ヘテロ並列計算環境のためのタスクスケジューリング手法のサーベイ

須田 礼仁†

高速ネットワークの普及にともない，ヘテロな計算機から構成されるネットワーク計算環境が現出しており，ヘテロ並列計算環境に適した並列化手法の必要性が高まっている．本稿ではタスク並列パラダイムにおける主要な問題であるスケジューリングアルゴリズムについて，ヘテロ並列計算環境に関した研究のサーベイを行う．大規模アプリケーションを想定して目的関数をスケジュール長（makespan）に絞るが，divisible load theory やマルチプロセッサタスクも含める．

A Survey on Task Scheduling for Heterogeneous Parallel Computing Environments

Reiji Suda†

Network computing environments with heterogeneous computers have emerged as results of speedups of computer networks, and needs of parallelization technologies for heterogeneous parallel computing environments are increasing. This paper surveys scheduling algorithms, which are the major issue of parallelization in the task parallel paradigm, for heterogeneous parallel computing environments. The objective is limited to the schedule length (makespan) assuming large scale applications, but divisible load theory and multiprocessor task are included.

1. はじめに

クラスタは広く普及し，MPI による並列プログラミングは今や常識である．効率的な並列化のために必要なのは，逐次性の回避，負荷の均衡化，通信量・回数の削減，通信遅延の隠蔽であるが，これらの知見も一般に広まり，並列化技術の基盤はすでに固まった．これらの技術の多くは，均一なプロセッサ仕様，占有的な利用と安定した動作，適度に効率的で十分な並列性を持つ通信メカニズムなどを仮定している．これらの仮定は，効率的な並列処理を許容可能な複雑さのプログラミングで実現するためには必須である．メタコンピューティング，マルチクラスタやグリッドなど，深い並列性階層を持つアーキテクチャも急速に広まっているが，そのような場合であっても基礎としては従来型の均一な並列環境が用いられ続けるであろう．

しかし，ネットワークの急速な普及と今後期待されるさらなる性能向上は，あらゆる計算機がネットワークを通じて相互に結合されることを意味する．このような環境では，接続されているコンピュータの仕様が一様であるという仮定は明らかに制約が強すぎる．グリッド環境も当然のことながら非一様なハードウェア・ソフトウェアを想定している．また，ユビキタスあるいはパーベイシブな環境を大規模計算に利用しようとすれば，当然ヘテロな並列プロセッサを想定しなければならない．このように，ヘテロな仕様のプロセッサが結合した並列計算環境は，今後いっそう重要性を増してくるものと考えられる．以下ではこのような計算環境を「ヘテロ並列計算環境」と呼ぶことにする．タスク並列とデータ並列は並列化の 2 大パラダイムであるが，本稿ではこのうち**タスク並列**の主要問題である**スケジューリング**のアルゴリズムを対象とする．

本サーベイの目的は，ヘテロ並列計算環境のための既存の並列化技術を概観し，現在までに得られている知見を総合するとともに，現在欠けている技術を明確にして，今後の研究開発の方向性を示唆することである．この分野の裾野は意外に広く，しかもここ数年非常に多くの研究がなされており，日々その地平線は広がりつつある．このため以下に述べるように本稿で取り上げる範囲は相当に制限せざるをえないが，それでも将来にわたるこれらの技術の必要性に鑑み，今後の研究開発にわずかでも資することができれば幸いと考

† 東京大学大学院情報理工学系研究科コンピュータ科学専攻
Department of Computer Science, Graduate School of Information Science and Technology, the University of Tokyo

える．

後述のように本稿の中心はGrahamらのサーベイ[88]を基点とするスケジューリング理論であるが，ずっと後になりBraunら[38]が独立に諸概念を再定義しており，それに従っている研究もいくつかある．両者はヘテロ性を意味する用語が異なり，一方しか知らないと他方の論文を検索することすら難しいため，これらの用語を紹介することも本稿の役割の一部である．またヘテロ並列計算環境のためのタスクスケジューリングについては日本人の貢献がきわめて少ないので，多くの方々にこの分野を知っていただき，利用あるいは研究していただきたいと願うものである．

1.1 本稿で扱うテーマと本稿の構成

タスク並列のパラダイムでは，計算全体は**タスク**または**ジョブ**と呼ばれる独立に処理できる部分に分割されており，これらのタスクをどの**マシン**（**プロセッサ**ともいう）でいつ実行するかという**スケジュール**を決定することが主要な問題であるとする．タスクスケジューリングの問題には，タスク間にはまったく依存関係がない**独立タスクスケジューリング**と，あるタスクは別のタスクの終了，あるいは別のタスクから送られてくるデータの受信を待たなければならない**DAGスケジューリング**がある．さらに，スケジューリングの前にすべてのタスクの情報が得られている**オフラインスケジューリング**と，スケジュールをするに従ってタスクの情報が与えられる**オンラインスケジューリング**の違いが大きい．本稿では2章でヘテロ性のモデルについて概観した後，3章でタスクが任意のサイズに分割できると仮定するdivisible load theory（DLT），4章ではタスクの実行を中断し別のマシンで再開できるプリエンプティブスケジューリングを取り上げる．これらのモデルではタスクが分割できるという自由度を利用して効率的なスケジュールを得ることが可能である．その後分割できないタスクのスケジューリングとして，5章で独立タスクのオフラインスケジューリング，6章でDAGのオフラインスケジューリング，7章でオンラインスケジューリングを取り上げる．さらに8章では各タスクが複数のプロセッサを使用するマルチプロセッサタスクのスケジューリングを取り上げる．最後に9章は全体のまとめとする．今後の課題などは原則として各章の中で論ずる．

タスクスケジューリングには多様な問題設定があるが，本稿では目的関数を**makespan**（**スケジュール長**ともいう）に絞る．これは科学技術計算や最適化問題など，大規模で計算量の多いひとまとまりの計算を分割し，多数の計算機に分散して処理させることにより，短時間で終了させたい場合に適した目的関数である．実際には，多数のタスクが流れるなかで総体として効率良くかつ公平にシステムを運用することもきわめて重要なのであるが，本稿でのスタンスは，それを追求する前提として単一の計算を効率的に処理することが必要だということである．したがって，待ち行列理論や，ウェブサービスや電話・放送網などを含む平均応答時間を主な目的関数とする研究は原則として除外する．また，耐故障性やデッドラインのある実時間処理に関する研究も重要であるが，主たる目的関数が異なってしまうので除外することとした．メモリ量やI/O能力，2次記憶やファイルなどのリソースが制約条件や目的関数に含まれる研究も多数あり，実用上も重要ではあるが，これはそれだけでサーベイを行っても大変なほど深さと広さのある問題であり，ここではやむをえず割愛する．

2. ヘテロ並列計算環境のモデル

並列処理の所要時間を考えたとき，最も基本的な要素は計算と通信である．以下本章では，計算と通信のヘテロ性のモデルについて論ずる．

まず，本稿において共通に用いる表記を導入する．ある一定のまとまりの処理を表す**タスク**（**ジョブ**とも呼ばれる）は n 個あり，$T_1, T_2, \cdots, T_n$ で表す．**マシン**（**プロセッサ**とも呼ばれる）は m 台あり，$M_1, M_2, \cdots, M_m$ で表す．タスク T_i をマシン M_p で処理したときの**所要時間**を t_{ip} とする．**以下では n と m はそれぞれつねにタスクとマシンの数を表す**．さらに，**特に断ることなく，i，j はタスクの添字，p，q はマシンの添字であるとする**．また，「タスク T_i」とあるいは「i 番目のタスク」というべきところを，省略して「タスク i」という．

2.1 計算時間のモデル

Graham ら[88]は1979年当時のスケジューリング理論のサーベイにおいて，並列処理における所要時間モデルを3つに分類している．これはスケジューリングの分野では25年以上の歴史を有する確立した概念なので，本稿でもこれに従うことにする．

所要時間 t_{ip} がマシン p によらないとき，これらのマシンは **identical** であるという．これはホモな並列計算環境に対応する．

任意の i，p に対して所要時間が

$$t_{ip} = w_i / s_p$$

と書けるとき，これらのマシンは **uniform** であるという．w_i はタスク i の**仕事量**，s_p はマシン p の**速度**と呼ばれる．

所要時間に上記のような制約を課さない場合，これらのマシンは **unrelated** であるという．Unrelated の場合には，マシン p でタスク i が処理できない場合を $t_{ip} = \infty$ として扱うことができるとする．

このように，計算性能がヘテロな並列計算環境は uniform な場合と unrelated の場合の 2 つに細分化される．Identical，uniform，unrelated という用語はハイパフォーマンスコンピューティング分野での語感にマッチしないが，スケジューリング分野では完全に定着している．そのため，本稿でもこれをそのまま採用する．

Uniform という仮定は理論的には便利な仮定で，後に見るように unrelated と uniform では問題の難しさがまったく違う．しかしその一方で，現実のヘテロなプロセッサが uniform という条件を満たすというのは非常に期待しにくい．任意のタスクに対して所要時間がマシン速度に反比例するためには，マシンのアーキテクチャが同一で，メモリやディスクへのアクセス所要時間も一様に速度に反比例し，かつキャッシュやメモリなどのサイズは同一でなければならない．唯一 uniform が現実的なのは，(マシンではなく) タスクの処理内容がすべて同一の場合である．この場合，各タスクの処理時間の逆数をマシン性能と定義できる．この場合タスクの仕事量は 1 と仮定して一般性を失わないので，このような問題を **UET**（Unit Execution Time）と呼んでいる（本来 identical の場合にのみふさわしい用語であるがヘテロの場合にも流用されている）．

タスク i がどのマシンに割り当てられるかを示す**割当て変数**（assignment variable）は

$$x_{ip} = 1 \quad \text{タスク } i \text{ をマシン } p \text{ に割り当てる}$$
$$= 0 \quad \text{それ以外}$$

と定義される．マシン p における計算時間は通常

$$C_p^X = \sum_{i=1}^{n} x_{ip} t_{ip} \tag{1}$$

とモデル化される．

2.2 通信時間のモデル

計算時間について参照した Graham らの論文[88] は通信に関しては何も述べていない．通信を考慮したスケジューリングの理論はその後展開された[153] が，残念ながらヘテロなマシンに対してはほとんど結果がない．本稿でもほとんど取り上げることはないが，今後の発展を期待してひととおり紹介しておく．

スケジューリング理論では，先行制約があるタスク i，j があったとき，タスク i がマシン p に，タスク j がマシン $q \neq p$ に割り当てられると，タスク i の完了後ある時間（通信時間）が経過しないとタスク j が開始できない，とモデル化する場合が多い．主流のモデルではこの通信の所要時間をタスクとマシンで決まる c_{ijpq} とし（このまま制約をもうけなければ **unrelated な通信性能**），さらに

$$c_{ijpq} = c_{ij}/b_{pq}$$

（**uniform な通信性能**）と簡単化する場合が多い．ここでは c_{ij} は**通信量**，b_{pq} は**バンド幅**である．さらにバンド幅がマシンによらず $b_{pq} = 1$ とする場合も多い（**identical な通信性能**）．

複数のタスクが同じデータを必要とする場合も多く，そのような場合に重複してデータを転送するのは無駄である．そこで，タスク i が生成する**データセット** D_i^{gen} と必要とする**データセット** D_i^{req} を定義し，マシン p からマシン q には

$$D_{pq}^{com} = \left(\bigcup_{i:x_{ip}=1} D_i^{gen} \right) \bigcap \left(\bigcup_{j:x_{jq}=1} D_j^{req} \right)$$

だけのデータが送信されるものとモデル化する場合がある．この場合，データセットの各要素にサイズを定義し，identical または uniform な通信性能を仮定して通信所要時間をモデル化することが多い．

2.3 Graham らの 3 項表記

本稿では Graham ら[88] により導入された 3 項表記法を若干修正して利用する．これはスケジューリング理論では定着しているものであるが，取り上げている問題を簡潔かつ明確に表現でき，問題のバリエーションがはっきりするので未解決問題も明らかになり，非常に価値が高い．

3 項表記は 3 つの項を縦棒 | で区切った $\alpha|\beta|\gamma$ という形をしている．最初の α はヘテロ性とマシン数に関する仮定，中央の β は各種の条件，最後の γ は最適化における目的関数である．

最初の α の項としては，記号 P，Q，R を identical，uniform，unrelated の意味で用いる．マシン数が固定の場合にはそれをこれらの記号の後に書く．たとえば $Q2$ と書けば uniform なマシン 2 台という問題設定を意味する．また慣例として，Qm と書く場合には，マシン数 m は任意だが，計算量を考えるときには固定されて「定数」と見なされることを意味する．もっとも，現在ではマシン数 m は非常に多くなってきているので，Qm に対するアルゴリズムにどれだけの意味があるかは再検討の必要がある．

中央の β の項としては，各種の条件が入る（以下のような条件が何もなければ空文字列となる）．本稿

で Graham らの論文から引用するものとしては，プリエンプションを意味する *pmtn* と，先行制約の存在を意味する *prec* がある．先行制約が tree などの構造に制約される場合には *prec* の代わりにその構造が書かれる．UET の問題では *UET*，オンラインの問題では *online* を入れる．通信が入ればそのモデルを記述するが，本稿で出てくるのは Unit Communication Time（UCT）のみで，この場合 *UCT* と書くことにする．このほか必要に応じて適宜導入する．

最後の γ は目的関数である．これは本稿では makespan のみであり，C_{max} と書かれる．

2.4 今後の課題

UET 以外で完全に uniform な計算環境というのは考えにくいが，他方タスクの処理時間のマシン依存性に何の制限もない unrelated という仮定は，汎用計算機から構成されるシステムを考えると，過度に一般化されている．これに対して一部の研究[122)]では，所要時間を

$$t_{ip} = \sum_{k=1}^{l} w_{ik}/s_{kp}$$

のようにモデル化している．ここではタスクは l 種類の**演算**の組合せからなり，w_{ik} はタスク i に含まれる演算 k の仕事量，s_{kp} はマシン p が単位量の演算 k を処理する際の速度である．このモデルは uniform と unrelated を結ぶモデルとして興味深いが，残念ながらこれまではこの仮定がアルゴリズムの設計の役には立っていない．

