組合せ爆発を乗り越える
最先端アルゴリズム技術とその実装

西野 正彬 (NTTコミュニケーション科学基礎研究所)

NTTデータ数理システムユーザコンファレンス

組合せ爆発を乗り越える
最先端アルゴリズム技術とその実装

西野 正彬 (NTTコミュニケーション科学基礎研究所)

NTTデータ数理システムユーザコンファレンス

自己紹介

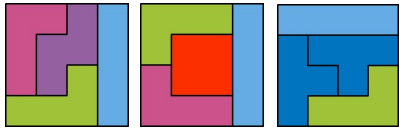
- 西野 正彬 (Masaaki Nishino)

- 日本電信電話株式会社
  NTT コミュニケーション科学基礎研究所
  特別研究員

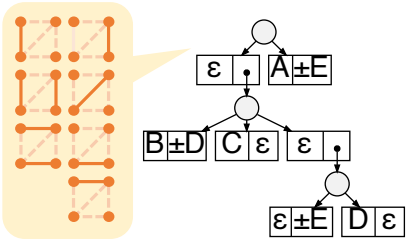
- アルゴリズム、自然言語処理の基礎研究に従事

本日のトピック

- 組合せを扱う最先端のアルゴリズム研究の紹介



- 基礎研究におけるアルゴリズムの実装において
  NTTデータ数理システム社に
  どのように協力いただいているかの紹介



アウトライン

1. アルゴリズムとは
2. 組合せ爆発を乗り越えるアルゴリズム技術
3. アルゴリズムの実装とその価値

アルゴリズムとは

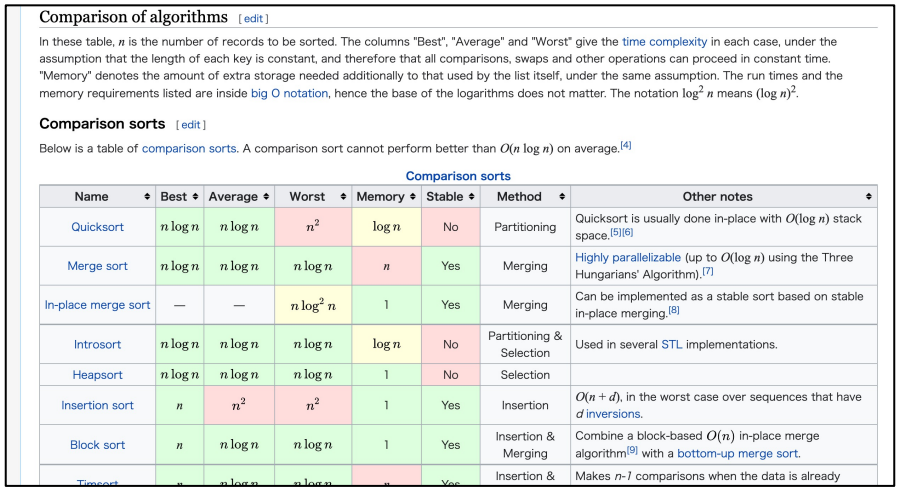
計算機を用いて問題を解くための計算手続き

様々な問題に対して、その問題を解くためのアルゴリズムが存在

例：ソート（並べ替え）問題

アルゴリズムの特徴１

- １つの問題を解くための方法が多数存在
  - Wikipediaには40種類以上のソートアルゴリズムが掲載
- アルゴリズム間の違い
  - 実行時間
  - 使用メモリ量



Comparison of algorithms [edit]

In these table, n is the number of records to be sorted. The columns "Best", "Average" and "Worst" give the time complexity in each case, under the assumption that the length of each key is constant, and therefore that all comparisons, swaps and other operations can proceed in constant time. "Memory" denotes the amount of extra storage needed additionally to that used by the list itself, under the same assumption. The run times and the memory requirements listed are inside big O notation, hence the base of the logarithms does not matter. The notation $\log^2 n$ means $(\log n)^2$.

Comparison sorts [edit]

Below is a table of comparison sorts. A comparison sort cannot perform better than $O(n \log n)$ on average.[4]

Comparison sorts

Name	Best	Average	Worst	Memory	Stable	Method	Other notes
Quicksort	$n \log n$	$n \log n$	n^2	$\log n$	No	Partitioning	Quicksort is usually done in-place with $O(\log n)$ stack space.[5][6]
Merge sort	$n \log n$	$n \log n$	$n \log n$	n	Yes	Merging	Highly parallelizable (up to $O(\log n)$ using the Three Hungarians' Algorithm).[7]
In-place merge sort	—	—	$n \log^2 n$	1	Yes	Merging	Can be implemented as a stable sort based on stable in-place merging.[8]
Introsort	$n \log n$	$n \log n$	$n \log n$	$\log n$	No	Partitioning & Selection	Used in several STL implementations.
Heapsort	$n \log n$	$n \log n$	$n \log n$	1	No	Selection	
Insertion sort	n	n^2	n^2	1	Yes	Insertion	$O(n + d)$, in the worst case over sequences that have d inversions.
Block sort	n	$n \log n$	$n \log n$	1	Yes	Insertion & Merging	Combine a block-based $O(n)$ in-place merge algorithm[9] with a bottom-up merge sort.
Timsort	n	$n \log n$	$n \log n$	n	Yes	Insertion &	Makes n-1 comparisons when the data is already

https://en.wikipedia.org/wiki/Sorting_algorithm

アルゴリズムの違いが１万倍以上の速度差を生むことも

アルゴリズムの特徴２

- 汎用的
  - １つのアルゴリズムが様々な場面で用いられる
  - アルゴリズムを改善することで、幅広い場面で
    計算の高速化や消費エネルギーの削減などに貢献

