実践講座

スケジューリング問題の新解法

(1)：スケジューリング問題と計算の複雑さ

茨木　俊秀

1. はじめに

冒頭から恐縮であるが、この「実践」講座のしょっぱなに、計算の複雑さの話を持ってくるのはまずいという気がしないではない。というのは、計算の複雑さの理論の主要なテーマは、「 ... の問題を解くことが本質的に困難であることを数学的に証明する」というものであるからである。残念ながら、スケジューリング問題についても、ごく簡単なものを除いて、ほとんどが「本質的に難しい」ことが分かっている。こう書くと、「最適解を求めるのは困難でも、現実には近似解でよいのだから心配ない」とおっしゃる方もあろう。たしかにこれは問題の一面を正しく捉えているが、かなりの問題について、理論的に保証された性能を持つ近似アルゴリズムを作ることが、最適解を求めるアルゴリズムを作るのと同じくらい難しい、ということも判明してきている。

したがって、「難しくても何であっても、ともかくスケジュールを作らねばならない」という現場にたいし、はなはだ情けない情報しか提供できないことになってしまう。しかし、ものは考え様である。困難であることが真実ならば、それを正しく認識することから現実への対処が始まる。そのための関門である、と捉えていただければ、私の気も軽くなる。以下、その精神で、効率よく解ける問題についての説明、また困難な問題についてそれを克服するために試みられている最近の動きにも触れるようにしたい。オーガナイザーの木瀬洋先生の意図もそこにあるのだと思う。

いばらき　としひで　京都大学工学部
〒606 京都市左京区吉田本町

2. スケジューリング問題の分類

「分かる」ことと「分ける」ことが通じているように、物事の科学的理解は対象の体系的な分類から始まる。スケジューリングも例外ではなく、複雑さの理論を展開するには分類法の確立がまず必要であった。この目的に大きな貢献をしたのは、E. Lawler, J. K. Lenstra および A. H. G. Rinnooy Kan らである。彼らは、待ち行列のケンドールの記号にならって、

$$\alpha|\beta|\gamma$$

という記法を考案した。

一般に、スケジューリング問題では、m 台の機械 (machine) で n 個の仕事 (job) を処理する。機械 M_i で仕事 J_j を処理するために要する時間 p_{ij} (1機械ならば単に p_j)、および仕事の重要度を示す重み w_j は既知とする。(これらを確率変数と考えると別の話題になる。) 上の記法の α は機械と仕事の関わり具合を記述するもので、普通 $\alpha = \alpha_1\alpha_2$ の二つのフィールドからなっている。まず、

$$\alpha_1 \in \{\circ, P, Q, R, O, F, J\}$$

であるが、それぞれは

- $\circ$：1機械（記号 $\circ$ は他の文字とつなぐときは省略）
- P：同一並列機械（同じ性能の機械が m 台）
- Q：一様並列機械（機械の性能は一つのパラメータで決まる）
- R：無相関並列機械（m 台の機械の性能は互いに独立）
- O：オープンショップ
- F：フローショップ
- J：ジョブショップ

などの意味を持つ。(紙数の都合上、それぞれの用語の説明は省略するが、スケジューリング理論の標準的なものばかりなので、お許し願いたい。)

α_2 には1とか2とか機械の台数が入るが、変数ならば m と書くか省略する。

つぎの β は、仕事の特性や、処理順序に関する制約などを記述するものである。代表的な記号をあげると、

$pmtn$ (preemption 仕事の中断と続行が許される)、
$prec$ (precedence relation 仕事間の先行関係(半順序)が制約として与えられる)、
r_j (仕事 J_j は準備時間 r_j を持つ)、
$p_j = 1$ (すべての仕事の処理時間は単位時間1)、
...