別のアイデアとしては，uniform からの乖離度を制約するというものが考えられる．すなわち所要時間を

$$t_{ip} = \omega_{ip} w_i / s_p$$

のように uniform な場合の時間に速度低下率 ω_{ip} をかけた形でモデル化しておき，速度低下率の範囲を

$$1 \leq \omega_{ip} \leq \bar{\omega}$$

のようにマシンやタスクによらない定数で制約するのである．さらに上記 2 つのモデルを組合せ，定数 $\bar{\omega} \geq 1$ と基準速度 s_p があって，要素速度 s_{kp} が $s_p/\bar{\omega} \leq s_{kp} \leq s_p$ のように抑えられるとするモデルも考えられる．このようなモデルを工夫して，従来は unrelated と分類されていた問題に対して効率的なアルゴリズムを開発することが望まれる．

また，複数のタスクが割り当てられたときの所要時間を式 (1) のように表現するのも，現実の所要時間を適切に表現しているとはいいにくい．実際の計算環境では命令・データキャッシュの影響などで，このような単純な所要時間になることはほとんど期待できない．現実の所要時間により近く，しかも現実的な計算量のスケジューリングアルゴリズムが構築できるような所要時間モデルの開発が期待される．

通信時間についてはさらに事態は深刻である．Sinnen ら[146)]が指摘するように，単一のメッセージをマシン p からマシン q に送信するだけでも，通信が計算速度に影響するかどうか，計算と通信がオーバラップできるかどうか，送信命令が受信命令よりも先に出された場合に通信はいつおきるのか，どういうルーティングがとられるのか，他の通信（q から p への通信，p から他のマシンへの通信，他のマシン間での通信）との衝突が生じた場合にどれだけのペナルティがかかるのかなど，通信時間に影響するファクタはあまりにも多い．ネットワークの接続位相はいかようにも定義ができる．そして，ハードウェア・ソフトウェアの作りにもよるが，一般にメッセージ長と通信所要時間との関係は非常に複雑である．

今後，通信スケジューリングの問題は現実世界においてさらに重要性を増してくることは確実である．そのため，今後さまざまな通信時間モデルが提案されることであろう．上記の問題のうちのいくつかを解決するモデルを構築することは比較的容易であるが，ハードとソフトの現状および未来の可能性に適応できる汎用性を持ち，しかも計算量的な複雑さを適度に抑えたアルゴリズムの構築を許すようなモデルができるかどうかと問われると予測は難しい．一方では現実的な所要時間にあうように数学的な定式化を行うことと，他方ではそれに近いモデルで現実的な計算量と性能を達成するスケジューリングアルゴリズムを開発することの，両方が必要である．

現実の計算環境では，所要時間がすべて分かっているということは少ない．予測される所要時間がある程度の精度を持っていれば，予測値でスケジューリングすることは十分現実的である．この場合，厳密な最適解を得ることは重要ではなく，適当な近似解で足りることはもちろんである．

一部の研究では実行時間が確率的な挙動をすると仮定して，実行時間の不確定性をモデル化しようとしている．マシンの計算時間の最大値である makespan の平均値は，各マシンの平均計算時間の最大値以上[103)]となる．そのため，タスクの実行時間を平均値で置き換えて makespan を最小化しても，必ずしも平均 makespan を最小化しない．そこで平均 makespan を評価し，それを最小化するという研究もある[62)]．

他方，マシン性能が不確定という問題設定も考えられる．Unrelated モデルで，マシン性能が不確定であっ

ても一定であれば，最初に何かのタスクを試行してみて性能を測定してから他のタスクのスケジュールをするのが自然である．性能が時間的に変化しても，比較的ゆっくりであれば，似たようなことで対応できる．3.3 節では一部そのような研究を紹介するが，他の分野ではまだ非常に少ない．

さらに極端な状況である所要時間が完全に未知の場合については，オンラインスケジューリングの章で一部触れる．

3. Divisible Load とマスタ・ワーカ

本章ではスケジューリング手法のサーベイの最初として，**DLT**（**Divisible Load Theory**．定着した和訳がなく，綴ると長く読みにくいので，以下では比較的通用する略語である DLT を用いることにする）と，それに関係の深いマスタ・ワーカ（マスタ・スレーブともいう）型のスケジューリングについて述べる．DLT については比較的最近のよくまとまったサーベイとして Bharadwaj らのもの[27] があり，T. Robertazzi のウェブページ `http://www.ece.sunysb.edu/~tom/dlt.html` に網羅的な文献リストがあるので，ここではいくつかの論文を紹介するにとどめる．

DLT では，タスクは任意のサイズに分割することができ，各部分は任意のプロセッサで独立に処理することができる．通常マスタ・ワーカモデルを採用し，初期状態ではマスタと呼ばれる 1 台のマシンにすべてのタスクが置かれていて，ネットワークを通じてタスクをワーカに配分して処理を行う．マシンは一般にヘテロであるが，タスクは均質であるため必然的に uniform である．DLT は UET モデルでタスク数を無限に多くした極限と見なすことができる[23]．Partitionable あるいは bag-of-tasks と呼んでいる研究者もいる．

このようにモデルが簡単であるので，通信時間を無視してしまうと最適解は自明である．すなわち，タスク量をマシン性能に比例して割り当ててやればよい．しかし通常は通信コストを適当に設定して，通信と計算をあわせてスケジューリングする．その際ネットワークトポロジを考慮するのが一般的で，デイジーチェイン[48]，バス結合[1]，木[49] のほか多様なモデルが取り上げられている．最初はマスタからワーカへのタスクの転送が 1 度だけの**シングルラウンド**（**single installment** または **single round**）であったが，その後複数回に分けて転送する**マルチラウンド**[22]（**multi-installment** または **multi-round**）も考慮されるようになった．シングルラウンドではワーカに割り当てるタスク量が比較的簡単な式で陽に表現できる場合が多い．

DLT のモデルは一見すると簡単すぎるように思われるかもしれない．しかし上記とよく似た結果はさまざまな場面で再発見[4),15),55),133] されていて，そのことは DLT の有用性を示唆している．また応用としては，初期の論文はセンサから得られたデータをワーカに分散して分析させることを目的にしていたが，その後，マルチメディア[25] や画像処理[4] などにも応用されている．

3.1 タスクの転送順序の最適化

初期の研究ではワーカの順序が固定されていたが，その後ワーカにタスクを送る順序の最適化が行われた．この節の結果はいずれもシングルラウンドを仮定している．

Bharadwaj ら[21] は，タスクの転送時間はサイズに比例するが，係数はヘテロでもよいと仮定した．ネットワークはマスタを中心とするスター結合だがシングルポートで一度に 1 つのワーカにしかタスクを送れないとする．このとき彼らは**通信のバンド幅が広いワーカに先にタスクを転送する**のが最適であることを示した．ネットワークが木でも同様である[105]．

一方，Barlas[12] は，通信時間がタスクサイズに依存しない定数の場合について，結果の収集も含めて考察した．やはりスター結合であるが，所要時間モデルとして，ワーカ M_p へタスクを送信する時間を α_p，M_p がサイズ ρ のタスクを処理する時間を $\gamma_p\rho$，M_p からマスタに結果を返送する時間を $\kappa\alpha_p$（κ は定数）とした．このとき，$\alpha_p\gamma_p$ が小さいマシンから順に転送し，結果の受信はその逆順にするのが最適である．したがって，通信速度がマシンによらなければ，**高速なマシンに先にタスクを送る**ということになる．バス結合でも同様である[24]．

すなわち，通信時間が線形なら通信性能，定数なら計算性能に従ってワーカを順序付けるのが最適ということになる．しかしながら通信時間としては定数項＋線形項というモデル（このような所要時間モデルを**アフィン**なモデルという）が用いられることが多い．Beaumont ら[17] が特殊な場合について最適化を考察しているが，Drozdowski ら[67] はアフィンな計算時間での最適化は NP 困難であることを証明し，アフィンな通信時間でも NP 困難であろうと予想（conjecture）している．Genaud ら[81] は通信時間が非線形の場合も含めた最適化を試みている．

3.2 マルチラウンドスケジューリング

Beaumont ら[11),15),17] は計算時間と通信時間がア

フィンなモデルのマルチラウンドスケジューリングに対する漸近最適スケジューリングを提案した．そこではスケジュールの最初と最後以外は，同一スケジュールからなる**定常ラウンド**を反復するものと仮定する．そして，所要時間の定数項を無視したモデルに対する定常ラウンドの最適スケジューリングを求める．このときラウンド数 k に対して定常ラウンドのスケジュール長を T/k とおくと，T は所要時間の定数項を含めた場合の makespan の下限となる．また，本来のアフィンな所要時間モデルにおける定常ラウンドのスケジュール長はある定数 A を用いて $T/k + A$ で抑えられる．そして，スケジュールの最初と最後の部分を含め，全体の makespan は $T + T/k + Ak$ 以下となる．ここで $k \approx \sqrt{T/A}$ とおけば $C_{max} \approx T + 2\sqrt{TA}$ になる．最適な makespan C^{opt}_{max} は T 以上であったので $C_{max}/C^{opt}_{max} \approx 1 + 2\sqrt{A/T}$ となり，A が定数なので $T \to \infty$ において $C_{max}/C^{opt}_{max} \to 1$ となる．これを**漸近最適**と彼らは呼んでいるが，マシン数を固定してタスク量だけを増やした話なので，計算量理論における最悪性能比（後述）とは異なる概念である．なお，Ooshita ら[128] は上記の系列に属す従来研究の一部における線形計画法の定式化は，閉路を含むネットワークではシングルポートの制約を満たす通信スケジューリングができないことを指摘している．

一方，Yang ら[159] はラウンドサイズを等比級数的に大きくする UMR なる手法により，所要時間の定数項によるオーバヘッドを軽減できるとしている．彼らは後の論文[160] でセルフスケジューリングの手法である factoring を参照し，途中から等比級数的にラウンドサイズを小さくする手法を RUMR と呼び，これにより所要時間のモデルからのずれを吸収しうるとしている．また Yamamoto ら[158] は UMR を拡張した PTUMR を提案している．UMR をはじめとするこれらの研究ではワーカがアイドルになる時間がないという仮定のもとにアルゴリズムを設計しているが，他方，前述の Beaumont らの定常ラウンドの最適解ではアイドルになるワーカが必要となる．

計算時間も通信時間も線形であれば，Beaumont らの定常ラウンドスケジューリングでラウンド数を無限に増やした極限が最適となる．しかしアフィンな計算時間モデルでは厳密な最適解は一般に定常ラウンドや等比級数にはならず，それを求めることは現実的でない．Drozdowski ら[66] はタスク量とワーカの選択を組合せ最適化として定式化し，分枝限定法や GA で近似的に最適化することを試みている．

3.3 マスタ・ワーカのその他の結果

ここでいったん DLT を離れ，UET タスクに対するマスタ・ワーカのスケジューリングについて概観する．

Beaumont ら[16] は，マスタ・ワーカモデルを仮定して，通信を考慮したスケジューリング問題の厳密な最適化問題の複雑さについて検討している．マスタはシングルポートとし，通信時間がタスク量に依存しないとする．計算前に通信が必要な場合にはグラフの重み付きマッチング問題に帰着させて $O(n^{5/2})$ で最適解が得られる．計算後にも通信が必要な場合は強 NP 困難となるが，計算前と計算後を独立に最適化する手法に対してある種の性能バウンドを得ている．通信時間がタスク数に比例する場合については，DLT と同様の漸近最適スケジューリングを示している．一般に，タスクサイズを有理数に制限することにより，DLT の定常ラウンドスケジューリングは UET に適用することができる[11),128]．

Dutot[68] はネットワークがデイジーチェインの場合は $O(nm^2)$ の計算量で最適解が得られることを示した．これと上記の Beaumont らの結果を合わせると spider（マスタ以外は子孫が 1 つ以下である木）に対する最適解も得られることを指摘している．さらに Dutot[69] はネットワーク構造が tree で store-and-forward の場合には最適スケジューリング問題は強 NP 困難であることを示した．

マスタ・ワーカ型の処理は動的負荷分散に用いられることが多い．Factoring などの古典的な（identical マシン向けの）セルフスケジューリング手法をそのまま適用する場合も多いが，ヘテロ向けの手法もいくつか提案されている．Flynn Hummel ら[76] はチャンクサイズをマシン速度に比例させる weighted factoring を提案した．Clarke ら[57] は実行結果からマシン性能を推定してチャンクサイズを修正する手法を提案している．Banicescu も同時期に同様の手法である adaptive factoring および adaptive weighted factoring を提案している（共同研究者[41] の記述が明確）．Chronopoulos ら[53] の DTSS（distributed trapezoid self-scheduling）も同様であるが，システムの実行キューの長さから実効性能を評価している．Ferreto らの Adaptive Time Factoring[73] は計算量ではなく「推定所要時間」を factoring することで自然にヘテロな環境に対応できる手法である．Cheng ら[50] は最初にタスクのうち一定の割合を性能比（原著では「クロック比」）で分割し，その後セルフスケジューリングに切り替えることを提案している．

3.4 マスタやタスクの種類が複数の問題

ほとんどの DLT の論文はマスタ・ワーカを仮定しているが，Haddad[89),90)] は初期状態で任意のマシンがタスクを持っていてよい場合について論じている．Suda ら[149)] はこれを Multi-Master Divisible Load と呼んでいるが，スケジューリングの結果として他のマシンにタスクを送るものがマスタ，受け取るものがワーカとなるのであって，事前に区別されているわけではない．特殊な場合として従来のシングルマスタのモデルを含んでいる．

Haddad の論文はいずれもワーカはタスクの受信待ちを起こさない場合に限って解析されている．「受信待ちを起こさない」という仮定は従来のシングルマスタを含まない厳しい制約であるが，シングルラウンドで最適解が得られるという利点がある．また通信時間はタスク量に比例する簡単なモデルとなっている．