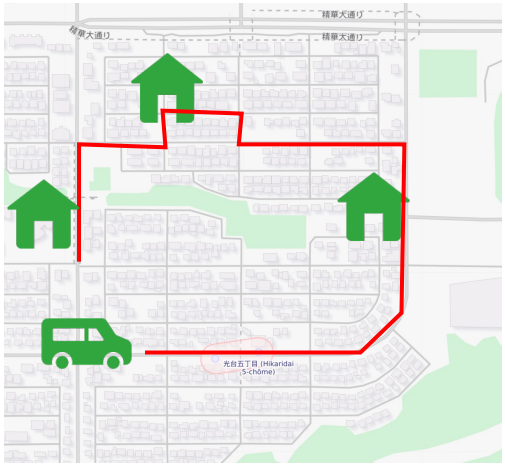
効率的なアルゴリズムを考案することで
広く世の中に貢献できる

アウトライン

1. アルゴリズムとは
2. 組合せ爆発を乗り越えるアルゴリズム技術
3. アルゴリズムの実装とその価値

組合せアルゴリズムとは

- 離散的な対象の組合せを扱う問題を解くアルゴリズム



最適な経路の発見

https://www.openstreetmap.org/

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

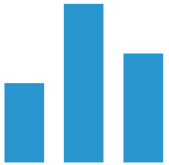
パズル

https://ja.wikipedia.org/wiki/%E6%95%B0%E7%8B%AC

組合せ問題の例

制約充足
条件を満たす組合せ
があるかを調べる

最適化
条件を満たす
もっともよい
組合せを見つける



数え上げ
条件を満たす
組合せの数を
数える

組合せ爆発のため計算が困難になりやすい

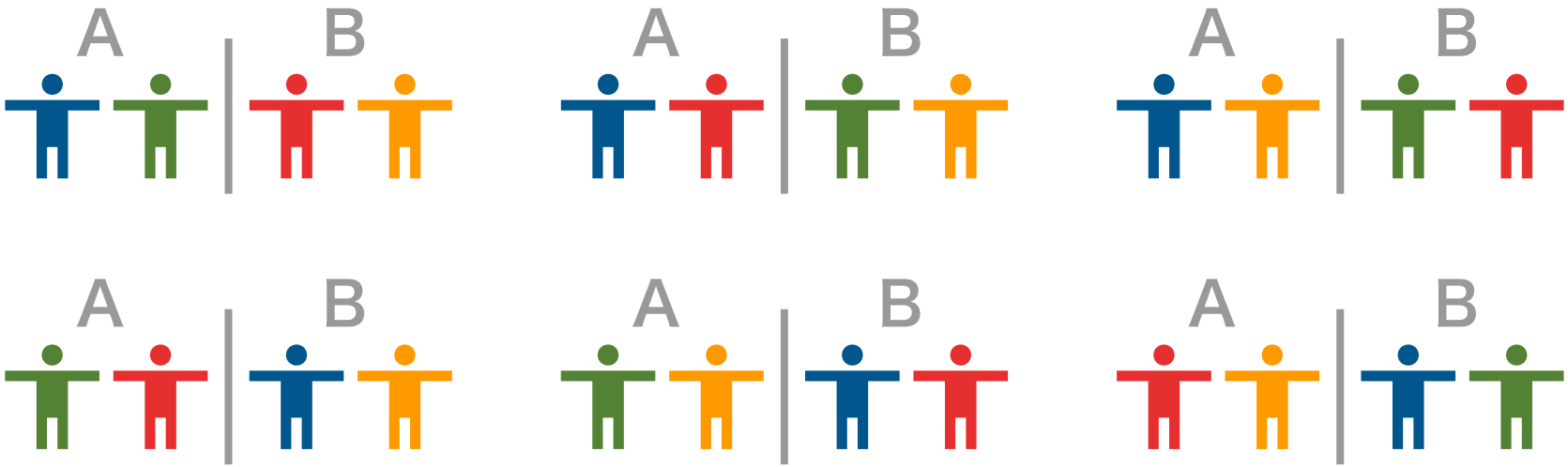
グループ分けは何通り？

4人の従業員を2人ずつの２つのグループ
（A組、B組）に分ける方法は何通り？

答え

NTT

6通り



グループ分けは何通り？

10人の従業員を5人ずつの２つのグループ
（A組、B組）に分ける方法は何通り？

答え

252通り

従業員が増えると…

126,410,606,437,752 通り
（126兆）

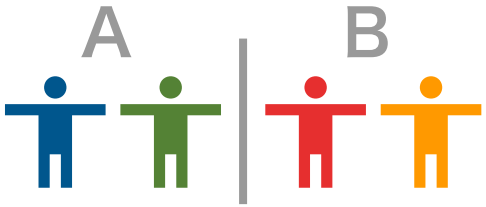
100,891,344,545,564,
193,334,812,497,256 通り

