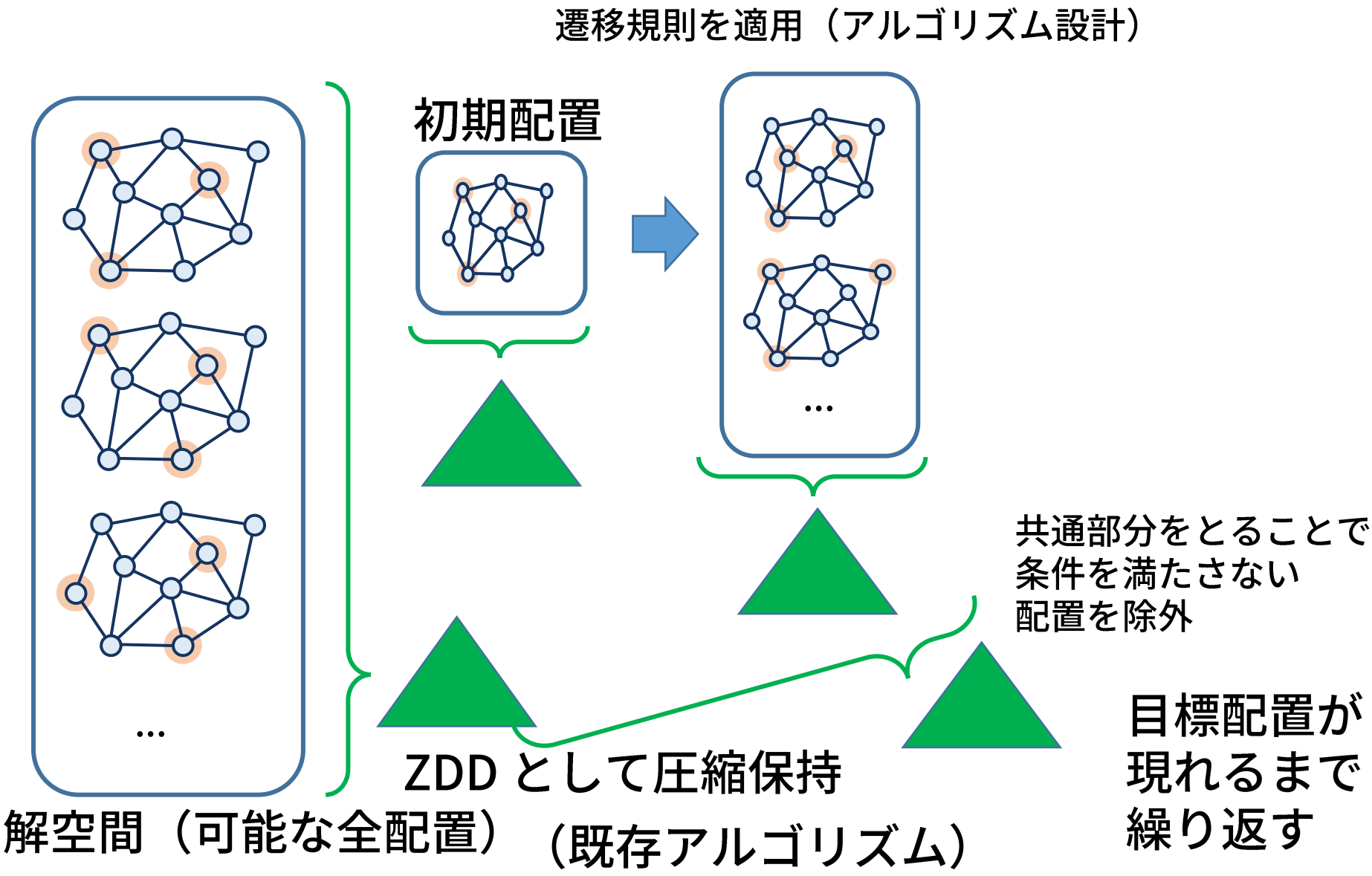
ZDD として圧縮保持

解空間（可能な全配置）　（既存アルゴリズム）

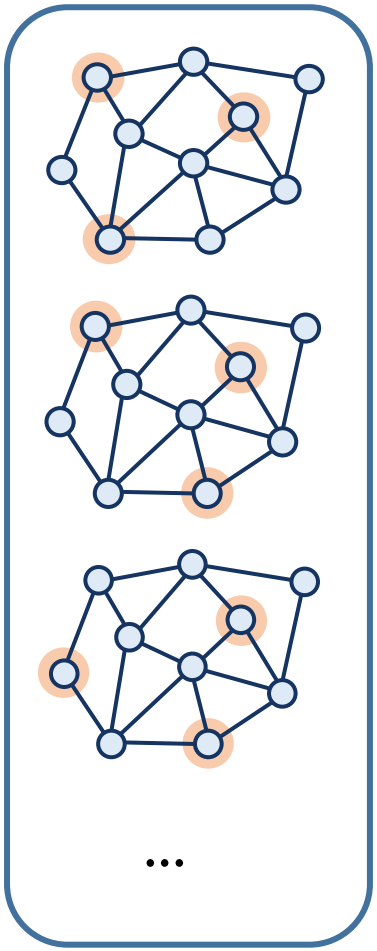
ZDD フレームワーク全体図



ZDD フレームワーク全体図



ZDD フレームワークでできること



解空間（可能な全配置）

ZDD として圧縮保持

解空間を ZDD で表せるなら
遷移対象にできる

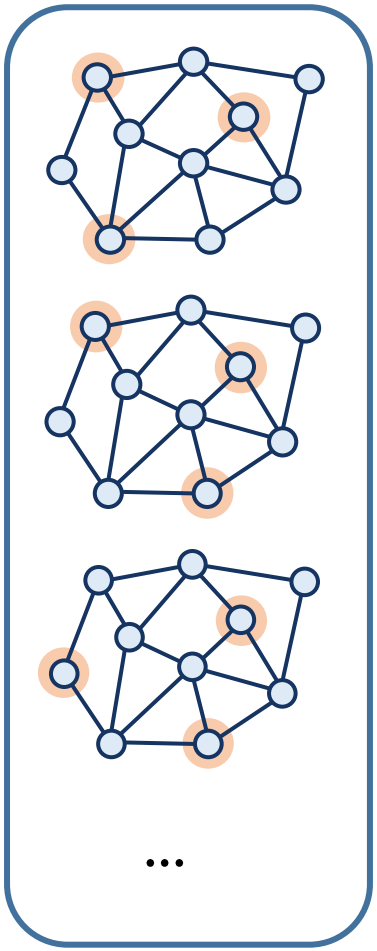
- 各頂点の次数
- 各頂点の連結性
- 連結成分の個数
- サイクルの有無
- 辺重みの総和

の組合せで指定できる部分グラフ集合

独立集合、クリーク、集合被覆、頂点被覆、木、森、全域木、全域森、虹色全域木、根付き全域森、シュタイナー木、パス、ハミルトンパス、サイクル、ハミルトンサイクル、マッチング、完全マッチング、*k*-正則グラフ、（次数が任意の）正則グラフ、次数指定部分グラフ、スターグラフ、クリーク、二部グラフ、JRの片道きっぷ、弦グラフ、弦二部グラフ、メニエルグラフ、*d*-爪自由グラフ、偶穴（奇穴）自由グラフ、誘導パス、誘導マッチング、誘導正則グラフ

minimal, maximal, minimum, maximum,
connected, X-free, induced なども、ZDD 演算で可能

ZDD フレームワークに関する業績



解空間（可能な全配置）

ZDD として圧縮保持

CoRe Challenge 2022（追実験）
sp019 インスタンス（|V| = 247, |E| = 1,578）、
最短遷移長が 5,767,157 を 151,567 秒で解いた

伊藤 健洋, 川原 純, 宋 剛秀, 鈴木 顕, 照山 順一, 戸田 貴久,
ZDDを用いた組合せ遷移ソルバーについての考察,
2021年度冬のLAシンポジウム, 2022.

