数式処理を用いたアニーリングマシン向け プログラミング言語およびその処理系の開発

- 専門的な知識がないユーザーにもアニーリングを -

1　背景

社会的な課題を解決するために新しいコンピュータの需要が高まっている。例えば、工場内での物資の運搬経路[*1]や、金融資産をどのような割合で保持すべきか決定するポートフォリオ最適化問題[*2]、Web サービスにおける広告配信[*3]等、さまざまな社会的もしくは実務的な要請に基づいてアニーリングマシンの活用が模索されている。

これらのアニーリングマシンによって効率的に解くことができると期待される問題は組合せ最適化問題とよばれ、制約条件および最適化関数によって定式化される。しかしながら、アニーリングマシンはこれらを直扱うことはできない。よって、現状でアニーリングマシンを用いて組合せ最適化問題を解くためには、以下に示す手順を踏む必要がある。

1. 解きたい組合せ最適化問題を制約条件および目的関数として定式化する。
2. (1) の式をハミルトニアンの形式に変形する。
3. (2) の式をイジングモデルの形に変形する。
4. 制約条件のパラメータを決め、(3) の式から QUBO を生成する。
5. アニーリングマシンを用いて最適解を求める。
6. (5) で得られた解が (1) で定式化した問題の制約を満たすことを確認する。
7. (6) で満たされない場合、パラメータを調整し (4) からやり直す。

図 1　アニーリングマシンによる組合せ最適化問題の求解手順

このようにアニーリングマシンの活用には複雑かつ専門的な操作が必要であり、これはアニーリングマシンを使うすべての人にとって簡単であるとは言えない。

以上の手段を簡略化するためのライブラリとして、現在 PyQUBO が広く用いられている。PyQUBO はハミルトニアンの形式で記述された問題をイジングモデルの形式に変換し、QUBO を生成し、アニーリングマシンを用いて解を求める。また、得られた解において制約条件が満たされているかを確認する。すなわち、上の工程における (3), (4) の一部, (5), (6) を自動的に行う。逆に、(2) 及び (6) は利用者自身が行う必要があり、この事から PyQUBO はアニーリングや組合せ最適化問題に関する詳しい知識を持ち、自力で問題をハミルトニアンの形式で記述することができ、また制約を満たす正しい解が求まらなかった場合に自力でパラメータを調整することができる人をターゲットにしていると言える。

しかしながら、このような知識を持つユーザーは、現状では大学で物理学等の専門分野を学んでいる人に限られると考えられる。今後アニーリングマシンがさらに普及するためには、アニーリングマシンを活用するユーザーが増えることが重要である。そのためには、研究者や専門教育を受けた人だけでなく、アニーリングマシンを用いて解決したい課題を持つすべての人にとってアニーリングマシンを活用できる環境を整備することが重要であると考えられる。

2　目的

前章で述べたように、アニーリングマシンを用いて解決したい課題を持つすべての人 (図 2 における三角形の全体) にとってアニーリングマシンが活用できる環境を整備することが重要であると考えられるが、そのための足がかりとして本プロジェクトでは特に以下の属性を

[*1] https://www.tohoku.ac.jp/japanese/2019/11/press20191125-01-Com.html, 2020/2/18 閲覧

[*2] 田中 宗 et al., 量子アニーリングの基礎と応用事例の現状, 低温工学, 53 5 (2018)

[*3] リクルート コミュニケーションズ株式会社, https://www.rco.recruit.co.jp/pressrelease/2018/180129_RCOpress.pdf, 2020/2/18 閲覧

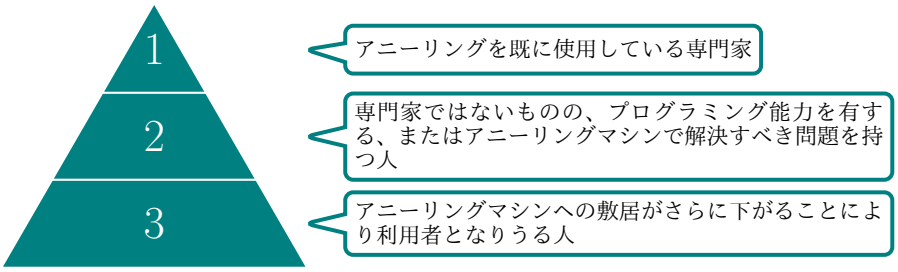

図 2　アニーリングマシンの潜在的利用者

持ったユーザー (図 2 における 2 に相当) を対象とする。

■プログラミングを行う能力を持つ人。　昨今のプログラミング業界は国内外を問わず盛況を極めている。21 世紀に入ってから様々なプログラミング言語が生み出され、その中で新しいパラダイムが生み出されている。時代に取り残されないためには、常に新しいプログラミング言語やプログラミングパラダイムを研究し続ける必要があり、プログラマーの多くはそのような能力に長けているはずである。アニーリングマシンの活用方法も、新たなプログラミングパラダイムの一種であると考えることができるため、これはプログラマーにとっては受け入れやすいものであると考えられる。