Banino ら[11)] は線形計画を用いて定常ラウンドの漸近最適スケジューリングを導いている（しかしシングルポートではスケジュールできない場合が生じる[128)]）．Marchal ら[120)] はさらにマスタごとにタスクの性質が異なる場合について考察しているが，定常ラウンドに限っても NP 困難な定式化となっており，線形計画緩和と貪欲算法による近似アルゴリズムを提案している．ここでは最初からマルチポートを想定している．

一方，Suda ら[149)] は均一な完全結合ネットワークの場合は通信時間がアフィンなモデルに対して漸近最適解が $O(m \log m)$ の計算量で求められることを示した（シングルポートを仮定している）．また，ラウンドサイズを可変として，定常ラウンドの前のプロローグと最後のエピローグも含めてスケジューリングを最適化した．これは受信待ちがなくてもよい場合は Haddad の解に，マスタ・ワーカの場合は Beaumont らの定常ラウンドスケジューリングの改良になっている．

なお，上述の Marchal ら[120)] の研究では異質な複数種類（types）のタスクが同時に処理されるというモデルになっている．一方 multiple divisible load と呼ばれる研究の多く[26),65)] は 1 種類のタスクを処理してから次の種類に移るというモデルになっており，従来モデルとあまり差がない．

3.5 今後の課題

本章では Graham の 3 項表記を用いなかった．これは DLT ではタスクモデルが単純な分，マシンとネットワークの条件が複雑であるためである．これまでどのような問題設定で解かれてきたのか分析してモデルを整理し，未解決問題を明確にする必要がある．

既存のアルゴリズムのいくつかは大規模な線形計画問題を解く必要がある．より高速な近似アルゴリズムが構築されることが望ましい．また，3.4 節で論じた，マスタやタスクの種類が複数の問題についてはまだ研究が始まったばかりで，さらなる開発が必要である．

4. プリエンプティブスケジューリング

古典的なタスクスケジューリングにおける**プリエンプション**では，タスクは任意の時点で中断することができ，すぐに別のプロセッサで再開することができる．中断・再開やタスクの転送のオーバヘッドは考慮されていない．このような単純なモデルでは，以下に見るように，最適解が容易に得られる．

4.1 独立タスクスケジューリング

$Q|pmtn|C_{max}$ については，以下のような最短 makespan が解析的に出る．あらかじめタスクは仕事量の降順，マシンは速度の降順に番号を振りなおし，$W_i = \sum_{j \le i} w_i$，$S_p = \sum_{q \le p} s_q$ を定義しておく．最大のタスク 1 つだけという問題を考えると，それを最も速いマシンで処理するのが最適であり，その所要時間は $w_1/s_1 = W_1/S_1$ である．全タスクに対する makespan はこれより短くはならない．タスクが 2 つであれば，速いほうから 2 台のみを使えばよく，所要時間は合計仕事量を合計速度で処理する W_2/S_2 以上はかかる．同様にして $\min\{n, m\}$ 個の makespan の下限が得られる．最後に $m \le n$ なら全タスクを全プロセッサで処理する W_n/S_m も makespan の下限となる．これらをあわせると

$$C_{max} \ge \max\left\{\frac{W_1}{S_1}, \frac{W_2}{S_2}, \cdots, \frac{W_{m-1}}{S_{m-1}}, \frac{W_n}{S_m}\right\}$$

となるが，実は最適スケジューリングでの makespan はこの右辺に一致する．

このことは最初 Horvath ら[96)] により示されたが，彼らの手法は本来 DAG スケジューリングである．すぐに Gonzalez ら[87)] により独立タスクに適したアルゴリズムが示された．後になって Shachnai ら[140)] がプリエンプションの数が最小となるスケジューリングアルゴリズムを示している．

$R|pmtn|C_{max}$ の解は Lawler ら[114)] により示されている．これは τ_{ip} をマシン p でタスク i を処理する時間として線形計画問題

$$\begin{aligned} &\text{minimize} \quad t \\ &\text{subject to} \quad \sum_p \tau_{ip}/t_{ip} = 1 \qquad \forall i \\ &\qquad\qquad \sum_i \tau_{ip} \le t \qquad \forall p \end{aligned}$$

$$\sum_p \tau_{ip} \le t \qquad \forall i$$
$$0 \le \tau_{ip}$$

を解き，同じ論文で示されているオープンショップ問題（下記）に帰着させる．線形計画は多項式時間で解けるとはいえ計算量が多いとして Jansen ら[99] は高速近似アルゴリズムを提案している．

なお，$Q||C_{max}$ の最適解 C^{opt}_{max} と，同じ問題のプリエンプションを許した最適解 C^{pmtn}_{max} との間には，$C^{opt}_{max} \le (2-1/m)C^{pmtn}_{max}$ の関係が[154]，$R||C_{max}$ の場合は $C^{opt}_{max} < 4C^{pmtn}_{max}$ の関係がある[116]．

ここで参考までに**ショップスケジューリング**とは何かを Graham らのサーベイ[88] に基づいて説明しておく．タスクスケジューリングとの決定的な違いは，各タスクはサブタスクに分かれていて，(1) サブタスクはどのマシンで実行されるかが決まっていることと，(2) 同じタスクに属すサブタスクは（異なるマシン上でも）同時に処理できないことである．サブタスク間の先行制約によりいくつかの種類に分けられるが，そのうち**オープンショップ**はサブタスク間に先行制約がないものである．プリエンプティブスケジューリングにおいて，どのタスクをどのマシンにどれだけ割り当てるかが決まってしまえば，独立タスクスケジューリング問題はオープンショップ問題になるのは明らかであろう．ショップスケジューリングはタスクスケジューリングとは制約が相当に異なり並列化に直接利用することは難しいが，このように場合によっては最適解・近似解が得られる可能性があり，研究動向を見守っておく価値がある．なお，(1) の制約のみの問題を **dedicated** なマシンと呼んでいることがある．

4.2 DAG およびオンラインスケジューリング

$Q|pmtn, prec|C_{max}$ に対しては，上述の Horvath ら[96] が最悪性能比が $\sqrt{3m/2}$ となる近似アルゴリズムを提案している．ここで**最悪性能比**（worst-case performance ratio）とは，近似解の makespan C^{app}_{max} と最適解の makespan C^{opt}_{max} との比 $C^{app}_{max}/C^{opt}_{max}$ の上限である．すなわちどのような問題に対しても最適解の $\sqrt{3m/2}$ 倍以内の makespan の解が得られるということである（論文によっては漸近的な上限，すなわち問題サイズが大きい極限での上限を指すこともある）．最悪性能比が σ の近似アルゴリズムを簡単に σ **近似アルゴリズム**ともいう．また，最悪性能比と後述の競合比をあわせて**性能バウンド**と呼ぶことにする．

Horvath らのアルゴリズムは典型的な**レベルスケジューリング**である．タスク T_i に対して**レベル** $L(T_i)$ を

$$L(T_i) = w_i + \max_{T_i \prec T_j} \{L(T_j)\}$$

（ただし後続タスクがない場合は $L(T_i) = w_i$）と定義し，このレベルの高いものから順にマシンに割り付ける．プリエンプションは処理中のタスクが均等な速度で処理されるように行われるが，非常に高頻度である．2 台ならこれは最適解を与え，3 台以上であれば $\sqrt{3m/2}$ 近似アルゴリズムとなる．Jaffe[98] は $O(\sqrt{m} + O(m^{1/4}))$ 近似アルゴリズムを与えている．

$R|pmtn, prec|C_{max}$ の研究はほとんど見当たらない．Blazewicz ら[33] は依存グラフが interval order のときに多項式時間で解けることを示しているが，これはきわめて特殊な条件である．

オンラインでプリエンプティブというスケジューリングも少ない．Epstein ら[71] は $Q|pmtn, online|\ C_{max}$ に対するランダマイズアルゴリズムの競合比の下限が 2 であることを示したが，これは下限のみで具体的なアルゴリズムは示していない．

4.3 今後の課題

これらの研究で仮定されているプリエンプションのあり方はあまりにも非現実的であり，中断・転送・再開のコストを考慮した研究が望まれる．

これらのコストやシステムの実装の大変さを考慮するとプリエンプティブなタスクの研究は一見無意味なようにも見える．しかし，以下に見るように他のモデルではほとんど最適解など望めないのに対比して，きわめて高速に最適解が得られる点は重要である．上記のモデルで適度に近似できるのであれば，プリエンプティブなタスクはもっと広く積極的に利用されてしかるべきである．

実用的な観点からは，プリエンプションしてもマシンを移動せず，同じマシンで再開する**制約されたプリエンプション**でどれだけの結果が得られるかが興味深いが，自然にこの制約が課されるショップスケジューリング以外にはあまり結果がない．

5. 独立タスクのオフラインスケジューリング

本章からは分割できないタスクのスケジューリングを見てゆく．本章はすべてのタスクが独立な問題に対するスケジューリング手法のサーベイである．論文によっては独立なタスクを **meta-task** と呼び，独立タスクのスケジューリングを **meta-task mapping** と呼んでいる．また，論文によっては**負荷分散**（**load balancing**）と呼んでいることがある．タスクの「所要時間」を「負荷量」と読みかえると，各マシンでの「処理完了時刻」が「マシン負荷」に相当するのでこ

の類推が得られる．

5.1 $Q|UET|C_{max}$

UET は特別に簡単である．Graham らのサーベイ[88]では $O(n^3)$ のアルゴリズムが示されているが，Boudet ら[36]が $O(m^2)$ で解けることを示している．ここでは後者のアルゴリズムを紹介しておく．まず

$$\tilde{x}_p = \left\lfloor \frac{1/s_p}{\sum_q 1/s_q} n \right\rfloor$$

として，マシン p に割り当てるタスク数の近似を得る．これは $n - m \leq \sum_p \tilde{x}_p \leq n$ を満たすので，割り当てられていないタスク数はたかだか m 個である．これらを 1 つずつ，そのタスクの処理を完了する時刻が最も早いマシンに割り当ててゆく（このようなタスクを割り当てるマシンの選択法を以下では **Earliest Completion Time**（**ECT**）と呼ぶ）．ヒープを用いて容易に $O(m \log m)$ の計算量にできる（が，Boudet らの論文では指摘されていない）．

マスタ・ワーカモデルを仮定した通信を含むスケジューリングは 3.3 節で紹介している．

5.2 $Q||C_{max}$

次に uniform でタスクの仕事量が一般の場合について論ずる．最初に厳密アルゴリズムと，厳密解にいくらでも近い解を得ることができる (F)PTAS を概観する．そのあと近似アルゴリズム（**発見的アルゴリズム**あるいは**ヒューリスティクス**とも呼ばれる）を紹介する．

5.2.1 厳密アルゴリズムと (F)PTAS

Horowitz ら[95]は，$R||C_{max}$ に対する動的計画法による厳密解と $Rm||C_{max}$ に対する FPTAS を与えている．ここで **FPTAS**（**Fully Polynomial Time Approximation Scheme**）とは，任意の $\varepsilon > 0$ に対して最悪性能比が $1+\varepsilon$ 以下となるような解を与えるアルゴリズムであって，計算量が問題サイズと $1/\varepsilon$ に対して多項式となるものである．このアルゴリズムは FPTAS の典型的な例なので $Q2||C_{max}$ の場合について概要を紹介しておく．

$W = \sum_i w_i$，$s_1 = 1$，$s_2 = S < 1$ とする．まず明らかに $W/2 \leq C^{opt}_{max} \leq W$ であることに注意する．タスクを $\varepsilon W/2$ を境にして large と small の 2 種類に分ける．そして large のタスクだけについてのスケジューリングのうち W 以内に終わるものすべてを列挙する．ただしこのとき所要時間は $\varepsilon^2 W/4$ の整数倍に切り捨てて，M_1 の所要時間ごとに M_2 の所要時間が最短のもののみ残す．これにより列挙されるスケジューリングの数は最大でも $4/\varepsilon^2$ となり，列挙の計算量は $O(n/\varepsilon^2)$ ですむ．その結果最小 makespan（これを C^1_{max} とすると $C^1_{max} < C^{opt}_{max}$ である）を与えるものを選び，切り捨てた所要時間をもとに戻す（結果を C^2_{max} とする）が，M_1 には $2/\varepsilon$ 個，M_2 にはそれ以下の large タスクしか入らないので，$C^2_{max} \leq C^1_{max} + \varepsilon W/2$ である．最後に small タスクを ECT で詰めてゆくが，その結果得られた makespan C^3_{max} は，C^2_{max} に等しいか，さもなくば完全均衡解 $W/(1+S)$ より $w_n S/(1+S)$ しか遅くない．いずれにせよ $C^3_{max} \leq (1+\varepsilon)C^{opt}_{max}$ である．

Hochbaum と Shmoys[93] は $Q||C_{max}$ に対する PTAS を示した．ここで **PTAS**（**Polynomial Time Approximation Scheme**）とは，FPATS から計算量が $1/\varepsilon$ に関する多項式である条件を除いたもので，Hochbaum らのアルゴリズムの計算量は $O((2/\varepsilon^2)mn^{(10/\varepsilon^2)+3})$ と評価されている．彼らのアルゴリズムは Horowitz らのアルゴリズムよりも一段複雑であるが，基本的なアイデアは同じである．

$Q||C_{max}$ には FPTAS は知られていない．これは $P||C_{max}$ すら強 NP 困難であるためで，$P \neq NP$ であれば FPTAS は存在しないのである．したがって $Qm||C_{max}$ への FPTAS と $Q||C_{max}$ への PTAS は望むべき限界なのである．理論的にはすばらしいのであるが，m ないし $1/\varepsilon$ に対する指数的な依存性のため，非実用的といわざるをえない．したがって，実用的には以下に述べる近似アルゴリズムを用いることになる．

5.2.2 LPT

Uniform の場合にはヒューリスティックな手法でかなり良い近似解が得られる．よく知られたものとして，**LPT**（**Longest Processing Time first**）と MULTIFIT がある．