LA/EATCS-Japan発表論文賞

Takehiro Ito, Jun Kawahara, Yu Nakahata, Takehide Soh, Akira Suzuki, Junichi Teruyama, Takahisa Toda, “ZDD-Based Algorithmic Framework for Solving Shortest Reconfiguration Problems,” in Proceedings of the 20th International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research, 2023.

arXiv:2207.13959

CPAIOR 2023 採録決定

ソースコード公開
https://github.com/junkawahara/ddreconf

B-2. 実装技術基盤：組合せ遷移ソルバーとその技術

B-2.3. BMCベースソルバー

戸田 貴久
(電気通信大学)

Combinatorial Reconfiguration

BMCソルバー開発の背景

個別の研究分野・コミュニティ

組合せ遷移

主にグラフに現れる計算問題の複雑さ

モデル検査

システムのデバッグ・検証のための理論と実践

・・・・

共通する概念

“状態”　“遷移関係”

“到達可能性”

・・・・

分野ごとに違いがあるが、抽象レベルでは共通の概念

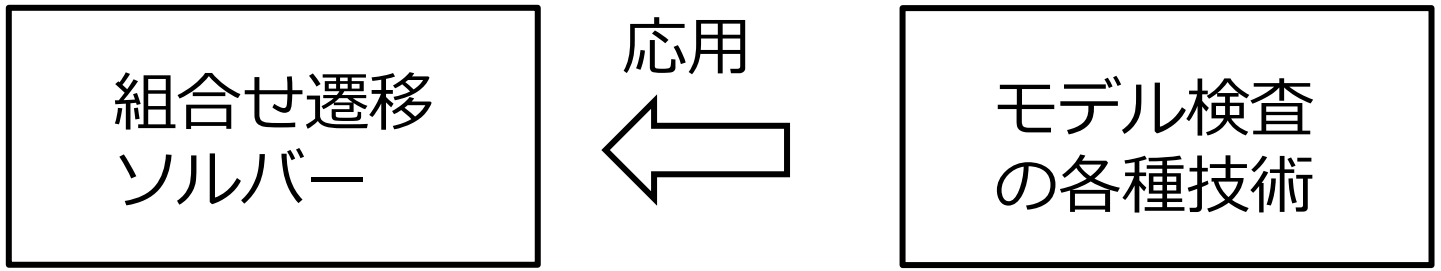
モデル検査器で十分？不十分？

- **目標：組合せ遷移ソルバーの開発**
  - **モデル検査器の技術（BMC）**に基づくソルバ
- BMCで十分？不十分？
  - BMC側には**各種ソフトウェアが整備**
  - 個々のグラフ遷移問題は“頑張れば”解けそう
  - ソフトウェアはグラフをそのまま扱えない

私の研究のチャレンジ

① グラフ概念をどう扱うか？

② 問題の個別の扱いを統一できないか？



グラフ頂点集合の性質と記述

グラフの一階述語論理による
頂点集合の性質の表現可能性

独立集合

$$\bigvee_{1 \le i < j \le k} \neg edg(x_i, x_j)$$

支配集合

$$\forall w \bigvee_{1 \le i \le k} w = x_i \vee edg(w, x_i)$$

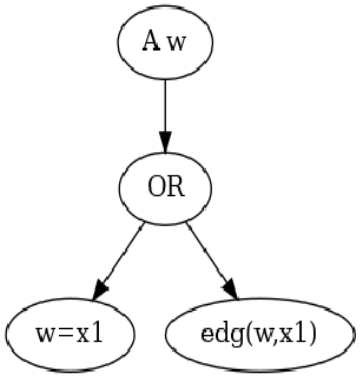
頂点被覆

$$\forall v \forall w \; edg(v, w) \Longrightarrow \bigvee_{1 \le i \le k} v = x_i \vee w = x_i$$

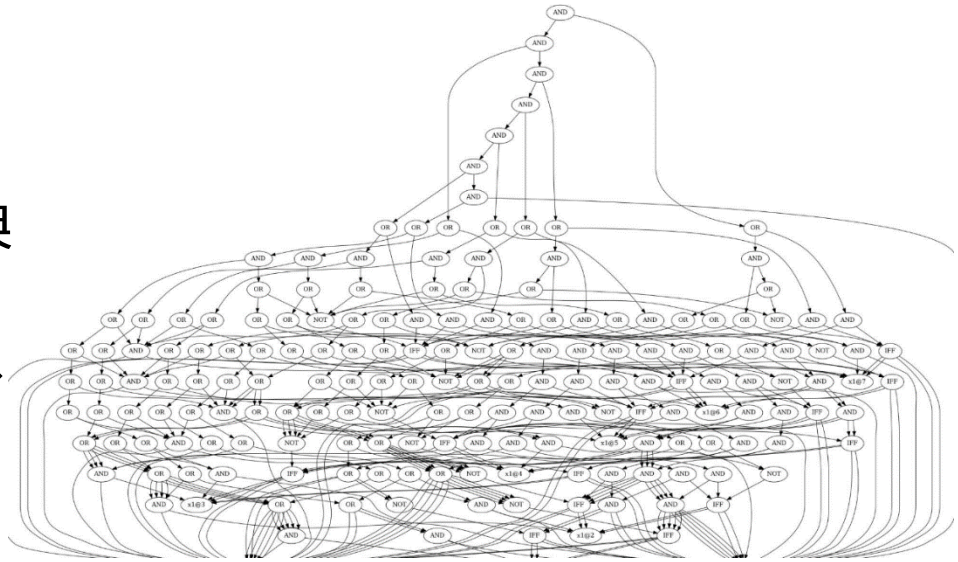
Pythonライブラリの開発

グラフの一階論理で表現可能なあらゆる
頂点集合の性質を命題論理式に変換可能

$$\forall \boldsymbol{w}\ \boldsymbol{w} = \boldsymbol{x_1} \vee \boldsymbol{edg}(\boldsymbol{w}, \boldsymbol{x_1})$$

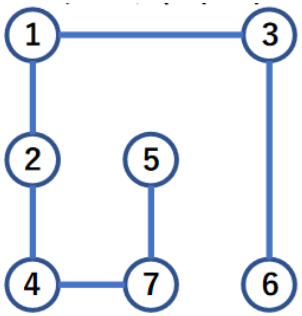


自動変換



等価な命題論理式

グラフ



Pythonライブラリの開発

グラフの一階論理で表現可能なあらゆる
頂点集合の性質を命題論理式に変換可能

```
<expr> ::= <atom>
         | “(” <unop> <expr> “)”
         | “(” <expr> <binop> <expr> “)”
         |  “(” <qf> <var> “)”

<unop> ::= “!”

<binop> ::= “&” | “|” | “->” | “<->”

<qf> ::= “A” | “E”
```

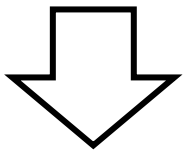
論理式をテキストで記述するだけでOK！

利点

- 抽象度を下げることなしに、
  グラフ概念を記述可能
- 記述言語の学習コストがほぼゼロ
- 充足可能性の最新技術の恩恵を享受可能

例：サイズ高々２の支配集合

A w ((x1=w | edg(x1,w)) | (x2=w | edg(x2,w)))