グループ分けの種類数が急増

組合せ爆発とは

問題サイズに対して組合せの数が

指数的に増加する現象



グループ分け

移動経路

並べ替え

普遍的な現象

組合せ爆発の困難さ

- 多くの問題で、問題を解くのに組合せの
  総数に比例した計算時間が必要

- 問題サイズが少し大きくなっただけで、現実的な時間で
  問題を解くことが困難になる

組合せ爆発を避ける

全ての組合せを扱わずに問題を解く

一般的な手法

組合せ爆発を乗り超える

膨大な数の組合せを**小さく圧縮**することで問題を解く

私達が研究している手法

圧縮に基づくアプローチの利点

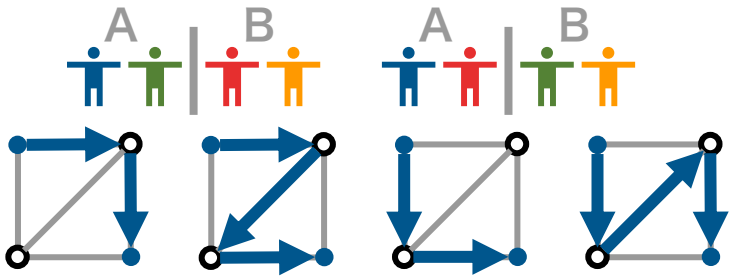
組合せを全て保持している

- 条件に沿った組合せを効率的に取り出すなどの操作が柔軟に実行可能
- 組合せに関する正確な値を算出できる

組合せ集合

- **組合せ**を要素とする**集合**
  - 1, 2, 3の組合せ集合:　　{{1,2}, {1,3}, {2,3}},　{{1}, {1,2,3}}

- さまざまなものが組合せ集合として表現できる
  - グループ分けの集合
  - 移動経路の集合



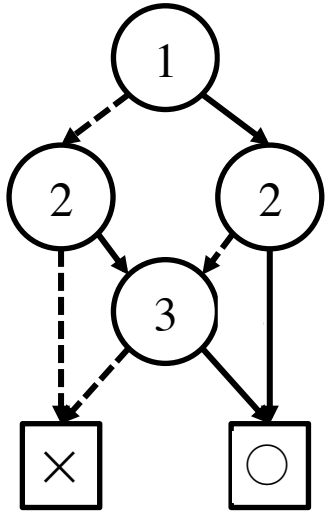
二分決定グラフ

組合せ集合を**グラフ**として表現する方法

（頂点と辺からなる図形）

{{1,2}, {1,3}, {2,3}}

組合せ集合



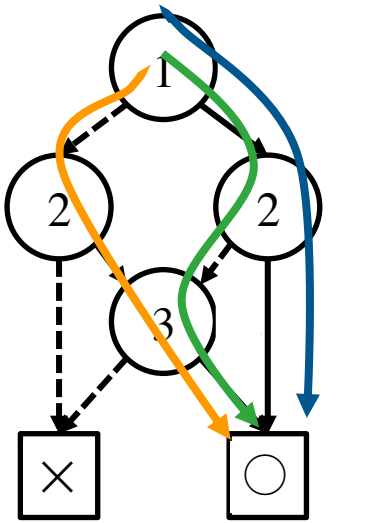
二分決定グラフ

二分決定グラフの解釈

各組合せが根から◯までの経路に対応する

{{1,2}, {1,3}, {2,3}}

組合せ集合



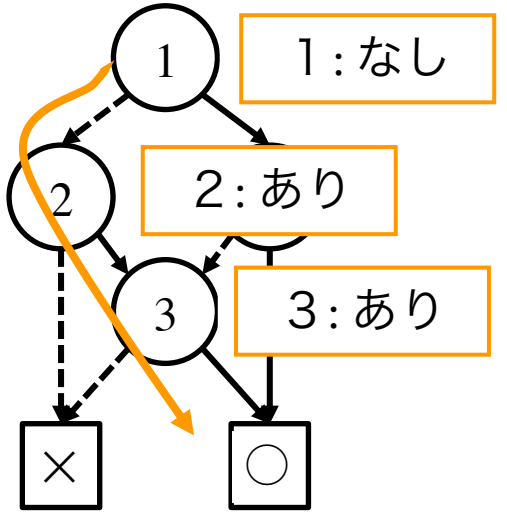
二分決定グラフ

二分決定グラフの解釈

破線は要素が「ない」ことを、実線は要素が「ある」ことを表す

{{1,2}, {1,3}, {2,3}}

組合せ集合



二分決定グラフ

二分決定グラフの特徴

1. 組合せ集合を**圧縮して表現**できる

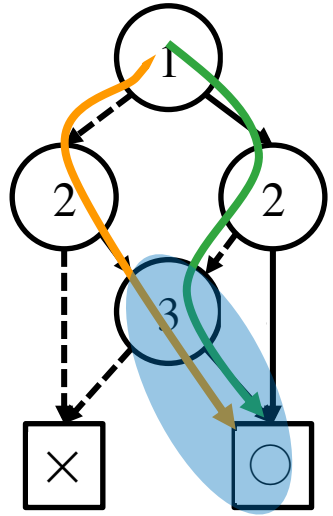