■簡単に組合せ最適化問題を解きたいと考えている企業ユーザー。　現状では、民間企業と大学の連携により、アニーリングマシンを用いた様々なプロジェクトが行われている。しかしながら、余力のない企業の場合、外部機関と連携したり、専門家を雇用することは容易ではない。そもそも、企業の抱える問題がアニーリングマシンによって解決できるものであるかを判別することが困難である。よって、専門家ではなくても気軽にアニーリングマシンを活用して組合せ最適化問題を解くことができる環境を用意できれば、問題解決のため、または問題解決方法の検証のために活用されることが期待される。

本プロジェクトの目的は、これら属性をもった人にとって敷居の低いアニーリングマシンの利用環境を提供することである。そのために、組合せ最適化問題を簡潔に記述するためのプログラミング言語を設計し、それを解釈して実行するための処理系を実装する。この処理系を用いれば、図 1 における (3), (4) の一部, (5), (6) に加え、(2), (4), (7) を自動で行うことができる。これにより、アニーリングに関する専門的な知識がなくても、最小の手間でアニーリングマシンが利用できるようになると考えられる。

さらに、本プロジェクトの長期的な目的として、アニーリングコミュニティを発展させることを目指している。情報技術の革新のためには、一部の専門家だけでなく、それを活用したいと考える多くの人によって利用されることが重要であると考えられる。アニーリングにおいても、アニーリングマシンを利用するためのツールを整備し、利用人口を増やすことにより、技術革新がより進むと考えている。

3　ソフトウェア開発内容

本プロジェクトでは、アニーリングに関する専門的な知識を持たないユーザーがアニーリングマシンを活用できるよう、問題を直接記述することのできるプログラミング言語およびその処理系を実装した。この処理系は以下のような特徴をもつ。

- アニーリングマシンによって高速に解くことができると期待されている組合せ最適化問題を、従来手法より直観的に記述することができる。
- 上記の通り記述した組合せ最適化問題を実際に解き、解を得ることができる。このとき、従来手法ではユーザーが行う必要があった解の検証やパラメタサーチは自動で行われ、利用者には追加の手間を求めない。

実装した処理系の構造を図 4 に示す。なお、本プロジェクトでは以下に示す 2 つの処理系を作成している。

■初期実装　今回実装するプログラミング言語は、アニーリングマシン向けの言語であり、先鋭的な試みである。よって、実際にそのような言語を実装できるか、またその実装を用いて問題を解決できるかを実験するため、初期実装として JavaScript を用いた処理系を実装した。この初期実装によって、実際にアニーリングマシン向けの言語を実装し、正しく問題の解を求められることを確認できた。この実装は公開していない。

■二次実装　初期実装により、アニーリングマシン向けのプログラミング言語を実装することが可能であることがわかった。しかしながら、プロジェクト期間後も含め、処理系の機能を今後も継続して拡張してゆくことを考慮すると、初期実装では拡張性に問題があった。よって、よりモジュール化をすすめ、拡張性の高い処理系を実装するために、Rust 言語を用いた二次実装を行った。二次実装では他にも、コマンドラインヘルプや図 3 に示す分かりやすいエラー表示も実装した。

また、以下のような問題を解くことのできるサンプルコードも作成した。

- 整数分割問題
- グラフ分割問題 (図 5)
- 集合詰め問題 (図 6)
- 頂点被覆問題 (図 7)
- 最小最大マッチング問題
- グラフ彩色問題 (図 8)
- ハミルトン経路問題
- 巡回セールスマン問題

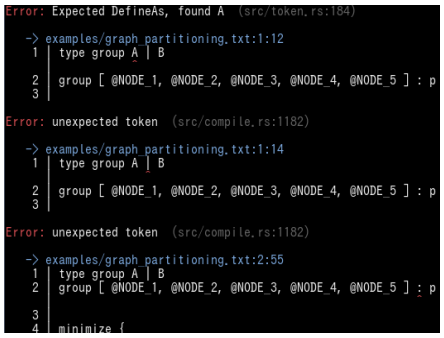

図 3　エラー表示

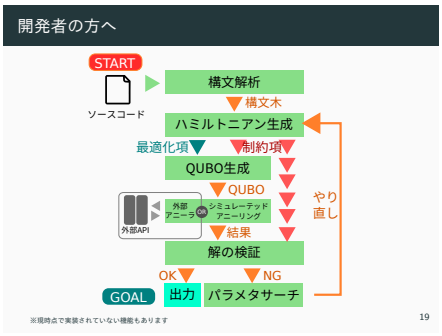

図 4　SpoonQ の内部構造

4　新規性・優位性

1 章で述べたとおり、現状ではアニーリングは専門家 (図 2 における 1) でなければ利用が困難である。本プロジェクトで開発する言語は、アニーリングに関する専門的な知識は必要とせず、プログラミングの経験が合ったり、すでに解きたい問題を持っているユーザー (図 2 における 2) に対してアプローチすることができる。これにより、より多くの人をターゲットとすることができる。