グラフを指定して
SATに変換

SATの充足解（ここでは支配集合）のサンプリング

グラフの頂点集合の遷移問題の統一的な扱いを可能にする枠組

グラフの１階述語論理式の組$(\phi, \psi, \iota, \gamma)$で表現可能なグラフ遷移問題

- ϕ　状態集合を記述する式
- ψ　状態の遷移関係を記述する式
- ι　初期状態（の集合）を記述する式
- γ　最終状態の集合を記述する式

１階述語論理式で表現された一種のNFA

⇨ **統一的な計算方法も与えた**

利点

いろいろな遷移問題に対する統一的な解法を実現できるようになった。例えば、

- Graph Pebbling
- Tower of Hanoi
- Snake

科研費 学術変革領域研究(B) 2020-2022年度
組合せ遷移の展開に向けた計算機科学・工学・数学によるアプローチの融合

B-2. 実装技術基盤：組合せ遷移ソルバーとその技術

B-2.4. SATベースソルバー

宋 剛秀
（神戸大学）

2023年2月21日(火) 最終報告会

Combinatorial Reconfiguration

研究目的と研究内容

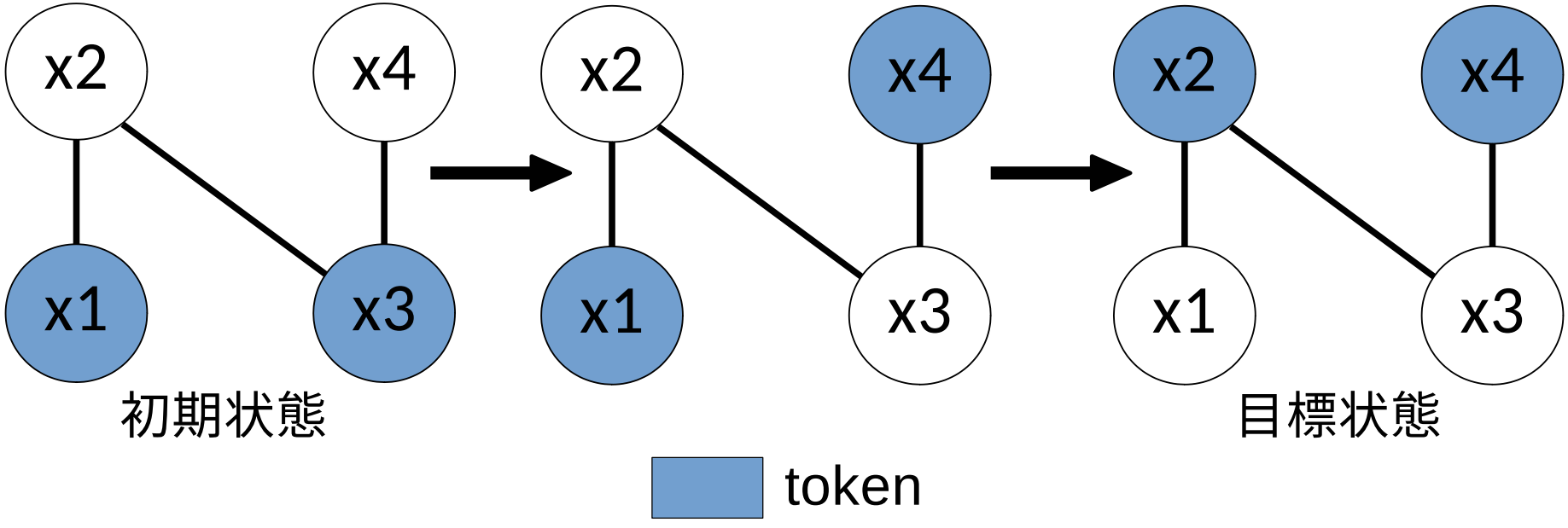
研究内容

- 汎用記述言語 (CoReDL)
  - 組合せ遷移問題の記述に特化した言語を制約プログラミング言語 Sugar [田村ら, 2008] を基に実現した.
- 処理系
  - SAT 符号化と SAT ソルバーを用いて実現した.

研究目的と研究内容

研究内容

- 汎用記述言語 (CoReDL)
  - 組合せ遷移問題の記述に特化した言語を制約プログラミング言語 Sugar [田村ら, 2008] を基に実現した.
- 処理系
  - SAT 符号化と SAT ソルバーを用いて実現した.



独立集合遷移問題の遷移 (遷移毎，1 トークン移動)

汎用記述言語 CoReDL の例

```
(StateVar
 (int x1 0 1)
 (int x2 0 1)  ...
)
(SolutionSpace
 (or (ne x1 1) (ne x2 1))
 (or (ne x1 1) (ne x2 1))  ...
)
(Transition
 (int l1 0 1)
 (int g1 0 1)...
 (iff (lt x1′ x1″) (eq l1 1))
 (iff (gt x1′ x1″) (eq g1 1))
 ...
 (eq (add l1 l2 l3 l4) 1)
 (eq (add g1 g2 g3 g4) 1)
)
```

組合せ遷移問題の入力

1. 基となる組合せ問題:
   - `StateVar`
   - `SolutionSpace`
2. 遷移制約: `Transition`
3. 初期状態: `InitState`
4. 目標状態: `GoalState`

```
(InitState
 (eq x1 1)
 (eq x2 0)...
)
(GoalState
 (eq x1 0)
 (eq x2 1)...
)
```

汎用記述言語 CoReDL の例

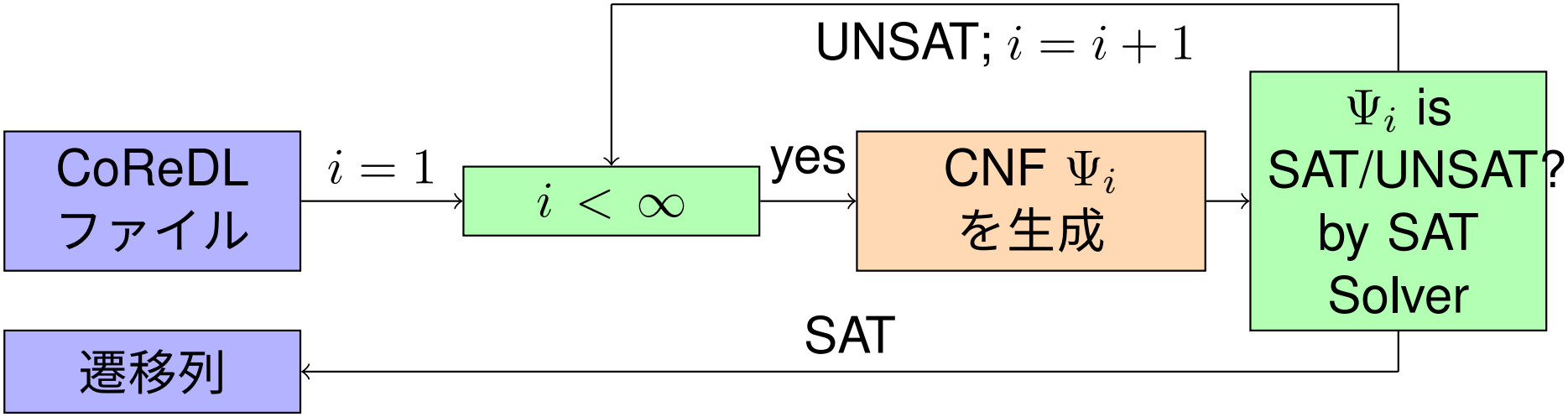
```
(StateVar
 (int x1 0 1)
 (int x2 0 1)   ...
)
(SolutionSpace
 (or (ne x1 1) (ne x2 1))
 (or (ne x1 1) (ne x2 1))   ...
)
(Transition
 (int l1 0 1)
 (int g1 0 1)...
 (iff (lt x1′ x1″) (eq l1 1))
 (iff (gt x1′ x1″) (eq g1 1))
 ...
 (eq (add l1 l2 l3 l4) 1)
 (eq (add g1 g2 g3 g4) 1)
)
```

組合せ遷移問題の入力

1. 基となる組合せ問題:
   - StateVar
   - SolutionSpace
2. 遷移制約: Transition
3. 初期状態: InitState
4. 目標状態: GoalState

```
(InitState
 (eq x1 1)
 (eq x2 0)...
)
(GoalState
 (eq x1 0)
 (eq x2 1)...
)
```

CoReDL の SAT ソルバーを用いた処理系



1. CoReDL ファイルを受け取る. $i = 1$ で初期化する.
2. 「遷移長 i 以内に遷移列が存在するかどうか」を表す SAT 問題 Ψ_i を生成する (SAT 符号化には Sugar を利用).
3. SAT ソルバーで Ψ_i の充足可能性を判定する.
   - SAT であれば遷移列を出力して終了する.
   - UNSAT であれば $i = i + 1$ として手順 2. に戻る.

この手続きは有界モデル検査と同様であり， 遷移列が存在しないことは基本的には示せない.

提案方法の評価

記述言語の評価

- 代表的な組合せ遷移問題である，独立集合遷移問題，グラフ彩色遷移問題，3SAT 遷移問題を実際に記述できることを確認した．
- 既存の制約プログラミング言語を用いた場合， 遷移長に応じて記述が長くなるが， 提案方法では遷移長に依存しない書き方ができることを確認した．

処理系の性能評価

- 独立集合遷移問題を対象とした国際競技会 Core Challenge 2022 のベンチマークと結果と比較した．
- 環境が異なるため厳密な比較は難しいが， 4 位に相当する成績であった (369 問中 1 位が 238 問を解き， 提案システムは 189 問を解いた)．

