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

ー制約充足問題をアニーリングで解こうー

1　背景

私は、様々な社会的問題を解決するための新たな手段として、アニーリングを用いる方法を普及させたいと考えている。2019 年度のプロジェクトで開発した SpoonQ は、主に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手軽に解く方法を提供する。しかしながら、SpoonQ を用いた方法は、古典コンピュータによる既存の問題解決手法とは全く異なるものであり、既に既存の SAT ソルバを活用しているユーザーに対して、SpoonQ を用いたアニーリングの活用を提案することは困難である。

今年度のプロジェクトでは、制約充足問題を入力し、アニーリングを用いて効率的に求解できるソルバを実装する。制約充足問題は古典コンピュータを用いた既存の問題解決手法において広く使われている。制約充足問題のソルバを提供することで、既存の手法を活用しているユーザーに対しても、アニーリングを利用するためのツールを提供することができる。その結果、2019 年度の成果物と合わせて、より広いユーザー層に対して問題解決のための手段を提供できるようになることが期待される。さらに、SpoonQ からこのソルバを活用できるようにすることにより、SpoonQ を用いる場合でも、より効率的に問題を解くことができるようになると考えられる。

2　目的

(a) 制約充足問題の例　　(b) SAT ソルバを用いた制約充足問題の解き方

図 1　制約充足問題について

図 1(a) に示すように、世の中に存在する様々な問題は制約充足問題で記述される。そのため、制約充足問題を解く方法は古くから研究されている。そのうち最も有名なものは古典コンピュータ上における SAT ソルバを用いる方法である。SAT ソルバを使う場合、図 1(b) に示すように、制約充足問題を制約記述言語によって記述し、制約記述言語処理系を用いて CNF 形式の問題に変換し、さらに SAT ソルバを用いて解を

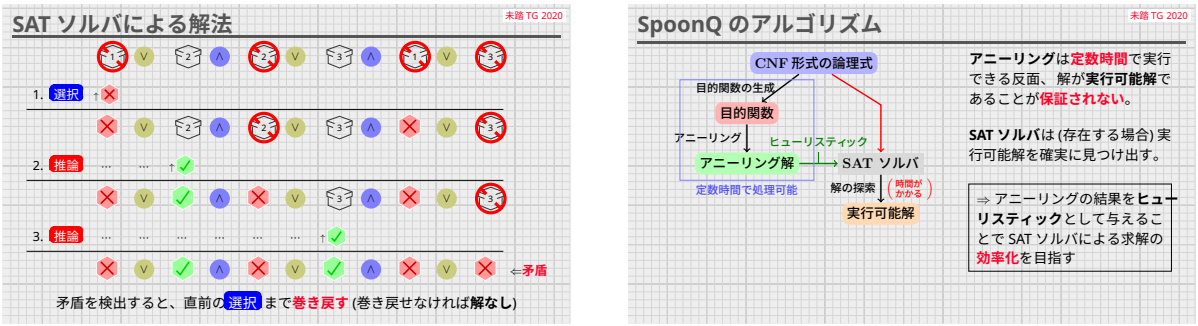
得る。

本年度のプロジェクトは、既存の古典コンピュータ上で動作するSATソルバが扱うものと同じ形式で与えられる制約充足問題を対象とする。これにより、既に古典コンピュータで制約充足問題を扱っていたユーザーをアニーリングの世界に取り込むことが期待される。

3　ソフトウェア開発内容

本プロジェクトは以下の3つのプロジェクトによって構成される。

■SpoonQ　昨年度のプロジェクトで開発したSpoonQに対して、CNF形式で与えられた制約充足問題を処理する機能を追加した。これにより、ユーザーは本プログラムを用いて「SpoonQ言語」を利用するか、もしくはCNF形式の制約充足問題を利用するかの両方を選ぶことができる。具体的には、SpoonQに与えるファイルによって自動的に両者が区別される。SpoonQはファイルの中身を読み込み、有効なCNFファイルのヘッダを検出すればそれをCNFファイルと判断し、そうでなければ「SpoonQ言語」のファイルであると判断する。このとき、ファイルの拡張子は判定に用いられない。



(a) 古典 SAT ソルバのアルゴリズム　　(b) SpoonQ のアルゴリズム

図 2　アルゴリズムの詳細

古典SATソルバのアルゴリズムを簡単にしたものを図2(a)に示す。古典SATソルバでは、入力されたCNF形式の論理式に対して、「選択」と「推論」の各フェーズを繰り返すことにより、総当り的に解を探索している。しかしながら、この探索には時間がかかる場合がある。