LPT は最初 $P||C_{max}$ に対して提案された手法であるが，$Q||C_{max}$ に対する LPT のアルゴリズム[86]では，最初にタスクを仕事量の大きいものから順に並べておき，この順序で ECT によりタスクを順次マシンに割り当ててゆく．

上記の LPT を定義した Gonzalez ら[86]は，その最悪性能比が 1.5 から 2 の間であることを示した．LPT の最悪性能比は，さらに Dobson[61] により 1.512 以上 1.584 以下，そして Friesen[79] により 1.52 以上であることが示されている．

なお，最も速いマシンと最も遅いマシンの速度比が小さいと，もっと良い最悪性能比が得られる[126]．また，LPT のようにタスクを割り付ける順序を決めておくアルゴリズムを一般に**リストスケジューリング**と呼ぶが，$Q||C_{max}$ に対する任意のリストスケジューリン

グの最悪性能比は $m > 2$ に対して $(1+\sqrt{2m-2})/2$ 以下であることが Cho ら[51] によって示されている．

ごく初期の研究である Liu ら[117] は，ECT ではなく，タスクを割り当てるマシンを「最初にアイドルになるマシン」としてアルゴリズムを定義し，解析を行っている．限りなく遅いマシンが 1 台ある場合，それにタスクを割り当てないほうがよいが，Liu らのアルゴリズムでは m 番目のタスクを必ず割り当ててしまう．このため最悪性能比はマシン性能に依存するものしか得られない．

5.2.3 MULTIFIT

MULTIFIT[78] は LPT とは逆に，makespan を先に決めるアルゴリズムとなっている．やはりタスクは大きさの順に並べておき，それぞれのタスクは与えられた C_{max} に収まる最初のマシンに割り当てる（**FFD: First Fit Decreasing**）．最初に与えた C_{max} が小さすぎるとスケジューリングに失敗するので，二分探索法でスケジューリングできる最小の C_{max} を決定する．これも最初は $P||C_{max}$ に対して提案された手法である．

$Q||C_{max}$ に対する MULTIFIT の最悪性能比は Friesen ら[78] により 1.341 から 1.4 の間，さらに Chen[45] により 1.341 から 1.382 の間であることが示されている．

このように $Q||C_{max}$ については高速な近似アルゴリズムが知られている．MULTIFIT のように makespan を決めておけば，$Q||C_{max}$ はマルチナップザック問題（実際はより簡単なマルチサブセットサム問題）に帰着されるが，これに対するアルゴリズムも多数知られている．これらのことから，$Q||C_{max}$ についてはおよそ十分な知見が得られていると考えてよい．これ以外に台数やマシン速度に強い制限を設けた研究がいくつかあるが，汎用性が低いので省略する．

5.3 $R||C_{max}$

Unrelated になると uniform に比べて格段に問題が難しくなる．$Q||C_{max}$ に対するナップザック問題に対応する $R||C_{max}$ の相手は GAP（Generalized Assignment Problem）と呼ばれているが，これもナップザック問題よりも格段に難しい．

5.3.1 厳密アルゴリズムと (F)PTAS

$Q||C_{max}$ のところで説明した Horowitz らの論文[95] は本来 $R||C_{max}$ に関する論文であり，前述のとおり動的計画法で最適解が得られることを示している．Graham ら[88] によれば Stern が分枝限定法で $R||C_{max}$ を解いている．Martello ら[121] は分枝限定法を用いて，Mokotoff ら[125] は切除平面法を用いて，それぞれ厳密解を求めている．最後の論文ではマシン数とタスク数の積 mn が数百程度までは数秒で結果を得ているが，$mn > 1000$ では 1 時間以上かかる．所要時間は多項式では抑えられないから，これより大きな問題では近似アルゴリズムが必須である．

Horowitz ら[95] の $Rm||C_{max}$ に対する FPTAS の計算量は $O(nm(nm/\varepsilon)^{m-1})$ である（論文中の計算量の評価は間違っている）．この後 $Rm||C_{max}$ に対する (F)PTAS の研究がいくつかある[74),99),115)] が，いずれも m に関しては指数的である．$Q||C_{max}$ に対しては Hochbaum と Shmoys の PTAS があったが，$P \neq NP$ であれば $R||C_{max}$ に対して性能比が 3/2 未満の多項式時間近似アルゴリズムは存在しないことを Lenstra ら[115] が示している．

5.3.2 性能バウンドがある近似アルゴリズム

Ibarra ら[97] は $R||C_{max}$ に対して以下の 5 つの簡単な近似アルゴリズムを定義している．

アルゴリズム A は ECT によるリストスケジューリングであるが，タスクの順序は任意である．

アルゴリズム B も同様であるが，LPT にならって，タスクを $\min_p\{t_{ip}\}$ の順に並べておく．

アルゴリズム C も同様であるが，タスクを $\max_p\{t_{ip}\}$ の順に並べておく．

アルゴリズム D は未スケジューリングのタスクのうち，最も早い完了時刻が最も早いタスクを選んで ECT でスケジュールする．

アルゴリズム E は未スケジューリングのタスクのうち，最も早い完了時刻が最も遅いタスクを選んで ECT でスケジュールする（論文ではこのアルゴリズムの定義が間違っている）．

彼らはアルゴリズム D 以外の 4 つのアルゴリズムの最悪性能比がプロセッサ数 m であることを示したが，後に Spinrad[147] がアルゴリズム D の最悪性能比も m であることを示し，最悪性能比ということでは同じであることが判明した．

次の進歩は Davis ら[59] によるアルゴリズムであり，これは $O(mn\log n)$ の計算量で最悪性能比 $2\sqrt{m}$ を達成する．まず**効率**という指標を

$$ef_{ip} = \min_q\{t_{iq}\}/t_{ip}$$

で定義したうえで，最も早くアイドルになるマシンに，未割当てのタスクで最も効率の良いもの（効率が同じなら最も所要時間の長いもの）を割り当てる．ただし，効率が $1/(2-\sqrt{2})\sqrt{m}$ 以下なら割り当てない．また，効率が同じなら時間がかかるタスクを優先する．シンプルかつ有効であり，しかも ECT ではない点が大変

興味深い．

Hariri ら[91] は線形計画と ECT による $2+\lfloor \log_2(m-1) \rfloor$ 近似アルゴリズムを提案している．

さらに Lenstra ら[115] が 2 近似アルゴリズムの存在を示した．このあと多数の 2 近似アルゴリズムが続く[8),80),142),161)]．2 を下回ったのは Shchepin ら[141] による $2-1/m$ 近似アルゴリズムのみのようであるが，これも m が大きくなると 2 に漸近する．前述のように最悪性能比の既知の下限は 3/2 であるが，2 が限界なのかもしれない．

5.3.3 その他の近似アルゴリズム

Ibarra らによるアルゴリズムは繰り返し再発見されている[5),39),77),119)]．アルゴリズム A は Fast Greedy あるいは Minimum Completion Time（MCT），アルゴリズム B は Heavy Tasks First（HTF），アルゴリズム D は Min-min，アルゴリズム E は Max-min などとも呼ばれている．しかし最悪の場合に最適解の m 倍ということはいわば「並列化効率」が $1/m$ ということであり，ほとんど並列化されていないに近い，きわめてひどいスケジュールであることに厳重な注意が必要である．

さらに明らかにこれらと同程度以下のものとして，最初に空くマシンにタスクを割り付けるリストスケジューリング **OLB**（Opportunistic Load Balancing），t_{ip} を最小にするマシンに割り付ける **MET**（Minimum Executimon Time）= LBA（Limited Best Assignment）= UDA（User Directed Assignment）などがある．OLB は 5.2.2 項で説明した Liu ら[117] と同じ理由で最悪性能比を抑えることができない．MET は m-近似アルゴリズムである．1 台だけが微妙に速い uniform な環境では最適解の m 倍に限りなく近いスケジュールを出す．一方でマシンの稼働時間の合計を C_{sum} とすると，$C_{sum}^{MET} \leq C_{sum}^{opt}$ となる．すると $C_{max}^{MET} \leq C_{sum}^{MET} \leq C_{sum}^{opt} \leq mC_{max}^{opt}$ となって，最悪性能比は m 倍以内であることが分かる．

多少良くなる可能性があるが最悪性能比が評価されていないものとして，Min-min と Max-min の良い方を選択する **Greedy**[39]，負荷の不均衡さに応じて MCT と MET を切り替える **Switching Algorithm**[119]，各タスクについて処理時間が短いほうから k% のマシンに割当て先を絞る **K-Percent Best**[119]，他のマシンに割り当てたときの所要時間の増加が少ないものは後回しにする **Sufferage**[119]，その拡張版でクラスタ単位で Sufferage を計算する **XSufferage**[42]，t_{ip} の平均・最大・最小などで N 個のクラスに分けたうえで Min-min を使う **Segmented Min-min**[156]，所要時間と完了時刻とそれらの平均値との比でタスクの順序を決める **Relative Cost**[157] などがある．Min-min と Max-min の最悪性能比を出すタスクセットは異なっているので，Greedy の最悪性能比が m 未満である可能性はある．しかし，K-Percent Best は $100\,m/k$ 台の非常に遅いマシンを追加すれば実質的に $R||C_{max}$ になってしまうので最悪性能比を改善しないであろう．すべてのマシンがすべてのタスクを実行できないと Relative Cost は定義できない．

また，Simulated Annealing，タブーサーチ，Genetic Algorithm，降下法，A*，分枝限定法，ビームサーチなどの汎用探索手法を用いた実装例も多数[39),83),84),124),129)] ある．さらにはアルゴリズムのパラメータを機械学習で最適化する提案[82] もある．

5.4 今後の課題

独立タスクスケジューリングでは，既存の枠組みに対しては理論的成果はほぼ揃っている．

Lenstra ら[115] によれば，所要時間が $t_{ip} \in \{1, \infty\}$ および $t_{ip} \in \{1, 2\}$ のように限られる場合には多項式時間で解けるが，$t_{ip} \in \{t, t'\}$ で $t < t'$ かつ $2t \neq t'$ の場合には NP 困難であるという．しかし，所要時間が 2 つの値しかとらなくても，m 台のマシンに対して 2^m 種類のタスクがありうる．実用的にはむしろタスクやマシンの種類が限られる場合に関する研究がほしい．これに関して，Thomas McCormick ら[151] がタスクが 2 種類の場合は uniform でも unrelated でも多項式時間で解けることを示している．このような問題は **high multiplicity** problem と呼ばれるが，計算量理論上微妙な問題を含んでおり[40]，成果があまり多くないのが残念である．

本章ではタスクはどのマシンにでも自由に割り当てられるとしているが，現実にはどこかに存在するタスクを当該マシンに移送して実行しなければならない．このコストを次章で論じる DAG スケジューリングの枠組みでモデル化すること自体は可能であるが，DAG が高さ 1 の木のみからなる森というきわめて特異な問題であるので，一般的な DAG スケジューリングの手法を使うのは無駄である．特殊な場合については 3.3 節で触れたが，今後さらに研究が進むことを期待する．

6. DAG のオフラインスケジューリング

次に，タスク間に先行制約（precedence constraints）がある場合についてのオフラインスケジューリングを概観する．すなわち，$T_i \prec T_j$ であれば，T_j は T_i が完了した後でないと開始できないという条件

が課される．

先行制約の最も一般的な形はDAGであるが，DAGの構造を制限する場合がある．応用で時折見かけるのが *tree* である．全順序になったものを *chain* というが，並列処理とはいえないので省略する（最適解は後述ようにBILで与えられる）．Chainが多数ある場合は *chains* と書くが，これは並列処理として意味があるので取り上げる．

6.1 性能バウンドのあるアルゴリズム

最初にuniformの場合について概観する．$Q|prec|C_{max}$ の初期の論文の1つは5.2.2項でも紹介したLiuらの論文[117]であるが，性能比が大きいと最悪性能比が抑えられないのは前述のとおりである．しかしJaffe[98]は最速マシンよりも $\sqrt{m}$ 倍以上遅いマシンは使わないことにすれば最悪性能比が $1+2\sqrt{m}$ となることを示した．さらに $\sqrt{m}+O(m^{1/4})$ 近似のアルゴリズムも示している．

Chudakら[54]は $Q|prec|C_{max}$ に対する $O(\log m)$ 近似アルゴリズムを提案している．まずマシン速度が K 種類しかないと仮定して，$O(K)$ 近似アルゴリズムを構築する．そして，最速マシンの m 倍以上遅いマシンは切り捨て，残ったマシンの速度を最速マシンの $(1/2)^k$ 倍に切り捨てる．これにより $K=\log m$ とできるが，makespanは4倍しか伸びないことが示される．よって $O(\log m)$ 近似アルゴリズムとなる．Chekuriら[44]はChudakらのものよりもやや簡明な $O(\log m)$ 近似アルゴリズムを提案しているが，DAGをchainsに分解する点が大きな特徴である．Chudakら[54]はLenstraとRinnooy Kanの論文を参照し，$P\neq NP$ であれば $Q|prec|C_{max}$ には近似比が4/3未満の近似アルゴリズムは存在しないとしている．

Chekuriらは上記の論文[44]で $Q|chains|\ C_{max}$ に対する6近似アルゴリズムも提出している．この問題に対してはWoeginger[154]が2近似アルゴリズムを示している．そこでは各chainを1つのタスクにまとめた $Q||C_{max}$ とそのプリエンプティブ版 $Q|pmtn|C_{max}$ を構成し，これらの最適解のmakespanが $C^{pmtn}_{max}\leq C^{chains}_{max}\leq C^{indep}_{max}\leq 2C^{pmtn}_{max}$ を満たすことを利用する．

$R|prec|C_{max}$ に関しては結果が非常に少ない．Macularら[118]は問題が整数計画に帰着できることを示しており，Collら[58]は変数・制約ともより少ない整数計画での定式化を示しているが，いずれもそれを解くためのアルゴリズムの提案はない．最近Anil Kumarら[2]は $R|forest|C_{max}$ に対してpolylogarithmic近似アルゴリズムを提出しており，今後の展開を期待したい．

