システム/制御/情報, Vol. 54, No. 10, pp. 385–391, 2010

解　説

1機械スケジューリング問題に対する厳密解法の研究動向

田中　俊二*

1. はじめに

機械や設備などの資源を用いて与えられた仕事を処理する際の作業スケジュールを策定する問題は生産スケジューリング問題とよばれ，古くから盛んに研究されてきた．生産スケジューリング問題は一般に組合せ最適化問題として扱うことができるが，そのほとんどがNP困難であり，厳密解が求まるのは小規模な問題のみであった．しかし近年，解法の研究の発展と計算機性能の向上により，ある程度現実的な規模の問題に対しても厳密解が求まるようになってきている．そこで本稿では，生産スケジューリング問題の中でもとくに研究が発展している1機械スケジューリング問題について，厳密解法に関する最近の研究動向および筆者らの取り組みを紹介する．

2. 1機械スケジューリング問題

まず，本稿で扱う1機械スケジューリング問題の概要を述べておく．

1台の機械(machine)を用いて，与えられたn個の仕事(job)を処理することを考える．各仕事i $(1\leq i\leq n)$について，処理にかかる時間（処理時間; processing time）p_iはあらかじめ与えられているものとする．また，機械は同時に複数の仕事を処理できず，さらに，仕事の処理は中断・分割できない(nonpreemptive)とする．このとき，何らかの目的関数を最適化するよう各仕事iの完了時刻(completion time) C_iを決定する問題が，対象とする1機械スケジューリング問題である．ただし，これは1機械スケジューリング問題の基本的な要件であり，実際には，処理開始可能時刻(release date)や段取り時間(setup time)など，付加的な要件を考慮する場合が多い．これらに関しては後述することにする．また，処理時間が変更可能(controllable)である問題[1]や処理時間に学習効果(learning effect)を考慮した問題[2]なども研究されているが，本稿では扱わない．

1機械スケジューリング問題で用いられる目的関数は，完了時刻C_iに依存するコスト関数を$f_i(C_i)$として，その和を取ったもの$\sum_i f_i(C_i)$と最大を取ったもの$\max_i f_i(C_i)$に大別される．いずれの場合でも最小化を行うことが一般的であるため，ここでは，前者をmin-sum型，後者をmin-max型とよぶものとする．min-sum型の代表例としては，

- 完了時刻和(total completion time) $\sum_i C_i$
- 納期遅れ和(total tardiness) $\sum_i T_i$

などが，一方，min-max型としては，

- 最大完了時刻(makespan) $\max_i C_i$
- 最大納期遅れ[1](maximum lateness) $\max_i L_i$

などがある．ただし，

$$T_i=\max(C_i-d_i,0),\quad L_i=C_i-d_i \tag{1}$$

であり，d_iは仕事iの納期(duedate)を表す．多機械問題であるジョブショップ(jobshop)やフローショップ(flowshop)問題などではmin-max型の最大完了時刻が用いられることも多いが，1機械問題ではmin-sum型の目的関数がよく用いられる．この理由として，min-sum型よりもmin-max型の方が一般に最適解を求めるのが容易であることがあげられる．たとえば，処理開始可能時刻や段取り時間など付加的な条件がない場合，最大完了時刻の最小値は自明$(\sum_i p_i)$であり問題として成立しない．一方，完了時刻和の最小値は，処理時間の非減少順(Shortest Processing Time; SPT)に仕事を並べることで求まるものの，少なくとも自明ではない．

そこで，本稿でもmin-sum型の目的関数を用いることにして，つぎに正規(regular)・非正規(nonregular)という分類を与えておく．各仕事の完了時刻C_i $(1\leq i\leq n)$に関して単調非減少である目的関数は正規目的関数，それ以外は非正規目的関数とよばれる．先にあげた完了時刻和，納期遅れ和などはいずれも正規目的関数である．正規目的関数を最小化する際には，前詰めのスケジュール，すなわち，できるだけ仕事を早く処理するスケジュールのみを考えればよい．よって，仕事の処理順序を決定すればスケジュールは一意に定まる．一方，非正規目的関数の場合，機械が仕事を処理していない時間（遊休時間; idle time）をあえて作ることで目的関数値を改善できる可能性がある．このため，仕事の処理順序と遊休時間の両方を決定する必要がある．非正規目的関数の代表例に

* 京都大学大学院　工学研究科

Key Words: single-machine scheduling problem, exact algorithm, Lagrangian relaxation, dynamic programming.

[1] latenessは「納期ずれ」と訳すことが多いが，earliness-tardinessとの混同を避けるため，ここでは「納期遅れ」としておく．

は納期ずれ和 (total earliness-tardiness) $\sum_i(E_i+T_i)$ がある．ただし，E_i は納期進み[2](earliness) であり，

$$E_i=\max(d_i-C_i,0) \tag{2}$$

で定義される．納期ずれ和を最小化するには，各仕事をなるべく納期ちょうどに完了すればよい．

最後に，Graham らの three field notation[3] を紹介しておく．これは，生産スケジューリング問題の分類に標準的に用いられる記法で，「$\alpha|\beta|\gamma$」の α の部分に機械の条件を，β には仕事の条件を，そして γ に目的関数を記述する．たとえば，1機械納期遅れ和最小化問題は $1||\sum_i T_i$ と表される．また，各仕事 i に対し，処理を開始できる時刻 (処理開始可能時刻) $r_i\geq 0$ が与えられている場合には $1|r_i|\sum_i T_i$ となる．