などである。デフォルト値は、preemption なし、先行関係なし(任意の処理順序が許される)、準備時間0、処理時間 p_{ij} は任意、重み w_j は任意、などのように定められている。

最後の γ は目的関数を記述するものである。仕事 J_j の納期 d_j と完了時刻 C_j に基づいて、

$$L_j = C_j - d_j \text{ (lateness 遅れ時間)}$$
$$T_j = \max\{0, C_j - d_j\} \text{ (tardiness 延滞時間)}$$
$$U_j = 0 \text{ if } C_j \le d_j, 1 \text{ if } C_j > d_j$$

を定義し、目的関数

$C_{\max} = \max_j C_j$ (makespan 最大完了時刻、メイクスパン)
$L_{\max} = \max_j L_j$ (maximum lateness 最大納期遅れ)
$\sum C_j$ (flow time 滞留時間)
$\sum w_j C_j$ (重み付き滞留時間)
$\sum w_j T_j$ (重み付き総延滞時間)
$\sum w_j U_j$ (重み付き遅れ仕事数)
...

などがよく用いられる。もちろん、これらを最小化するスケジュールが要求されるのである。

これらのパラメータを組み合わせると、たとえば、つぎのような例が得られる。

$1|\text{prec}|L_{\max}$: 1機械スケジューリング問題で、最大納期遅れを最小にする。ただし、仕事間には、技術的な制約から定まる先行関係が付されている。

$F2||C_{\max}$: 2機械フローショップ問題。すべての仕事が完了する時刻(メイクスパン)の最小化が求められている。

つまり、α, β, γ を適当に選ぶだけで、きわめて多数のスケジューリング問題を簡単かつ正確に表現できるのである。必要ならば新しい記号を導入すれば、さらに多様な問題を扱うことも可能である。Lawler, Lenstra と Rinnooy Kan らは、この分類法にしたがって、それぞれの問題の計算の複雑さを明らかにする作業にとりかかり、組合せスケジューリング理論という一つの分野を作り上げていった。

しかし、この記法で、いわゆるスケジューリング問題の全体を網羅できているわけではないことも強調しておきたい。CIM や FMS などのキーワードとともに語られる最近の生産システムには、α, β, γ のパラメータだけでは記述できない複雑な側面がある。たとえば、生産スケジューリングにおける目的関数として重要である中間在庫の量を記述するには、機械と機械の間に位置するバッファや機械間の仕事の移動をつかさどる搬送車を定式化に含めねばならないだろう。さらに、これらの機械の操作や管理にたずさわる作業者に関するマンパワーのスケジューリング、生産ロットの適正サイズを決定する問題、機械に装着する工具の取り換え問題、等々考慮すべきファクターはいくらでもある。上の記法を拡張することで、これら広い対象をある程度含めることは可能であろう。しかし、生産システムは刻々変化していく生き物のようなものであるので、一つの言語で捉えるのは所詮不可能と達観するのが正解という気もする。

以下、本講座では、上の記法の枠内にある「古典的な」スケジューリング問題のうち、いくつかを取りあげ、計算の複雑さの観点から眺めることにする。

3. 効率よく解けるスケジューリング問題

スケジューリング問題の中には、簡単なルールで正確な最適スケジュールが求まるものも結構ある。本節でそのいくつかを紹介する。証明は省略するが、いずれも組合せ最適化問題の練習問題として手ごろなものであるので、試みてみられることをおすすめする。

$1||\sum w_jC_j$：1 機械スケジューリング問題。各仕事 J_j は処理時間 p_j と重み w_j をもつ。仕事間の先行関係や準備時間はない。重み付き滞留時間を最小にする。

答：n 個の仕事 J_j を p_j/w_j の小さいものから順に処理する。（これは Smith のルールと呼ばれている。簡単のため $w_j=1$ の場合を考えると、要するに早く済むものから片づけることを意味する。）

$1||L_{\max}$：上の問題と同様。ただし、各仕事は納期 d_j をもち、最大納期遅れを最小にする。

答：d_j の小さいものから順に処理する。（これは、我々が日常的に置かれている状況である。多くの人はこのルールを体験的に会得している。しかし、これはいくつかの仕事の納期遅れが避けられないという厳しい状況では、遅れ仕事数を最小にするとは限らない。次項参照。）