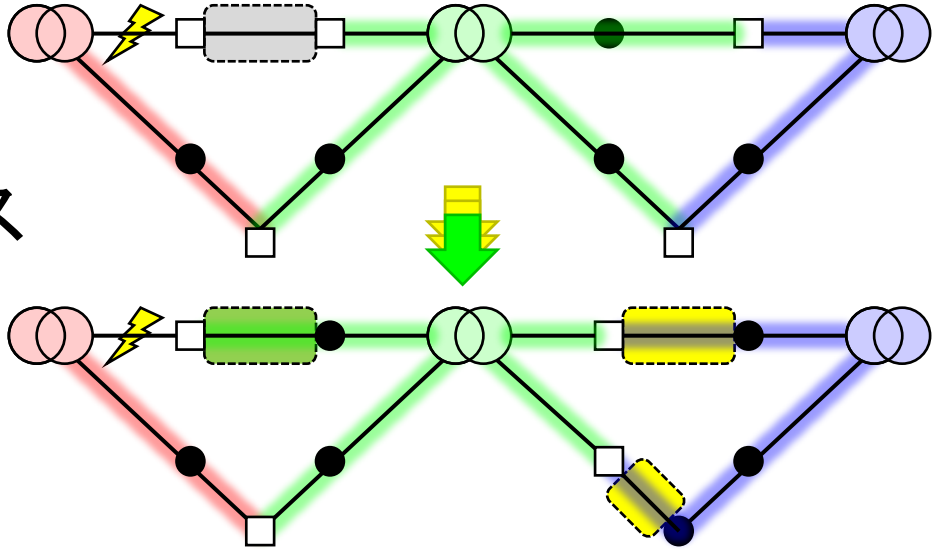
B-3. 実装技術基盤の配電制御への展開

飯岡 大輔
（中部大学）

鈴木 顕
（東北大学）

Combinatorial Reconfiguration

社会への発信

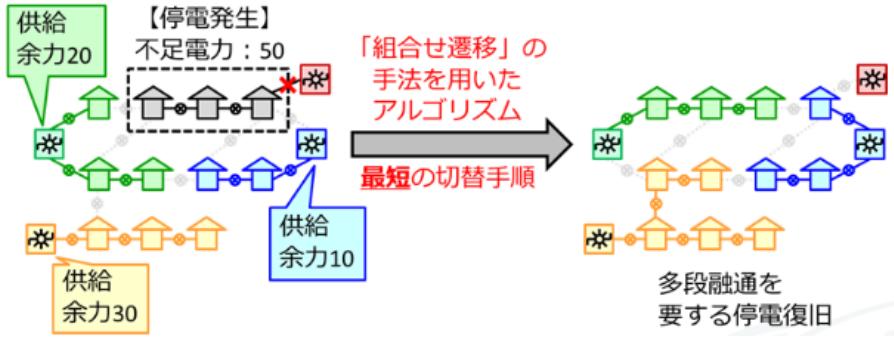


2022年11月7日　プレスリリース

2022年 | プレスリリース・研究成果

停電復旧の最短手順を算出するアルゴリズムを開発 多段融通にも対応、より広域な配電運用への活用に期待

2022年11月 7日 14:00 | プレスリリース・研究成果



- 特許共同出願中
- 2022年11月8日　電気新聞

川原 純	京都大学
山岡宙太	京都大学
伊藤健洋	東北大学
鈴木 顕	東北大学
飯岡大輔	中部大学
杉村修平	明電舎
後藤誠弥	明電舎
田邊隆之	明電舎

2023年3月17日　電気学会　全国大会
「停電復旧の最短手順を算出するアルゴリズム」

- 産学連携研究

配電線に大電流が流れ続けると？

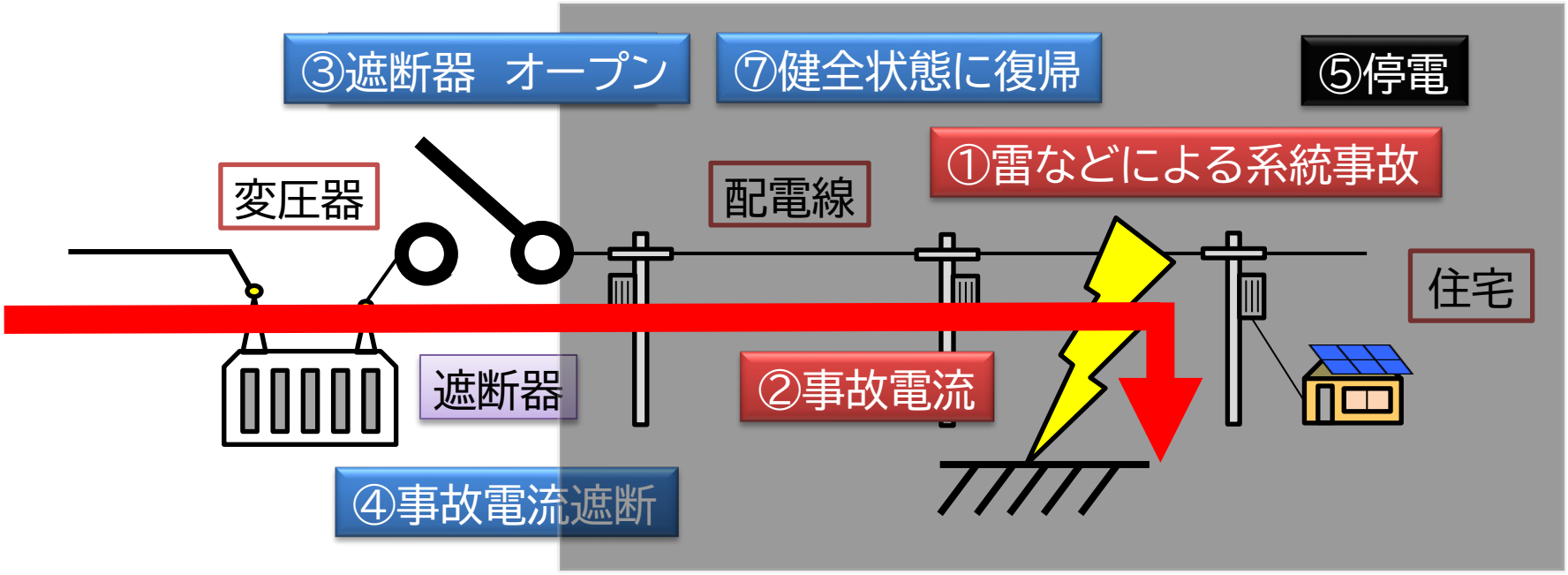
過電流が遮断されずに流れ続けると

設備が損壊，復旧に多くの時間が必要

https://www.youtube.com/watch?v=gOT8jx4jEzM&t=1s

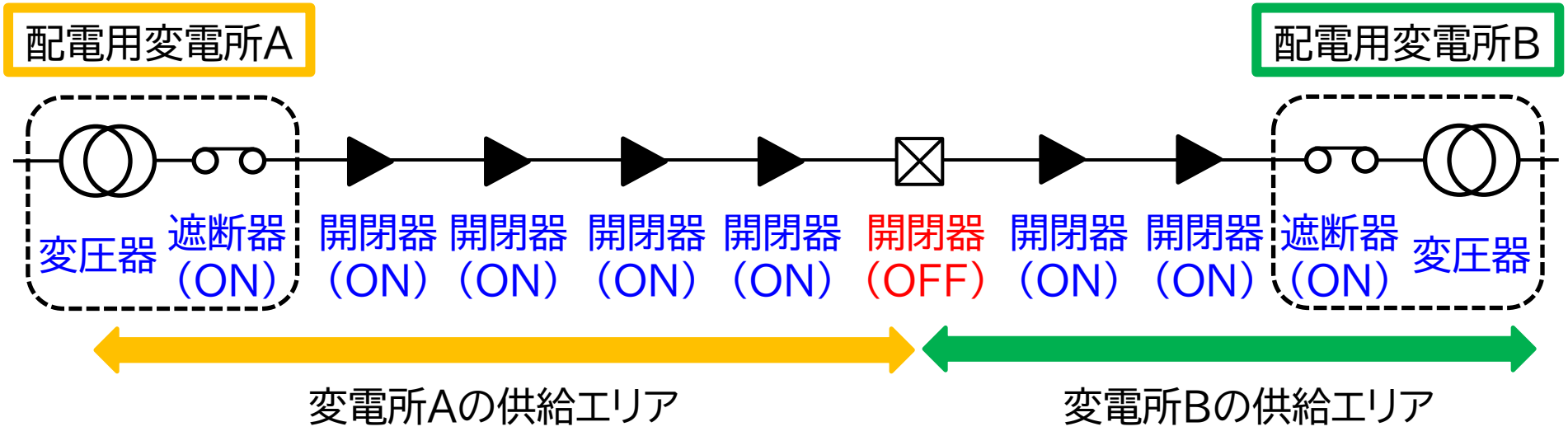