以上のアルゴリズムはいずれも通信コストを無視したモデルとなっている．スケジューリング理論の分野では，通信コストとして，先行タスクが完了してから通信に対応する時間が経過すると後続タスクが開始できるとするモデルが多い．これは完全結合ネットワークで通信と計算が完全にオーバラップできることを意味する．通信を含むタスクスケジューリングの理論はそれなりの論文数があるが，ほとんどがidenticalマシンである．Kubiakら[108]がかろうじて $Q|UET,chains,UCT|C_{max}$ のアルゴリズムで $C_{max}\leq C^{opt}_{max}+2m-1$ となるようなアルゴリズムを示している（最速マシンの速度を1とする）．彼らは $Q|UET,chains,UCT|C_{max}$ も $Q|UET,chains|C_{max}$ も強NP完全であることを示しており，$P\neq NP$ ならFPTASは存在しない．このほか2台に限定した結果がいくつかあるが，Kubiakの論文に述べられていることもあり省略する．

6.2 性能バウンドのないアルゴリズム

理論的な研究とは別に，実装ベースの研究は多数行われている．1995年ごろまではいくつかの基本的な提案がなされていたが，その後急激に提案が増え，著者が論文を入手できた範囲でも60件以上の提案がある．論文によっては所要時間モデルやアルゴリズムの細部が明記されていない場合もあり，相互比較も十分に行われていないので，本稿では代表的な手法と特徴的な点のある手法をいくつか選んで紹介することにする．

圧倒的に多いのはレベルスケジューリングに基づく手法である．古典的な手法はElRewiniらによるMapping Heuristic（MH）[70]，Sihらによる Generalized Dynamic Level（GDL）[145]，Ohらによる Best Imaginary Level（BIL）[127]の3つである．MH[70]ではidenticalなマシンを想定してレベルを定義し，ECTでスケジューリングするのを基本としているが，通信の衝突やinsertion，duplicationなどを含む改良もあわせて提案している．GDL[145]はアルゴリズムの途中における状況を反映させてレベルを動的に修正する．Best Imaginary Level（BIL）[127]は

$$L_i=\min_p L_{ip}$$
$$L_{ip}=t_{ip}+\min_q\{c_{ijpq}+L_{jq}\mid T_i\prec T_j\}$$

で定義されるが，これは先行制約が単一chainの場合の最小makespanを与え[148]，identicalの場合のボトムレベルのヘテロ拡張になっている．Menasceら[122]は基本的な手法のバリエーションについて実験的比較を行った初期の研究である．

比較的引用回数が多い Heterogeneous Earliest-Finish-Time（HEFT）[152] や Levelized Duplication Based Scheduling（LDBS）[63] は所要時間の平均値を用いてレベルを定義しており，unrelated なマシンでの性能には限界がある．このほかレベルの定義としては，ボトムレベルとトップレベルを両方参照して優先度を決めるもの[9),43),132)]，クリティカルパスのノードに高い優先度を割り当てるもの[13),131)]，数タスク分先読みをするもの[155)]，さらにはレベルをメタヒューリスティクスで最適化するもの[35),104),134)]，ランダムなタスク順序でリストスケジューリングを繰り返すもの[37)]など，多様なものが提案されている．このほかレベルスケジューリングだけで 30 件以上の論文がある．

レベルスケジューリング以外にも多様な手法が提案されている．Clustering for Heterogeneous Processors（CHP）[34] は関連の強いタスクどうしをクラスタリングしてからクラスタ単位でマシンに割り当てるが，uniform なマシンしか想定していない．割り当てるマシンを特定せずにタスクをクラスタとしてまとめる戦略には unrelated の場合には合理性がないため必然的に uniform に限られる．このほかに Cluster-M [72] や Triplet [56] がクラスタリングを行うが，これはタスクのマシンへの割当てまでしか行わず，スケジュールは作成しないため，DAG スケジューリングといってよいかどうか多少疑問が残る．各タスクがどのマシンに割り当てられるかが決まっても，先行制約があるためにタスクの実行順序により makespan が異なり，その最適化問題は強 NP 困難である[20)]．

クリティカルパスをできるだけ 1 台の速いマシンに割り当てる方針でスケジュールを構成する手法には Heterogeneous Critical Path Fast Duplication（HCPFD）[111)]，Bubble Scheduling and Allocation（BSA）[112)]，Critical-Path-on-a-Processor（CPOP）[152)]，(Enhanced) Critical-Path-On-a-Cluster（CPOC）[106)] などがある．Uniform な場合にはクリティカルパスを 1 台のマシンに割り当てるのは自然な選択だが，unrelated な場合に適用するにはここでもやはり限界がある．

Hybrid Heuristic [135)] は適当な優先度順に，互いに独立なタスクからなるグループに分割し，グループごとに独立タスクスケジューリングを適用する手法である．Herrmann ら[92)] は $R|chains|C_{max}$ に対して，上記のいずれにも分類しにくい（かなり計算量が多い）数個の近似アルゴリズムを提案・比較している．

分枝限定法[85)]，A* [52)]，タブーサーチ[130)]，Genetic Algorithm [14),144)]，Simulated Annealing [92)]，メトロポリス法[137)] などの一般的な最適化手法を用いた例も多い．Task duplication も考慮して混合整数線形計画に帰着させ，厳密な最適化を行った例[18)] もあるが，ごく小規模な問題にしか適用できない．

6.3 今後の課題

$R|prec|C_{max}$ に対する理論的な結果は貧弱で，しかも通信について考慮したものはほとんどない．容易に解決できるとは思えないが，真摯な努力が必要である．

その一方で実装ベースの研究は実験条件が不明確だったり，比較実験の対象が少なかったりして，絶対評価も相対評価も十分にはされていない．それでも $R|prec|C_{max}$ に対しては理論的に性能が保証された結果はほとんどないため，これらの提案は無視できない．今後システムのモデルとアルゴリズムの記述を統一し，一貫した説得力のある実験方法により比較評価し，知見を整理することが必要である．

DAG スケジューリングでは，あるタスクの開始時刻は，先行制約関係にあるタスクの終了時刻に依存している場合と，先行制約がないが同じマシンに割り当てられているタスクの終了時刻に依存している場合とがありうる．このためタスクの所要時間に不確定性がある場合，それが makespan に及ぼす影響は独立タスクの場合よりも格段に複雑になり，その解析だけでも研究課題となりうる[102),136)]．

なお，マシンや通信性能が未知の場合に対して，Kwok ら[113)] は実行中には計算量の大きいスケジューリングアルゴリズムを使うとオーバヘッドになることから，代表的なパラメータ値であらかじめオフラインスケジューリングを行っておき，実際の状況に最も近い仮定に基づくスケジュールを選択することにより，オンラインスケジューリングよりも良好な結果が得られるとしている．

7. オンラインスケジューリング

次にオンラインスケジューリングについて概観する．完全な**オンライン**というのは，タスクが 1 つずつ与えられ，そのタスクをスケジュールしてしまわないと次のタスクに関する情報が何も得られないものをいう．タスクの数や所要時間の上限・下限などの情報が事前に与えられている場合を**セミオンライン**と呼ぶこともあるようである．なお，プリエンプティブな場合については 4.2 節ですでに紹介している．

オンラインアルゴリズムの評価では，オフラインの最適解に対する性能比が広く用いられているが，これは**競合比**（competitive ratio）と呼ばれている．

なお，動的スケジューリングや動的負荷分散と呼ば

れるものと，オンラインスケジューリングとは異なる概念である．「動的スケジューリング」は，実行中にその場の情報を利用して行うスケジューリング手法を指しているが，複数のタスクの情報を蓄えてスケジューリングされることが多い．「動的負荷分散」という用語は，データ並列の場合や，負荷情報の獲得やタスクの移送などのシステム的な部分も含む広い概念である．

7.1 オンラインスケジューリング

$Q|online|C_{max}$ については，すでに紹介したとおり Cho らの論文[51] が $m > 2$ に対するリストスケジューリングが $(1+\sqrt{2m-2})/2$ 近似になることを示しているが，ここではタスクの順序が自由であるためオンラインスケジューリングと見なすことができる．

Aspnes ら[6] は $Q|online|C_{max}$ に対する競合比が 8 の，$R|online|C_{max}$ に対する競合比が $O(\log m)$ のアルゴリズムを示している．これはオンラインアルゴリズムの典型的な例になっているので $R|online|C_{max}$ のアルゴリズムの概略を紹介する．

まず，オフライン最適 makespan の見積り $C \geq C^{opt}_{max}$ と競合比の目標値 κ，1 以上の定数 a を仮定する．また，タスク i までスケジューリングした時点でマシン p に割り当てられているタスクの処理時間の合計を $C_p(i)$ とする．アルゴリズムは非常に単純で，$a^{(C_p(i-1)+t_{ip})/C} - a^{C_p(i-1)/C}$ を最小にするマシン p にタスク i を割り当てる．このとき $C_p(i-1)+t_{ip} > \kappa C$ となったら目標の競合比を超えているためスケジューリングは「失敗」であるが，以下のようにして決して失敗しない条件を導くことができる．すなわち**ポテンシャル関数**を

$$\Phi(i) = \sum_p a^{C_p(i)/C}(\gamma - C^{opt}_p(i)/C)$$

と定義する．オフライン最適解がタスク i をマシン p に割り当てるとすると，比較的簡単な計算の後

$$\Phi(i) - \Phi(i-1) \leq a^{C_p(i-1)/C}(\gamma(a^{t_{ip}/C} - 1) - t_{ip}/C)$$

が得られる．明らかに $t_{ip} \leq C^{opt}_{max} \leq C$ であるから，$a = 1 + 1/\gamma$ とおけば右辺は正にならない．すなわち $\Phi(i)$ は非増加関数であって，

$$\sum_p a^{C_p(i)/C}(\gamma - 1) \leq \Phi(0) = \gamma m$$

となる．よって

$$C_{max} = \max_p C_p(n) \leq C \log_a \left(\frac{\gamma}{\gamma - 1} m\right)$$

となる．このことは $\kappa = \log_a(m\gamma/(\gamma-1))$ とおけば $C \geq C^{opt}_{max}$ であればアルゴリズムが失敗しないことを示している．実際には適当な C で開始し，失敗したら C を 2 倍に再設定（**doubling** という）し，それまでのスケジュールはすべて忘れてスケジューリングをやりなおす．スケジューリングが完了したとき，全体の makespan は $2\kappa C$（C は最後のスケジューリングで用いたもの）未満であり，$C/2$ で失敗したことは $C/2 < C^{opt}_{max}$ を意味しているから，競合比は $4\kappa = O(\log m)$ 以下であることが分かる．

なお，$Q|online|C_{max}$ に対する競合比 8 のアルゴリズムは，$\kappa = 2$ として，makespan が $2C$ を超えない範囲で最も遅いマシンにタスクを割り当てるほかは，上記のアルゴリズムと同様である．

$Q|online|C_{max}$ については，その後 Berman ら[19] が競合比 5.828 の決定的アルゴリズムと 4.311 のランダマイズアルゴリズムとともに，競合比の下限として決定的アルゴリズムは 2.4380，ランダマイズは 1.8372 を示している．ランダマイズのほうの下限は前述の Epstein ら[71] により 2 に改善されている．また少数のマシンに対する特別な結果もいくつかある．$R|online|C_{max}$ の競合比は $O(\log m)$ が限界[7] であり，Aspnes の結果はオーダとしては最適である．

7.2 関連する問題

Sgall によるオンラインスケジューリングのサーベイ[138] は identical な場合なども含めて有用な知見をまとめており，一読の価値がある．また上記とは異なる問題設定の研究についても紹介されている．

タスクがある時刻（**release time** という）にならないとスケジュールできるようにならないような問題に対しては，上述の doubling に似た**バッチスケジューリング**という手法により，オフライン問題の性能比 σ の近似アルゴリズムを利用して競合比 2σ のアルゴリズムを構成することができる[143]．DAG スケジューリングにおいて，先行制約のあるタスクが完了した時点を release time と見なせば，この手法は DAG のオンラインスケジューリングとして見ることが可能である（ただし競合比は release time 固定の問題としての評価となり，DAG スケジューリングとしての競合比とは異なる）．

また，タスクの所要時間が不明という問題に対しては，uniform でも競合比は $O(\sqrt{m})$ より良くできない．これに対して Jaffe[98] の $Q||C_{max}$ に対する $\sqrt{m} + O(m^{1/4})$ 近似アルゴリズムと，Davis ら[59] の $R||C_{max}$ に対する $O(\sqrt{m})$ 近似アルゴリズムは，いずれも効率が既知であればタスクの所要時間が未知でも適用できる．

7.3 今後の課題

紹介した範囲の結果は理論的な限界と実際のアルゴ

リズムの性能が近く，満足のゆくものといえる．しかし，DAG やプリエンプティブスケジューリングの研究はほとんどないので，今後の展開が望まれる．

8. マルチプロセッサタスクスケジューリング

最後にマルチプロセッサタスクのスケジューリングについて論じる．ここで**マルチプロセッサタスク**とは，個々のタスクが複数のマシンを同時に使用して処理されるものである．MPI などを用いデータ並列などの手法ですでに並列化がされたプログラムが多数あって，それらのプログラムをタスクと見なしてスケジューリングするような状況を想定すると分かりやすい．このような問題設定は **space sharing** とか **mixed (data and task) parallelism** とか呼ばれることもある．

マルチプロセッサタスクは，個々のタスクがどのように並列処理できるかによっていくつかの種類に分けられる．ある流儀では，タスクを実行するマシン数が固定の場合を **rigid**，実行するマシン数を実行直前に選択できる場合を **moldable**，実行中にも変えられる場合を **malleable** と呼ぶ．

