標準タスクグラフセットを用いた実行時間最小マルチプロセッサスケジューリングアルゴリズムの性能評価

飛田高雄[†,††] 笠原博徳[†,††]

本論文では，強 NP 困難な組合せ最適化問題である実行時間最小マルチプロセッサスケジューリング問題に対するヒューリスティックアルゴリズム，逐次および並列最適化アルゴリズムの性能評価のための“標準タスクグラフセット”と名付けたランダムタスクグラフ集を提案するとともに，それを用いたアルゴリズムの評価について述べる．従来のマルチプロセッサスケジューリングアルゴリズムの研究においては，評価に使われたランダムグラフが提案アルゴリズムに都合のよいランダムグラフである，あるいは他の研究者が検証しようとしても使われたグラフが手に入らずどのアルゴリズムが真に良いのかを比較できないという問題があった．提案する標準タスクグラフセットはこの問題を解決するため，従来の論文で使用された種々のランダムグラフ生成法を用い多種のランダムタスクグラフを生成するとともにタスクグラフとその解などの情報を Web サイトで公開しており，これを用いることで今後スケジューリングアルゴリズム研究者が各種アルゴリズムも同一条件下で公平に評価し，アルゴリズムの性能を比較することが可能となる．また本論文ではこの標準タスクグラフセットのタスク数 50 から 5,000 までのタスクグラフ 2,700 例を用いてアルゴリズムの評価を行い，標準タスクグラフセットの有効性の評価も行う．この評価ではヒューリスティックアルゴリズム CP，CP/MISF ではそれぞれ全問題の 68.22%，68.46%に，逐次最適化アルゴリズム DF/IHS では 600 秒の探索上限時間内に 85.79%，並列最適化アルゴリズム PDF/IHS では 4 プロセッサ SMP 上で 89.60%に最適解が得られることが確かめられ，提案する標準タスクグラフセットがヒューリスティックおよび逐次・並列最適化アルゴリズムの評価に有効であることが確認された．

Performance Evaluation of Minimum Execution Time Multiprocessor Scheduling Algorithms Using Standard Task Graph Set

Takao Tobita[†,††] and Hironori Kasahara[†,††]

This paper proposes a “Standard Task Graph Set” (STG) to evaluate performance of heuristic and optimization algorithms for the minimum execution time multiprocessor scheduling problem, which is known as a strong NP-hard combinational optimization problem, and describes evaluation results by applying them to several algorithms. In the previous researches on multiprocessor scheduling algorithms, there exists a problem that it is not able to compare the performance to decide which algorithm is better, because the task graphs fit for the algorithm proposed in each paper or were not available to the other researchers. To cope with this problem, STG makes possible the fair evaluation and comparison of the algorithms under the same conditions for every researchers by giving many kinds of random task graphs based on various task graph generation methods used in the literature with their scheduling results, and making them available from Website. This paper evaluates several algorithms using 2,700 task graphs with 50 to 5,000 tasks from STG and evaluates its effectiveness. The performance evaluation confirms that heuristic algorithms CP and CP/MISF could obtain optimal schedules 68.22% and 68.46% of tested cases, 85.79% by a sequential optimization algorithm DF/IHS, and 89.60% by a parallel optimization algorithm PDF/IHS on a SMP with 4 processor elements within 600 seconds upper limit. It was also confirmed that the proposed STG is useful for evaluation of the heuristic and the optimization scheduling algorithms.

1. はじめに

マルチプロセッサシステム上で効率良く並列処理を行うには，一般にきわめて難しい最適化問題である実行時間最小マルチプロセッサスケジューリング問題，すなわち処理すべきタスク集合をどのようにプロセッ

† 早稲田大学理工学部電気電子情報工学科
Department of Electrical, Electronics and Computer Engineering, School of Science and Engineering, Waseda University
†† アドバンスト並列化コンパイラ共同体
Advanced Parallelizing Compiler Project