3. 1機械スケジューリング問題に対する厳密解法の研究動向

前節で述べた1機械スケジューリング問題はそのほとんどがNP困難であるため，最適解を高速に求めるのは難しいと考えられている．実際，厳密解法の研究が盛んに行われてきたものの，数十仕事程度の規模の問題を解くのがやっとであった．しかし最近になって，より規模の大きい問題に適用可能な厳密解法が提案されている．本節では，代表的な1機械スケジューリング問題について，これまでに提案されている厳密解法を紹介する．

3.1 納期遅れ和最小化 ($1||\sum_i T_i$, $1|r_i|\sum_i T_i$)

納期遅れ T_i の和 $\sum_i T_i$ を最小化する問題である．処理開始可能時刻 r_i に制限がない，すなわち，$r_i=0$ である問題 $1||\sum_i T_i$ に関しては，古くからさまざまな研究が行われている．実際，2仕事間の処理順序関係の最適性に関する十分条件が Emmons により示された [4] のは1960年代である．これらの条件は Emmons の優越条件 (Emmons' dominance conditions) とよばれ，その後のスケジューリング問題に対する研究全般に多大な影響を与えている．たとえば，分枝限定法による厳密解法において，問題に応じた優越条件が分枝数の削減に利用される．

$1||\sum_i T_i$ に対するもう一つの重要な結果は Lawler による分解定理 [5](Lawler's decomposition theorem) である．本定理によりもとの問題を部分問題へ分解することが可能となり，求解の困難さを大幅に低減することができる．Szwarc らの分枝限定法 [6] は，問題分解と優越条件を巧みに利用することで最大500仕事の問題に対して最適解を求めることに成功している．

一方，処理開始可能時刻 r_i に制限がある $1|r_i|\sum_i T_i$ は，問題分解を適用することができず，$1||\sum_i T_i$ ほど効率よく最適解を求めることができない．最近提案された文献 [7] の分枝限定法では，過去に生成された子問題を記憶しておき優越条件のテストに利用する，という工夫を施しているが，最適解が求まるのは最大でも100仕事程度の問題である．

3.2 重み付き納期遅れ和最小化 ($1||\sum_i w_i T_i$, $1|r_i|\sum_i w_i T_i$)

納期遅れ T_i の加重和 $\sum_i w_i T_i$ を最小化する問題である．前項で述べた Lawler の分解定理は，重み w_i が agreeable，すなわち，$p_i<p_j$ なら $w_i\geq w_j$ という条件を満たす問題でしか成立しない．このため，この条件を満たさない一般的な $1||\sum_i w_i T_i$ については，数十仕事程度の問題しか解けない時期が長く続いたが，2007年に Pan と Shi の分枝限定法 [8] が OR-Library[9] の100仕事のベンチマーク問題をすべて解くのに成功した (計算時間は Pentium4 2.8GHz のコンピュータ使用で最長9時間)．この解法は，時間の離散化に基づく定式化 (time-indexed formulation) の連続緩和問題を等価な問題 (max-min transportation problem) に変換することで下界値計算の高速化を図っている点に特徴がある．なお，時間の離散化に基づく定式化は **4.1** で詳述する．

一方，処理開始可能時刻 r_i に制限のある $1|r_i|\sum_i w_i T_i$ についてはあまり研究が進んでいない．Jouglet らの分枝限定法 [10] では，各種の優越条件のテストを組み込んで効率化をはかっているものの，求解可能な問題の規模は最大30仕事程度である．

3.3 重み付き完了時刻和最小化 ($1|r_i|\sum_i w_i C_i$)

完了時刻 C_i の加重和 $\sum_i w_i C_i$ を最小化する問題である．処理開始可能時刻 r_i に制限がない場合，w_i/p_i の非増加順 (WSPT; Weighted Shortest Processing Time) に仕事を並べたスケジュールが最適となることはよく知られている．しかし，処理開始可能時刻に制限がある場合はNP困難となる．この問題に対しては，動的計画法と分枝限定法，さらには制約伝搬の考え方を組み合わせた解法 [11] が提案されており，最大200仕事の問題を解くことに成功している．ただし，最長で4日近くを要している (Pentium4 2.8GHz のコンピュータを使用)．

3.4 順序依存段取り時間を考慮した完了時刻和最小化 ($1|s_{ij}|\sum_i C_i$)，納期遅れ和最小化 ($1|s_{ij}|\sum_i T_i$)

段取り時間 (setup time) が順序依存 (sequence-dependent) とは，段取り時間が直前に処理した仕事と今処理しようとしている仕事に依存することを指す．たとえば，仕事 i を処理する際には，直前に処理した仕事を仕事 j とすると，段取り時間 s_{ji} が必要となる．また，仕事 i が最初に処理する仕事の場合，段取り時間は s_{0i} で与えられる．

Bigras ら [12] は，このような段取り時間を考慮した完了時刻和最小化問題 ($1|s_{ij}|\sum_i C_i$)，納期遅れ和最小化

[2] earliness の訳語は定まっていないようであるが，ここでは「納期進み」としておく．納期余裕 (slack) と同じ意味をもつため，「納期余裕」とよばれることもある．

問題 $(1|s_{ij}|\sum_i T_i)$ に対し，複数の下界値計算法の比較を行っている．その下界値計算方法は少々複雑であるが，簡単にまとめると

- 集合分割問題として定式化する
- その連続緩和問題を列生成法を適用して解く
- 列を生成する際に制約を追加して下界値を改善
- 主問題にカットを追加することで下界値を改善