Moldable または malleable なタスクの場合，マシンがヘテロだと所要時間をモデル化するのが難しい．比較的一般的なものには，タスクがどのマシンセットを使用するかが固定されている **dedicated**（3 項表記では fix）と，マシンセットごとに所要時間が指定されている **general multiprocessor task**（3 項表記では set）がある．なお，タスクが使うマシン数がある値 k で固定のとき $fix = k$ あるいは $set = k$ のように，タスクが使うマシン数に上限 k があるとき $fix \leq k$ あるいは $set \leq k$ と書く．

これらは実質的に unrelated なマシンを意味するので，**本稿では 3 項表記の最初の項を R とするが，既存の文献では P となっている**．Uniform な性能モデルの試みとしては，Błażewicz ら[32] と Shachnai ら[139] が一応扱っているが，前者はホモなサブセットしか使わず，後者は理想的な速度向上を仮定していて，いずれも満足のゆくモデルとはいえない．Identical な場合にはマシン数と所要時間の関係を与えればよいが，この場合は**パラレルタスク**と呼ばれる．Drozdowski のサーベイ[64] はやや古いがパラレルタスクを含むこの分野の網羅的なサーベイで，非常に参考になる．

なお，マルチプロセッサタスクのスケジューリング問題は通信のスケジューリングとも関係が深い．集団通信のような多数の通信からなる問題を考え，衝突がおきないような最短時間で完了するスケジューリングを求めたいとする．すると，競合をおこすリソースをマシン，個別の通信をタスク，それぞれの通信が使用するリソースをタスクが使用するマシンにそれぞれ置き換えることにより，$R|fix|C_{max}$ に帰着することができる．ルーティングに自由度があれば $R|set|C_{max}$ となる．プリエンプションや先行制約も自然に翻訳できる．

8.1 $R|fix|C_{max}$

まず計算困難性についての結果を述べる．Krawczyk らの論文[107] はマルチプロセッサタスクの最初期の論文の 1 つであるが，ここでは $R|fix = 2, UET|C_{max}$ が NP 困難であることが示されている．さらに Kubale[109] は同じ問題が強 NP 困難であることを示している．また Błażewicz ら[30] は $R3|fix|C_{max}$ も強 NP 困難であるとしているが，後に証明を修正している[31]．Hoogeveen ら[94] は $R|fix, UET|C_{max}$ および $R3|fix|C_{max}$ はいずれも強 NP 困難であることを示した．$P \neq NP$ であれば，Kubale[110] は $R|fix = 2|C_{max}$ に対して，Hoogeveen ら[94] は $R|fix, UET|C_{max}$ に対して，それぞれ性能比が 4/3 より良い多項式近似近似アルゴリズムが存在しないことを示している．Fishkin ら[75] はさらに強く，$R|fix, UET|C_{max}$ には $\sqrt{m}$ より良い近似アルゴリズムが存在しないことが示せると述べている．

マシン数を固定した場合については，Kubale[110] は $Rm|fix = 2, UET|C_{max}$ が $O(n)$ の計算量で解けること，Hoogeveen らは上記の論文[94] で $Rm|fix, UET|C_{max}$ が多項式時間で解けることを示している．Amoura ら[3] は $Rm|fix|C_{max}$ に対する PTAS を示している．

$R|fix|C_{max}$ については，Dell'Olmo ら[60] が $R|fix = 2|C_{max}$ に対して $3s/(2s+1)$ 近似アルゴリズム（$m = 2s+1$）を与えている．さらに Bampis ら[10] は $R|fix \leq k, online, UET|C_{max}$ に対する k 近似，$R|fix, online, UET|C_{max}$ に対する $2\sqrt{m}$ 近似，$R|fix|C_{max}$ に対する $3\sqrt{m}$ 近似，$R|fix, online|C_{max}$ に対する $6\sqrt{m}$ 近似アルゴリズムを提出している．前述のとおり $R|fix|C_{max}$ には $\sqrt{m}$ より良い近似アルゴリズムはないので，Bampis らのアルゴリズムの性能比はオーダが最適である．

プリエンプティブにしても問題はあまり易しくならない．Bianco ら[28] は $R|fix, pmtn, UET|C_{max}$ が NP 困難であることを Kubale が 1990 年に示しているとしている．Jansen ら[101] は $R|fix = 2, pmtn|C_{max}$ は多項式時間で解けるが，$R|fix = 3, pmtn|C_{max}$ は強 NP 困難であることを示した．マシン数を固定した $Rm|fix, pmtn|C_{max}$ に対しては Bianco ら[28] が線形

計画による解，Amoura ら[3] が線形時間の解を示している．

8.2 $R|set|C_{max}$

Bianco ら[29] は $R2|set|C_{max}$ と $R3|set|C_{max}$ を解いているが，$R|set|C_{max}$ にはほとんど自明な m 近似アルゴリズムしか得ていない．その後 Chen ら[46] は $Rm|set|C_{max}$ に対して $m/2+\varepsilon$ 近似アルゴリズムを提示し，さらに Chen ら[47] および Jansen ら[100] がいずれも $Rm|set|C_{max}$ に対する PTAS を提出した．

しかしマシン数を変数に扱った $R|set|C_{max}$ については目立った成果がない．上記の Bianco らの m 近似アルゴリズムほかには，Miranda ら[123] が，最悪性能比が定数となるような多項式近似アルゴリズムが存在しないことを証明している．

Bianco らの上述の論文[29] は $Rm|set, pmtn|C_{max}$ が線形計画を用いて多項式時間で解けることを示した．しかし m を変数扱いした場合については Jansen と Porkolab[101] が $R|set = 2, pmtn|C_{max}$ でも強 NP 困難であることを示している．

8.3 今後の課題

$R|fix|C_{max}$ についてはある程度の結果が得られており，通信のスケジューリングなどに利用できる可能性がある．$R|set|C_{max}$ については否定的な結果が多く，1 つのタスクが使用できるマシン数を制約するなど工夫してアルゴリズムを構築することが望まれる．実装ベースの研究はヘテロではきわめて少ない[150]．

また，DAG スケジューリングと同様に，同一のマシンセットでのみ実行を再開できる制限されたプリエンプションがマルチプロセッサタスクでも役に立つ．マルチプロセッサタスクを移送するのは逐次タスクの移送よりもさらに大変であろうから，このような制限は現実的である．Drozdovsky によるサーベイ[64] では $pmtn$ はこの制限つきのものを指しており，マシンセットを変更するものは variable profile と呼んでいる．しかし制限つきのプリエンプションの結果は，4 台以下のものと private communication のみとなっており，これも今後の研究課題である．

9. おわりに

本稿ではヘテロ並列計算環境のためのタスクスケジューリング手法についてサーベイを行った．DLT (Divisible Load Theory)，プリエンプティブスケジューリング，独立タスクスケジューリング，DAG スケジューリング，オンラインスケジューリング，それにマルチプロセッサタスクスケジューリングについて，これまで知られている結果について概観し，今後の課題についても考察した．

本稿に対し，理論に偏っていて不満を感じた読者もいるかもしれない．DLT 以外の理論分野では通信コストを考慮した研究が（ヘテロなマシンに対しては）ほとんどないので，満足できる問題設定と解とはいい難いのは事実である．しかしながら DAG スケジューリングで見たように，理論的評価のない実装実験のみの評価も十分に行われていない．理論・実践の両分野の歩み寄りが必要であるが，それは必ずしも容易なことではない．今後さらにさまざまな問題設定の工夫がなされて理論と実践との距離が近づくことを願いたい．また，この分野における日本人の貢献が少ないので，多くの方々にこの分野を知っていただくことには価値があると思う．本稿がわが国の並列処理のさらなる発展にわずかながらでも寄与できれば幸甚である．

謝辞 論文を改良する貴重なご意見をくださった匿名査読者の方々に感謝を申し上げます．

本研究は文部科学省 21 世紀 COE プログラムおよび科学研究費による支援を受けています．

参考文献

1) Agrawal, R. and Jagadish, H.V.: Partitioning Techniques for Large Grained Parallelism, *IEEE Trans. Comput.*, Vol.37, No.12, pp.1627–1634 (1988).
2) Anil Kumar, V.S., Marathe, M.V., Parthasarathy, S. and Srinivasan, A.: Scheduling on unrelated machines under tree-like precedence constraints, *Proc. Approx. Rand. Comb. Optim.*, LNCS☆ 3624, pp.146–157 (2005).
3) Amoura, A.K., Bampis, E., Kenyon, C. and Manoussakis, Y.: Scheduling independent multiprocessor tasks, *Proc. 5th Eur. Symp. Alg.*, LNCS 1284, pp.1–12 (1997).
4) Aritsugi, M., Fukatsu, H. and Kanamori, Y.: Several partitioning strategies for parallel image convolution in a network of heterogeneous workstations, *Par. Comp.*, Vol.27, No.3, pp.269–293 (2001).
5) Armstrong, R., Hensgen, D. and Kidd, T.: The relative performance of various mapping algorithms is independent of sizable variances in run-time predictions, *Proc. Workshop Hetero. Proc.*, pp.79–87 (1998).
6) Aspnes, J., et al.: On-line routing of virtual circuits with applications to load balancing and machine scheduling, *J. ACM*, Vol.44, No.3, pp.486–504 (1997).

☆ LNCS: Lecture Notes on Computer Science, Springer

7) Azar, Y., Naor, J. and Rom, R.: The Competitiveness of On-Line Assignments, *J. Alg.*, Vol.18, No.2, pp.221–237 (1995).
8) Azar, Y. and Epstein, A.: Convex programming for scheduling unrelated parallel machines, *Proc. Ann. Symp. Theory of Comp.*, pp.331–337 (2005).
9) Bajaj, R. and Agrawal, D.P.: Improving Scheduling of Tasks in a Heterogeneous Environment, *IEEE Trans. Par. Distr. Sys.*, Vol.15, No.2, pp.107–118 (2004).
10) Bampis, E., et al.: Scheduling of independent dedicated multiprocessor tasks, *Proc. Ann. Int'l Symp. Alg. Comp.*, LNCS 2518, pp.391–402 (2002).
11) Banino, C., et al.: Scheduling strategies for master-slave tasking on heterogeneous processor platforms, *IEEE Trans. Par. Distr. Sys.*, Vol.15, No.4, pp.319–330 (2004).
12) Barlas, G.D.: Collection-aware optimum sequencing of operations and closed-form solutions for the distribution of a divisible load on arbitrary processor trees, *IEEE Trans. Par. Distr. Sys.*, Vol.9, No.5, pp.429–441 (1998).
13) Baskiyar, S. and Dickinson, C.: Scheduling directed a-cyclic task graphs on a bounded set of heterogeneous processors using task duplication, *J. Par. Distr. Comp.*, Vol.65, No.8, pp.911–921 (2005).
14) Baxter, M.J., Tokhi, M.O. and Fleming, P.J.: Investigation of the heterogeneous mapping problem using genetic algorithms, *Proc. UKACC Int. Conf. Control.*, pt.1, pp.448–453 (1996).
15) Beaumont, O., et al.: Bandwidth-centric allocation of independent tasks on heterogeneous platforms, *Proc. Int'l Par. Distr. Proc. Symp.*, pp.633–638 (2002).
16) Beaumont, O., Legrand, A. and Robert, Y.: The master-slave paradigm with heterogeneous processors, *IEEE Trans. Par. Distr. Sys.*, Vol.14, No.9, pp.897–908 (2003).
17) Beaumont, O., Legrand, A. and Robert, Y.: Scheduling divisible workloads on heterogeneous platforms, *Par. Comp.*, Vol.29, No.9, pp.1121–1152 (2003).
18) Bender, A.: MILP based task mapping for heterogeneous multiprocessor systems, *Proc, EURO-DAC '96*, pp.190–197 (1996).
19) Berman, P., Charikar, M. and Karpinski, M.: On-line load balancing for related machines, *Proc. Workshop Alg. Data Str.*, LNCS 1272, pp.116–125 (1997).
20) Bernstein, D., Rodeh, M. and Gertner, I.: On the complexity of scheduling problems for parallel/pipelined machines, *IEEE Trans. Comput.*, Vol.38, No.9, pp.1308–1313 (1989).
21) Bharadwaj, V., Ghose, D. and Mani, V.: Optimal sequencing and arrangement in distributed single-level tree networks with communication delays, *IEEE Trans. Par. Distr. Sys.*, Vol.5, No.9, pp.968–976 (1994).
22) Bharadwaj, V., Ghose, D. and Mani, V.: Multi-installment load distribution in tree networks with delays, *IEEE Trans. Aero. Electr. Sys.*, Vol.31, No.2, pp.555–567 (1995).
23) Bharadwaj, V. and Viswanadham, N.: Suboptimal solutions using integer approximation techniques for scheduling divisible loads on distributed bus networks, *IEEE Trans. Sys. Man, Cyb. A*, Vol.30, No.6, pp.680–691 (2000).
24) Bharadwaj, V., Li, X. and Chung, K.C.: On the influence of start-up costs in scheduling divisible loads on bus networks, *IEEE Trans. Par. Distr. Sys.*, Vol.11, No.12, pp.1288–1305 (2000).
25) Bharadwaj, V. and Barlas, G.: Access time minimization for distributed multimedia applications, *Multimedia Tools and Applications*, Vol.12, No.2-3, pp.235–256 (2000).
26) Bharadwaj, V. and Barlas, G.: Efficient scheduling strategies for processing multiple divisible loads on bus networks, *J. Par. Distr. Comp.*, Vol.62, No.1, pp.132–151 (2002).
27) Bharadwaj, V., Ghose, D. and Robertazzi, T.G.: Divisible Load Theory: A New Paradigm for Load Scheduling in Distributed Systems, *Cluster Computing*, Vol.6, No.1, pp.7–17 (2003).
28) Bianco, L., Błażewicz, J., Dell'Olmo, P. and Drozdowski, M.: Scheduling preemptive multiprocessor tasks on dedicated processors, *Perf. Eval.*, Vol.20, No.4, pp.361–371 (1994).
29) Bianco, L., Błażewicz, J., Dell'Olmo, P. and Drozdowski, M.: Scheduling multiprocessor tasks on a dynamic configuration of dedicated processors, *Ann. Oper. Res.*, Vol.58, No.7, pp.493–517 (1995).
30) Błażewicz, J., Dell'Olmo, P., Drozdowski, M. and Speranza, M.G.: Scheduling multiprocessor tasks on three dedicated processors, *Info. Proc. Lett.*, Vol.41, No.5, pp.275–280 (1992).
31) Błażewicz, J., Dell'Olmo, P., Drozdowski, M. and Speranza, M.: Corrigendum: Scheduling multiprocessor tasks on three dedicated processors, *Info. Proc. Lett.*, Vol.49, No.5, pp.269–270 (1994).
32) Błażewicz, J., Drozdowski, M., Schmidt, G. and de Werra, D: Scheduling independent mul-

