講演 3

最適化とAI ―計算理論の視点から―

牧野　和久（数理解析研究所　教授）

よろしくお願いいたします。私は牧野でして、本日は「最適化とAI ―計算理論の視点から―」というタイトルでお話させていただきます。

今回のタイトルにあります「最適化とAI」というのは、ここにキーワードが出ていますが、時間が30分と実は短いので、その基礎を成すアルゴリズム論という、太字で書いたところに焦点を絞って、お話したいと思います。

これが本講演の流れです。アルゴリズム、計算の話をして、最後に未解決で有名なP vs NP問題を紹介し、終わりたいと思います。

さて、「アルゴリズム」、この言葉を20年前に知っていた方が、どれぐらいおられるでしょうか。高校生の方は生きていないので、知る由がないと思いますが、20年前、アルゴリズムという言葉は、専門家を除いて、多くの皆さん知らない状況でした。

ちょうど20年前にNHKで「ピタゴラスイッチ」という番組がスタートしまして、その番組中に「アルゴリズムたいそう」というものがあります。実は十数年前まではアルゴリズムというと「アルゴリズムたいそう」を連想される人が日本ではたくさんいました。

しかし今や、これは先月の２月７日のYahoo!ニュースの見出しですが、「ウーバー配達員の業務上過失致死罪、AIのアルゴリズムが遠因か」と。記事の内容はさておき、記事のタイトルにアルゴリズムが出てくるというぐらい、世の中にアルゴリズムが浸透しています。

このアルゴリズムは一体何かというと、問題や課題を解決するための計算手順や処理手順です。よく似た言葉にプログラムというのがあります。プログラムというのはアルゴリズムを計算機に分かるように記したもの。もちろんアルゴリズムは計算機、コンピューターの出現によって、その重要度は増して急速に発展していますが、概念としては、アルゴリズムとコンピューターは別物だと思ってください。

ですから、例えば高校生の皆さんが、数学の問題をどうやって解こうかと考える。その解き方はアルゴリズムそのものです。あるいは料理の作り方。「さしすせそ」の順に調味料を入れて、こっちで茹でながら何かを切るとか、効率良く、おいしい料理を作るというのも、アルゴリズムそのものだと思います。

実社会では、コンピューター自身がアルゴリズムで動いていますから、当然ありとあらゆるところでアルゴリズムは使われているんですけれども、こと最適化に関して見てみますと、配送計画、スケジューリング、基盤配線。基盤配線というのは、集積回路を作るときにはできるだけ小さいものを作りたいですから、どの素子を基盤のどこに置くかということを考え

る。あるいは、コンテナの詰め込み。コンテナにどうやって商品を詰め込むか、詰め込み方が輸送費に直結します。あるいはタンパク質の構造解析などなど、様々な場面で最適化アルゴリズムが使われます。

アルゴリズムの歴史は、というと非常に古くて、紀元前2500年のバビロニア、エジプトで始まったとされています。これらは算術的なアルゴリズムが多いです。もうちょっとアルゴリズムっぽいのは古代ギリシャで、ユークリッドの互除法みたいなものが出てきました。アルゴリズムの名前は、９世紀の数学者のアル・フワーリズミーによるそうです。すみません、このへんはこの講演のために調べたので、私自身が慣れていませんが。

もう一つ、古代ギリシャの数学で、目盛りのない定規とコンパスで何ができるかを問う問題があります。例えば、いま見るように正六角形をつくることができます。

このように定規とコンパスで何が作図でき、何ができないか、という研究がされていました。例えば皆さん、中学校の１年生？ぐらいで習ったと思いますが、角の二等分線、これはいまお見せしているように、定規とコンパスで引くことができます。それでは定規とコンパスを用いて、角の三等分線を引くことはできるでしょうか？　実は一般にはできないということが知られている、証明されています。ただ、紀元前ぐらいに考えられていたのですが、証明自身は最近、最近といっても1837年に与えられています。このような定規とコンパスでの作図もまさにアルゴリズムです。

ここでアルゴリズムを考えてみましょう。簡単な例ですが、１から1+2+3……と、1000まで足しましょう。どうやってやりますか、皆さん。何かありましたら。

そうですね、ありがとうございます。今、会場の方が答えていただきました。ただ、小学校の低学年の人はどうするかというと、根性を出すわけですね。最初の１と２を足して３を出す。出てきた３と次の３を足して６を出す。その６と次の４を足して10を出す、と頑張るわけですね。そうすると、何回、計算しましたかと。足し算、引き算、掛け算、割り算がありますけども、何回の演算でできましたかというと、この場合は足し算を999回使えばできますよね。