という方法である．彼らは，この下界値を分枝限定法に組み込むことで，$1|s_{ij}|\sum_i C_i$ は最大50仕事，$1|s_{ij}|\sum_i T_i$ は最大45仕事の問題を，それぞれ解くことに成功している．ただし，後者の問題を解く際には最長で1週間以上かかっている（Pentium4 3.4GHzのコンピュータを使用）．

なお，$1|s_{ij}|\sum_i T_i$ に対するBigrasらの解法は重み付きの問題 $1|s_{ij}|\sum_i w_i T_i$ に対しても適用可能であるものの，数値実験は行われていない．

3.5 重み付き納期ずれ和最小化 $(1||\sum_i(\alpha_i E_i+\beta_i T_i)$, $1|r_i|\sum_i(\alpha_i E_i+\beta_i T_i))$

納期ずれの加重和 $\sum_i(\alpha_i E_i+\beta_i T_i)$ を最小化する問題である．非正規目的関数の代表例として，厳密解法の研究が盛んに行われている．最近の研究であるSourdの分枝限定法 [13] は，処理開始可能時刻 r_i の制限にかかわらず50〜60仕事程度の問題まで最適解を求めることができる．また，すべての仕事の納期が同一 (common duedate) である問題 $1|d_i=d|\sum_i(\alpha_i E_i+\beta_i T_i)$ については，1000仕事の問題を解くことに成功している．この解法では，時間の離散化に基づく定式化をLagrange緩和し，さらに制約条件を加えることで下界値の改善を図っており，これは後述する筆者らの解法と同様である．

3.2 の最後で述べた $1|r_i|\sum_i w_i T_i$ は，$1|r_i|\sum_i(\alpha_i E_i+\beta_i T_i)$ において $\alpha_i=0$ とした問題である．したがって，文献 [13] の解法は $1|r_i|\sum_i w_i T_i$ にも適用可能であり，数値実験は行われていないものの，文献 [10] の解法より効率がよい可能性がある．

4. SSDP法に基づく厳密解法

前節で見てきたように，代表的な1機械スケジューリング問題に対してさまざまな厳密解法が提案されている．一般に，広範な問題を扱える解法は性能が不十分であり，一方，優越条件など問題固有の情報を用いて高速化を図った解法は汎用性に難がある．たとえば，**3.5** の最後でも述べたとおり，$1|r_i|\sum_i(\alpha_i E_i+\beta_i T_i)$ は，$\alpha_i=0$ とすることで $1|r_i|\sum_i w_i T_i$ を，また，$d_i=0$ とすることで $1|r_i|\sum_i w_i C_i$ を，それぞれ表すことができるため，文献 [13] の解法は **3.1**，**3.2**，**3.3** の問題にも適用可能である．しかし，**3.1** の $1|r_i|\sum T_i$ に対しては文献 [7] の解法の方が効率がよい．

このような状況をふまえ，筆者らは1機械スケジューリング問題に対する汎用的かつ高速な厳密解法の開発に取り組んでいる [14–16]．そして，$1||\sum_i w_i T_i$，$1|r_i|\sum_i w_i T_i$，$1|r_i|\sum_i w_i C_i$，$1||\sum_i(\alpha_i E_i+\beta_i T_i)$，$1|r_i|\sum_i(\alpha_i E_i+\beta_i T_i)$ のすべての問題に対し，この解法が前節の既存解法よりも効率的であることを示している．さらに，順序依存段取り時間を考慮した $1|s_{ij}|\sum_i w_i T_i$ に対しても，最適解が未知であったベンチマーク問題に対して最適解を与えることに成功している [17,18]．

この解法の基本となるのは，時間の離散化に基づく定式化であり，そのLagrange緩和（状態空間緩和，state-space relaxationともよばれている [19,20]）を用いたSSDP (Successive Sublimation Dynamic Programming) 法 [21,22] である．SSDP法は動的計画法ベースの厳密解法であり，

(1) 緩和問題を動的計画法で解く
(2) 不要な動的計画法の状態を削減する
(3) もとの問題の情報（制約）を緩和問題に追加する

を下界値・上界値間のギャップが0になるまで繰り返す，というものである．SSDP法では，動的計画法における状態を記憶しておく必要があるのだが，(3) において制約を追加することで状態数が増大してしまうため，いかに状態数を削減するかが重要となる．そこで筆者らは，文献 [22] の解法をもとに，いくつかの改善を組み込むことで状態数を削減し，効率の向上を実現している．

以下，仕事 i を時刻 C_i に完了した際に発生するコストを $f_i(C_i)$ と表すものとして，順序依存段取り時間 s_{ij} が存在しない問題 $1|r_i|\sum_i f_i(C_i)$ に対する解法を説明する．

4.1 時間の離散化に基づく定式化

時間の離散化に基づく定式化 (time-indexed formulation) は古くから知られている（たとえば文献 [23]）が，連続緩和により良好な下界値が得られることが示され [24] て以降，頻繁に用いられるようになっている．この定式化では，各仕事 i の完了時刻 C_i が取りうる範囲を整数（あるいは，ある刻み幅の整数倍）に限定する必要がある．正規目的関数の場合には，処理時間 p_i，処理開始可能時刻 r_i が整数であれば C_i も整数となる．一方，非正規目的関数である（重み付き）納期ずれ和を最小化する場合は，さらに納期 d_i も整数である必要がある．この整数性の仮定のもと，$1|r_i|\sum_i f_i(C_i)$ は次の問題 (P) として定式化できる．

