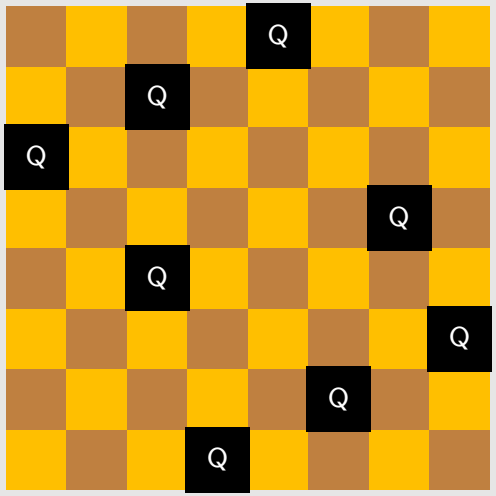
制約充足問題

制約充足問題

変数と制約関数が与えられた時、制約関数を満たすような変数の値を求める問題。

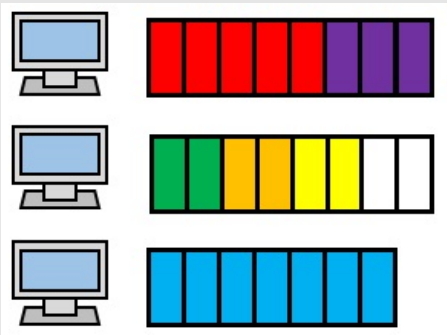
N クイーン問題



グラフ彩色問題

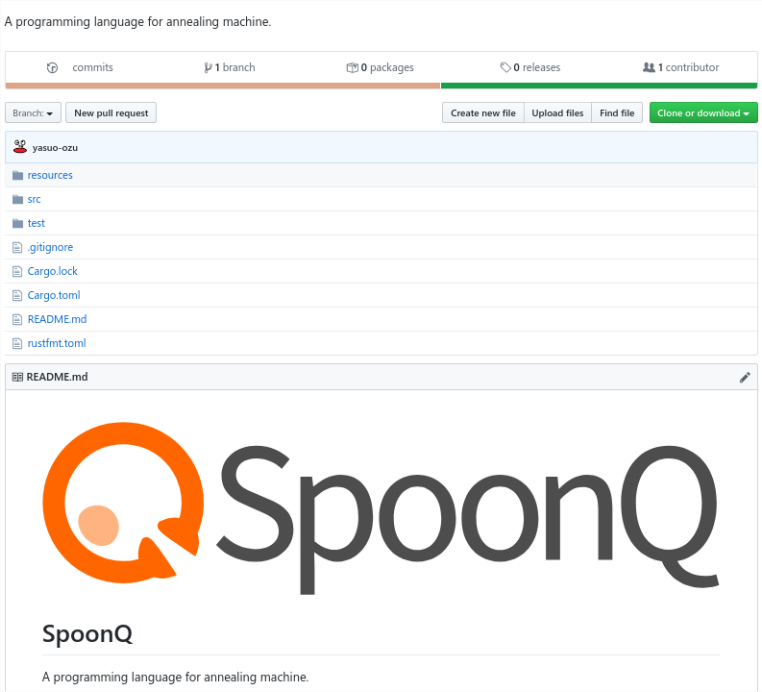
(Wikipedia より)

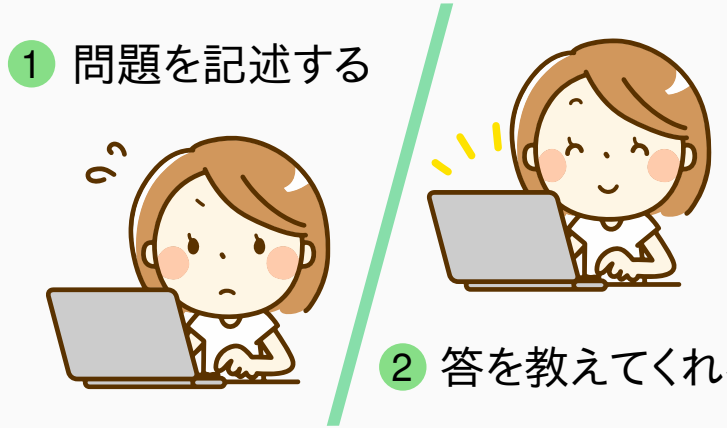
スケジュール問題



2019 年度のプロジェクト「SpoonQ」

Github で公開 ⇒ https://github.com/SpoonQ/SpoonQ





初心者のユーザーがアニーリングに関する専門的な技術を学ぶこと無く、問題をアニーリングによって解くことができる

SpoonQ コードの例

N-Queens 問題

```
type location := 1 .. 5
location [@c1, @c2, @c3, @c4, @c5] : rows
different(rows)
solve(rows)
print(rows)
```