2. **圧縮したまま**様々な計算を実行できる

圧縮して表現する

共通な要素に対応する箇所をまとめることで
小さく表現できる

{{1,2}, {1,3}, {2,3}}

組合せ集合



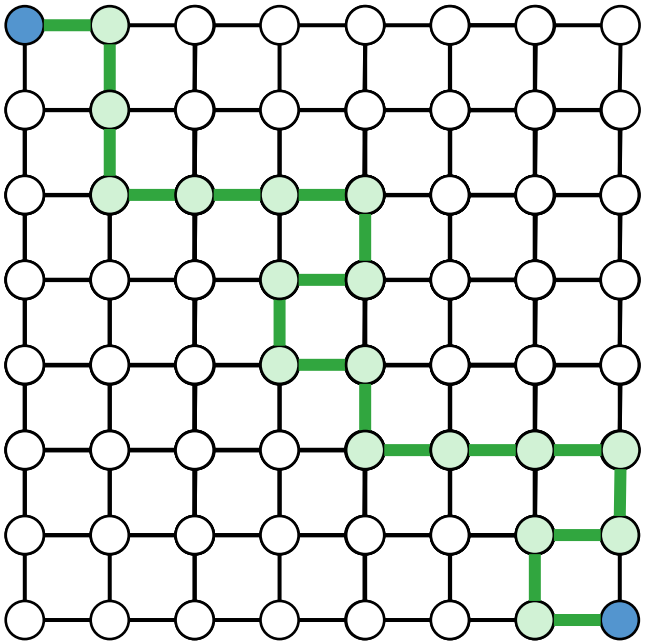
3をまとめて
圧縮

二分決定グラフ

組合せ爆発を圧縮する

同じ点を二度通らないスタートからゴールまでの経路

スタート



ゴール

経路の総数

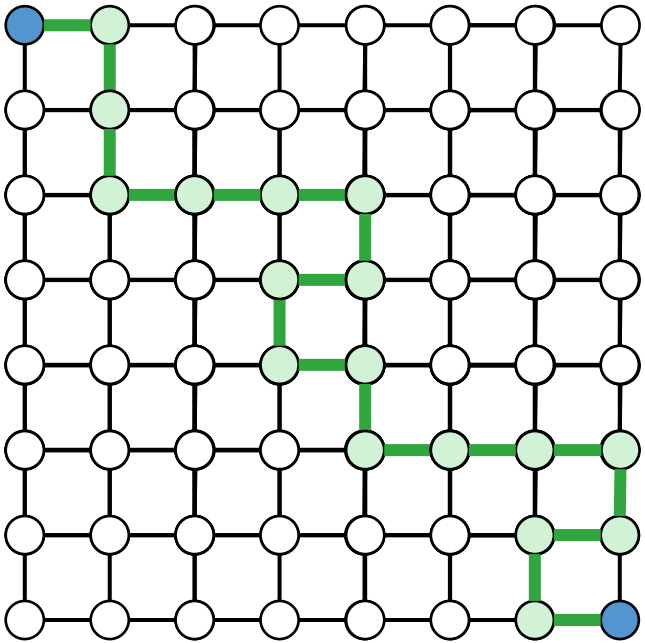
789,360,053,252通り

組合せ爆発！

組合せ爆発を圧縮する

同じ点を二度通らないスタートからゴールまでの経路

スタート



ゴール

経路の総数

789,360,053,252通り

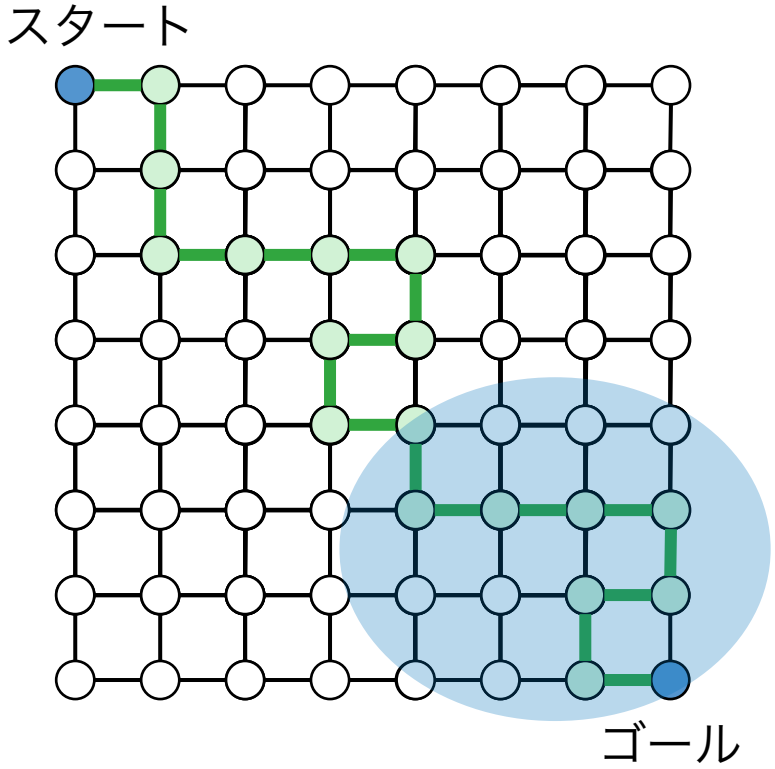
二分決定グラフの頂点数

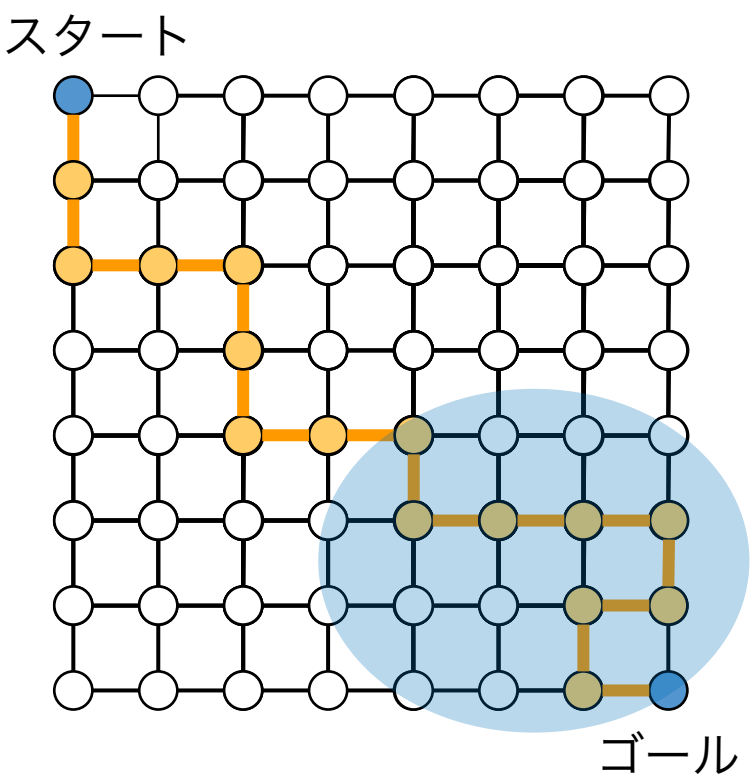
33,580個

大幅に圧縮！

組合せ爆発を圧縮する

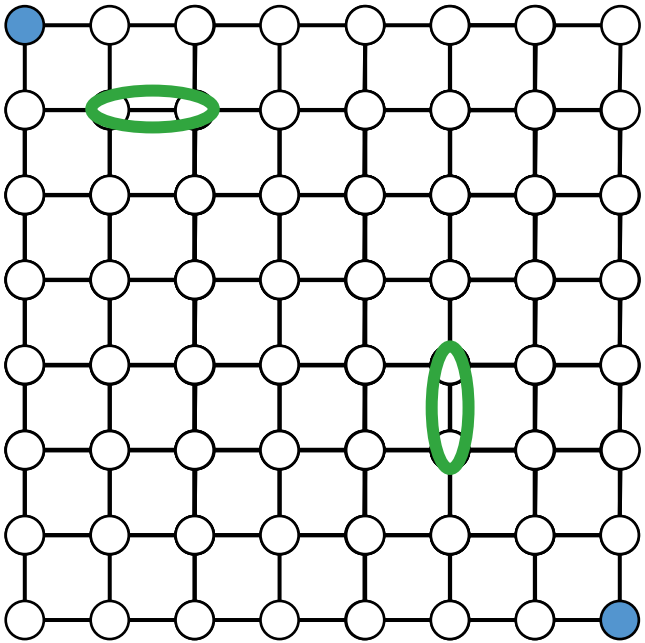
似た経路が無数に存在 → 二分決定グラフで効果的に圧縮できる





圧縮して計算

スタート



ゴール

特定の辺を通る経路の数を求めたい

素朴な方法：経路の総数

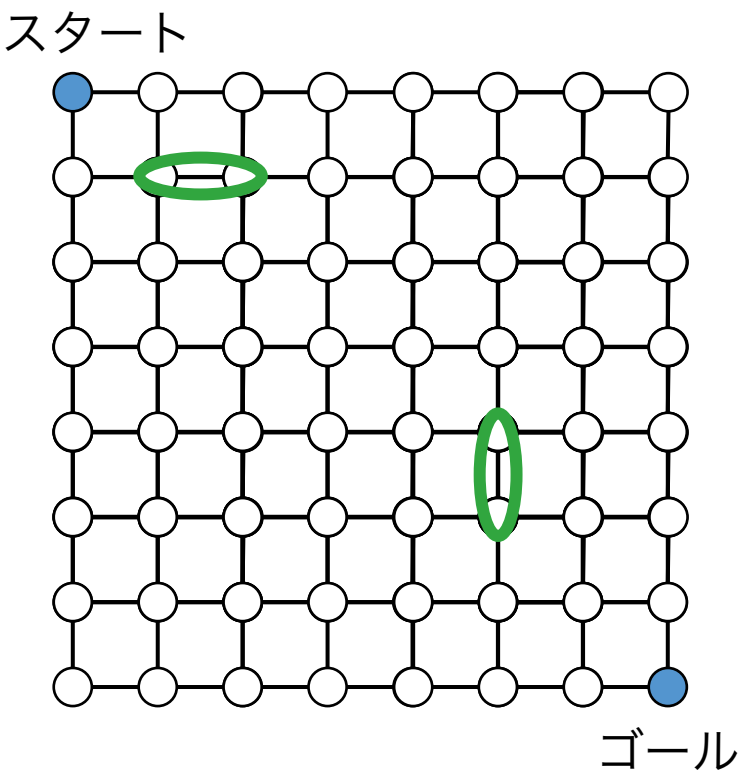
（789,360,053,252通り）

に比例する時間

組合せ爆発！

計算に膨大な時間がかかる

圧縮して計算



特定の辺を通る経路の数を求めたい

素朴な方法：経路の総数
（789,360,053,252通り）
に比例する時間

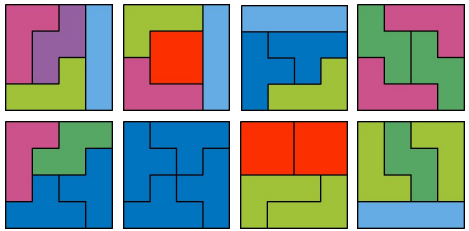
二分決定グラフの頂点数
（33,580個）
に比例する時間で計算可能

現実的な時間で計算可能に！

研究事例紹介

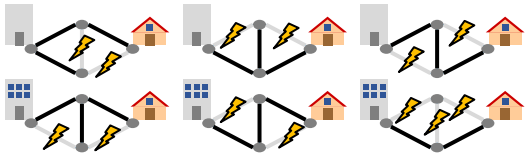
- 敷き詰め方の列挙

  （計算の大幅な高速化、柔軟な処理）