5　期待されるユーザー価値と社会へのインパクト

これまで、専門的な知識がないために、アニーリングマシンを利用できなかった人が、本言語によりアニーリングマシンを用いて問題を解決することができるようになる。また、アニーリングマシンを活用する人口が増えることにより、アニーリング業界の技術革新が期待できる。さらに、処理系を使用することにより、利用者は問題を記述することのみに専念できる。問題の記述体型を言語によって定義することで、ユーザにアニーリングマシンに対する新しい視点を与えることができ、新たな技術創発を促すきっかけになると期待している。

6　氏名（所属）

小津 泰生 (名古屋大学 理学研究科)
(参考) https://github.com/SpoonQ

```
type group A | B | C
group [ @NODE_1, @NODE_2, @NODE_3, @NODE_4,
    @NODE_5, @NODE_6, @NODE_7, @NODE_8 ] : p
minimize {
 1 : @NODE_1 == @NODE_2
 1 : @NODE_1 == @NODE_3
 1 : @NODE_1 == @NODE_5
 1 : @NODE_1 == @NODE_6
 1 : @NODE_1 == @NODE_8
 1 : @NODE_2 == @NODE_4
 1 : @NODE_2 == @NODE_5
 1 : @NODE_2 == @NODE_7
 1 : @NODE_3 == @NODE_4
 1 : @NODE_3 == @NODE_5
 1 : @NODE_3 == @NODE_6
 1 : @NODE_3 == @NODE_8
 1 : @NODE_4 == @NODE_5
 1 : @NODE_4 == @NODE_7
 1 : @NODE_5 == @NODE_7
 1 : @NODE_6 == @NODE_7
}
solve(p)
print(p)
```

図 5　グラフ分割問題の記述例

```
type colored := yes | no
colored [ @v1, @v2, @v3, @v4, @v5,
      @v6, @v7, @v8 ] : vertices
@v1 == yes, @v4 == yes
@v2 == yes, @v5 == yes
@v4 == yes, @v5 == yes
@v5 == yes, @v7 == yes
minimize {
 1 : @v1 == yes
 1 : @v2 == yes
 1 : @v3 == yes
 1 : @v4 == yes
 1 : @v5 == yes
 1 : @v6 == yes
 1 : @v7 == yes
 1 : @v8 == yes
}
solve(vertices)
print(vertices)
```

図 7　頂点被覆問題の記述例

```
type bool := yes | no
bool [ @v1, @v2, @v3, @v4, @v5,
      @v6, @v7, @v8 ] : subsets
@v1 != yes || @v4 != yes
@v1 != yes || @v6 != yes
@v2 != yes || @v5 != yes
@v2 != yes || @v8 != yes
@v3 != yes || @v7 != yes
@v4 != yes || @v5 != yes
@v4 != yes || @v8 != yes
@v5 != yes || @v7 != yes
@v7 != yes || @v8 != yes
maximize {
 1 : @v1 == yes
 1 : @v2 == yes
 1 : @v3 == yes
 1 : @v4 == yes
 1 : @v5 == yes
 1 : @v6 == yes
 1 : @v7 == yes
 1 : @v8 == yes
}
solve(subsets)
print(subsets)
```

図 6　集合詰め問題の記述例

```
type color red | orange |pink
color [@BC, @YK, @NW, @AB, @SK, @NV, @MT,
      @ON, @QB, @NB, @NS, @PE, @NL] : p
@BC != @AB, @BC != @YK, @BC != @NW
@YK != @BC, @YK != @NW
@NW != @YK, @NW != @BC
@AB != @BC, @AB != @NW, @AB != @SK
@SK != @AB, @SK != @NW, @SK != @MT
@NV != @NW, @NV != @MT
@MT != @NV, @MT != @SK, @MT != @ON
@ON != @MT, @ON != @QB
@QB != @ON, @QB != @NB, @QB != @NL
@NB != @QB, @NB != @NS
@NS != @NB
@NL != @QB
solve(p)
print(p)
```

図 8　グラフ彩色問題の記述例