$$\min_{\boldsymbol{x}} \sum_{i=1}^{n}\sum_{t=1}^{T} f_i(t)x_{it} \tag{3}$$

$$\text{s.t.} \sum_{i=0}^{n}\sum_{s=t}^{\min(t+p_i-1,T)} x_{is}=1, \quad 1\le t\le T \tag{4}$$

$$\sum_{t=1}^{T} x_{it}=1, \quad 1\le i\le n \tag{5}$$

$$x_{it}\in\{0,1\}, \quad 0\le i\le n,\ 1\le t\le T \tag{6}$$

ここで，x_{it} は，t が仕事 i の完了時刻 C_i に一致するとき1，そうでないとき0となる決定変数である $(t<r_i+p_i$

では $x_{it}=0$ とする). ただし，長さ1の遊休時間に対応するダミーの仕事0 ($p_0=1$, $r_0=0$, $f_0(t)\equiv 0$) を導入しており，機械が $[t-1,\ t)$ の期間で遊休状態のとき $x_{0t}=1$ となる．また，計画対象期間は $[0,\ T)$ であり，T は最適性が損なわれないよう十分大きくとる．たとえば，$1|r_i|\sum_i(\alpha_i E_i+\beta_i T_i)$ の場合は，

$$T=\max\left\{\max_{1\le i\le n} r_i,\ \max_{1\le i\le n} d_i\right\}+\sum_{i=1}^{n} p_i \tag{7}$$

とすれば十分である．

この定式化において，(4) 式は，$[t-1,\ t)$ の期間に処理可能な仕事は最大一つであるという容量制約を，また，(5) 式は，各仕事 i をちょうど一回ずつ処理しなければならないという処理回数制約を，それぞれ表している．なお，ダミーの仕事を用いずに定式化した場合，(4) 式は不等式制約となる．

4.2 最短経路問題への変換

前項で定式化した問題 (P) は，適切なネットワーク構造 $G=(V,A)$ を導入することで，制約付きの最短経路問題へ変換できる．前項の定式化における決定変数と同様，仕事 i を時刻 t に完了することを節点 v_{it} に対応づける．すなわち，節点集合 V として

$$V=\{v_{it}|0\le i\le n,\ r_i+p_i\le t\le T\}\cup\{v_{n+1,0}\}\cup\{v_{n+1,T+1}\} \tag{8}$$

を考える．ただし，もう一つのダミー仕事 $n+1$ ($p_{n+1}=1$, $f_{n+1}(t)\equiv 0$) を考え，対応する節点 $v_{n+1,0}$, $v_{n+1,T+1}$ をそれぞれネットワークの始点，終点とする．一方，枝集合 A は

$$A=\{(v_{j,t-p_i},v_{it})|v_{j,t-p_i},v_{it}\in V\} \tag{9}$$

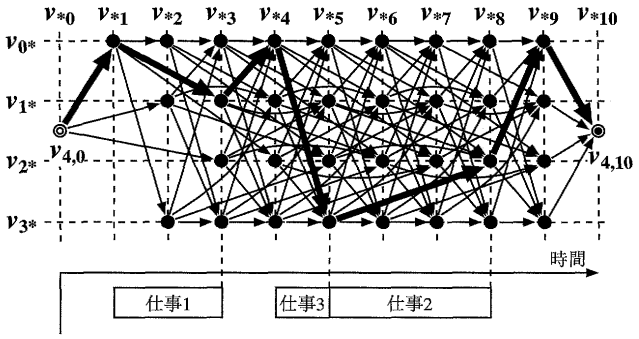
とる．さらに，枝 $(v_{j,t-p_i},v_{it})$ の長さ $c_{(v_{j,t-p_i},v_{it})}$ を $c_{(v_{j,t-p_i},v_{it})}=f_i(t)$ とする．すると，前項の問題 (P) は，「各 i $(1\le i\le n)$ に対して $v_{it}\in V$ をちょうど一回ずつ通る」という制約のもとで，$v_{n+1,0}$ から $v_{n+1,T+1}$ までの最短経路を求める問題と等価となる．そして，最適目的関数値は最短経路長で与えられる（第1図）．そこで，$v_{n+1,0}$ から $v_{n+1,T+1}$ への経路（通過する節点の集合）を $\mathcal{P}$，その長さを $L(\mathcal{P})$ と表すものとする．すなわち，

$$L(\mathcal{P})=\sum_{v_{it}\in\mathcal{P}} f_i(t)=\sum_{\substack{v_{it}\in\mathcal{P}\\ 1\le i\le n}} f_i(t) \tag{10}$$

とする．さらに，$\mathcal{V}_i(\mathcal{P})$ を $\mathcal{P}$ が v_{it} $(1\le t\le T)$ を通過する回数，つまり，

$$\mathcal{V}_i(\mathcal{P})=|\{v_{it}|v_{it}\in\mathcal{P}\}| \tag{11}$$

とする．これらを用いることにより，最短経路問題としての定式化 (N) は以下で与えられる．



第1図 ネットワーク G の例 ($p_1=2$, $p_2=3$, $p_3=1$, $r_1=r_2=0$, $r_3=1$, $T=9$)

$$\min_{\mathcal{P}}\ L(\mathcal{P}) \tag{12}$$

$$\text{s.t. } \mathcal{V}_i(\mathcal{P})=1,\quad 1\le i\le n \tag{13}$$

4.3 Lagrange緩和問題

問題 (P)（あるいは等価な問題 (N)）はそのままでは解くのが困難であるので，(5) 式（(13) 式）の制約条件を Lagrange 緩和する．すなわち，各仕事を必ず1回ずつ処理しなければならないという制約を緩和し，何回処理してもよいとする（1回も処理しなくてもよい）．このタイプの緩和は文献 [20,22,13] などで用いられている．また，(P) の連続緩和問題を列生成法により解く際にも現れる [25]．一方，Lagrange 分解調整法 [26] の枠組では (4) 式の Lagrange 緩和が用いられる．いずれの緩和方法でも（Lagrange 乗数を適切に設定すれば）得られる下界値は同じであるが，前者の緩和は，次項の方法で制約を付加して下界値を改善できる点に特長がある．