配電自動化システムによる事故復旧

配電自動化システムによる事故復旧



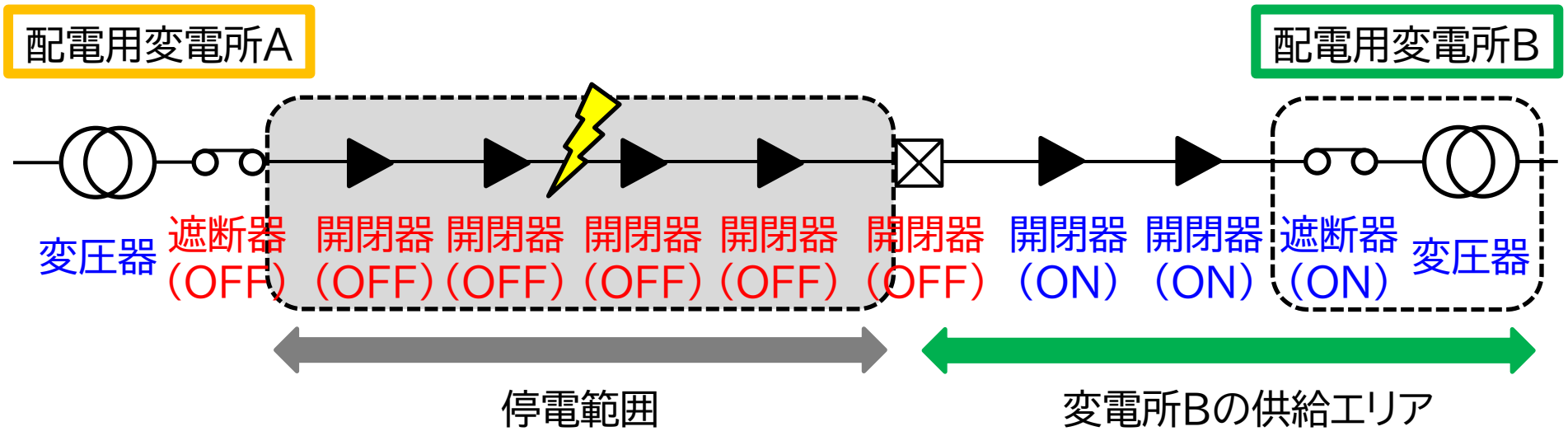
実際のシステム：　停電範囲を極小化するための制御が加えられている

事故復旧動作（停電範囲の極小化）



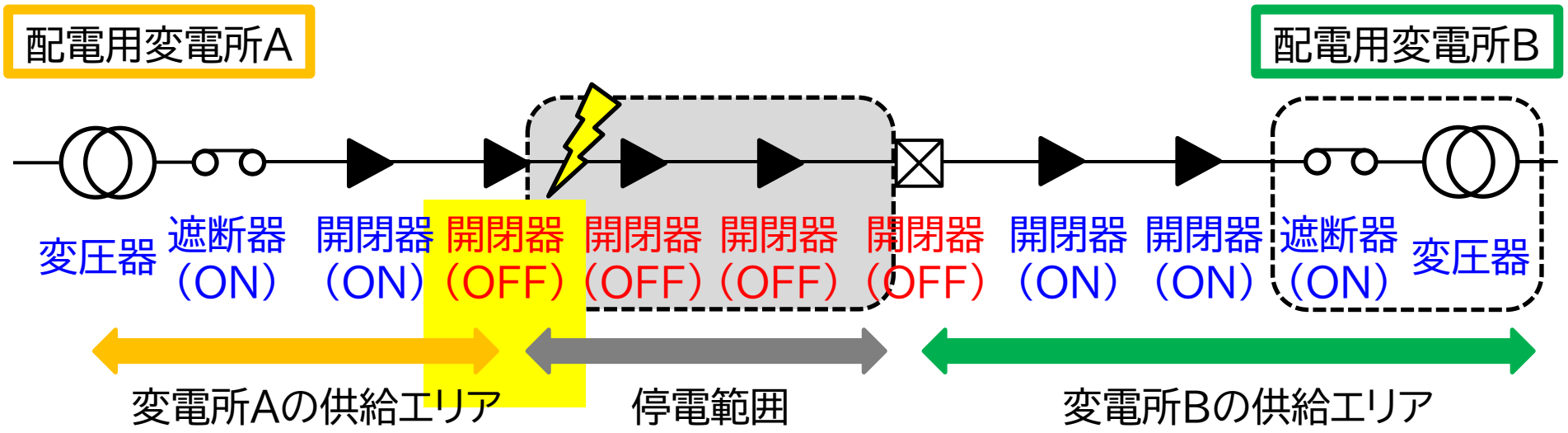
1. 平常時：
   - 電力の供給エリアが分かれている
     - 変電所Aのエリア
     - 変電所Bのエリア
   - 開閉器で線路が複数区間に分割されている

事故復旧動作(停電範囲の極小化)



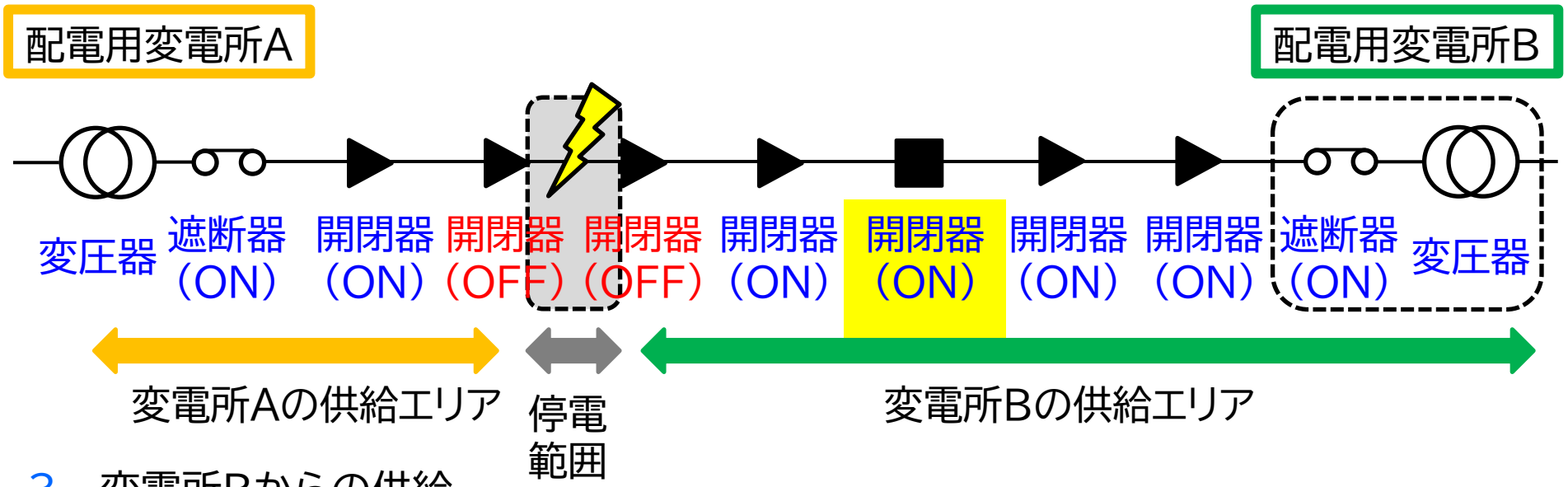
2. 事故発生時:
   - 遮断器が過電流を検出して遮断。変電所Aの供給エリアが停電。
   - 各開閉器をOFFに切替
   - 停電範囲を極小化するために, 電源側から順にスイッチを ON する

事故復旧動作(停電範囲の極小化)



3. 遮断器・開閉器の再投入：
   - 変電所遮断器を自動で再投入(ON)
   - 順次, 開閉器を投入(ON)
     - 時限順送と呼ばれる操作で, タイマーで順番に自動投入
       ⇒ 停電区間の極小化が進む
   - 【問題点】事故継続状態の場合, これ以降の開閉器をONにできない
   - さらに停電範囲を小さくするためには, 変電所Bから電力を供給

事故復旧動作(停電範囲の極小化)