先ほどありましたが、皆さんは公式で、ここでは証明しませんが、こんなふうにできますよね。初項と最後の項を足したものと項数を掛けて２で割ると計算できます。この方法は、掛け算１回、足し算１回、割り算１回ということで、３回で計算できます。そうすると、999回と３回の、どっちが楽ですか？　だから皆さんは学校で公式を習うわけですね。よろしいでしょうか。

それではもう一つ。正の整数ａとｎが与えられます。そしてａのｎ乗を計算するという問題を考えましょう。簡単なものですね。これも先ほどみたいに単純な方法を用いると、最初にa^2を求めて、次にa^2とａを掛けて、a^3を求めてやる、というふうに計算していくと、a^nですから、ｎ－１回の掛け算を行えば、計算ができます。

この問題を工夫して計算できるでしょうか？　ここでは説明を簡単にするために、ｎを２

のべき乗、２のｋ乗とします。ｋは正の整数です。そうすると最初は一緒ですよね、ａとａを掛けて、a^2を出すと。先ほどはa^2と次のａを掛けていましたけど、せっかくa^2を計算して求めたので、a^2とa^2を掛けてやる、そうするとa^4ができますよね。次に、a^4を知っているのだから、このa^4とa^4を掛けて、a^8を出せばいい、と計算する。

ここで、肩に載っている指数部分を見てやると、２は2^1で、４は2^2で、ということで、指数のところが２の何乗になっていますから、ｎが2^kのときは、上から順番に１、２、３……と、ｋ回やれば、ｎが2^kですから、log ｎ回掛け算を行うとa^nが計算できます。

先ほどの一番最初はｎ－１回でしたから、単純な方法をすると2^k－１回、後のやり方でいうとｋ回で、計算の効率としては随分違いますよね。ｋが10だとしても、2^{10}－１は1023で、10回計算するのと、1023回を計算するのと、どっちがいいかということです。

このように問題が一つでも解き方、アルゴリズムはいっぱいあります。その中でどのアルゴリズムを使うかというのは非常に重要な問題になります。次の例はアルゴリズムというか、暗号の例ですけれども、自転車に付けるような、鍵です。何回回せば、鍵が開けらるか、という問題です。

暗号のポイントは、知っている人と知らない人で大きなギャップがあるということです。鍵の番号を知っている人は、この鍵は４桁ありますけど、各桁を時計回りか、反時計回りか、高々５回、カチカチカチと回せば揃えられるわけですから、５×桁数４で、たかだか20回回せば、鍵を開けられます。

一方、番号を知らない人はというと、もう頑張るしかないですよね。10000回、全部試す。期待値でも5000回ですね。だから20回と10000回で、これは大きな違いです。今は４桁でしたが、ｎ桁だとすると、5nと10のｎ乗と、すごいギャップがあるわけですね。これが暗号の一つの基本になっています。

それではもう一つ。323を素因数分解しなさい。皆さんできますでしょうか？　意外とできないですよね。私はもちろん答えを知っているんですが、意外と難しいと思います。答えは17×19です。ここでのポイントは、17と19を知っていれば、２桁の掛け算ですが、323は簡単にできます。一方、323を知っていても、それを素因数分解することは、ちょっと難しい。難しそうだ。いやいや、323ぐらい、僕は簡単に解けたよという人が、オンラインの方でもいらっしゃるかもしれませんが、それでは、これを考えましょうか。

これは五つの数字があるのではなくて、５行にわたる一つの数字、174桁ある数字です。これを素因数分解しましょう。答えはこのようになります。

実はこの問題は、2003年にRSA社が、これを解いたら１万ドルをあげるよと。100万円ちょっとですね、そういう懸賞を出した問題です。当時は３カ月ぐらいやって解けたという問題です。だからそんなに簡単ではない。もちろん今のコンピューターは性能が良くなっていますけど、それでもなかなか難しい問題です。

これが公開鍵暗号と呼ばれる、暗号方式の一つの基礎で、つまり二つのものＡ、Ｂがある

とすると、片方のＡから片方のＢを計算することが易しくて、ＢからＡを計算するのが難しい。一方向性、one-way性と呼ばれるものです。それが公開鍵暗号の基礎で、素因数分解は、ある意味でRSA暗号の基礎になっています。