ここでは，問題 (N) における (13) 式を Lagrange 乗数 μ_i $(1\le i\le n)$ を用いて緩和するものとする．得られる緩和問題を (LR) とすると，その目的関数は

$$L(\mathcal{P})+\sum_{i=1}^{n}\mu_i(1-\mathcal{V}_i(\mathcal{P}))=L_{\mathrm{R}}(\mathcal{P};\boldsymbol{\mu})+\sum_{1\le i\le n}\mu_i \tag{14}$$

となる．ただし，$L_{\mathrm{R}}(\mathcal{P};\boldsymbol{\mu})$ は

$$L_{\mathrm{R}}(\mathcal{P};\boldsymbol{\mu})=\sum_{\substack{v_{it}\in\mathcal{P}\\ 1\le i\le n}}(f_i(t)-\mu_i) \tag{15}$$

である．(14) 式より，ある与えられた Lagrange 乗数に対して (LR) を解くには $L_{\mathrm{R}}(\mathcal{P};\boldsymbol{\mu})$ を最小化すればよい．また，(15) 式より，この $L_{\mathrm{R}}(\mathcal{P};\boldsymbol{\mu})$ を最小化する問題は，G の枝の長さ $c_{(v_{j,t-p_i},v_{it})}$ を $c_{(v_{j,t-p_i},v_{it})}=f_i(t)-\mu_i$ と変更した（$\mu_0=\mu_{n+1}=0$ とする）ネットワーク上での（無制約の）最短経路問題となる．この問題は，動的計画法を用いて $O(nT)$ の計算量で解くことができる [20]．

4.4 緩和問題への制約の付加

前項の Lagrange 緩和問題 (LR) に制約（カット）を付加することで下界値の改善をはかる．なお，原問題の最適解（の少なくとも一つ）はこれら制約を満たすことが

保証されているため，制約を付加することで緩和問題としての妥当性は損なわれないことに注意する．

4.4.1 k-cycle elimination

まず，巡回セールスマン問題 (Traveling Salesman Problem; TSP) や配車問題 (Vehicle Routing Problem; VRP) などで広く用いられている k-cycle elimination 制約に対応する，以下の制約を加える．

> 任意の i $(1\leq i\leq n)$ に対し，$v_{it}\to v_{a_1t_1}\to\cdots\to v_{a_lt_l}\to v_{is}$ という経路は禁止．ただし，$l\leq k-1$ で，$a_j\neq i$ $(1\leq j\leq l)$.

すなわち，$k=1$ では $v_{it}\to v_{i,t+p_i}$ という経路が，$k=2$ ではさらに $v_{it}\to v_{j,t+p_j}\to v_{i,t+p_i+p_j}$ という経路が禁止されることになる．

(LR) にこの k-cycle elimination 制約を付加した問題を (LR_k) と表すものとすると，(LR_k) は動的計画法により $O(n^kT)$ の計算量で解くことができる [20]．とくに (LR_1) は，G から枝 $(v_{i,t-p_i},v_{it})$ $(1\leq i\leq n)$ を取り除いて得られる第 2 図のようなネットワーク $G_{\mathrm{S}}=(V,A_{\mathrm{S}})$ 上での無制約の最短経路問題に帰着されるが，(LR_2) は G_{S} 上でも 2-cycle elimination 制約付きの最短経路問題となる．ここでは，$k\leq 2$，すなわち，(LR_1)，(LR_2) のみを考える．

4.4.2 隣接 2 仕事間の優越関係による制約

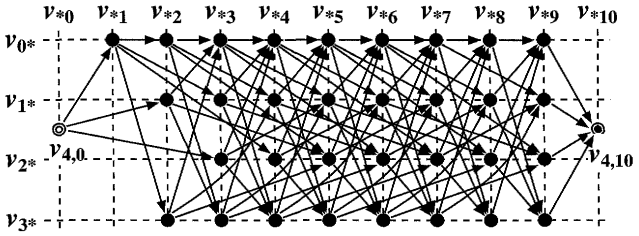
この制約は，下界値を改善するだけでなく緩和問題の求解速度を向上させることのできる強力な条件である．筆者ら [14] とは独立に，Sourd も同様の方法を提案している [13]．

二つの仕事 i，j $(0\leq i\leq n,\ i\neq j)$ を時刻 t に完了するようこの順に連続して処理したとき，そのコストの和は $f_i(t-p_j)+f_j(t)$ となる．逆に，仕事 j，i の順に処理した場合は $f_j(t-p_i)+f_i(t)$ となる．いまもし，$f_i(t-p_j)+f_j(t)<f_j(t-p_i)+f_i(t)$ が成り立つなら，最適スケジュールにおいて $(C_j=t-p_i)\wedge(C_i=t)$ となることはありえない（この 2 仕事の処理順序を入れ替えれることで，他の仕事に影響を与えることなく目的関数値を減少させることができる）．よって，対応する枝 $(v_{j,t-p_i},v_{it})$ を通ることを禁止する．逆に $f_i(t-p_j)+f_j(t)>f_j(t-p_i)+f_i(t)$ なら枝 $(v_{i,t-p_j},v_{jt})$ を通ることを禁止する．$f_i(t-p_j)+f_j(t)=f_j(t-p_i)+f_i(t)$ の場合はいずれか一方の枝を通ることを禁止する．