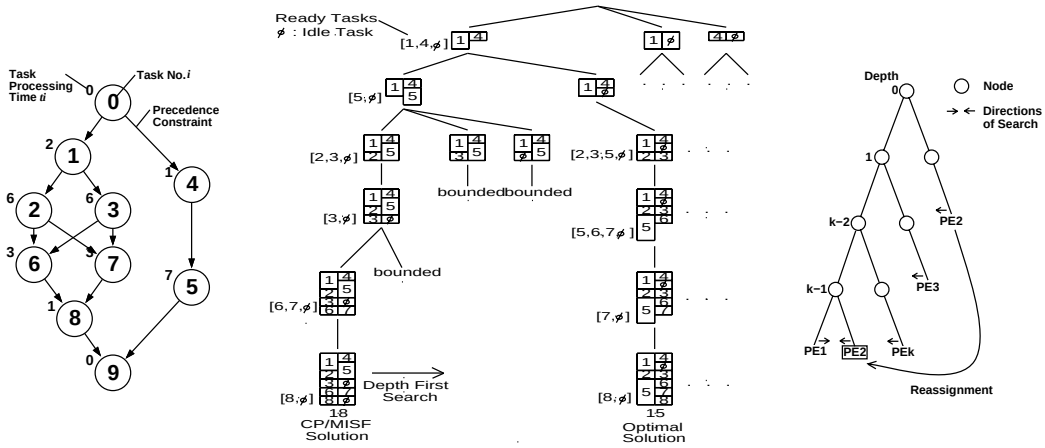
サに割り当て，どのような順序で実行すれば実行時間を最小にできるか，という問題を解かなければならない[1)~3)]．特にタスク間の先行制約が任意形状で，タスク処理時間やプロセッサ台数が任意である問題は，プロセッサ間データ転送オーバーヘッドが無視できるとした理論的な問題でさえ強 NP 困難な難しい（intractable）問題で，P ≠ NP ならば擬多項式時間最適化アルゴリズム，ならびに両完全多項式時間近似スキームの構築も不可能である[4)]ことが知られている．このためこの種の問題では，つねに短時間で最適解を得られる保証はないが多くの問題に対して最適解を短時間で得ることができる実用的（アルゴリズム用語における “practical”）なアルゴリズムが提案されている[5)~7)]．本論文ではマルチプロセッサ上での並列処理の基礎研究であるこの理論的なマルチプロセッサスケジューリング問題に対するアルゴリズムの評価手法を議論する．

実用的なアルゴリズムの例であるヒューリスティックアルゴリズムとしては，タスクグラフの出口ノードから各タスクまでの最長パス長（以下 CP 長）が長いタスクから割り当てる HLFET（Highest Levels First with Estimated Times）[8)]や CP（Critical Path）[1)]，CP 長が長いタスクを優先するが，同一の CP 長を持つタスクが複数存在すれば直接後続タスク数が多いものを優先する CP/MISF（Critical Path/Most Immediate Successors First）[2)]などが提案されている．また最適解，すなわち最小のスケジュール長を持つ解を求める実用的な最適化アルゴリズムとしては，Ramamoorthy らが DP（Dynamic Programming）を用いてプロセッサ数を 2，タスク数を約 40 と限定し最適解を得ている[9)]ほか，筆者らは CP/MISF により初期上限値を得，またそのタスク割当て優先度を用いて探索順を決定するとともに，シャープな下界[10)]により限定操作を行う深さ優先分枝限定法である逐次最適化アルゴリズム DF/IHS（Depth First/Implicit Heuristic Search）[2)]と，それを並列処理用に拡張した並列アルゴリズム PDF/IHS（Parallelized DF/IHS）を提案している[11)]．

また関連してデータ転送オーバヘッドが大きい場合のスケジューリングでは，データ転送も考慮し最も早く実行できるタスクとプロセッサの組を優先する ETF（Earliest Task First）[12)]，CP 長が長いタスクを局所的なデータ転送時間が最小となるプロセッサへ割り当てるが，最小のデータ転送時間となる候補が複数あれば直接後続タスク数の多いものを優先する CP/DT/MISF（Critical Path/Data Transfer/Most Immediate Successors First）[3)]をはじめ，MH（Mapping Heuristics）[13)]，DT/CP（Data Transfer/Critical Path）[14)]，RCP（Ready Critical Path）[15)]，DCP（Dynamic Critical Path）[16)]などのヒューリスティックが提案されているほか，最適化アルゴリズムとしては，無限個のプロセッサがあると仮定し，全タスクを 1 つずつプロセッサに配置した後，DS（Dominant Sequence）のタスクから同じプロセッサに割り当てデータ転送を除去することで，fork，join，粗粒度 tree や一部の細粒度 tree 形状のタスクグラフで最適解を得られる DSC（Dominant Sequence Clustering）[5),6)]や，やはり無限個のプロセッサを仮定し，タスクグラフの出口ノードに近い方から順に割り当て，先行タスクからのデータ転送時間よりその先行タスクの処理時間が短ければ後続タスクと同じプロセッサで先行タスクを実行するなど，タスクの複製を行って fork / join 型のタスクグラフでタスク処理時間とデータ転送時間が一定の条件[7)]を満たせば最適解を得られる TDS（Task Duplication based Scheduling）[7)]などが提案されている．