(注意) 現在公開しているもの (Rust 版) は昨年度の中核実装
(JavaScript 版) のサブセットであるため、一部動かない命令がある

本年度プロジェクト

2020 年度のプロジェクト

アニーリングを用いた効率的な

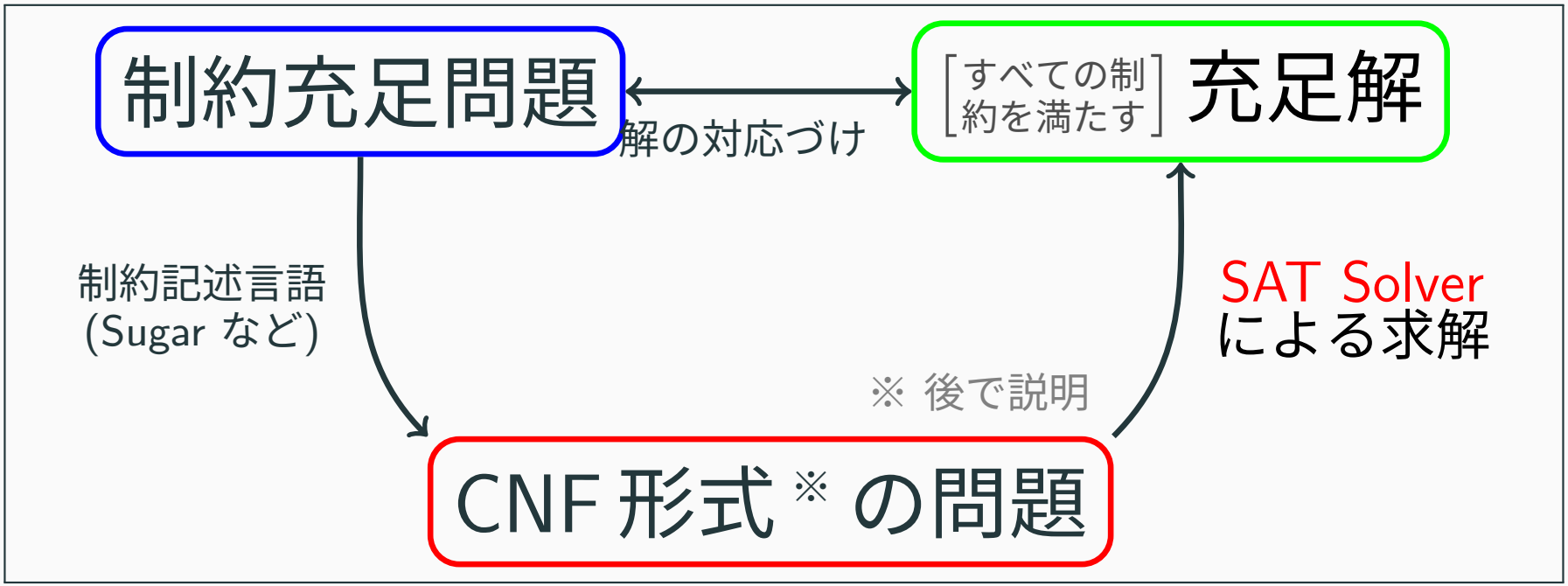
制約充足問題ソルバの実装

SpoonQ の機能拡張として、SAT ソルバを実装する

SAT ソルバとは

制約充足問題を解くために広く使われている。

CNF 形式の問題を解くためのソルバ



現在使われている SAT ソルバは, 古典コンピュータ上で動作 (古典 SAT ソルバ)