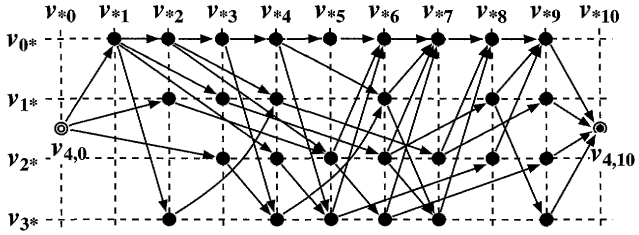
文献 [13] では，(LR_1) にこの制約を付加した Lagrange 緩和問題により下界値を求めているが，その計算量は $O(n^2T)$ であり，(LR_2) にこの制約を付加した場合と同じである．そこで，ここでは (LR_2) に付加した緩和問題 $(\widehat{\mathrm{LR}}_2)$ を考えるものとする．この問題は，通過禁止の枝を G_{S} から削除して得られるネットワーク $\widehat{G}_{\mathrm{S}}$ 上で，2-cycle elimination 制約付きの最短経路問題となる（第 3 図）．

4.4.3 (13) 式の制約条件

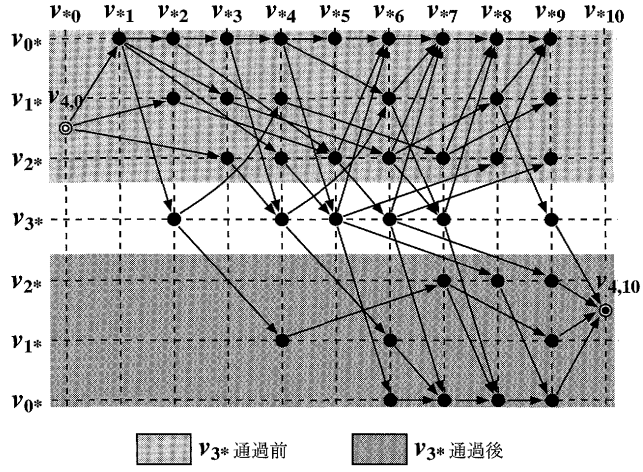
Lagrange 緩和した制約条件 (13) の一部を緩和問題 $(\widehat{\mathrm{LR}}_2)$ に戻してやることで下界値を改善する．仕事の



第 2 図　ネットワーク G_{S}



第 3 図　ネットワーク $\widehat{G}_{\mathrm{S}}$



第 4 図　ネットワーク $\widehat{G}_{\mathrm{S}}^{\mathcal{M}}$ $(\mathcal{M}=\{3\})$

部分集合 $\mathcal{M}\subseteq\{1,\ldots,n\}$ を考え，$\mathcal{M}$ に属する仕事は必ず 1 回ずつ処理しなければならないとする．この制約を $(\widehat{\mathrm{LR}}_2)$ に付加した緩和問題を $(\widehat{\mathrm{LR}}_2^{\mathcal{M}})$ と表すことにする．したがって，$\mathcal{M}=\{1,\ldots,n\}$ なら $(\widehat{\mathrm{LR}}_2^{\mathcal{M}})$ は原問題 (N) に一致する．緩和問題 $(\widehat{\mathrm{LR}}_2^{\mathcal{M}})$ の計算量は $O(n^2 2^{|\mathcal{M}|}T)$ であり，対応するネットワーク $\widehat{G}_{\mathrm{S}}^{\mathcal{M}}$ は第 4 図のようになる．よって，$|\mathcal{M}|$ が大きくなると，ネットワークのサイズが増大することになる．

4.5 解法の構成

前項の緩和問題および制約を用いて SSDP 法による厳密解法を構成する．具体的な構成は以下のとおりである．

(i) 緩和問題 (LR_1) に対し劣勾配法などで Lagrange 乗数 μ_i を最適化する．

(ii) (i) で求まった Lagrange 乗数を初期値として，今度は緩和問題 $(\widehat{\mathrm{LR}}_2)$ に対し Lagrange 乗数を最適化する．

(iii) Lagrange乗数は(ii)で求まったものに固定し，緩和問題$(\widehat{\mathrm{LR}}_2^{\mathcal{M}})$を$\mathcal{M}$に仕事を追加しながら解く．上界値と下界値のギャップがなくなれば終了．

4.6 ネットワーク縮減

緩和問題$(\widehat{\mathrm{LR}}_2^{\mathcal{M}})$に対応するネットワーク$\widehat{G}_{\mathrm{S}}^{\mathcal{M}}$は，$\mathcal{M}$に仕事を追加するにつれて増大する．$(\widehat{\mathrm{LR}}_2^{\mathcal{M}})$を動的計画法で解く際には一本の枝が一つの状態に対応するため，記憶容量の圧迫と計算時間の増大につながる．このため，不要な枝や節点を削除してネットワークの縮減を行うことが重要となる．筆者らの解法では，いくつかの縮減方法を適用しているが，ここでは，その中でもっとも効果的な上界値を用いる方法[22]を説明する．このネットワーク縮減は(LR_1)，$(\widehat{\mathrm{LR}}_2)$に対応する$G_{\mathrm{S}}$，$\widehat{G}_{\mathrm{S}}$に対しても行うため，$\widehat{G}_{\mathrm{S}}$を例に述べる．