これに対して、SpoonQによって充足可能性問題を解く場合のアルゴリズムを図2(b)に示す。このアルゴリズムでは、古典SATソルバの入力としてアニーリングによって得られた解を与えている。これにより、SATソルバの探索時間を短くすることを目指している。

■RustQUBO　RustQUBOは、Rust言語から最適化問題の目的関数や制約充足問題の制約式を扱うためのライブラリである。これらをアニーリングマシンで扱うためには、QUBOと呼ばれる高々2次の目的関数に変換する必要がある。RustQUBOは目的

関数を Rust 言語上で記述する方法を提供し、また後述する石川の方法に基づいた次元削減を行う。RustQUBO 及び後述する annealers を組み合わせることにより、目的関数をアニーリングマシンまたはシミュレーテッドアニーリングを用いて解くことができる。また、解のエネルギーや部分場を求める機能も含む。

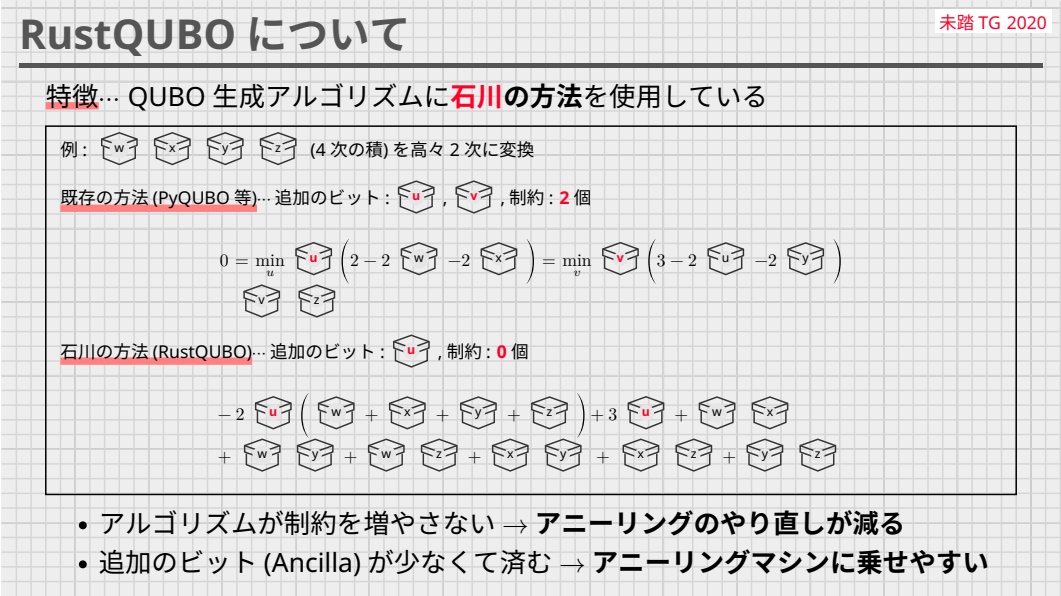


図 3　石川の方法による次元削減

RustQUBO では 3 次以上の高次項を含む目的関数を扱うことができるが、これを実際のアニーリングマシンに乗せるためには、次元削減によって高々 2 次の目的関数に変換する必要がある。図 3 に示すように、石川の方法は従来の手法より制約を増やさず、また追加のビット数も少なくて済むという特徴がある。これは、従来手法よりアニーリングに適した手法であると言える。

■annealers　annealers は、与えられた問題 (主に QUBO 形式) を内蔵されたシミュレーテッドアニーリングのアルゴリズムによって解くための Rust 向けライブラリである。前述した RustQUBO が出力した問題を解くことができる他、拡張が容易な設計となっており、将来的に外部アニーラに対応した場合、内蔵シミュレーテッドアニーラ及び外部アニーラを共通のインターフェースで扱うことができる。

4　新規性・優位性

新規性・優位性について、3 項目に分けて述べる。

■SpoonQ　制約充足問題の求解に使われる SAT ソルバは、これまで古典コンピュータ上で広く研究されていたが、アニーリングを SAT ソルバと組み合わせる方法は本プロジェクト独自のものである。現時点では既存の SAT ソルバの性能にはかなわないものの、SAT ソルバに対する入力の与え方を変更する等、簡単な変更を加えることでアルゴリズムを書き換えることができるため、アルゴリズムの効果の検証や、問題

に合わせたアルゴリズムの調整も可能である。また、今後さらに開発を進めることにより、古典 SAT ソルバとアニーリングによる SAT ソルバを競わせることができると期待される。また、古典コンピュータと戦わせることによって、アニーリング技術やハードウェアの発展を促すものと考えられる。