tiprocessor tasks on a uniform k-processor system, *Par. Comp.*, Vol.20, No.1, pp.15–28 (1994).
33) Blazewicz, J. and Kobler, D.: Review of properties of different precedence graphs for scheduling problems, *Eur. J. Oper. Res.*, Vol.142, No.3, pp.435–443 (2002).
34) Boeres, C., Filho, J.V. and Rebello, V.E.F.: A cluster-based strategy for scheduling task on heterogeneous processors, *Proc. Symp. Comp. Arch. High Perf. Comp.*, pp.214–221 (2004).
35) Boeres, C., Rios, E. and Ochi, L.S.: Hybrid evolutionary static scheduling for heterogeneous systems, *IEEE Cong. Evol. Comp.*, pp.1929–1936 (2005).
36) Boudet, V., Rastello, F. and Robert, Y.: PVM implementation of heterogeneous ScaLAPACK dense linear solvers, *Proc. Euro. PVM/MPI*, LNCS 1697, pp.333–340 (1999).
37) Boyer, W.F. and Hura, G.S.: Non-evolutionary algorithm for scheduling dependent tasks in distributed heterogeneous computing environments, *J. Par. Distr. Comp.*, Vol.65, No.9, pp.1035–1046 (2005).
38) Braun, T.D., et al.: A taxonomy for describing matching and scheduling heuristics for mixed-machine heterogeneous computing systems, *IEEE Symp. Reliable Distr. Sys.*, pp.330–335 (1998).
39) Braun, T.D., et al.: A comparison study of static mapping heuristics for a class of meta-tasks on heterogeneous computing systems, *Proc. Hetero. Comp. Workshop*, pp.15–29 (1999).
40) Brauner, N., Crama, Y., Grigoriev, A. and van de Klundert, J.: A framework for the complexity of high-multiplicity scheduling problems, *J. Comb. Opt.*, Vol.9, No.3, pp.313–323 (2005).
41) Carino, R.L. and Banicescu, I.: Dynamic scheduling parallel loops with variable iterate execution times, *Proc. Int'l Par. Distr. Proc. Symp.,* pp.239–246 (2002).
42) Casanova, H., Legrand, A., Zagorodnov, D. and Berman, F.: Heuristics for scheduling parameter sweep applications in grid environments, *Proc. Hetero. Comp. Workshop*, pp.349–363 (2000).
43) Chan, W.-Y. and Li, C.-K.: Scheduling tasks in DAG to heterogeneous processor system, *Proc. Euromicro Workshop Par. Distr. Proc.*, pp.27–31 (1998).
44) Chekuri, C. and Bender M.A.: An efficient approximation algorithm for minimizing makespan on uniformly related machines, *Proc. Conf. Integer Prog. Comb. Opt.*, LNCS 1412, pp.383–393 (1998).
45) Chen, B.: Tighter bound for MULTIFIT scheduling on uniform processors, *Discr. Appl. Math.*, Vol.31, No.3, p.227 (1991).
46) Chen, J. and Lee, C.-Y.: General multiprocessor task scheduling, *Naval Res. Logist.*, Vol.46, No.1, pp.57–74 (1999).
47) Chen, J. and Miranda, A.: A polynomial time approximation scheme for general multiprocessor job scheduling, *Proc. Symp. Theory of Computing*, pp.418–427 (1999).
48) Cheng, Y.C. and Robertazzi, T.G.: Distributed computation with communication delays, *IEEE Trans. Aero. Electr. Sys.*, Vol.24, No.6, pp.700–712 (1988).
49) Cheng, Y.C. and Robertazzi, T.G.: Distributed Computation for a Tree Network with Communication Delays, *IEEE Trans. Aero. Electr. Sys.*, Vol.26, No.3, pp.511–516 (1990).
50) Cheng, K.-W., Yang, C.-T., Lai, C.-L. and Chang, S.-C.: A parallel loop self-scheduling on Grid computing environments, *Proc. Int'l Symp. Par. Arch. Alg. Netw.*, pp.409–414 (2004).
51) Cho, Y. and Sahni, S.: Bounds for list schedules on uniform processors, *SIAM J Comput.*, Vol.9, No.1, pp.91–103 (1980).
52) Chow, K.-W. and Liu, B.: On mapping signal processing algorithms to a heterogeneous multiprocessor system, *Proc. Int'l Conf. Acoustics, Speech and Sig. Proc.*, Vol.3, pp.1585–1588 (1991).
53) Chronopoulos, A.T., Penmatsa, S., Xu, J.-H. and Ali, S.: Distributed loop-scheduling schemes for heterogeneous computer systems, *Concurrency Comp.: Pract. Exper.*, Vol.18, No.7, pp.771–785 (2006).
54) Chudak, F.A. and Shmoys, D.: Approximation algorithms for precedence-constrained scheduling problems on parallel machines that run at different speeds, *Proc. Symp. Discr. Alg.*, pp.581–590 (1997).
55) Cierniak, M., Zaki, M.J. and Li, W.: Compile-time scheduling algorithms for a heterogeneous network of workstations, *Computer J.*, Vol.40, No.6, pp.356–372 (1997).
56) Cirou, B. and Jeannot, E.: Triplet: a clustering scheduling algorithm for heterogeneous systems *Proc. Int'l Conf. Par. Proc. Workshop*, pp.237–244 (2001).
57) Clarke, S. and Dandamudi, S.P.: Usefulness of adaptive load sharing for parallel processing on networks of workstations, *Concurrency: Pract.*

Exper., Vol.11, No.8, pp.387–405 (1999).
58) Coll, P.E., Ribeiro, C.C. and de Souza, C.C.: Multiprocessor scheduling under precedence constraints: Polyhedral results, *Discr. Appl. Math.*, Vol.154, No.5, pp.770–801 (2006).
59) Davis, E. and Jaffe, J.M.: Algorithms for scheduling tasks on unrelated processors, *J. ACM*, Vol.28, No.4, pp.721–736 (1981).
60) Dell'Olmo, P., Giordani, S. and Speranza, M.G.: An approximation result for a duo-processor task scheduling problem, *Info. Proc. Lett.*, Vol.61, No.4, pp.195–200 (1997).
61) Dobson, G.: Scheduling independent tasks on uniform processors, *SIAM J. Comput.*, Vol.13, No.4, pp.705–716 (1984).
62) Dogan, A.: Stochastic scheduling of a meta-task in heterogeneous distributed computing, *Proc. Int'l Conf. Par. Proc. Workshop*, pp.369–375 (2001).
63) Dogan, A. and Ozguner, R.: LDBS: a duplication based scheduling algorithm for heterogeneous computing systems, *Proc. Int'l Conf. Par. Proc.*, pp.352–359 (2002).
64) Drozdowski, M.: Scheduling multiprocessor tasks — an overview, *Euro. J. Oper. Res.*, Vol.94, No.2, pp.215–230 (1996).
65) Drozdowski, M., Lawenda, M. and Guinand, F.: Scheduling multiple divisible loads, Tech. Rep., RA-007/04, Inst. Comp. Sci., Poznan U. Tech. (2004).
66) Drozdowski, M. and Lawenda, M.: On optimum multi-installment divisible load processing in heterogeneous distributed systems, *Proc. Int'l Euro-Par Conf.*, LNCS 3648, pp.231–240 (2005).
67) Drozdowski, M. and Lawenda, M.: The combinatorics in divisible load scheduling, *Found. Comp. Decision Sci.*, Vol.30, No.4, pp.297–308 (2005).
68) Dutot, P.-F.: Master-slave tasking on heterogeneous processors, *Proc. Int'l Par. Distr. Proc. Symp.* (2003).
69) Dutot, P.-F.: Complexity of master-slave tasking on heterogeneous trees, *Euro. J. Oper. Res.*, Vol.164, No.3, pp.690–695 (2005).
70) El-Rewini, H. and Lewis, T.G.: Scheduling parallel program tasks onto arbitrary target machines, *J. Par. Distr. Comp.*, Vol.9, No.2, pp.138–153 (1990).
71) Epstein, L. and Sgall, J.: A lower bound for on-line scheduling on uniformly related machines, *Oper. Res. Lett.*, Vol.26, No.1, pp.17–22 (2000).
72) Eshaghian, M.M. and Wu, Y.C.: Mapping Heterogeneous Task Graphs onto Heterogeneous System Graphs, *Proc. Hetero. Comp. Workshop*, pp.147–160 (1997).
73) Ferreto, T. and De Rose, C.: Scheduling divisible workloads using the adaptive time factoring algorithm, *Proc. Int'l Conf. Algo. Arch. Par. Proc.*, LNCS 3719, pp.232–239 (2005).
74) Fishkin, A.V., Jansen, K. and Mastrolilli, M.: Grouping techniques for scheduling problems: simpler and faster, *Proc. Eur. Symp. Alg.*, LNCS 2161, pp.206–217 (2001).
75) Fishkin, A., Jansen, K. and Porkolab, L.: On minimizing average weighted completion time of multiprocessor tasks with release dates, *Proc. Int'l Colloq. Automata, Lang. Prog.*, LNCS 2076, pp.875–886 (2001).
76) Flynn Hummel, S., Schmidt, J., Uma, R.N. and Wein, J.: Load-sharing in heterogeneous system via weighted factoring *Ann. ACM Symp. Par. Alg. Arch.*, pp.318–328 (1996).
77) Freund, R., Kidd, T., Hensgen, D. and Moore, L.: SmartNet: A scheduling framework for heterogeneous computing, *Proc. Int'l Symp. Par. Arch., Alg. Net.*, pp.514–521 (1996).
78) Friesen, D.K. and Langston, M.A.: Bounds for MULTIFIT scheduling on uniform processors, *SIAM J. Comput.*, Vol.12, No.1, pp.60–70 (1983).
79) Friesen, D.K.: Tighter bounds for LPT scheduling on uniform processors, *SIAM J. Comput.*, Vol.16, No.3, pp.554–560 (1987).
80) Gairing, M., Monien, B. and Woclaw, A.: A faster combinatorial approximation algorithm for scheduling unrelated parallel machines, *Proc. Int'l Colloq. Automata, Lang. Prog.*, LNCS 3580, pp.828–839 (2005).
81) Genaud, S., Giersch, A. and Vivien, F.: Load-balancing scatter operations for grid computing, *Par. Comp.*, Vol.30, No.8, pp.923–946 (2004).
82) Ghanbari, S. and Meybodi, M.R.: Learning automata based algorithms for mapping of a class of independent tasks over highly heterogeneous grids, *Proc. Euro. Grid Conf.*, LNCS 3470, pp.681–690 (2005).
83) Ghirardi, M. and Potts, C.N.: Makespan minimization for scheduling unrelated parallel machines: A recovering beam search approach, *Euro. J. Oper. Res.*, Vol.165, No.2, pp.457–467 (2005).
84) Glass, C.A., Potts, C.N. and Shade, P.: Unrelated parallel machine scheduling using local search, *Math. Comp. Modelling*, Vol.20, No.2, pp.41–52 (1994).