メリット

アニーリングを用いて SAT ソルバを実現するメリット

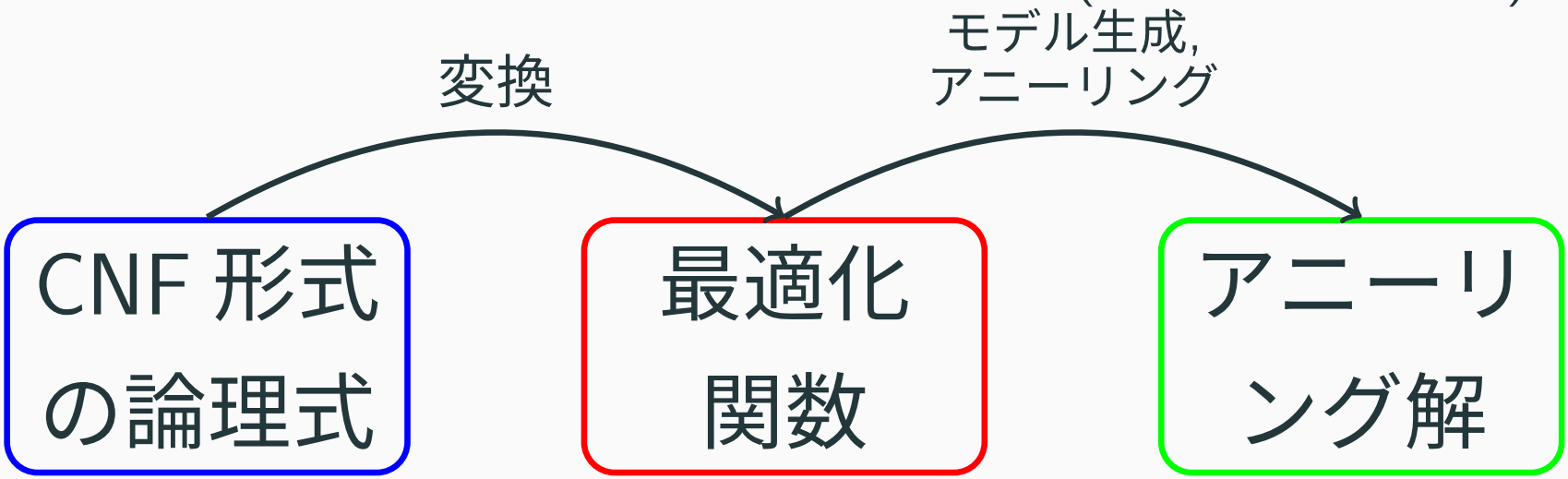
- すでに古典 SAT ソルバを活用している人に新しい求解方法を提案できる。
- 将来、アニーラの性能向上により古典 SAT ソルバと競わせることができる。(予想)

- SAT ソルバに興味を持っている人をアニーリングの世界に引き込める。

⇒ 既にアニーリングに親しんでいる人はメインターゲットではない

アニーリングを用いて SAT 問題を解くために

本プロジェクトで開発したアルゴリズム (単純バージョン)