■RustQUBO　目的関数をアニーリングによって解ける形式に変換することができる Rust 言語上のライブラリとして私の知る限り唯一のものである。また、石川のアルゴリズムを用いるなどの先進的な試みを行っている。さらに、annealers と連携することで、量子ビットの種類や実数値の解像度など、ハードウェアによって異なる仕様に適合した目的関数を生成できるという特徴をもつが、これも RustQUBO が持つ特徴である。

■annealers　各社が提供するアニーリング API を利用するためには、これまでは API 毎に別々のプログラムを作成する必要があった。annealers は 1 つのプログラムで複数のシミュレーテッドアニーリングやアニーリングマシンに対応することを目指しており、オープンソース業界では私の知る限り初の試みである。 複数のアニーリングマシンを共通のインターフェースで扱うために、annealers では図 4 に示すように、アニーリングマシンの特徴をモデル化し、抽象化を行った。これにより、同じプログラムで複数のアニーリングマシンを使うことと、それぞれのアニーリングマシンの機能を最大限に活用することを両立している。

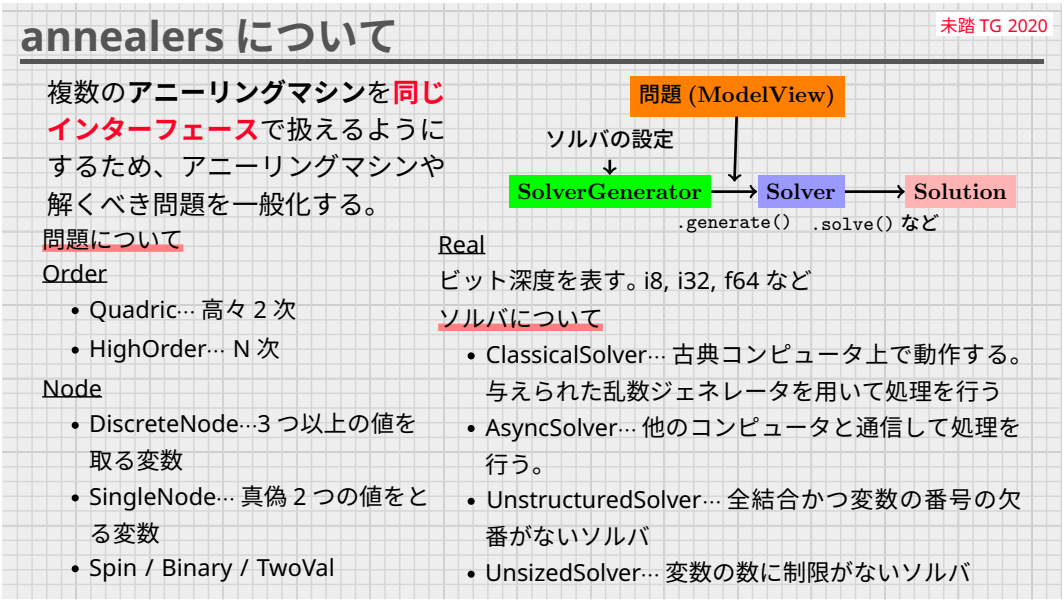


図 4　annealers によって実現される抽象化

5　期待されるユーザー価値と社会へのインパクト

今回のプロジェクトで開発した SpoonQ の SAT ソルバ機能は、アニーリング技術を活用したことにより、これまでの古典コンピュータ上の SAT ソルバとは大きく異な

るアルゴリズムとなっている。一方、入出力は古典コンピュータ上の SAT ソルバと同じ形式で行えるため、既に既存の SAT ソルバを活用しているユーザーが違和感なく使用することができる。これにより、既に古典コンピュータ上で制約充足問題を解くことに興味を持っているユーザーをアニーリングの世界に引き込むことができ、昨年度のプロジェクトと合わせることによって、より幅広い層のユーザーに対して、アニーリングに興味を持ってもらうためのきっかけを提供することができる。また、機械学習がそうであったように、アニーリングマシンの利用者が増えることによってアニーリングのコミュニティが活性化し、分野における技術開発のスピードも一層加速することが期待できる。

また、今回「RustQUBO」と「annealers」という 2 つの Rust 言語向けライブラリを実装した。これらは Rust 上で使用できるアニーリング向けのライブラリとして初めての試みである。Rust 言語は、これまで広く用いられていた言語 (Python など) に比べて、高速であり、可用性が高く、優れたパッケージ管理機能を持つ等、様々な優位性がある。また、型に厳密な言語であるため、Python のように実行することなく、多くの誤りに気づくことができる。アニーリング API の多くは有料であるため、プログラムのデバッグ時に余分に API を実行しないことは経済的にも重要である。

6 氏名（所属）

小津 泰生 (名古屋大学 理学研究科)

(参考) https://spoonq.github.io/