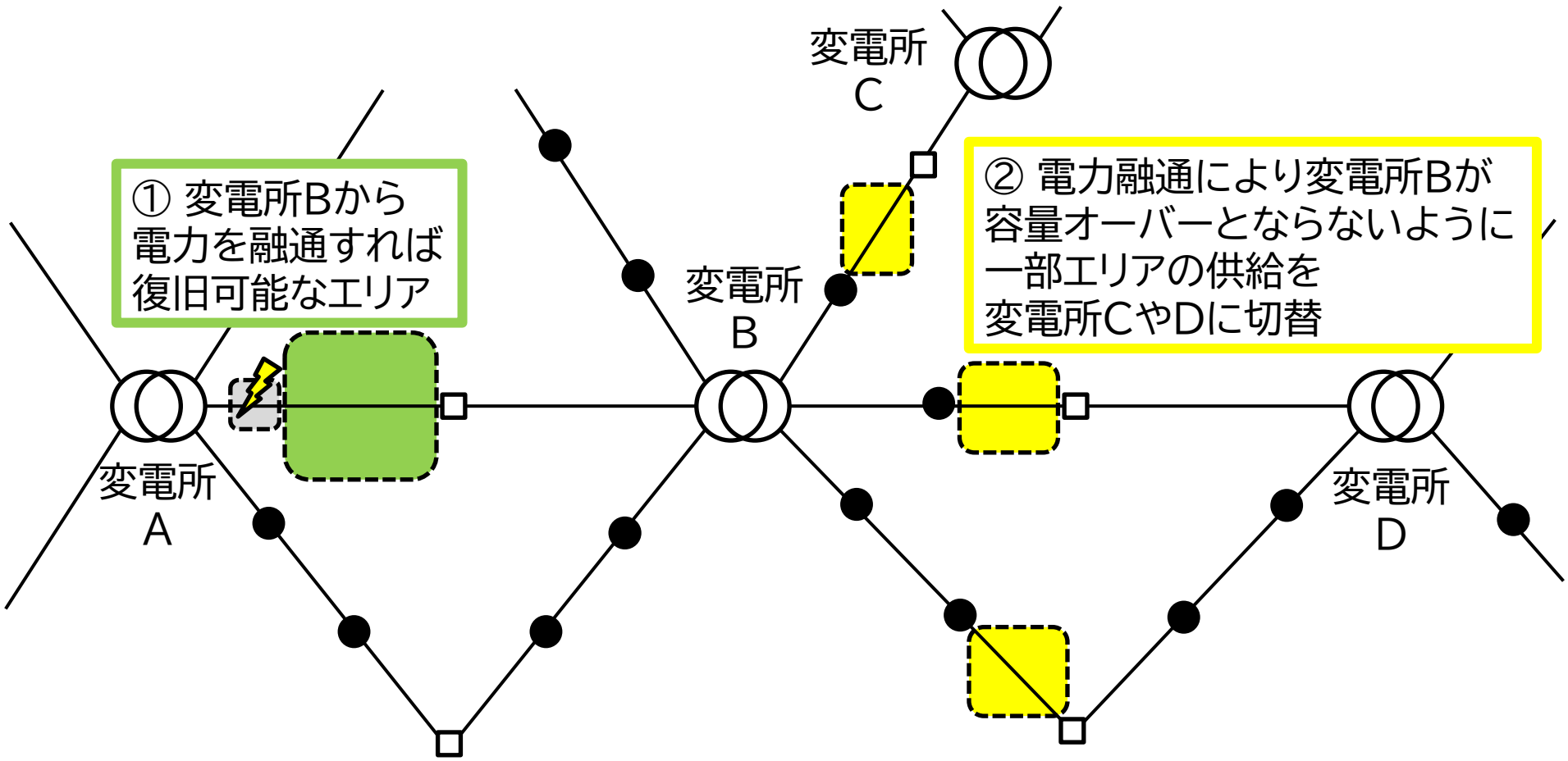


3. 変電所Bからの供給

- 通常時はOFFとなっている開閉器をONにする
  ⇒ 変電所Bからの電力供給が可能になる
- 順次，開閉器を投入(ON)
- 【ポイント】
  - 変電所Bの供給エリアが増えた分だけ，変電所Bとその供給エリア内の電線の負担が増える
  - 能力(定格容量)を超えないように調整しながら停電範囲を縮小する