CNF 形式の論理式

· · · 変数 ○ (もしくは ¬○) が OR(∨) で繋がった節が AND(∧) で繋がったもの

例

○ ∨ ¬○ ∨ ○ ∧ ○ ∨ ¬○ ∨ ○

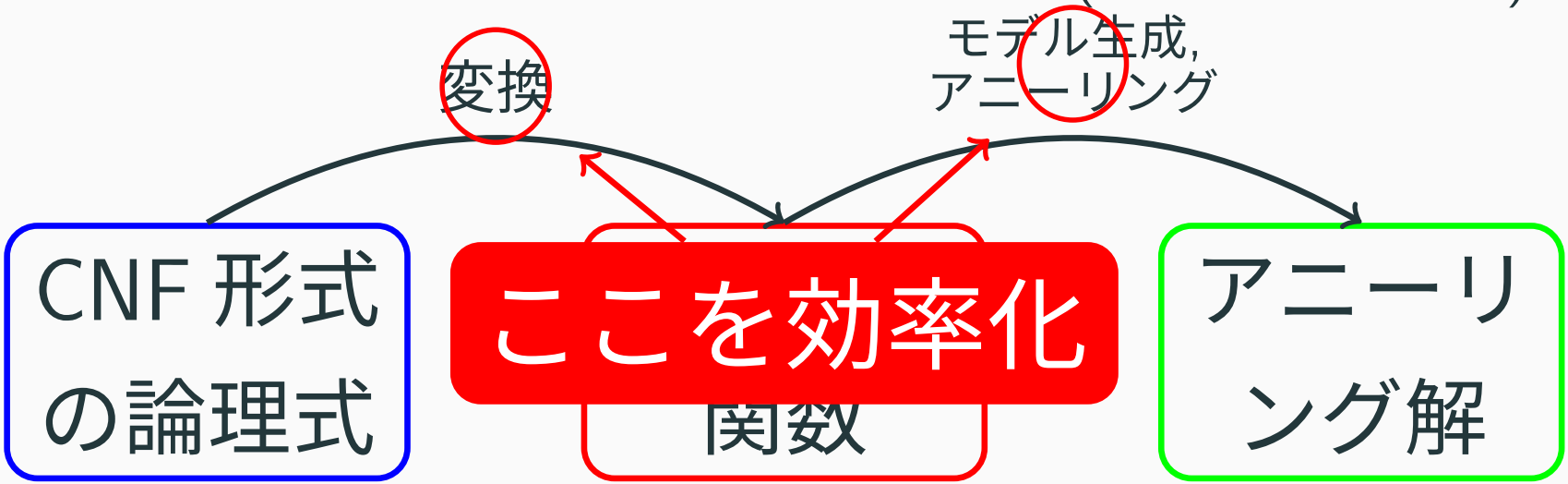
∧ ¬○ ∨ ○ ∨ ¬○ ∨ ○ ∧ ¬○

最適化関数 (目的関数、ハミルトニアン)

引数 $x_i \in \{True, False\}$ から実数値への写像。値が一定値以下のときに、引数が 充足解 であるといえる。

アニーリングを用いて SAT 問題を解くために

本プロジェクトで開発したアルゴリズム (単純バージョン)



CNF 形式の論理式

··· 変数 ○ (もしくは ¬○) が OR(∨) で繋がった節が AND(∧) で繋がったもの

例

○ ∨ ¬○ ∨ ○ ∧ ○ ∨ ¬○ ∨ ○

∧ ¬○ ∨ ○ ∨ ¬○ ∨ ○ ∧ ¬○

最適化関数 (目的関数、ハミルトニアン)

引数 $x_i \in \{True, False\}$ から実数値への写像。値が一定値以下のときに、引数が充足解であるといえる。

古典 SAT ソルバとアニーリングの違い

古典 SAT ソルバ

節の数、変数の数のバランスが重要

アニーリング

アニーラに乗せるため、変数の数を減らしたい

多くの CNF 問題は古典 SAT ソルバで効率的に求められるように生成される

具体例 ･･･ 8-Queen 問題による試算

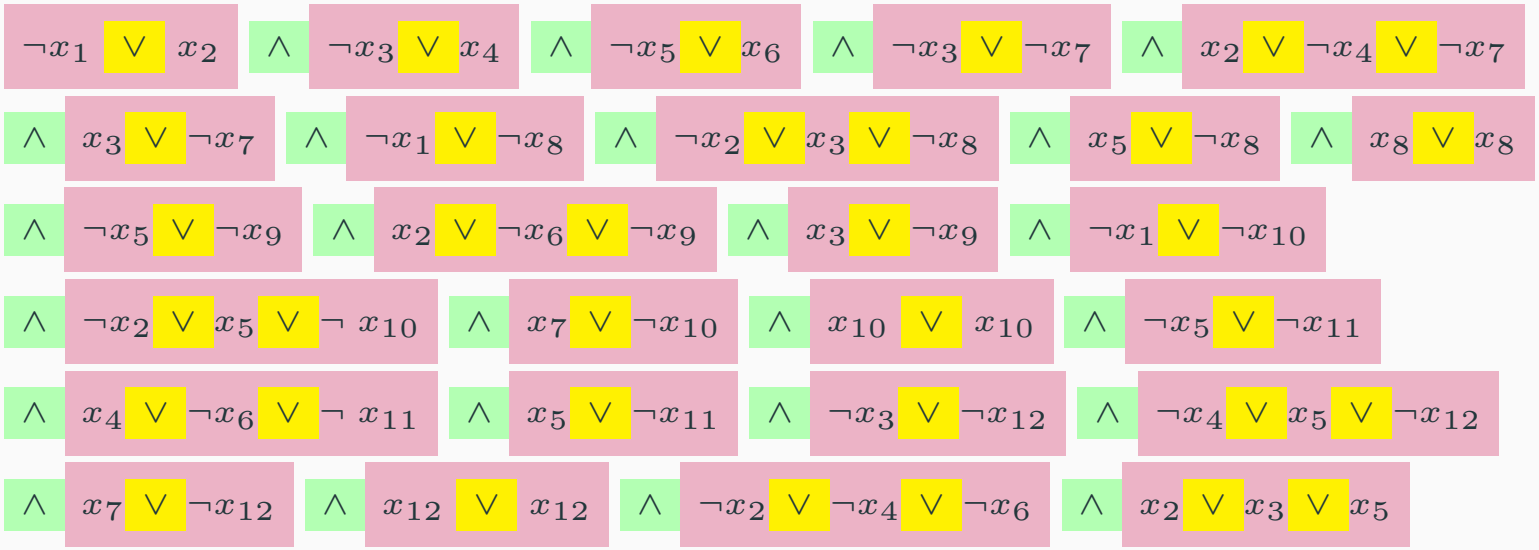
	節数	変数の数
CNF を直接用いる場合（現行手法）	519	460
理想的なハミルトニアン（これを目指す）	16	64