RSA暗号というのは、現在、使われている暗号の一つで、それをつくったRivest、Shamir、Adlemanの頭文字を取って、RSA暗号と呼ばれています。この暗号は理論的にも興味深いということで、彼らはこの暗号方式を提案したことにより計算機科学で重要なチューリング賞を受賞しています。

ただ、計算理論的にいうと、素因数分解は本当に難しいかどうかは、実はまだ分かっていません。現状では誰も速く解くことはできていないという問題です。

次は、計算の話をします。そもそも計算って何でしょうか？　計算、計算機、計算の可能性とは。この質問は、1930年代に盛んに研究されました。当時、いろんな妥当な計算モデルが提案されて、それに基づく計算の可能性が定義されました。計算のモデルというのは、計算機に対応します。

先ほどのコンパスと定規の話では、コンパスと定規は計算機ですよね。計算機というか、計算モデルが違えば、できることは変わってきますから、それぞれのモデルごとに計算可能性は定義できます。

問題ＡがモデルＭで計算可能であるとは、有限ステップで問題Ａを解く、モデルＭでのアルゴリズムが存在することです。モデルＭの下で問題Ａを解きたいが、有限ステップで終わるようなアルゴリズムがあるかないかが、計算可能性の定義です。

当時、いろいろと妥当なモデルが提案されましたが、全部等価だということが分かりました。これがChurch-Turingのテーゼです。当時の天才たち、Church、ゲーデル賞のGödel、Kleene、Post、Markov、先ほどのチューリング賞のTuringら、いろんな天才たちがいっぱいいて、様々な妥当な計算モデルと、それに基づく計算の可能性が提案されたんですが、全て等価であるということが分かりました。

等価というのは数学的な等価性なので、例えばチューリング機械とRAMとの等価性を証明するということは、チューリング機械で有限時間に停止するアルゴリズムがあったら、そのアルゴリズムからRAMのアルゴリズムをつくることができますよ、と証明する。逆にRAMのアルゴリズムがあったら、それからチューリング機械のアルゴリズムができますよ、ということを数学的に証明するということです。

当時、天才たちがいろいろとやっても、全部同じだということが分かったので、この「計算可能性」は非常に安定した概念ということで、現在も用いられています。

計算のお話をしましたけど、直感的には皆さんが持っているコンピューターでできることだと思ってください。もしプログラミングをしたことがある人でしたら、プログラミング言語は何でもいいですから、それで計算できること。つまり有限時間に止まるようなアルゴリズムが書けるか書けないか、と思ってください。もちろん現在の計算機を多少抽象化するこ

とにより数学的なモデルができる訳ですが、そんなに変わりません。

さて、計算の可能性が定義できたわけですがどんな問題でも計算できるか、性能が良いコンピューターを買えば何でも解ける、と思っている人が世の中には多いと思うんですけど、実はそうではありません。

その一例として、チューリングマシンの停止性問題と、ここに実は証明が書いてあるんですが、時間もないので省略します。もう一つ、Postの対応問題、これも計算できないことが知られています。この問題を紹介して、証明はしませんけど、なぜできないかという、説明をします。証明ではないです。説明をします。

問題は何かというと、文字列の集合ＡとＢが与えられます。ここに例を与えます。やりたいことは、ＡとＢそれぞれから文字列をi_1からi_mと、同じ順番で並べると、同じ文字列ができるか、あるいは、できないかを問う問題です。この例題では、YESとなります。

この問題は実は計算不可能です。もちろんこれは数学的に証明されていますから、どう頑張っても解けないんですけども、いや、できるじゃないかと思うかもしれませんよね。皆さんどうやって解きますか？　多くの人は並べる数ｍが１個だったら、a_1とb_1が等しいか、a_2とb_2は等しいかと、順番に見ていけばいい。

もし等しいものを見つければYESと答えればいいし、そうでなかったら、長さ１の文字列ではないから、長さ２だろうと、ｍを２にするわけですね。それでa_1a_2とb_1b_2が等しいか。a_1a_3とb_1b_3は等しいかと、一つ一つ試す。このように順番にやっていければ、解けそうに見えますよね。実際にちょうど等しくなるような並べ方があれば、どこかの段階で見つかるわけです。