組合せ遷移を適用

多段融通を実行する最短の切替手順



健全エリアへの電力供給を続けながら，
多段融通して停電復旧するためのシステムは存在しない
操作対象となる開閉器台数も多く，停電復旧まで多大な時間を必要としていた

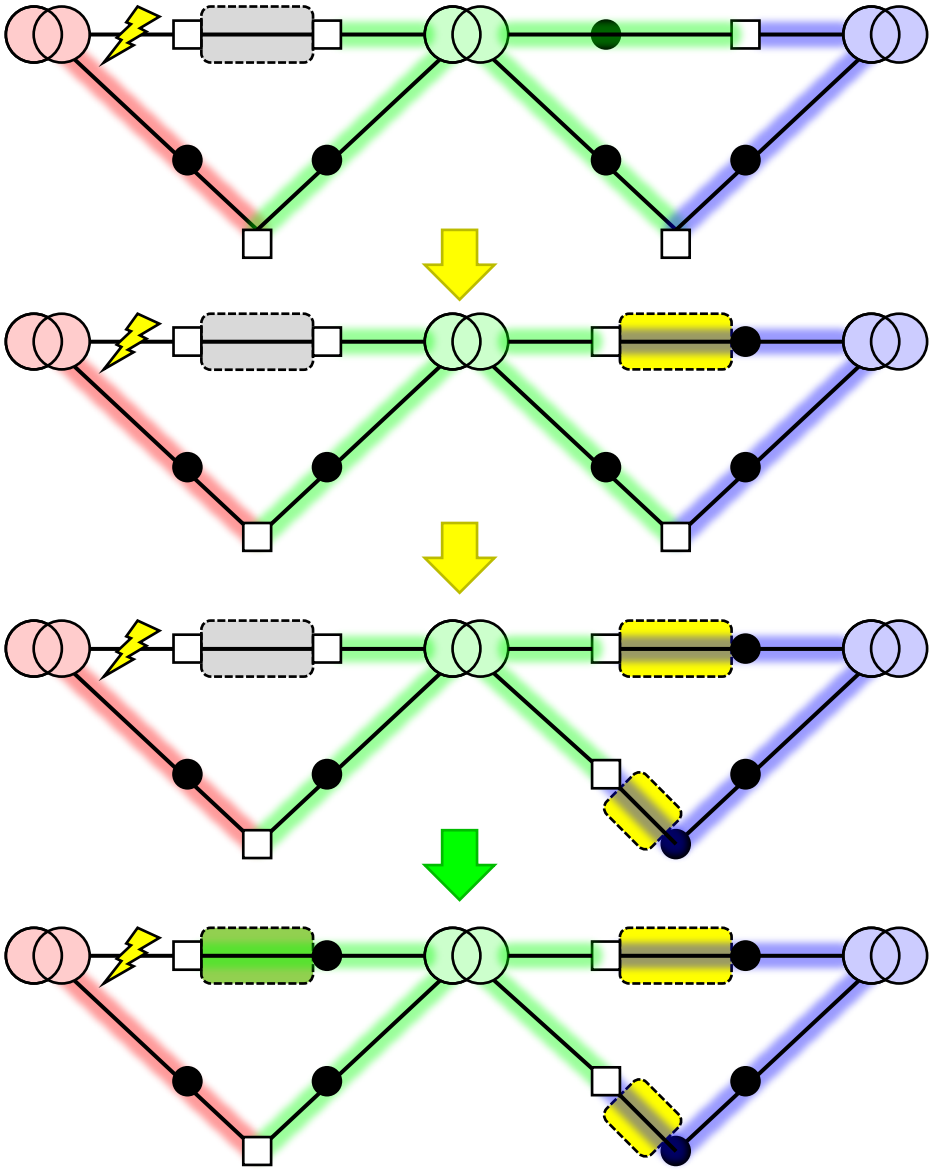
組合せ遷移の適用により最短の切り替え手順の算出に成功

アルゴリズムのポイント

停電復旧を実現する
切替手順を算出

1. 多段融通にも対応

2. 最短性を理論保証



私たちの貢献

従来手法

ZDDを用いて開閉状態を圧縮保持
（2014年井上ら）

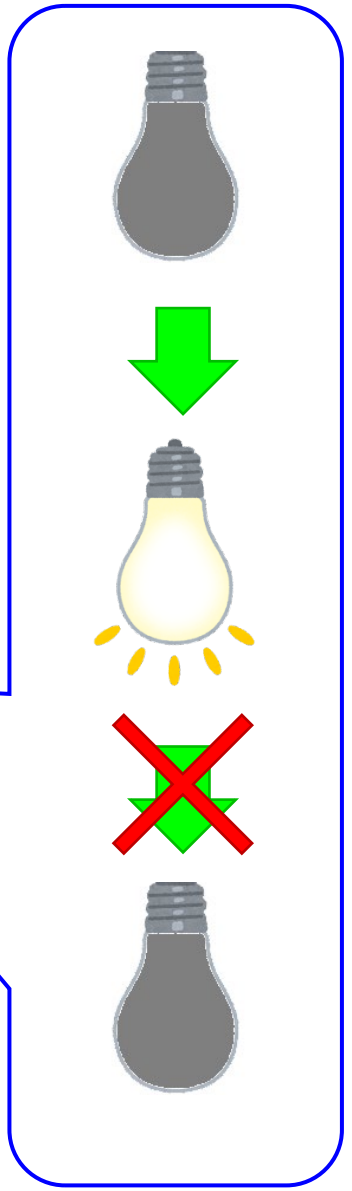
問題点

開閉状態は覚えられるが，
停電しているか否かを覚えられない．

一度停電を復旧させた箇所は，
再び停電させてはいけないという
実社会のルールに対応できない．

解決方法

開閉状態（辺）の情報だけでなく，
通電状況（点）の情報も保持するZDD



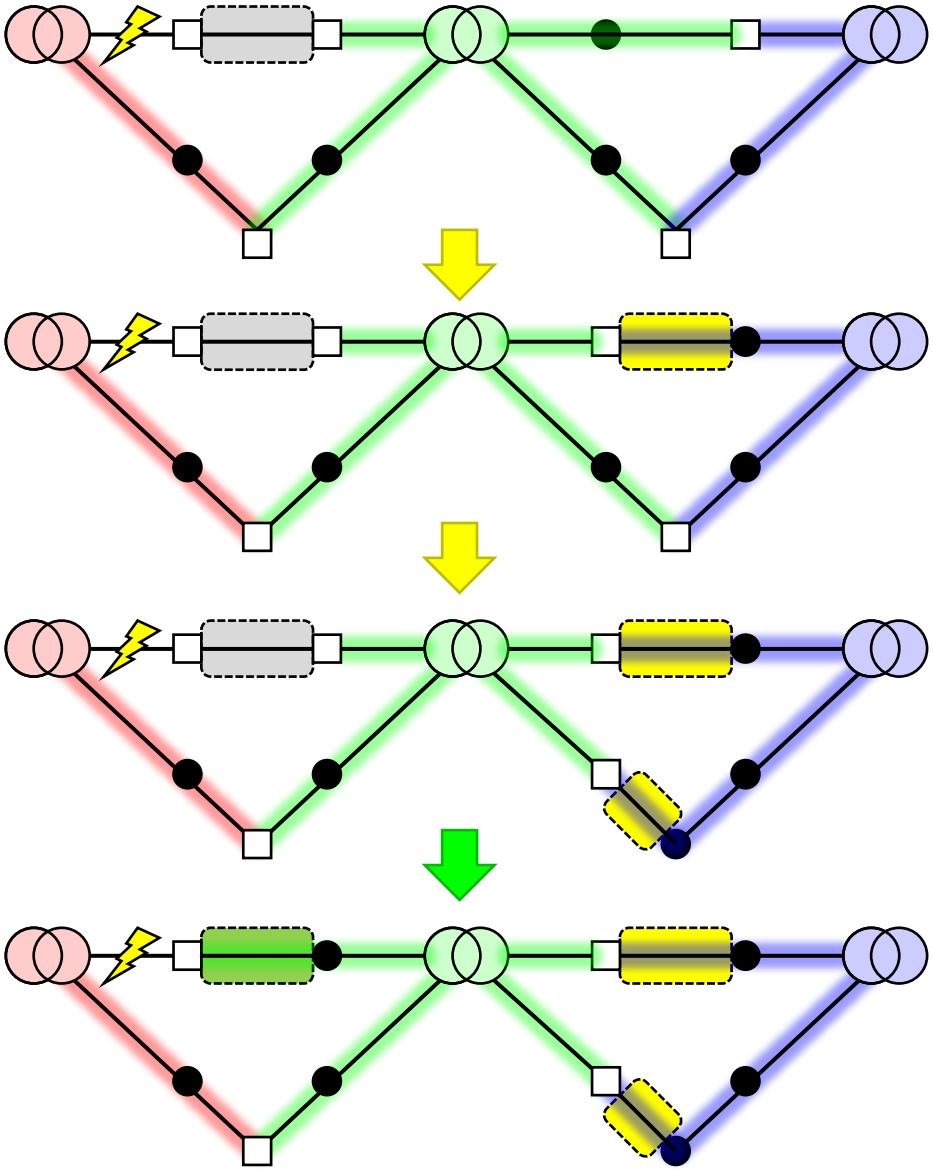
遷移ルール（許される操作）

・swap操作

開閉器を1つ閉じて，
別の開閉器を1つ開く操作

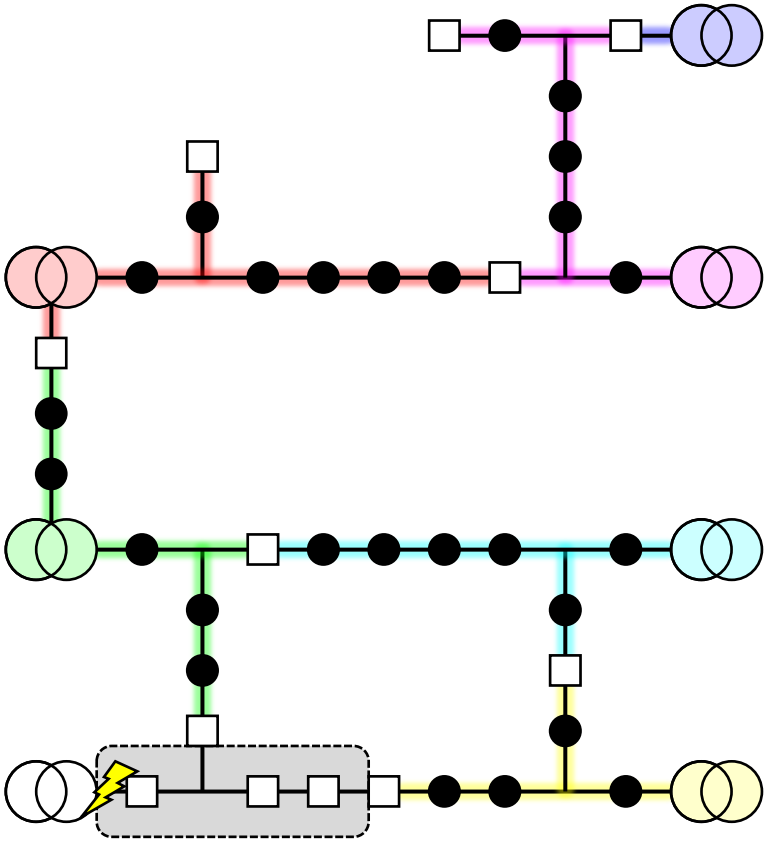
・add操作

開閉器を1つ閉じる操作