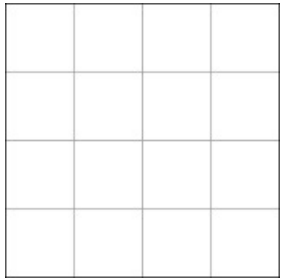
- 故障に強いネットワークの設計

  （計算の大幅な高速化）

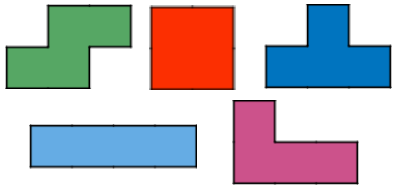


敷き詰め問題

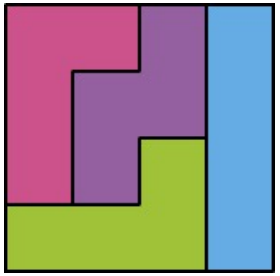
盤面に隙間なくピースを敷き詰める方法を見つける問題



盤面



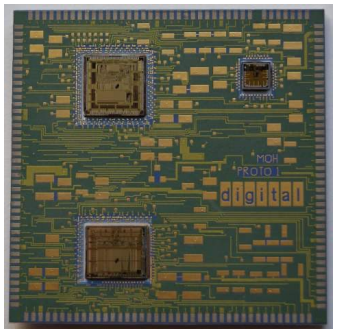
利用可能なピース



敷き詰めかたの例

敷き詰め問題の例

様々な現実の問題が敷き詰め問題として解ける



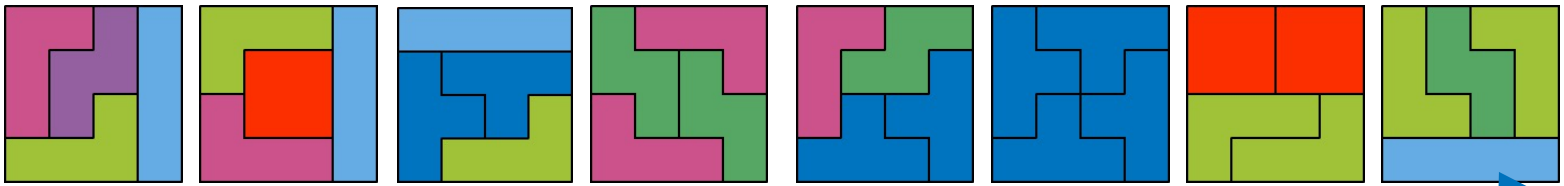
電子回路上の
モジュール配置

マンションの
間取り設計

https://ja.wikipedia.org/wiki/%E9%9B%86%E7%A9%8D%E5%9B%9E%E8%B7%AF#/media/%E3%83%95%E3%82%A1%E3%82%A4%E3%83%AB:Dec_alpha_small.JPG

問題の難しさ

- 解をひとつ見つけるだけでも難しい **(NP完全)**
- 膨大な数の解が存在する



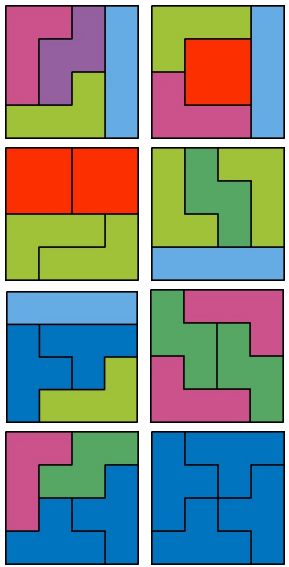
盤面サイズ	解の総数
4x4	117 通り
8x8	19,077,209,438 通り
12x12	13,664,822,582,333,502,156,627,512 通り

組合せ爆発！

敷き詰め方の列挙

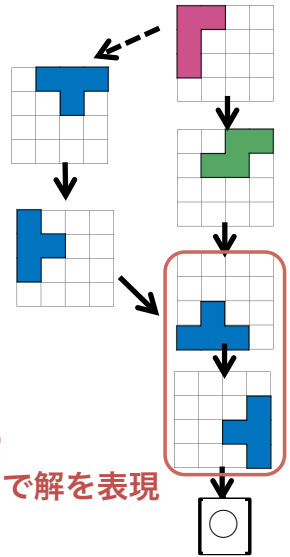
二分決定グラフを用いて敷き詰め方を全て見つける方法を開発

従来法：
敷き詰め方を順番に見つける



提案法：
敷き詰め方の集合を表す
二分決定グラフを作る

敷き詰め方は
ピースの置き方の
組合せとして表現



共通部分を圧縮
→小さいグラフで解を表現

手法の特徴

特徴1: 高速

例: ピース敷き詰め (8x8マス) の答えの列挙

既存法	提案法
16475秒	0.88秒

190億通りの解を1秒以内にすべて発見

なぜ高速？

ピース敷き詰め (10x10マス)