問題は、ない場合です。あるときは、どこかまでいけば、絶対に発見することができます。ただ、ない場合。例えばｍを１万まで試して、１万個を並べてみて揃わないから、もうこれでないだろうと言いきれるかというと、実はそうではなくて、ひょっとしたら１万１回目で発見できるかもしれない。

あるいは長さ１億までやったからもういいかと思うかもしれませんけど、そうではなくて、１億１回目に見つかるかもしれないということで、解がないときに、いつやめればいいかということが結論づけられない。だから計算できないということです。

これは証明ではなくて、あくまでも説明です。ということで、結局ずっとやめられないから、有限時間で停止しないということで、計算できない問題になります。

最後に、これは簡単ですけど、３以上の整数ｎを与えて、$x^n + y^n$がz^nになるような、正整数ｘ、ｙ、ｚがあればYES、なければNOと答える問題を考えます。ｎが２の場合は知っていますよね、ピタゴラスの定理、あるいは、三平方の定理と呼ばれるもので、$3^2 + 4^2 = 5^2$になっていると。ちょっと早口になっているのは、時間が押しているからで、お許しください。

ｎが３以上のとき、これもご存じの方が多いと思いますけれども、フェルマーの最終定理といって、実際にWilesによって1995年に解決されています。ｎが３以上の場合は存在しない、ということが知られています。証明は難しいですけど、私も読んだことはないですが。

この問題が計算可能か、計算不可能か、というと、これは簡単にできます。計算可能です。なぜならば、Print“NO”と、“NO”とだけ記述すればいいですね。だって、ないと知っているんですから。

ここで言いたかったことは、証明の難しさと計算の難しさとは全然別物ということです。もちろんフェルマーの最終定理を証明するということは非常に難しいことですが、いったん分かってしまえば、計算的には非常に簡単なものだということです。

さて、次に、計算の効率性。先ほど言いましたが、計算可能というのは有限ステップ、有限時間で終わるということです。有限時間ですが、有限ならばいいのでしょうか？　アルゴリズムの評価の仕方としては、二つの重要な指標があります。一つは時間量で、早く終わるという時間量。もう一つは領域量。メモリの量ですね、どれだけのメモリを使うかと。

例えばカーナビがつくられた当初というのは、メモリが非常に高価でしたので、できるだけメモリを使わないようなアルゴリズムをつくるか、ということが盛んに研究されましたけども、今はメモリが安価なので、あまり実用的に求められる場面が少なくなってきてはいます。

時間量について、今回はお話をします。もちろんアルゴリズムの良し悪しというのは、読みやすさとか、デバッグしやすさ。デバッグというのは、何か間違いがあったら、間違いを簡単に修正しやすい、そういう基準もあります。

今回お話する時間量ですが、この時間量は入力サイズを基準にして議論されます。例えば30人、高校生だと今はたぶん一クラス30人ぐらいかな。その30人ぐらいの「良い」座席表、ここで、何が「良い」かはさておき、30人の「良い」座席表を作ることと、100人のクラスの「良い」座席表を作ることを考えたら、100人のほうを作るほうが時間がよりかかるのは当然かと思います。この30、100という数字が入力サイズに対応します。この入力サイズを基準として、アルゴリズムが計算のために必要とする時間を見るということです。

これは私が好きな表です。ここに入力のサイズnが、縦に10、20……、1000と。実際の計算時間、入力nに対して、nステップでできますよとか、nの２乗ステップでできますよ、というのを横軸に書くと。

皆さんが大体持っておられる計算機は、10BIPS。BIPSというのは、billion instructions per secondということですから、１秒、カチッという間に10billion、100億回の計算をすると。コンピューターはやっぱりすごいですよね。１秒間に100億回計算する。

そうすると一番左の上は、nが10で、計算時間はnで10ですから、10を100億で割るということですね。だから１×10^{-9}秒でできる。すぐですね。nが20だったら、分かると思いますけど、ここが２になるわけですね。それではn^2は、というと、10^2は100ですから、100を100億で割ると。そうすると１×10^{-8}秒というふうに、ばーっと埋めると、こんな感じになっています。まあ許せる範囲ですよね。n^3は1000で0.1秒というのがありますけど、0.1秒でできると。

では、次に２のn乗を考えましょう。2^nという数字は、n人いて、n人の中からいいグループをつくりましょうとしたときに、グループの数というのは２のn乗個ありますよね。その