このように様々なマルチプロセッサスケジューリング問題に対するアルゴリズムが提案されてきたが，それらのアルゴリズムの性能評価においては，ランダムタスクグラフ[6),8),16),17)]やガウス消去，コレスキー分解などを簡略化した数例のタスクグラフ[5),6),16)]を用いて行うのが一般的であった．しかし一口にランダムといってもその論文で提案するアルゴリズムに有効なものであったり，少数のタスクグラフのみによる評価では性能が良くても他のタスクグラフでは有効でない場合があったりするなどの問題があった．また一般にこれらのタスクグラフは他の研究者の手に入らないため，他者が追試したり別の提案アルゴリズムとの性能比較を行うことも困難であった．

そこで本論文では，前述のような関連研究を含め，マルチプロセッサスケジューリングアルゴリズムの開発評価を行っている論文[6),8),17)]などを調査し，できるだけ公平なタスクグラフセットを作成するためそれらのタスクグラフ生成方法を取り入れた “標準タスクグラフセット” と名付けたタスクグラフ集を作成し，さらにタスクグラフだけでなくそのグラフの生成方法や性質，最適解情報も同時に公開し，これを研究者が共通で用いることで公平にアルゴリズムの性能評価・比較を行う手法を提案するとともに，この標準タスクグラフセット中のタスク数 50 から 5,000 のランダムタスクグラフ2,700 例を用いての性能評価例として，従来最良といわれているヒューリスティックアルゴリズム CP，



(a) A task graph (b) A partial search tree (c) Load distribution in PDF/IHS

図 1 タスクグラフと部分探索木，およびその PDF/IHS での割当て例
Fig. 1 A task graph, search tree and load distribution in PDF/IHS.

CP/MISF と，逐次最適化アルゴリズム DF/IHS，並列最適化アルゴリズム PDF/IHS の性能評価を行い，評価手法の有効性を検証する．

2. 対象スケジューリング問題の定義

本論文で扱うマルチプロセッサスケジューリング問題は，能力の等しい m 台のプロセッサで，任意の処理時間および任意形状の先行制約を持つ n 個のタスクからなるタスク集合 $T = \{T_1, T_2, \ldots, T_n\}$ をノンプリエンプティブ（処理割込みなし）に並列処理する際に，データ転送オーバヘッドが無視できる程度に小さいとして，その実行時間（スケジュール長）を最小にするスケジュールを求める問題である．このタスク集合 T はタスクグラフ $G(T, E)$（E は有向エッジの集合）として DAG（無サイクル有向グラフ）で記述され，このタスクグラフは 1 つの入口ノードと 1 つの出口ノードを持ち，すべてのノードは入口・出口ノードから到達できるものとする．

図 1 (a) は 8 個のタスクからなるタスクグラフの例で，図中の各ノードは 1 つのタスクを表し，ノード内の数字はタスク番号 i，ノード左上の数字は各タスクの処理時間 t_i（単位：unit time, 以下 [u.t.]）をそれぞれ表している．また図中の各アークはタスク間の先行制約を表し，ノード i からノード j へのアークは，タスク T_i がタスク T_j に先行するという半順序制約を表す．

3. マルチプロセッサスケジューリングアルゴリズム

本章では，本論文で性能評価を行うヒューリスティックアルゴリズム CP，CP/MISF と，最適化アルゴリズム DF/IHS および PDF/IHS について述べる．

3.1 ヒューリスティックアルゴリズム

2 章で述べた問題に対する代表的なヒューリスティックアルゴリズムに，CP と CP/MISF がある．CP（Critical Path）は，1 章で述べたように，CP 長が長いタスクから順に優先度リストを作成し，その優先度の高いタスクから順に空きプロセッサに割り当てるリストスケジューリングアルゴリズムである[1)]．また CP では同一 CP 長を持つタスクが複数ある場合にタスクの優先順位を一意に決定できないため，優先度リスト作成時に同一 CP 長を持つタスクが複数ある場合には直接後続タスク数の多いタスクを優先するものが CP/MISF（Critical Path/Most Immediate Successors First）である[2)]．