$1||\sum U_j$：上の問題と同様。ただし、遅れ仕事数を最小にする。

答：仕事を納期 d_j の小さいものから順に並べて行く。ただし、ある仕事 $J_{j'}$ が遅れることが判明すれば、$J_{j'}$ のかわりに、それまでにスケジュールされた仕事の中から処理時間最大のものを求め、遅れ仕事としてスケジュールから除く。以上の手順を、すべての仕事が考慮されるまで反復する。

$R|pmtn|C_{\max}$：並列機械問題。ただし、preemption を許す。

答：つぎの線形計画問題を解く。式中の x_{ij} は、仕事 J_j を機械 M_i で処理するために費やす時間を表す。

目的関数： $C_{\max} \longrightarrow$ 最小
制約条件

$$\sum_{i=1}^{m} x_{ij}/p_{ij}=1,\quad j=1,2,\ldots,n$$
$$\sum_{i=1}^{m} x_{ij}\le C_{\max},\quad j=1,2,\ldots,n$$
$$\sum_{j=1}^{n} x_{ij}\le C_{\max},\quad i=1,2,\ldots,m$$
$$x_{ij}\ge 0,\quad i=1,2,\ldots,m,\ j=1,2,\ldots,n$$

$F2||C_{\max}$：2 機械フローショップ問題。すなわち、各仕事 J_j は M_1 と M_2 の順序で機械を通る。処理時間はそれぞれ p_{1j} と p_{2j} である。

答：つぎのアルゴリズムにしたがって、仕事の順序 $\sigma=\sigma_1\sigma_2$ を定める。

1. $\sigma_1:=\epsilon$（空系列）, $\sigma_2:=\epsilon$, $P:=\{p_{1j},p_{2j}|j=1,2,\ldots,n\}$ と置く。

2. P の中で最小の値をもつ p_{ij} を見つける。もし $i=1$ ならば $\sigma_1:=\sigma_1J_j$ とする。一方、$i=2$ ならば $\sigma_2:=J_j\sigma_2$ とする。

3. p_{1j} と p_{2j} を P から除く（ただし、j はステップ 2 で得られたもの）。$P=\emptyset$ ならば終了。$P\neq\emptyset$ ならばステップ 2 へ戻る。 □

これは、$\min\{p_{1j},p_{2j'}\}\le\min\{p_{2j},p_{1j'}\}$ ならば、最適スケジュールにおいて、J_j が $J_{j'}$ に先行してよいという、いわゆる Johnson ルールに基づいている。（スケジューリング理論の古典的な結果で、彼の 1954 年の論文にある。）

簡単なルールで解けるこれらの問題は、最適スケジュールを求めるアルゴリズムの所要時間を n の多項式オーダー（ある定数 k を用いて $O(n^k)$）で抑えることができるという性質をもっている。たとえば、最初の二つは、あるパラメータにしたがって n 個の仕事を整列するだけであるから $O(n\log n)$ 時間で実行できる。$1||\sum U_j$ と $F2||C_{\max}$ は、それほど自明ではないが、やはりデータ構造を工夫すれば、$O(n\log n)$ 時間でよい。$R|pmtn|C_{\max}$ の線形計画問題も、Khachiyan や Karmarkar の結果としてよく知られているように、多項式時間で解ける。

多項式時間で解けるスケジューリング問題は、これら以外にもいろいろあるが、やはりかなり特殊で簡単な場合に限られるという印象である。

4. 困難なスケジューリング問題

いくつかの代表的なスケジューリング問題に対する計算の複雑さの結果を表 1 にまとめる。注目すべきは、前節の問題をすこし一般化すると、すぐ NP 困難になってしまうという事実である。NP 完全性や NP 困難性の概念についてはよく知られるようになってきているので、改めて説明しない。要は、ある問題が NP 困難であるとは、その問題のすべての問題例を効率よく解くアルゴリズムは存在しない（より厳密には、多項式オーダー時間では解けない）ことを意味する。（これも、正確には予想であって、証明されている訳ではない。有名な P ≠ NP 予想である。）