敷き詰め方の総数：　72,713,560,548,906,621通り
二分決定グラフの頂点数：　16,476,396個

既存法: **敷き詰め方の総数**に比例する計算

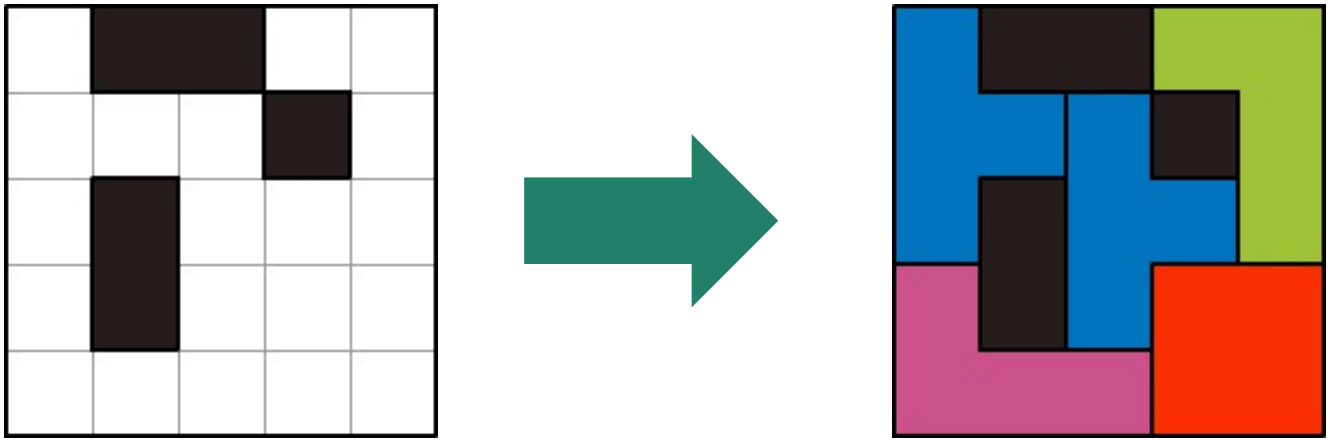
提案法: 二分決定グラフの**頂点数**に比例する計算

圧縮することで大幅な高速化を実現

手法の特徴

特徴２: 解をすべて圧縮して保存

条件に応じて適切な解を素早く取り出すことができる



デモ

マンションの間取りデザイン

- 3LDKのマンションの間取りをすべて列挙
- 条件に合わせて間取りを高速に絞り込むことができる

ネットワークの信頼性とは

電線が事故や災害の影響で切れるときに
通信可能である確率
故障に対する強さを表す

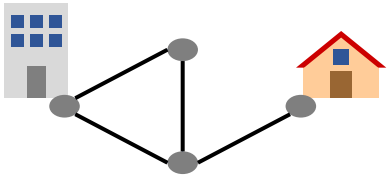
例：各電線が20%の確率で切れるときの信頼性

信頼性 80%

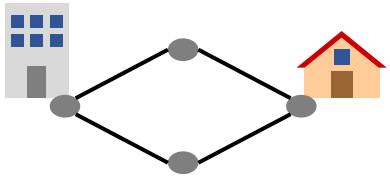
信頼性 96%

信頼性最大化問題

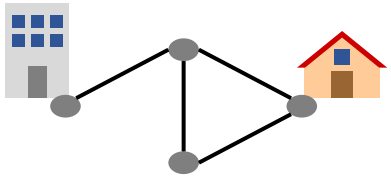
予算の範囲内で、信頼性を最大にする
電線の配置方法を見つける問題



信頼性 74%



信頼性 87%

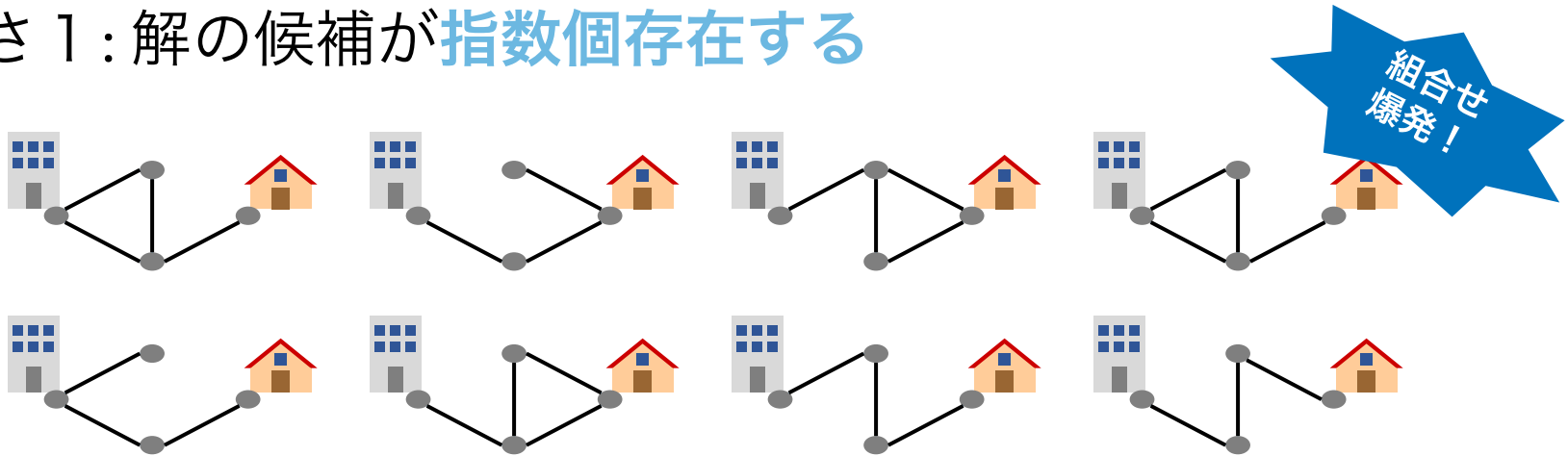


信頼性 74%

信頼性最大化問題と組合せ爆発

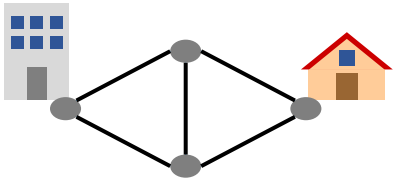
信頼性最大化問題は難しい

難しさ１：解の候補が**指数個存在する**

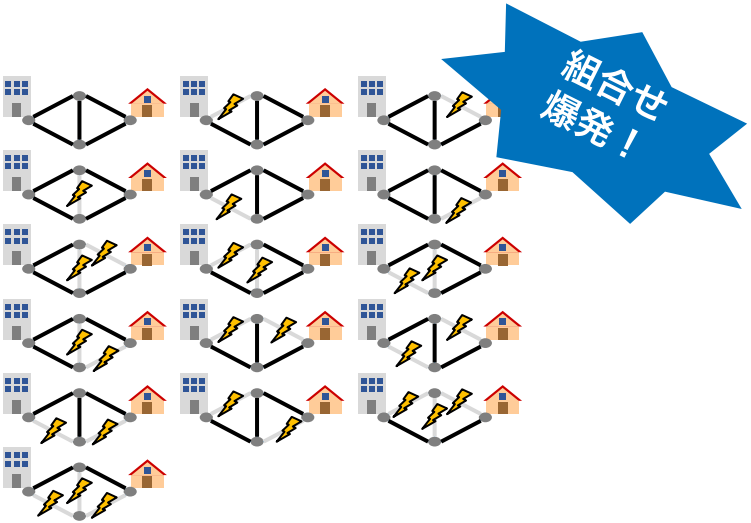


信頼性最大化問題と組合せ爆発

難しさ２：信頼性を調べるために指数通りの故障パターンを
調べ上げる必要がある



解の候補



故障パターン

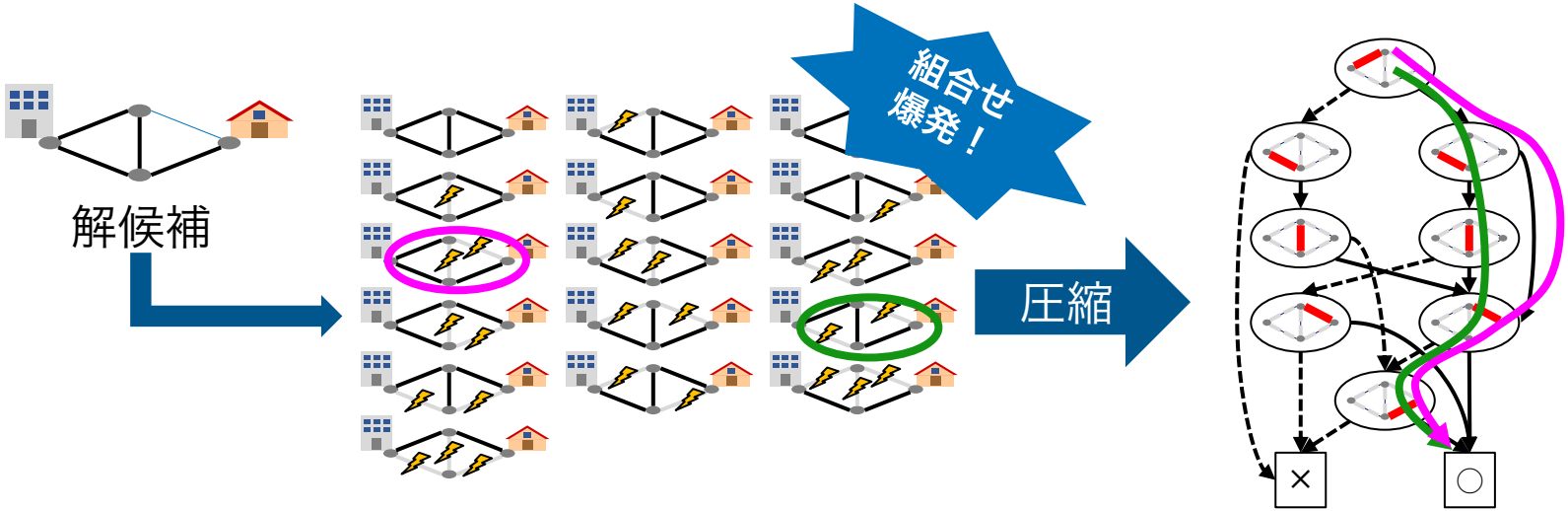
信頼性最大化と組合せ爆発