緩和問題$(\widehat{\mathrm{LR}}_2)$を順時間の動的計画法で解く際には，各枝$(v_{j,t-p_i},v_{it})$について，「$v_{n+1,0}$から枝$(v_{j,t-p_i},v_{it})$を通り$v_{it}$に到達する最短経路（ただし，2-cycle elimination制約を満たす）」を再帰的に求めることになる．逆時間の動的計画法を適用した場合は，「$v_{j,t-p_i}$から枝$(v_{j,t-p_i},v_{it})$を通り$v_{n+1,T+1}$に到達する最短経路（ただし，2-cycle elimination制約を満たす）」を求めることになる．したがって，これらを組み合わせることで「枝$(v_{j,t-p_i},v_{it})$を通らなければならない」という制約を$(\widehat{\mathrm{LR}}_2)$に付加した問題の解が求まる．その目的関数値が上界値よりも大きければ，原問題(N)の最適解が枝$(v_{j,t-p_i},v_{it})$を通ることはないので，$\widehat{G}_{\mathrm{S}}$から枝$(v_{j,t-p_i},v_{it})$を削除できる．

4.7 上界値計算

前項のネットワーク縮減が効率よく働くためには，良好な上界値，すなわち近似解が必要となる．そこで，動的計画法に基づく局所探索法であるdynasearch[27,28]を適用して近似解を求める．ただし，遊休時間が発生する問題には直接適用できないので，Sourdの拡張[29]を用いるものとする．このdynasearchを，緩和問題の解を実行可能化した解に適用して近似解を求める．

4.8 数値計算結果

以上の解法を例題に適用した結果を第1表にまとめる．いずれの例題も処理時間の最大値は100である．計算はPentium4 3.4GHz（メモリ1GB）のコンピュータ上で行った．ただし，$1|s_{ij}|\sum_i w_iT_i$のみ，Core2 Quad Q9650 3.0GHz（メモリ8GB）のコンピュータ上での結果である．表中のすべての問題に対し，既存解法よりも大規模な問題を解けることがわかる．なお，$1||\sum_i(\alpha_iE_i+\beta_iT_i)$，$1|r_i|\sum_i(\alpha_iE_i+\beta_iT_i)$に関しては，より大規模な問題にも適用可能であるが，数値実験を行っていない（ベンチマーク問題が存在しない）ため限界は不明である．一方，$1|r_i|\sum_i w_iT_i$はこの解法が苦手とする問題で，メモリ不足により80仕事問題を解くのがやっとである．納期遅れ和最小化問題では最適，あるいは最適に近い解が非常に多く存在し，ネットワーク縮減がうまく働かないのではないかと考えられる．しかし，$1||\sum_i w_iT_i$との差がこれほど大きい原因はわかっておらず，今後の検討課題である．また，順序依存段取り時間を考慮した問題については，**4.4.2**の制約を適用することができないため，効率が大幅に低下する．それでも，60仕事のベンチマーク問題120問中118問の最適解を求めることに成功している．

第1表 数値実験結果

問題のタイプ	最大仕事数	最大計算時間[秒]
$1\|\|\sum_i w_iT_i$	400	2,300
$1\|r_i\|\sum_i w_iT_i$	80	300
$1\|r_i\|\sum_i w_iC_i$	200	3,000
$1\|\|\sum_i(\alpha_iE_i+\beta_iT_i)$	150*	100
$1\|r_i\|\sum_i(\alpha_iE_i+\beta_iT_i)$	200*	700
$1\|s_{ij}\|\sum_i w_iT_i$	<60	100,000

* より大規模な問題も解くことができる

5. おわりに

本稿では，生産スケジューリング問題の基礎となる1機械スケジューリング問題をとりあげ，厳密解法の研究動向を筆者らの取り組みを中心に紹介した．厳密解法というと非実用的とのイメージを抱きがちだが，たとえば重み付き納期遅れ和最小化問題$1||\sum_i w_iT_i$の場合，最近のコンピュータを用いれば100仕事の問題が数秒で解けてしまう．場合によっては近似解法よりも高速であり，十分実用にたえるのではないかと考えている．とはいえ，さらなる高速化や多機械問題への対応など，まだまだ課題も多い．今後の研究の発展が待たれるところである．

筆者らの解法の情報はhttp://turbine.kuee.kyoto-u.ac.jp/~tanaka/SiPS/も参照されたい．

(2010年6月25日受付)

参考文献

[1] D. Shabtay and G. Steiner: A survey of scheduling with controllable processing times; *Discrete Applied Mathematics*, Vol. 155, pp. 1643–1666 (2007)

[2] D. Biskup: A state-of-the-art review on scheduling with learning effects; *European Journal of Operational Research*, Vol. 188, pp. 315–329 (2008)

[3] R. L. Graham, et al.: Optimization and approximation in deterministic sequencing and scheduling: A survey; *Annals of Discrete Mathematics*, Vol. 5, pp. 287–326 (1979)

[4] H. Emmons: One-machine sequencing to minimize certain functions of job tardiness; *Operations Research*, Vol. 17, pp. 701–715 (1969)

[5] E. L. Lawler: A "pseudopolynomial" algorithm for sequencing jobs to minimize total tardiness; *Annals of Discrete Mathematics*, Vol. 1, pp. 331–342 (1977)

[6] W. Szwarc, A. Grosso and F. Della Croce: Algorith-

mic paradoxes of the single-machine total tardiness problem; *Journal of Scheduling*, Vol. 4, pp. 93–104 (2001)