たとえば、1 機械のスケジューリング問題では、滞留時間を最小化するとき、仕事間の先行関係や準備時間が入ると NP 困難になる。ジョブショップやフローショップでは、機械数が 3 になるともう駄目である。並列機械の場合も、一般的には機械数 2 ですでに NP

困難である。

現実のシステムに登場するスケジューリング問題では、機械数も多く、さらに、いろいろな理由で複雑な制約条件が付加されるのが普通である。したがって、まず、NP困難であることを覚悟し、簡単なルールで厳密な最適スケジュールを得ることはあきらめた方がよさそうである。

ただ、効率よく解ける場合は非常に限られているとは言え、そのような問題についての十分な知識をもつことは、実用上も重要であることを強調しておきたい。現実の複雑なシステムであっても、皮相的な部分を単純化し、本質的なところのみを取り出せば、案外簡単な問題として捉えることができることが多いものである。もし、単純化された問題が厳密に解けるものであれば、その最適スケジュールをまず求め、そのあとで実際の制約や目的関数に合わせてスケジュールを手直しするという手順をとることができる。このようなアプローチが、実際に有効な手段になっている例は少なくない。

表1 代表的なスケジューリング問題の複雑さ

1機械	
問題	複雑さ
$1\|\|\sum w_jC_j$	$O(n\log n)$
$1\|\|L_{\max}$	$O(n\log n)$
$1\|\|\sum U_j$	$O(n\log n)$
$1\|\|prec\|C_{\max}$	$O(n^2)$
$1\|r_j,p_j=1\|L_{\max}$	$O(n\log n)$
$1\|pmtn,r_j\|\sum C_j$	$O(n\log n)$
$1\|pmtn,prec,r_j\|C_{\max}$	$O(n^2)$
$1\|\|\sum T_j$	NP困難
$1\|r_j\|\sum C_j$	NP困難
$1\|r_j\|L_{\max}$	NP困難
$1\|prec\|\sum C_j$	NP困難
$1\|prec,p_j=1\|\sum w_jC_j$	NP困難
$1\|pmtn,r_j\|\sum w_jC_j$	NP困難
並列機械	
問題	複雑さ
$P\|\|\sum C_j$	$O(n\log n)$
$P\|p_j=1\|\sum w_jC_j$	$O(n^3)$
$Q\|p_j=1\|C_{\max}$	$O(n^3)$
$Q\|pmtn\|\sum C_j$	$O(n\log n+mn)$
$R\|pmtn\|\sum w_jC_j$	LP問題
$P2\|\|C_{\max}$	NP困難
$P2\|\|\sum w_jC_j$	NP困難
$P2\|pmtn\|\sum w_jC_j$	NP困難
$P\|prec,p_j=1\|C_{\max}$	NP困難
$P\|prec,p_j=1\|\sum C_j$	NP困難
ジョブショップ、フローショップなど	
問題	複雑さ
$J2\|\|C_{\max}$	$O(n\log n)$
$F2\|\|C_{\max}$	$O(n\log n)$
$O2\|\|C_{\max}$	$O(n)$
$J3\|\|C_{\max}$	NP困難
$F3\|\|C_{\max}$	NP困難
$O3\|\|C_{\max}$	NP困難

5. 厳密アルゴリズムと近似アルゴリズム

対象とする問題がNP困難であっても、どうしても厳密に解きたい、あるいは解かねばならないという状況もある。問題の持つ固有の構造を利用して計算効率の向上を図れば、NP困難問題であっても、現実規模の問題例を実用的に解き得ることが可能かもわからない。NP困難性は最悪の場合の解析に基づく理論なので、あらゆる問題例を効率よく解くことはできないとしても、目の前にある問題例を解き得る可能性が否定されたわけではないからである。この目的に用いられるアプローチに分枝限定法（branch-and-bound method）がある。