※ 8-Queen 問題の CNF 生成には Sugar を使用 ※節数 ･･･ トップレベルの論理和の数

最適化関数への変換の効率化

CNF 形式の論理式 → 最適化関数

例: 3-Queen 問題の CSP を Sugar で生成 · · · 12 変数, 26 節



最適化関数への変換の効率化

CNF 形式の論理式 → 最適化関数

既存手法 以下のルールによって最適化関数に変換する

- $True = 0, False = 1, \neg x = (1 - x)$
- $x_1 \vee \cdots \vee x_N = x_1 x_2 \cdots x_n$
- $x_1 \wedge \cdots \wedge x_N = \neg((\neg x_1) \vee \cdots \vee (\neg x_N)) = 1 - (1 - x_1) \cdots (1 - x_N)$
- このようなアイデアは複数の論文で見られる
- 一方、最適なハミルトニアンは変数 9 個、節数 6 個

最適化関数への変換の効率化

CNF 形式の論理式 → 最適化関数

提案手法 And, Or, Not の他に以下の論理式の追加

- $\mathrm{CountEq}(x_1, \ldots, x_n, cnt) \cdots x_1, \ldots, x_n$ の中で $True$ となる変数が cnt 個である条件
- $\mathrm{CountLeq}(x_1, \ldots, x_n, cnt) \cdots x_1, \ldots, x_n$ の中で $True$ となる変数が cnt 個以下である条件
- これらを用いれば、3-Queens 問題は 6 個の CountEq の AND として表される

最適化関数への変換の効率化

CNF 形式の論理式 → 最適化関数

CountEq の実装例

- $True = 0, False = 1$
- $CountEq(x_1, \ldots, x_N) = ((1 - x_1) + \cdots + (1 - x_N) - 1)^2$

問題点 以下の 2 つの論理式が存在することになる

- CountEq の場合 $\cdots$ $x = 0$ で True, $x > 0$ で False
- それ以外 $\cdots$ $x = 0$ で True, $x = 1$ で False
- これらを分けて考える必要がある