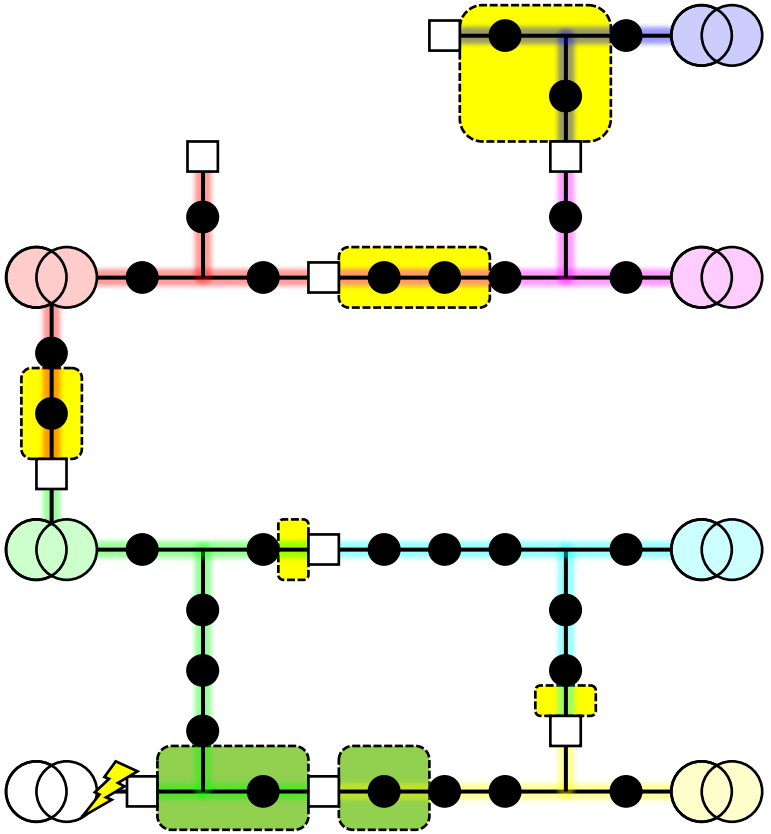


6段融通の例

停電発生時

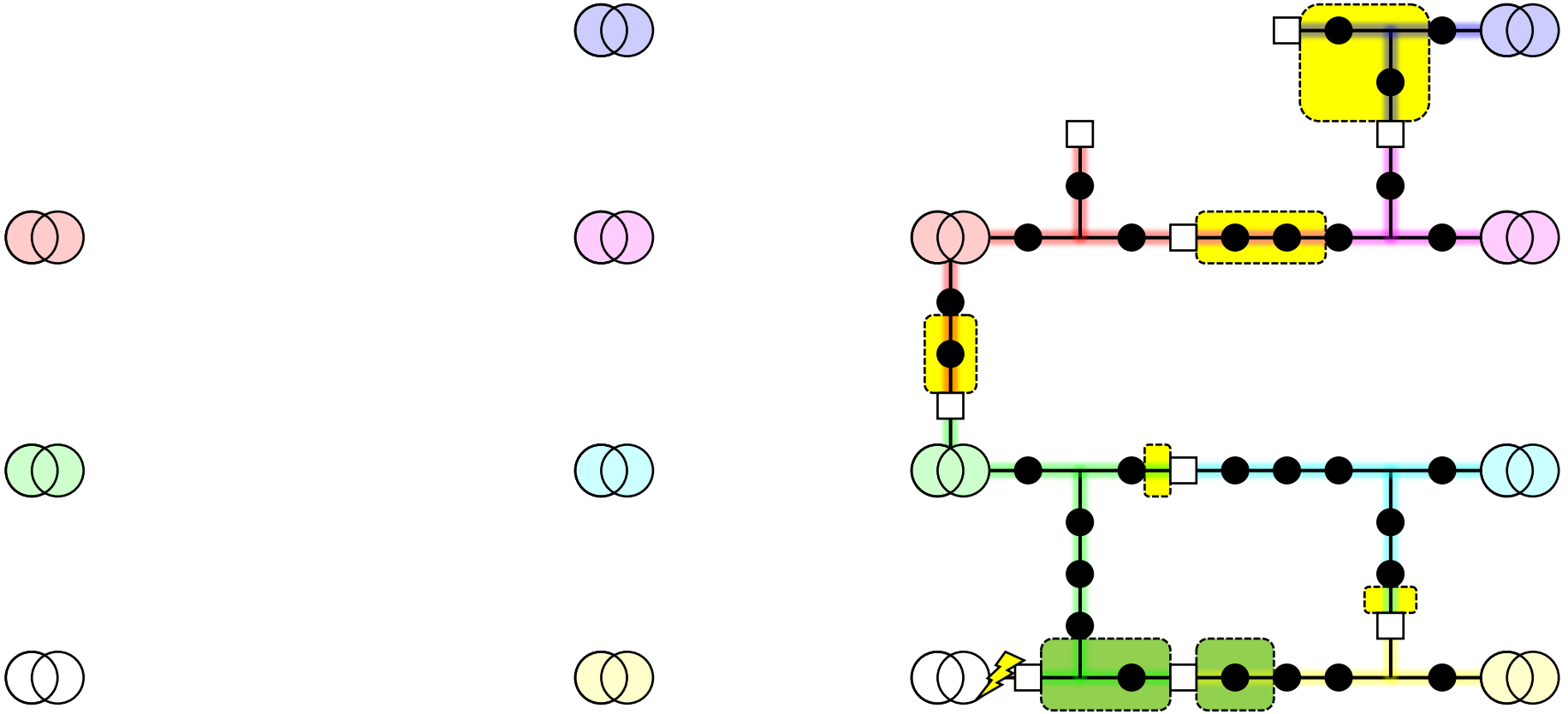


復旧後

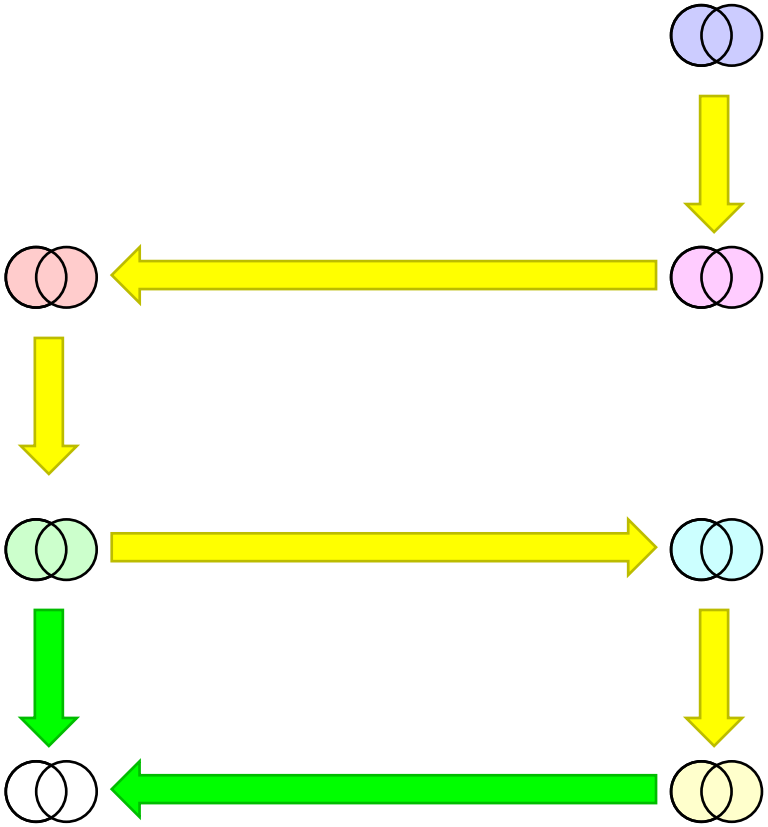


６段融通の例

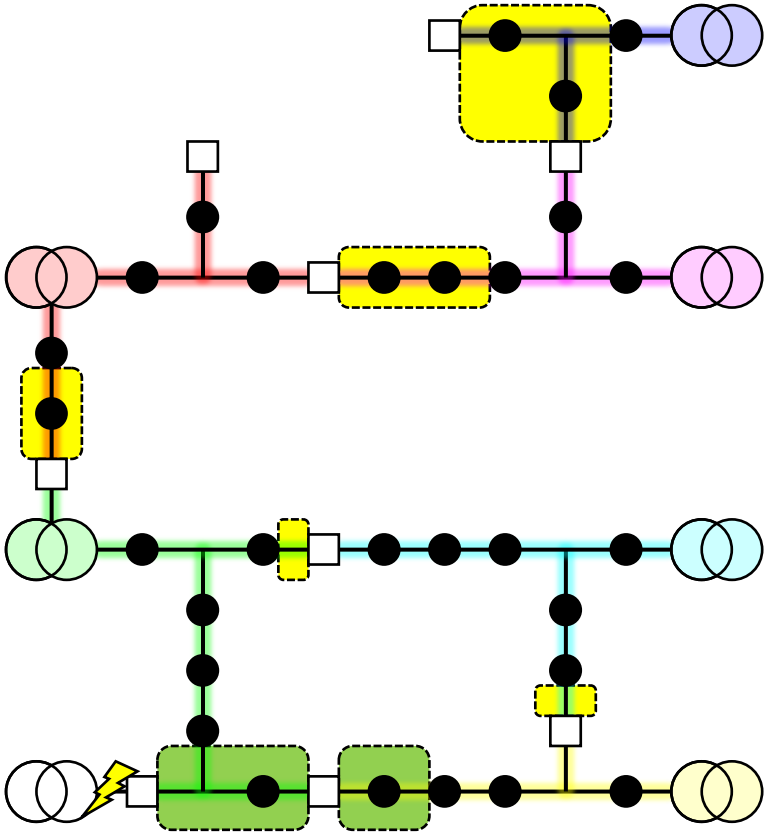
復旧後



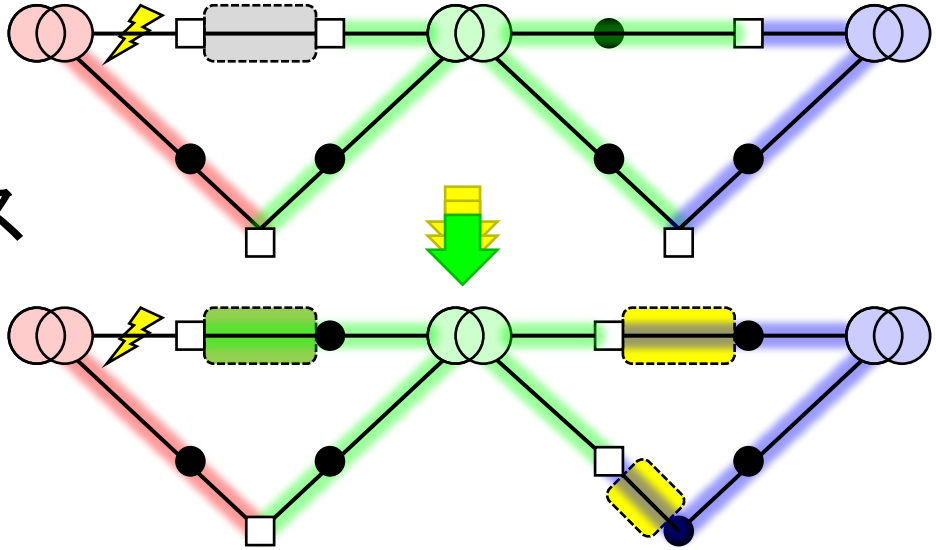
6段融通の例



復旧後



社会への発信

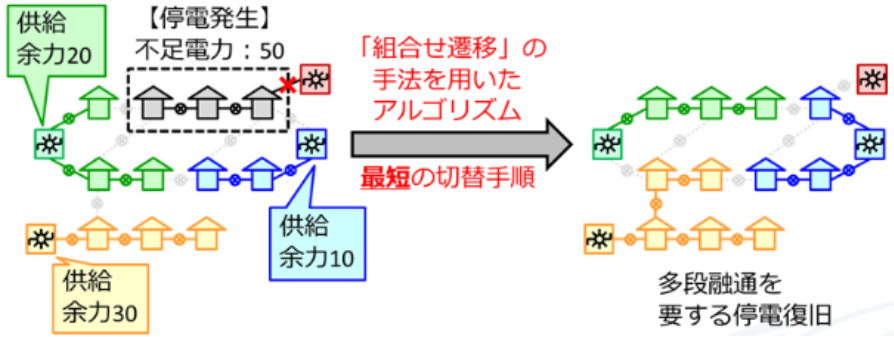


2022年11月7日　プレスリリース

2022年 | プレスリリース・研究成果

停電復旧の最短手順を算出するアルゴリズムを開発 多段融通にも対応、より広域な配電運用への活用に期待

2022年11月 7日 14:00 | プレスリリース・研究成果



- 特許共同出願中
- 2022年11月8日　電気新聞

川原 純	京都大学
山岡宙太	京都大学
伊藤健洋	東北大学
鈴木 顕	東北大学
飯岡大輔	中部大学
杉村修平	明電舎
後藤誠弥	明電舎
田邊隆之	明電舎

2023年3月17日　電気学会　全国大会
「停電復旧の最短手順を算出するアルゴリズム」

- 産学連携研究