85) Greenblatt, B. and Linn, C.J.: Branch and bound style algorithms for scheduling communicating tasks in a distributed system, *COMPCON Spring '87*, pp.12–17 (1987).
86) Gonzalez, T., Ibarra, O.H. and Sahni, S.: Bounds for LPT schedules on uniform processors, *SIAM J. Comput.*, Vol.6, No.1, pp.155–166 (1977).
87) Gonzalez, T. and Sahni, S.: Preemptive scheduling of uniform processor systems, *J. ACM*, Vol.25, No.1, pp.92–101 (1978).
88) Graham, R.L., Lawler, E.L., Lenstra, J.K. and Rinnooy Kan, A.H.G.R.: Optimization and approximation in deterministic sequencing and scheduling: a survey, *Annals of Discrete Mathematics*, Vol.5, pp.287–326 (1979).
89) Haddad, E.: Optimal dynamic redistribution of divisible load in distributed real-time systems, *Proc. IEEE Workshop Real-Time Appl.*, pp.21–26 (1994).
90) Haddad, E.: Real-time optimization of distributed load balancing, *Proc. Workshop Par. Distr. Real-Time Sys.*, pp.52–57 (1994).
91) Hariri, A.M.A. and Potts, C.N.: Heuristics for scheduling unrelated parallel machines, *Comp. Oper. Res.*, Vol.18, No.3, pp.323–331 (1991).
92) Herrmann, J., Proth, J.-M. and Sauer, N.: Heuristics for unrelated machine scheduling with precedence constraints, *Euro. J. Oper. Res.*, Vol.102, No.3, pp.528–537 (1997).
93) Hochbaum, D.S. and Shmoys, D.B.: A polynomial approximation scheme for machine scheduling on uniform processors: using the dual approximation approach, *Proc. Conf. Found. Software Tech. Theor. Comp. Sci.*, pp.382–393 (1986).
94) Hoogeveen, J.A., van de Velde, S.L. and Veltman, B.: Complexity of scheduling multiprocessor tasks with prespecified processor allocations, *Discr. Appl. Math.*, Vol.55, No.3, pp.259–272 (1994).
95) Horowitz, E. and Sahni, S.: Exact and approximate algorithms for scheduling nonidentical processors, *J. ACM*, Vol.23, No.2, pp.317–327 (1976).
96) Horvath, E.C., Lam, S. and Sethi, R.: A level algorithm for preemptive scheduling *J. ACM*, Vol.24, No.1, pp.32–43 (1977).
97) Ibarra, O.H. and Kim, C.E.: Heuristic algorithms for scheduling independent tasks on nonidentical processors, *J. ACM*, Vol.24, No.2 pp.280–289 (1977).
98) Jaffe, J.: Efficient scheduling of tasks without full use of processor resources, *Theor. Comp. Sci.*, Vol.12, No.1, pp.1–17 (1980).
99) Jansen, K. and Porkolab, L.: Improved approximation schemes for scheduling unrelated parallel machines, *Proc. Symp. Theor. Comp.*, pp.408–417 (1999).
100) Jansen, K. and Porkolab, L.: General multiprocessor task scheduling: approximate solutions in linear time, *Proc. Alg. Data Str.*, LNCS 1663, pp.110–121 (1999).
101) Jansen, K. and Porkolab, L.: Preemptive scheduling on dedicated processors: applications of fractional graph coloring, *Proc. Math. Found. Comp. Sci.*, LNCS 1893, pp.446–455 (2000).
102) Kamthe, A., Lee, S.-Y.: A stochastic approach to estimating earliest start times of nodes for scheduling DAGs on heterogeneous distributed computing systems, *Proc. Int'l Par. Dist. Proc. Symp.*, p.121b (2005).
103) Kidd, T. and Hensgen, D.: Why the mean is inadequate for accurate scheduling decisions, *Proc. Int'l Symp. Par. Arch. Algo. Netw.*, pp.262–267 (1999).
104) Kim, G.H. and Lee, C.S.G.: Genetic reinforcement learning for scheduling heterogeneous machines, *Proc. Int'l Conf. Robot. Autom.*, pt.3, pp.2798–2803 (1996).
105) Kim, H.J., Jee, G.-I. and Lee, J.G.: Optimal Load Distribution for Tree Network Processors, *IEEE Trans. Aero. Electr. Sys.*, Vol.32, No.2, pp.607–612 (1996).
106) Kim, J., Rho, J., Lee, J.-O. and Ko, M.-C.: Effective static task scheduling for realistic heterogeneous environment, *Proc. Int'l Workshop Distr. Comp.*, LNCS 3741, pp.428–438 (2005).
107) Krawczyk, H. and Kubale, M.: An approximation algorithm for diagnostic test scheduling in multiprocessor systems, *IEEE Trans. Comp.*, Vol.34, No.9, pp.869–872 (1985).
108) Kubiak, W., Penz, B. and Trystram, D.: Scheduling chains on uniform processors with communication delays, *J. Scheduling*, Vol.5, No.6, pp.459–476 (2002).
109) Kubale, M.: The complexity of scheduling independent two-processor tasks on dedicated processors, *Info. Proc. Lett.*, Vol.24, No.3, pp.141–147 (1987).
110) Kubale, M.: Preemptive versus nonpreemptive scheduling of biprocessor tasks on dedicated processors, *Euro. J. Oper. Res*, Vol.94, No.2, pp.242–251 (1996).
111) Kwok, Y.-K. and Ahmad, I.: Exploiting duplication to minimize the execution times of parallel programs on message-passing systems, *Proc.*

Int'l Par. Proc. Symp., pp.426–433 (1994).
112) Kwok, Y.-K. and Ahmad, I.: Link contention-constrained scheduling and mapping of tasks and messages to a network of heterogeneous processors, *Proc. Int'l Conf. Par. Proc.*, pp.551–558 (1999).
113) Kwok, Y.-K., et al.: A semi-static approach to mapping dynamic iterative tasks onto heterogeneous computing systems, *J. Par. Distr. Comp.*, Vol.66, No.1, pp.77–98 (2006).
114) Lawler, E.L. and Labetoulle, J.: On preemptive scheduling of unrelated parallel processors by linear programming *J. ACM*, Vol.25, No.4, pp.612–619 (1978).
115) Lenstra, J.K., Shmoys, D.B. and Tardos, E.: Approximation algorithms for scheduling unrelated parallel machines, *Proc. Ann. Symp. Found. Comp. Sci.*, pp.217–224 (1987).
116) Lin, J.H. and Vitter, J.S.: ϵ-approximations with minimum packing constraint violation, *Proc. Symp. Theor. Comp.*, pp.771–782 (1992).
117) Liu, J.W.S. and Liu, C.L.: Bounds on scheduling algorithms for heterogeneous computing systems, *Proc. IFIP Congr.*, pp.349–353 (1974).
118) Maculan, N., Porto, S.C.S., Ribeiro, C.C. and Carvalho de Souza, C.: A new formulation for scheduling unrelated processor under precedence constraints, *RAIRO Recherche Operationnelle*, Vol.33, No.1, pp.87–92 (1999).
119) Maheswaran, M., et al.: Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems, *Proc. Hetero. Comp. Workshop*, pp.30–44 (1999).
120) Marchal, L., Robert, Y., Yang, Y. and Casanova, H.: A realistic network/application model for scheduling divisible loads on large-scale platforms, *Proc. Int'l Par. Dist. Proc. Symp.*, p.48b (2005).
121) Martello, S., Soumis, F. and Toth, P.: Exact and approximation algorithms for makespan minimization on unrelated parallel machines, *Discr. Appl. Math.*, Vol.75, No.2, pp.169–188 (1997).
122) Menascé, D.A., Porto, S.C.S. and Tripathi, S.K.: Processor assignment in heterogeneous parallel architectures, *Proc. Int'l Conf. Par. Proc.*, pp.186–191 (1992).
123) Miranda, A., Torres, L. and Chen, J.: On the approximability of multiprocessor task scheduling problems, *Proc. Int'l Symp. Alg. Comp.*. LNCS 2518, pp.403–415 (2002).
124) Mokotoff, E. and Jimeno, J.L.: Heuristics based on partial enumeration for the unrelated parallel processor scheduling problem, *Ann. Oper. Res.*, Vol.117, No.1-4, pp.133–150 (2002).
125) Mokotoff, E. and Chretienne, P.: A cutting plane algorithm for the unrelated parallel machine scheduling problem, *Eur. J. Oper. Res.*, Vol.141, No.3, pp.515–525 (2002).
126) Morrison, J.F.: Note on LPT scheduling, *Oper. Res. Lett.*, Vol.7, No.2, pp.77–79 (1988).
127) Oh, H. and Ha, S.: A Static Scheduling Heuristic for Heterogeneous Processors, *Proc. Euro. Conf. Par. Proc.*, Vol.2, pp.573–577 (1996).
128) Ooshita, F., Matsumae, S. and Masuzawa, T.: Scheduling for Independent-Task Applications on Heterogeneous Parallel Computing Environments under the Unidirectional One-Port Model, *IEICE Trans. Info. Sys.*, to appear.
129) Piersma, N. and van Dijk, W.: A Local Search Heuristic for Unrelated Parallel Machine Scheduling with Efficient Neighborhood Search, *Math. Comp. Modelling*, Vol.24, No.9, pp.11–19 (1996).
130) Porto, S.C.S. and Ribeiro, C.C.: A tabu search approach to task scheduling on heterogeneous processors under precedence constraints, *Int'l J. High Speed Comp.*, Vol.7, No.1, pp.45–71 (1995).
131) Rădulescu, A. and van Gemund, A.J.C.: Fast and effective task scheduling in heterogeneous systems, *Proc. Hetero. Comp. Workshop*, pp.229–238 (2000).
132) Ranaweera, S. and Agrawal, D.P.: A task duplication based scheduling algorithm for heterogeneous systems, *Proc. Int'l Par. Distr. Proc. Symp.*, pp.445–450 (2000).
133) Rosenberg, A.L.: Sharing partitionable workloads in heterogeneous NOWs: greedier is not better, *Proc, IEEE Int'l Conf. Cluster Comp.*, pp.124–131 (2001).
134) Rzadca, K. and Seredynski, F.: Heterogeneous multiprocessor scheduling with differential evolution, *Proc. IEEE Congr. Evol. Comp.*, pt.3, pp.2840–2847 (2005).
135) Sakellariou, R. and Zhao, H.: A hybrid heuristic for DAG scheduling on heterogeneous systems, *Proc. Int'l Par. Distr. Proc. Symp.*, pp.111–123 (2004).
136) Sanlaville, E.: Sensitivity bounds for machine scheduling with uncertain communication delays, *J. Scheduling*, Vol.8, pp.461–473, (2005).
137) Scholz, P. and Harbeck, E.: Task assignment for distributed computing, *Proc. Conf. Adv. Par. Distr. Comp.*, pp.270–277 (1997).
138) Sgall, J.: On-line scheduling, *Online Algorithms: the State of the Art*, LNCS 1442, Fiat A., Woeginger G.J. (Eds), pp.196–231 (1998).

139) Shachnai, H. and Tamir, T.: Multiprocessor scheduling with machine allotment and parallelism constraints, *Algorithmica*, Vol.32, No.4, pp.651–678 (2002).
140) Shachnai, H., Tamir, T. and Woeginger, G.J.: Minimizing makespan and preemption costs on a system of uniform machines, *Proc. Ann. Eur. Symp. Alg.*, LNCS 2461, pp.859–871 (2002).
141) Shchepin, E.V. and Vakhania, N.: An optimal rounding gives a better approximation for scheduling unrelated machines, *Oper. Res. Lett.*, Vol.33, No.2, pp.127–133 (2005).
142) Shmoys, D.B. and Tardos, E.: Scheduling unrelated machines with costs, *Proc. Ann. Symp. Discr. Alg.*, pp.448–454 (1993).
143) Shmoys, D.B., Wein, J. and Williamson, D.P.: Scheduling parallel machines on-line, *SIAM J. Comp.*, Vol.24, No.6, pp.1313–1331 (1995).
144) Siegel, H.J., Wang, L., Roychowdhury, V.P. and Tan, M.: Computing with heterogeneous parallel machines: Advantages and challenges, *Proc. Int'l Symp. Par. Arch. Alg. Netw.*, pp.368–374 (1996).
145) Sih, G.C. and Lee, E.A.: Dynamic-level scheduling for heterogeneous processor networks, *Proc. Symp. Par. Distr. Proc.*, pp.42–49 (1990).
146) Sinnen, O., Sousa, L.A. and Sandnes, F.E.: Toward a realistic task scheduling model, *IEEE Trans. Par. Distr. Sys.*, Vol.17, No.3, pp.263–275 (2006).
147) Spinrad, J.: Worst-case analysis of a scheduling algorithm, *Oper. Res. Lett.*, Vol.4, No.1, pp.9–11 (1985).
148) Stone, H.S. and Bokhari, S.H.: Control of Distributed Processes, *IEEE Computer*, Vol.11, No.7, pp.97–106 (1978).
149) Suda, R. and Tomi, S.: Task Redistribution Scheduling using Multi-Master Divisible Load Model, *Proc. Par. Dist. Comp. Sys.*, to appear (2006).
150) Suter, F., Desprez, F. and Casanova, H.: From heterogeneous task scheduling to heterogeneous mixed parallel scheduling, *Proc. Euro-Par*, LNCS 3149, pp.230–237 (2004).
151) Thomas McCormick, S., Smallwood, S.R. and Spieksma, F.C.R.: A polynomial algorithm for multiprocessor scheduling with two job lengths, *Math. Oper. Res.*, Vol.26, No.1, pp.31–49 (2001).
152) Topcuoglu, H., Hariri, S.and Wu, M.-Y.: Task scheduling algorithms for heterogeneous processors, *Proc. Hetero. Comp. Workshop*, pp.3–14 (1999).
153) Veltman, B., Lageweg, B.J. and Lenstra, J.K.: Multiprocessor scheduling with communication delays, *Par. Comp.*, Vol.16, pp.173–182 (1990).
154) Woeginger, G.J.: Comment on scheduling on uniform machines under chain-type precedence constraints, *Oper. Res. Lett.*, Vol.26, No.3, pp.107–109 (2000).
155) Woo, N. and Yeom, H.Y.: K-depth look-ahead task scheduling in network of heterogeneous processors, *Int'l Conf. Info. Netw.*, pt.2, LNCS 2344, pp.736–745 (2002).
156) Wu, M.-Y., Shu, W. and Zhang, H.: Segmented min-min: a static mapping algorithm for meta-tasks on heterogeneous computing systems, *Proc. Hetero. Comp. Workshop*, pp.375–385 (2000).
157) Wu, M.-Y. and Shu, W.: A high-performance mapping algorithm for heterogeneous computing systems, *Proc. Int'l Par. Distr. Proc. Symp.* (2001).
158) Yamamoto, H., Tsuru, M. and Oie, Y.: Parallel transferable uniform multi-round algorithm for achieving minimum application turnaround times for divisible workload, *Proc. Int'l Conf. High Perf. Comp. Comm.*, LNCS 3726, pp.817–828 (2005).
159) Yang, Y. and Casanova, H.: UMR: a multi-round algorithm for scheduling divisible workloads, *Proc. Int'l Par. Distr. Proc. Symp.* (2003).
160) Yang, Y. and Casanova, H.: RUMR: robust scheduling for divisible workloads, *Proc. Int'l Symp. High Perf. Distr. Comp.*, pp.114–125 (2003).
161) Zhang, H.: Packing: scheduling, embedding, and approximating metrics, *Proc. Int'l Conf. Comp. Sci. Appl.*, pt.3, LNCS 3045, pp.764–775 (2004).

(平成 18 年 5 月 8 日受付)
(平成 18 年 9 月 5 日採録)

須田　礼仁（正会員）

昭和 43 年生．平成 5 年東京大学大学院理学系研究科情報科学専攻修士課程修了．同年同研究科助手．平成 9 年名古屋大学大学院工学研究科講師，平成 12 年同助教授，平成 14 年東京大学大学院情報理工学系研究科助教授．数値アルゴリズム，並列計算・高性能計算の手法に関する研究に従事．博士（理学）．平成 6 年度，平成 12 年度山下記念研究賞．日本応用数理学会会員．