Chapter 1.4
高柳奏和
2021/10/29

1.4
ここまでは、問題を分類する様々な方法について説明し、あえてソルバーについての説明にはふれなかった。結果的に、問題を見ただけでこれらのスキームの一つに従って問題を分類することができる。この節では、問題のカテゴリが、問題解決アルゴリズムの特性によって定義されるため、これが不可能な分類スキームについて説明する。このアプローチの背景にある動機は、問題をその難易度の観点（解くのが難しいか、もしくは容易か）から話そうという意図である。大雑把に言えば、基本的な考え方は、問題を、高速なソルバーが存在すれば容易、そうでなければ難しいということである。この問題の難易度に対する考え方は、計算複雑性の研究につながる。
先に進む前に、最適化問題において、対応する探索空間におけるオブジェクトのタイプによってさらなる区別をつけておく必要がある。もし探索空間 S が連続変数（すなわち実数）によって定義されている場合、数値最適化問題がある。もし S が離散変数（すなわちブール値もしくは整数）によって定義されているならば、組み合わせ最適化問題がある。難易度を表す様々な概念は組み合わせ最適化問題に対して定義される。注意してほしいのは、離散探索空間は常に有限、もしくは、最悪の場合、数えられるほどの無限であるということである。
計算複雑性についての完全な概略は、[180, 330, 331, 318]などの多くの書籍で扱われているため、ここでは取り上げない。むしろ、いくつかの重要な概念とその問題解決への影響、そしてよくある誤解について、簡単に説明する。さらに、この本にはふさわしくないため、このテーマを数学的に厳密には扱わない。ゆえに、アルゴリズム、問題の大きさ、ランタイムなどの重要な概念について、正確な定義はしないが必要に応じて例を上げてその意味を説明する。
計算複雑性における最初の重要な概念は、問題の大きさである。これは、問題の次元（すなわち変数の数）と問題変数の異なる値の数に基づいている。前述の例では、訪問する年の数や、ボード上に置くクイーンの数

などが問題の大きさを示す理にかなった尺度となる。2つ目の概念は、問題ではなくアルゴリズムに関するものだ。アルゴリズムの実行時間は、そのアルゴリズムが終了するまでに必要な基本的なステップ、つまり演算の数だ。計算複雑性の背後にある一般的な直観は、問題が大きければ大きいほど、解くのにより多くの時間を必要とするというものである。よく知られている問題の難易度の定義は、問題の大きさを、その問題の解くアルゴリズムの最悪の実行時間と関連付けている。この関係は、最悪の場合の実行時間の上限を問題の大きさの関数として指定する式で表現される。簡単に言うと、この式は多項式（比較的短い実行時間を示すと考えられている）もしくは、超多項式、例えば指数関数（長い実行時間を示す）である。最後の概念は、問題の削減だ。これは、ある問題を適切なマッピングによって別の問題に変換できるという考え方である。ただし、この変換は可逆的でないかもしれない。このように問題の変換や削減の考え方は少し複雑だが、前節で現実世界の問題はしばしば、異なるが同等の方法で形式化されることを見たように、まったく身近でないわけではない。問題の難易度に関してよく使われる概念は、イアkのように言い換えることができる。
多項式で解くことができるアルゴリズムが存在する場合、その問題はクラスPに属するという。つまり、問題に対するアルゴリズムが存在していて問題の大きさnに対する最悪の実行時間が、ある多項式Fにたいして F(n)よりも小さいアルゴリズムが存在する場合である。多項式Fよりも小さいアルゴリズムが存在する場合である。一般的な言葉で言えば、集合Pには、例えば、最小全域木のように簡単に解くことができる問題が含まれる。ある問題が、あるアルゴリズム（その実行速度については主張しない）で解くことができ、その解答を多項式時間以内で検証できる場合、その問題はNPクラスに属するという。これは多項式ソルバーは多項式時間の解法にも使われるPはNPの部分集合であるということに基づく。NP問題の例として、部分集合和問題がある。問題：整数の集合が与えられたとき、その集合の１つ以上の要素の和が０になる集合はあるか? 明らかに、この問題に否定的な答えを出すには、可能なすべての部分集合を調べる必要がある。残念ながら、可能な部分集合の数は、集合の大きさの多項式以上の数になる。しかし、解法が有効であることを検証するには、単に見つかった部分集合の内容を合計するだけである。

ある問題がNPというクラスに属し、NPの他の問題が多項式時間で実行されるアルゴリズムによってこの問題に還元できる場合、その問題はNP完全というクラスに属すると言われている。この問題は、実際には常に発生する難しい問題である。NP完全問題のよく知られた例は、インターネットで簡単に見つけることができる。
ここでは、コンピュータ・サイエンスにおける興味深い問題の大部分は、NP完全問題であることが判明している、という事以外の要約はしないことにする。
最後に、ある問題は、少なくともNP完全の問題と同じぐらい難しい場合、NP困難クラスに属すると言われる。（つまり、NP完全のすべての問題は、NP困難のひとつに減らすことができる）。NP完全のすべての問題は、NP困難の1つに還元できるが、その解が必ずしも多項式時間内に検証できない場合、NP困難クラスに属すると言われている。そのような例として、停止問題がある。
多項式時間で解が検証できない問題が存在することは、クラスPがクラスNP困難と同じではないことを証明している。不明なのは、2つのクラスPとNPが実際には同じあるかどうかです。これが事実であれば、コンピュータ・サイエンスや数学にとって非常な大きな意味を持ちます。難しいと思われていた問題にも高速なアルゴリズムが存在すると知られることになるからだ。したがって、P=NPであるかどうかは、複雑性理論の壮大な課題の一つであり、P=NPまたはP=NPであることを証明した場合には、100万ドルの報酬が提供されています。後者は多くの複雑な数学の対象となっているが、前者はNP不完全問題の高速アルゴリズムを作ることで簡単に証明できる。図1.4は、PとNPの等しさに応じた問題の難易度のクラスを示している。P=NPの場合、P=NP=NP完全の集合になるが。それでもNP困難の部分集合となる。
これは理論的なことのように聞こえるが、問題解決にとって非常に重要な意味を持っている。ある問題がNP不完全である場合、特定の事例を多項式時間で解くことはできても、すべての可能な事例について溶けるとは言えない。したがって、これらの問題に、問題解決手法を適用しようとすると、現在のところ、非常に小さな（あるいは簡単な）事例しか溶けないだろうと受け入れるか、厳密な解を提供するという考えを捨てて、近似やメタヒューリスティックに頼って十分な解を作るしかない。

これは、P にあることがわかっている問題とは対象的である。これらの問題の可能な解の数は指数関数的に増加する可能性があるが、解を見つけるアルゴリズムは存在し、その実行時間は事例のサイズに対して多項式に増加する。この節のまとめとして、実用的な問題の中には、調べてみると NP 完全と呼ばれる抽象的な問題の派生であることがわかるものが非常に多く存在する。このような問題がいくつかの例は簡単かもしれないが、ほとんどのコンピュータサイエンティストはこのような問題に対する多項式時間のアルゴリズムは存在しないと考えている。したがって、このような問題のどのような事例に対しても、受け入れられる解を作りたいと思うのであれば、近似やメタヒューリスティックスを使うことに目を向け、その事例に対して証明された最良の解を確実に見つけるという考えを捨てなければならない。