最適化関数への変換の効率化

CNF 形式の論理式 → 最適化関数

提案手法 1 つの論理式を複数の最適化関数に対応させる。この対応を戦略 (Strategy) と呼びたい。

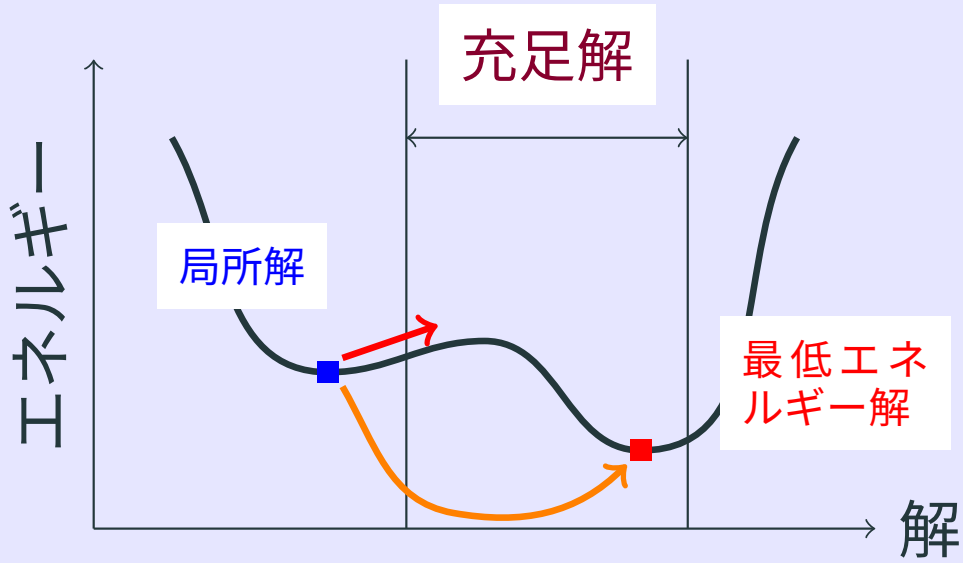
戦略	$x =$ *True*	$x =$ *False*
A	$x^{(A)} = 0$	$x^{(A)} = 1$
$\overline{A}$	$x^{(\overline{A})} = 1$	$x^{(\overline{A})} = 0$
B	$x^{(B)} = 0$	$x^{(B)} > 0$

$CountEq(x_1, \ldots, x_N)^{(B)} = (x_1^{(\overline{A})} + \cdots + x_N^{(\overline{A})} - 1)^2$ と整理される。

アニーリング解の後処理

最適化関数 ⟶ アニーリング解

アニーリング解 は最低エネルギーを取る解とは限らない。

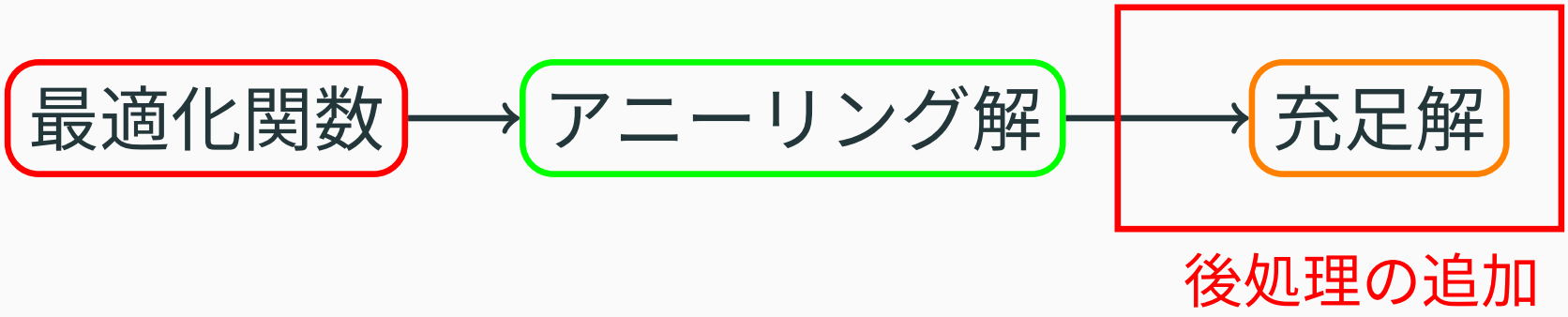


アニーリングでは最低エネルギー解が求まるとは限らない

→ その場合 アニーリングのやり直し が必要

⇒SpoonQ では後処理を行い、 直接充足解を目指す

アニーリング解の後処理



提案手法 アニーリング後に古典 SAT アルゴリズムによる後処理を行う

古典 SAT ソルバのアルゴリズム (単純化したもの)

- 文字が一個のみの節がある場合は、その文字を True とみなす
- 全節の中に肯定と否定の両方が含まれない文字がある場合、それを Trueとみなす
- 適当な文字を選択し、True の場合と False の場合でそれぞれ探索する

○○の部分でアニーリングの結果を用いる