- 信頼性最大化問題を解くためには二重の組合せ爆発に対処する必要がある
  - 解の候補数
  - 解ごとの故障パターン数

最適解を求める効率的な手法は知られていなかった

二分決定グラフを使った解法

二分決定グラフを用いて故障パターンの集合を表現し、
その上で探索を行うことで、信頼性最大化問題を解く



通信可能なすべての
故障パターン
（16通り）

すべての故障パターンを
表す二分決定グラフ
（10頂点）

提案法の特徴

- 従来より大規模なネットワークに対する最適解を求めることができる
  - 従来法:　10拠点程度
  - 提案法: 100拠点以上

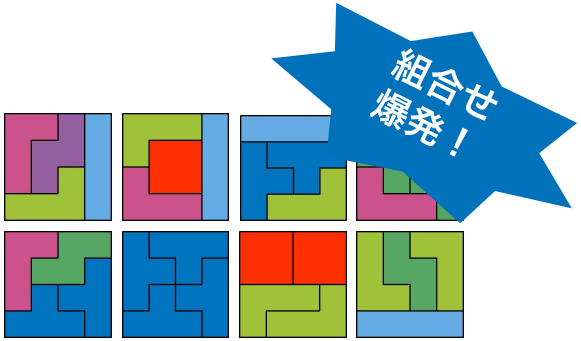
- 従来法より10万倍以上高速

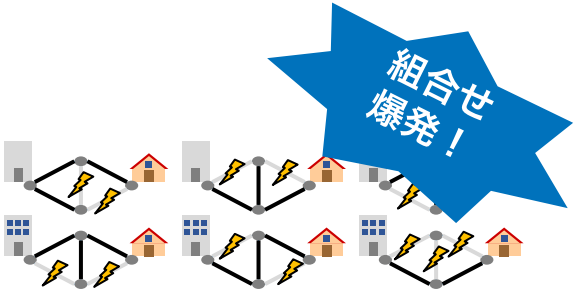
デモ

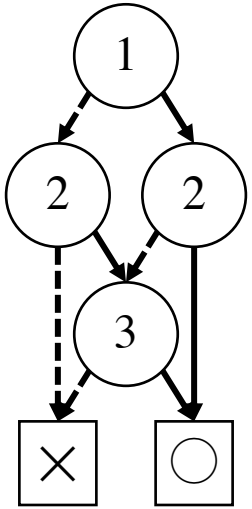
鉄道網の設計

・線路が故障しても、すべての駅に行き来できる確率が最大になるように線路を敷設する

ここまでのまとめ







二分決定グラフを用いて
組合せ爆発に対処

指数的に増大する対象を扱う
問題を解くことができる

アウトライン

1. アルゴリズムとは
2. 組合せ爆発を乗り越えるアルゴリズム技術
3. アルゴリズムの実装とその価値

アルゴリズム研究と実装

- アルゴリズムの実装は非常に重要
  - アルゴリズムの性能評価
  - 応用研究の推進

例：深層学習

強力なフレームワークが深層学習技術の
急速な発展を下支え

TensorFlow

PyTorch

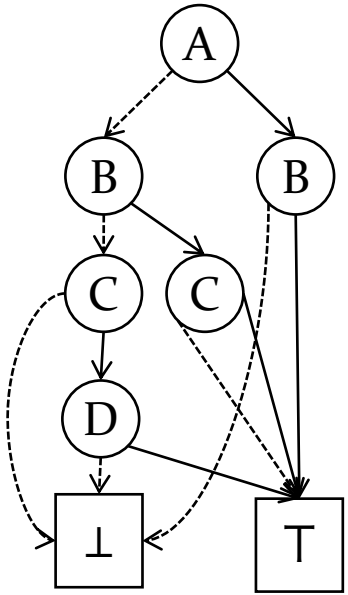
NumPy

mxnet

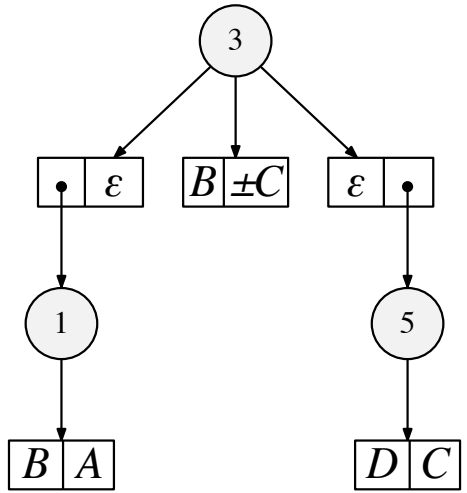
Keras

決定グラフ処理系の実装

2016年に新しい決定グラフであるZSDDを考案



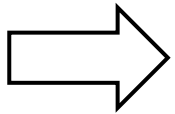