分枝限定法は、与えられた問題を、場合分けによって細分化していくことを基本にする、一種の列挙法である。ただし、細分化によって生成された部分問題に簡単なテストを加え最適解を与えないことが判明すればただちに終端するという限定操作によって、列挙の数を減らし、実用性を高める点に特徴がある。このために用いるテストを考案するため、条件を緩和することで解き易い問題に変形するという手段がよくとられる。この緩和問題の最適スケジュールが求まれば、それだけ限定操作の効果も上がるので、この目的にも「解ける」問題についての知識を蓄積しておくことが重要である。

分枝限定法によるスケジューリング問題の解法については、これまで多数の論文が書かれている。実際、1機械スケジューリング問題やジョブショップ、フロー

ショップなどでは、かなりの成果をあげていて、実用化されているケースもある。本講座でも取り上げられる予定になっている。

さて、いろいろ試みてみたが厳密に解くのはやはり難しい、あるいはともかく高速に解きたい、ということになると、いよいよ近似アルゴリズムの出番である。ヒューリスティック（発見的）アルゴリズムとも呼ばれるように、問題についての知識をうまく利用して、効率よく（多項式オーダー時間で）、最適解にできるだけ近い近似解を構成することを目的としている。

理論の立場からすると、この場合でも、近似アルゴリズムによって得られた値 OBJ と真の最適値 OPT の比

$$R = \mathrm{OBJ}/\mathrm{OPT} \quad (\geq 1)$$

についての何らかの保証がほしい。たとえば、ある定数 $\epsilon\ (>1)$ に対し、常に $R \leq \epsilon$ の成立することが言えれば有り難い。このようなアルゴリズムを ϵ-近似アルゴリズムと呼んでいる。

たとえば、$P||C_{\max}$ に対する有名な Graham のアルゴリズム（1969 年）は、処理時間 p_j の大きい仕事から順に、その時点での完了時刻最小の機械に割り当てていく、というものである。その結果、

$$R \leq \frac{4}{3} - \frac{1}{3m}$$

が証明できる。

あるいは、$R \leq \epsilon$ が常に成立することは言えなくても、確率的に保証された形で成立すれば、やはりかなり有り難い。このように、

厳密解 ⟶ 近似解 ⟶ 確率的近似解

という風に目標を緩めると、現実的に扱い得る問題の範囲が広がることは確かである。

しかし、最近の研究によれば、近似を許しても、ある意味では五十歩百歩であるという結果も出ている。たとえば、どのような定数 $\epsilon\ (>1)$ を定めても、ϵ-近似アルゴリズムを作ることはできない問題が存在する（$\mathrm{P} \neq \mathrm{NP}$ の前提で）。また、ある ϵ_1 という定数に対しては ϵ_1-近似アルゴリズムを作れるが、もう一つの定数 $\epsilon_2\ (<\epsilon_1)$ が存在して、それより小さい ϵ については ϵ-近似アルゴリズムを作れない、というタイプの問題もたくさん存在する。（このような問題のクラスは MAX SNP と呼ばれ、最近大きな話題になった。）いずれもどちらかといえば否定的結論である。

さらに、確率的挙動の研究についても、確率アルゴリズム (randomized algorithm) との関連のもとに、大きな進歩が見られる。最近のホットな話題の一つと言ってよい。しかし、これも、扱える範囲が若干広がるだけで、理論的な評価をきちんと行おうとすると、その先ではすぐ上と同様の否定的結論に至るようである。

6. メタ・ヒューリスティクス

スケジューリング問題を含む組合せ最適化の分野における最近のトレンドは、メタ・ヒューリスティクス (meta heuristics) にある。メタと付いているのは、近似アルゴリズムを実現するために有効な個々のヒューリスティクス（発見的知識）を有機的に統合するための枠組みを論じているからである。話題になっている枠組みには、