[7] G. K. Kao, E. C. Sewell and S. H. Jacobson: A branch, bound, and remember algorithm for the $1|r_i|\sum t_i$ scheduling problem; *Journal of Scheduling*, Vol. 12, pp. 163–175 (2009)

[8] Y. Pan and L. Shi: On the equivalence of the max-min transportation lower bound and the time-indexed lower bound for single-machine scheduling problems; *Mathematical Programming, Series A*, Vol. 110, pp. 543–559 (2007)

[9] OR-Library: http://people.brunel.ac.uk/~mastjjb/jeb/info.html

[10] A. Jouglet, P. Baptiste and J. Carlier: Branch-and-bound algorithms for total weighted tardiness; *Handbook of Scheduling, Algorithms, Models, and Performance Analysis* (J. Y-T. Leung, Ed.), CRC Press. Chapter 13 (2004)

[11] Y. Pan and L. Shi: New hybrid optimization algorithms for machine scheduling problems; *IEEE Transactions on Automation Science and Engineering*; Vol. 5, pp. 337–348 (2008)

[12] L.-P. Bigras, M. Gamache and G. Savard: The time-dependent traveling salesman problem and single machine scheduling problems with sequence dependent setup times; *Discrete Optimization*, Vol. 5, pp. 685–699 (2008)

[13] F. Sourd: New exact algorithms for one-machine earliness-tardiness scheduling; *INFORMS Journal on Computing*, Vol. 21, pp. 167–175 (2009)

[14] S. Tanaka, S. Fujikuma and M. Araki: An exact algorithm for single-machine scheduling without machine idle time; *Journal of Scheduling*, Vol. 12, pp. 575–593 (2009)

[15] S. Tanaka and S. Fujikuma: An efficient exact algorithm for general single-machine scheduling with machine idle time; *Proceedings of the 4th IEEE Conference on Automation Science and Engineering*, pp. 371–376 (2008)

[16] S. Tanaka and S. Fujikuma: A dynamic-programming-based exact algorithm for single-machine scheduling with machine idle time; 投稿中

[17] S. Tanaka and M. Araki: An exact algorithm for the single-machine total weighted tardiness problem with sequence-dependent setup times; *Proceedings of 2008 International Symposium on Flexible Automation*, CD-ROM (8 pages) (2008)

[18] S. Tanaka and M. Araki: An improved algorithm for the single-machine total weighted tardiness problem with sequence-dependent setup times; *Proceedings of International Symposium on Scheduling 2009*, pp. 97–102 (2009)

[19] N. Christofides, A. Mingozzi and P. Toth: State-space relaxation procedures for the computation of bounds to routing problems; *Networks*, Vol. 11, pp. 145–164 (1981)

[20] T. S. Abdul-Razaq and C. N. Potts: Dynamic programming state-space relaxation for single-machine scheduling; *Journal of the Operational Research Society*, Vol. 39, pp. 141–152 (1988)

[21] T. Ibaraki: Enumerative approaches to combinatorial optimization; *Annals of Operations Research*, Vols. 10 and 11 (1987)

[22] T. Ibaraki and Y. Nakamura: A dynamic programming method for single machine scheduling; *European Journal of Operational Research*, Vol. 76, pp. 72–82 (1994)

[23] A. A. B. Pritsker, L. J. Watters and P. M. Wolfe: Multiproject scheduling with limited resources: A zero-one programming approach; *Management Science*, Vol. 16, pp. 93–108 (1969)

[24] M. E. Dyer and L. A. Wolsey: Formulating the single-machine sequencing problem with release dates as a mixed integer problem; *Discrete Applied Mathematics*, Vol. 26, pp. 255–270 (1990)

[25] J. M. van den Akker, C. A. J. Hurkens and M. W. P. Savelsbergh: Time-indexed formulations for machine scheduling problems: Column generation; *INFORMS Journal on Computing*, Vol. 12, pp. 111–124 (2000)

[26] P. B. Luh, et al.: Schedule generation and reconfiguration for parallel machines; *IEEE Transactions on Robotics and Automation*, Vol. 6, pp. 687–696 (1990)

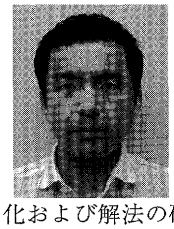
[27] R. K. Congram, C. N. Potts and S. L. van de Velde: An iterated dynasearch algorithm for the single machine total weighted tardiness scheduling problem; *INFORMS Journal on Computing*, Vol. 14, pp. 52–67 (2002)

[28] A. Grosso, F. Della Croce and R. Tadei: An enhanced dynasearch neighborhood for the single machine total weighted tardiness scheduling problem; *Operations Research Letters*, Vol. 32, pp. 68–72 (2004)

[29] F. Sourd: Dynasearch for the earliness-tardiness scheduling problem with release dates and setup constraints; *Operations Research Letters*, Vol. 34, pp. 591–598 (2006)

著者略歴

田中（たなか） 俊二（しゅんじ） (正会員)



1993年3月京都大学工学部電気工学第二学科卒業，2000年3月同大学院工学研究科電気工学専攻博士後期課程修了．同年4月同助手，2006年4月同助教となり現在に至る．生産スケジューリング問題，エレベータ群管理問題など最適化問題のモデル化および解法の研究に従事．博士（工学）．IEEE，計測自動制御学会，電気学会，日本オペレーションズ・リサーチ学会，スケジューリング学会各会員．