ZDD
（既存の決定グラフ）



ZSDD
（新しい決定グラフ）

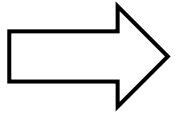
ZSDDの特徴

長所：
- ZDD（既存の決定グラフ）よりも高圧縮
- ZDDと互換性がある



多くの課題で
性能UPが見込める

短所：
- ZDDよりも複雑
- 理解しにくい
- 実装が難しい



実用における
障壁

ZSDD処理系実装までの経緯

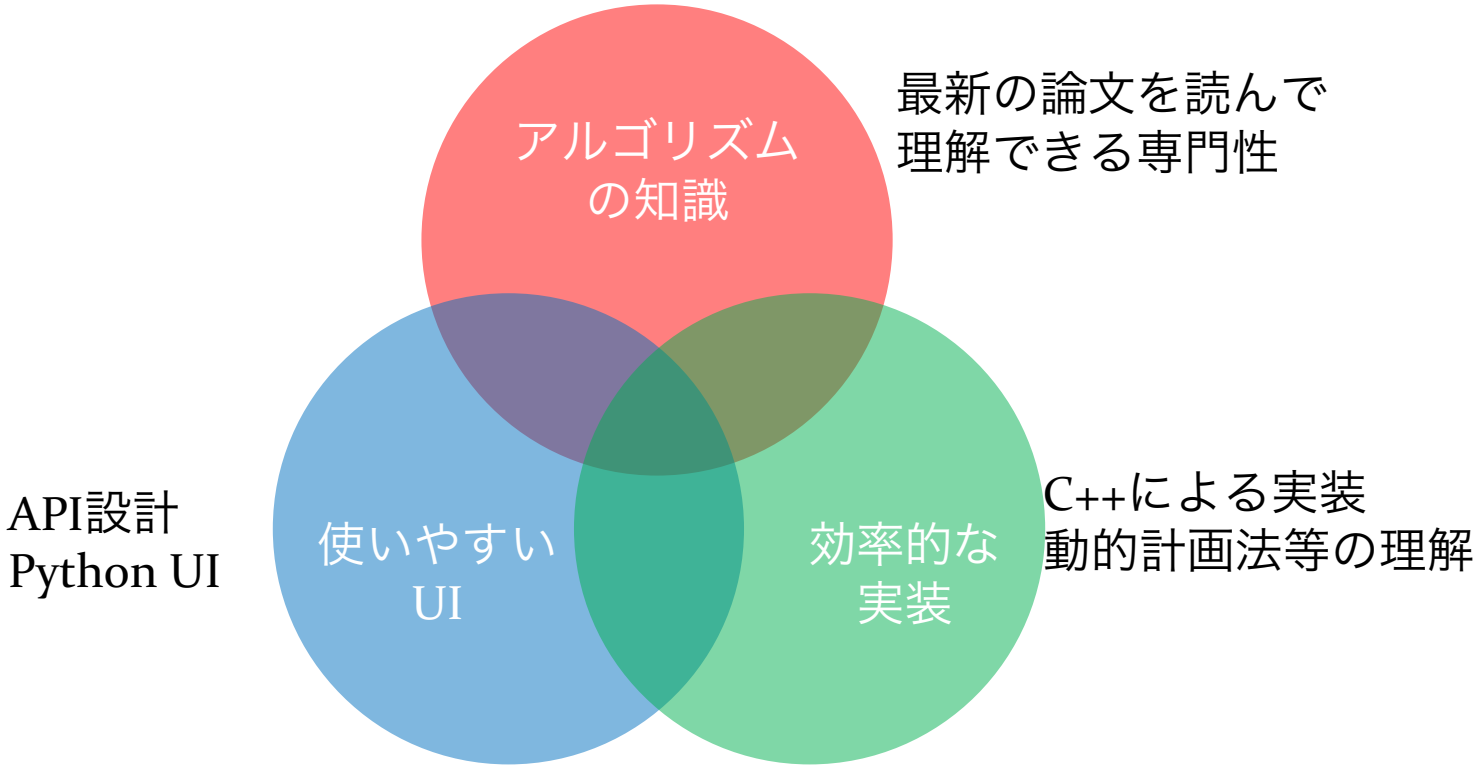
NTT

- ZSDDを考案したものの、プログラムは検証用に作成したもののみ（C++）

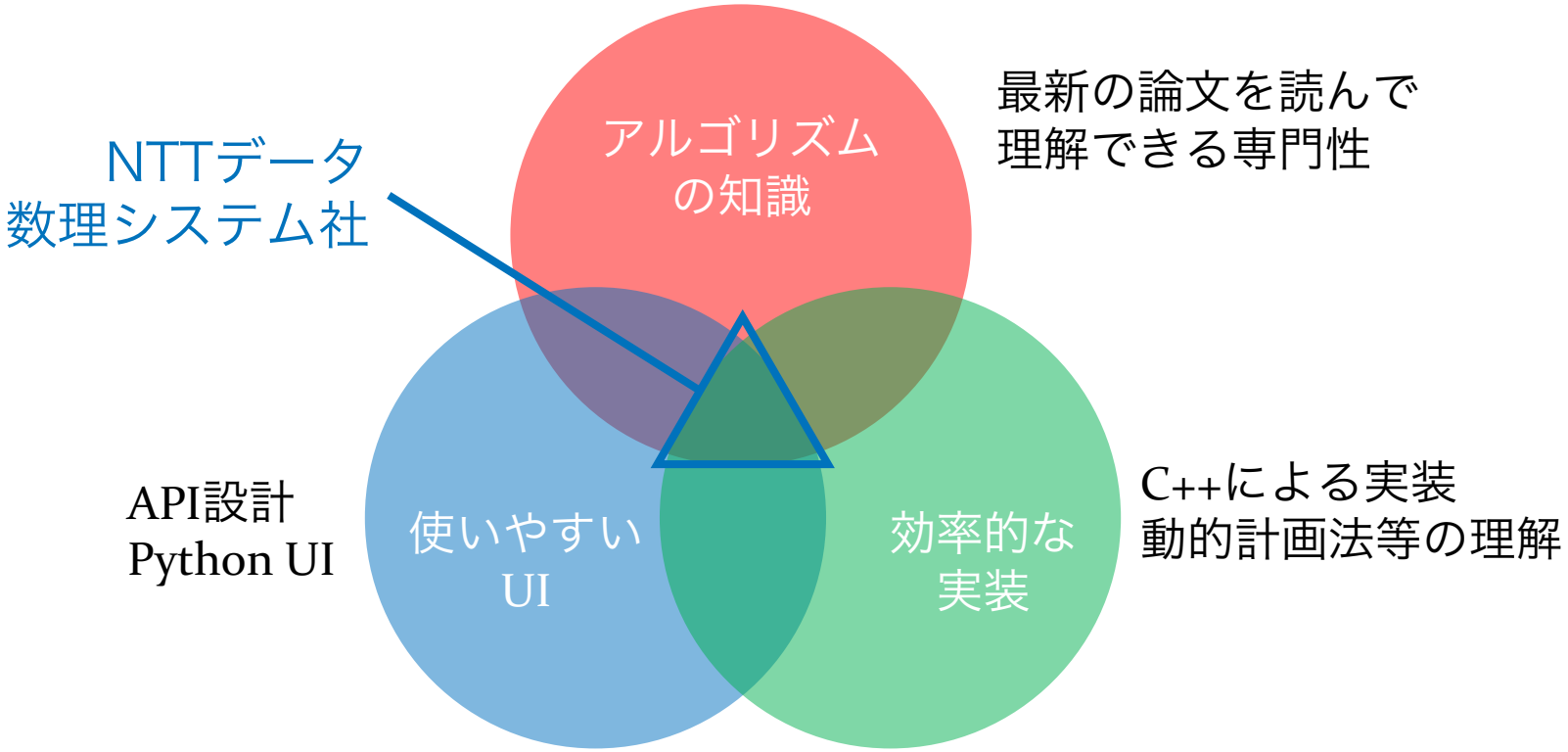
- 広く使ってもらうためには機能が不足
  - Pythonインタフェース
  - 多様な補助的演算（数え上げ、ガーベージコレクション、など）
  - C++による効率的な実装

自分たちだけで実装するのは困難

実装をお願いしたいが…



実装をお願いしたいが…



NTTデータ数理システム社

- 最先端アルゴリズムの実装をお願いできる貴重な会社
- 当初、ZSDDの実装をお願いできる会社があるとは
  思っていなかった
- 論文とサンプル実装をお渡ししただけで、
  中身を理解して、ZSDD処理ライブラリを実装いただけた

NTTデータ数理システム社

- C++での効率的な実装、Python UIの設計などにも幅広く対応
- 仕様や論文に怪しいところがあると正確に指摘
- 実装で悩むアルゴリズム研究者にお勧めしたい

まとめ

- 決定グラフを用いた最先端の組合せアルゴリズムの紹介
  - 膨大な組合せを圧縮することで組合せ爆発を乗り越える
- アルゴリズム実装の重要性とその困難さについて
  - 研究成果を活用するために精度の高い実装が不可欠
  - 最先端技術を活用するソフトウェアの開発は困難
  - NTTデータ数理システム社は最先端アルゴリズムの実装に必要な資質を備える、重要なパートナー