局所探索法 (local search)
アニーリング法 (simulated annealing)
タブー探索法 (tabu search)
遺伝アルゴリズム (genetic algorithm)
ニューラルネットワーク (neural network)

などがある。これらの詳細は、本シリーズでも、スケジューリングとの関わりのもとに逐次扱われることになっているので、ここでは名前だけにしておく。

ただし、計算の複雑さの理論の観点からこれらの枠組みを眺めると、ほとんど何も解明されていないと言わざるを得ない。まず、これらの枠組みに立って作られたアルゴリズムの計算量がどの程度になるのか分からないし（少なくとも多項式オーダー時間であるとは言えそうもない）、ごく部分的な結果を除いて、計算される近似解の精度についての定量的な評価がなされているわけでもない。得られているのは、計算実験による経験的なデータのみであって、それも不完全なものが多い。理論の立場からするとこれは大変残念な状況である。ただ、報告されている実験データには有望なものが多い。物事の最初はすべてこのようなものであるかもしれないが、今後、これらのデータの客観的評価を可能にするような理論が発展していくことを期待したいものである。

いずれにしても、これら新しい枠組みの中に本当に役立つものがあるのか、もしそうなら、どれが有効なのか、スケジューリング問題に限ればどうなのかなど、基本的な疑問も含めて、この分野はしばらく話題

の中心になりそうである。

7. むすび

スケジューリング、さらにそれを含む組合せ最適化の分野において、計算の複雑さの理論は何を教えてくれているのだろうか。最適解はもとより、近似解を求めることすら困難であるという事実は、逆説的に言えば、組合せ最適化がそれ自体の内部構造として、非常に豊富で深い内容を秘めていることを示しているのだと言って良いだろう。他の数学分野、たとえば、ソリトン、フラクタル、カオスといった魅力ある話題をつぎつぎと提供し続けている非線形システムがそうであるように、汲めども尽きない豊かな泉であるに違いない。スケジューリング理論が始まってまだせいぜい半世紀、ゆっくり時間をかけて楽しもうではありませんか。

参考文献

まず、スケジューリング理論の教科書として、古典ともいうべき [1,3] と最近の [2,9] などをあげておく。[9] はスケジューリング問題の近似アルゴリズム（ヒューリスティック解法）にも詳しい。スケジューリング問題の分類についての詳細な内容は [8] にある。NP困難性および計算の複雑さの理論は [6,7] などを参照のこと。[5] では近似アルゴリズムの最近の理論的成果を解説した。メタ・ヒューリスティクスの解説はたとえば [4] にある。

[1] Baker K.R. : "Introduction to Sequencing and Scheduling", Wiley, New York (1974).

[2] Blazewicz J., Ecker K., Schmidt G. and Weglarz J. : "Scheduling in Computer and Manufacturing Systems", Springer-Verlag, Berlin (1993).

[3] Conway R.W., Maxwell W.L. and Miller L.W. : "Theory of Scheduling", Addison-Wesley, Reading Mass. (1967).

[4] 茨木俊秀: "組合せ最適化の手法 —巡回セールスマン問題の例から—", 電気学会論文誌C, pp. 411-419(1994).

[5] 茨木俊秀: "組合せ最適化と計算量", 人工知能学会誌, pp. 335-341(1994).

[6] 茨木俊秀: "アルゴリズムとデータ構造", 昭晃堂(1989).

[7] 笠井琢美, 戸田誠之助: "計算の理論", 共立出版(1993).

[8] Lawler E., Lenstra J.K. and Rinnooy Kan A.H.G.: "Recent developments in deterministic sequencing and scheduling: A survey", Deterministic and Stochastic Scheduling, edited by Dempster M.A.H., Lenstra J.K. and Rinnooy Kan A.H.G., Reidel Publishing Co. (1982).

[9] Morton, T.E. and Pentico D.W.: "Heuristic Schduling Systems", John Wiley, New York (1993).