中から一番いいグループをつくるときに、全部試したと思ってください。そうしたらば、30までは許せますよね、1.1×10^{-1}秒。nが40になったら、2の40乗を100億で割ると、約1.8分になる。ちょっと許せなくなってきますよね。予約システムでカチッと押して1.8分待つというのは、嫌ですよね。

nが50だったら、31時間。もう許せないという人がいると思いますけれども、それは場合によっては許せるわけで、例えば工場のスケジューリングなどでは、1週間分を日曜日にスケジュールするということはよくあって、それは丸一日、計算機をぶん回して解を出して、次の1週間のスケジュールを出すということはあるので、そういう意味では、31時間は許せますけど、ちょっと許せない感じにはなってきました。

じゃあnが100だったら。生きていないですよね。2の100乗を100億で割ると、こんな数字です。カチッとやって、解けないかなと待っていたら、とんでもないですよね。

n!に至っては、こんな感じです。n!というのは、n人がいたら、全員の並べ方で一番いい並べ方をしましょうといったときに、全部を試したらこうなるということです。nが20で、20!で7.5年と。もう許せないですよね。30でもこんな、全然使い物にならない数字になっています。

この表を見ると、ここで違いがあるだろうと。nに対して多項式、nのc乗になっているのか、指数になっているのかということで、速いというのは多項式時間、nのc乗、多項式で解けて、遅いというのは指数時間必要とする。このように計算理論は展開されています。

もちろん分野によっては、ゲノムとかを扱う分野、あるいは、データマイニング分野では、非常に大きなデータを扱うような分野では、nの2乗ではちょっと遅すぎますから、n×ポリログn、n×log（n）の何とか乗ぐらいまでのアルゴリズムをつくろうという研究がされています。

この表を皆さんが持っているコンピューターが悪いんだろうと、スパコンを使えばいいじゃないかと言うかもしれませんけれども、所詮、京は10^{16}ですから、富岳はもうちょっと速いですけども、全然、太刀打ちできないですよね。実際に解きたい問題は、数百万変数の問題を解きたかったりするわけですから、解けないということです。

あと残り時間が2分ぐらいですけれども、まとめると、日本はものづくりといって、どっちかというとハードウエア、いいコンピューターをつくることが重要だよと言っている人が多いんですが、実はソフトウエア、アルゴリズムも重要で、その進化のほうが、ハードウエアの進化よりも勝っている場合もあります。

もちろんアルゴリズムの進化だけでは不十分で、ソフトウエアとハードウエア、両方の進化がないといけない。いいコンピューターも必要だし、いいアルゴリズムも必要だよ、ということが言いたいです。

最後にこの問題を紹介して、終わりたいと思います。巡回セールスマン問題、traveling salesman problemと、頭文字を取ってTSPという問題です。この問題は何かというと、頂点、この場合は4点と、それを結ぶ辺、その辺には長さが20、6という数字が付いてて、距離あ

るいは、移動時間を表すものだと思ってください。セールスマンが、自分の事務所からいろんな顧客を周り、事務所に戻ってくる、その際に必要となるトータルの距離あるいは、時間が短い、ハミルトン閉路を求める問題です。ここで、ハミルトン閉路というのは、全ての頂点をちょうど1回ずつ周るような閉路です。

実は、もう省略しますが、この問題が多項式時間で解けるかどうかは分かっていません。これはClay数学研究所のホームページをコピーしたものです。このClay数学研究所のここにMillennium problemがあるので、ここをクリックすると、こうやって出てきます。

これを見てやると、Millennium、2000年の5月にパリで、数学で七つの未解決問題を出しましたと。七つのうち1題でも解ければ100万ドル、1億円ちょっとをあげますよという問題を出したというところです。その中の一つがP vs NP problemという問題です。実は7のうち1題はすでに解かれていますけども、まだ残っている問題です。

ここをクリックすると、読みませんけども、意訳すると、意訳というか、数学的には正しいことです。先ほどのTSPという問題が多項式時間、多項式というのはグラフのサイズ：頂点数と枝数に対して、多項式時間で解ける、あるいは解けないということを証明すれば、100万ドルがもらえるという問題です。何となくお金のためにやっているというふうに誤解して欲しくないです。それぐらい解くことが難しい問題ということです。

ということで、終わりたいと思います。今日の話は、計算とかアルゴリズムのお話をして、ハードウエアとソフトウエアの、両方の進歩は非常に著しい。ただし何でも解けるわけではなく、非常に難しい数学の問題を含んでいる、ということをお話させていただきました。本講演を終わりたいと思います。ありがとうございました。