3.2 逐次最適化アルゴリズム DF/IHS

2 章で述べた問題に対する逐次最適化アルゴリズム DF/IHS（Depth First/Implicit Heuristic Search）[2)] は，主に前処理部と分枝限定法による探索部から構成される．前処理部では CP/MISF の割当て優先度の高い順に $O(n^2)$ の計算量でタスク番号を付け替えるだけで探索木の左側に CP/MISF 的に良い解を集めることができ，探索部では単に DF/FIFO（Depth First/First In First Out）方式で探索木の左側から探索を行えば自然にヒューリスティック的に良い解から探索を行うことができる．この結果，初期解すなわち初期探索上限値が現在最良のヒューリスティック CP/MISF 解となるため，効果的に枝刈りを行え探索時間を大幅に削減できる．

図 1 (b) は，図 1 (a) のタスク集合を 2 プロセッサに割り当てる際の DF/IHS における部分探索木を示している．図中各ノード部分に示されたガントチャートはその時点までに割り当てられたスケジュールを表し，左側がプロセッサ 1，右側がプロセッサ 2 のスケジュールである．また，[ ] 内の数字はその時点でのレディタスク集合を表している．ただし，本論文では各タスクの処理時間 t_i が異なることを仮定しているため，レディタスクがある限りプロセッサにタスクを割り当てるリストスケジューリングでは最適解を得られない可能性がある[9]．したがって，分枝時にプロセッサを強制的にアイドルにするようなタスク（図 1 (b) 中 ϕ で表示）も含めてノードを生成する．

DF/IHS では前述のように初期解が CP/MISF 解（図 1 (b) 中左端）となるため，この精度の良い上限値により部分探索木の早期限定が行える．限定操作では下界の計算量を考慮して，まず単純な実行時間の下界，すなわち分枝により分解された部分問題 π_a に対し

$$t_{cr}(\pi_a) = \max_{i \in I(\pi_a)} l_i + t_0$$

あるいは

$$t_{div}(\pi_a) = \left\lceil \sum_{i \in I(\pi_a)} t_i / m \right\rceil + t_0$$

で限定できなかった場合に限り Fernández によって拡張された Hu の下界[10]

$$t_{hu}(\pi_a) = t_{cr}(\pi_a) + \lceil q(\pi_a) \rceil$$
$$q(\pi_a) = \max_{0 \le t_k \le t_{cr}(\pi_a) - t_0} \left(-t_k + \frac{1}{m} \int_0^{t_k} F(\bar{\tau}, t) dt \right)$$

を用いて限定操作を行う（図 1 (b) 中 bounded と表記）．ただし，ここで

$I(\pi_a)$：π_a 内の未割当てタスク番号の集合
l_i：出口ノードからタスク i までの最長パス長
t_0：考慮中のノードが表す割当て時刻
m：問題で対象とするプロセッサ台数
$\lceil x \rceil$：x より大きい最小の整数

であり，負荷密度関数 $F(\bar{\tau}, t)$ は

$$\bar{\tau}_j = t_{cr}(\pi_a) - l_j - t_0$$
$$f(\bar{\tau}_j, t) \begin{cases} = 1, & \text{for } t \in [\bar{\tau}_j, \bar{\tau}_j + t_j] \\ = 0, & \text{otherwise} \end{cases}$$
$$F(\bar{\tau}, t) = \sum_{j \in I(\pi_a)} f(\bar{\tau}_j, t)$$

である．

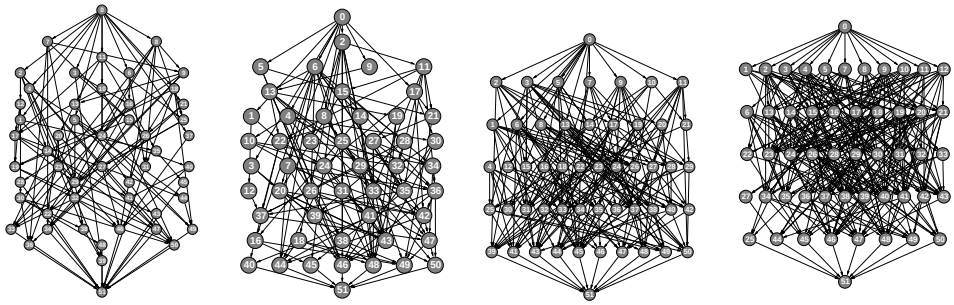
また DF/IHS は，SP（Selection Pointer）値と呼ばれるポインタを用い，バックトラック時にレディタスクテーブル（アクティブノード情報）を再生できるため，すべてのアクティブノードを生成・記憶しておく必要がなく，領域複雑度を $O(n^2 + mn)$ に抑えられる[2]．

3.3 並列最適化アルゴリズム PDF/IHS

DF/IHS の探索部を並列化した並列最適化アルゴリズムが PDF/IHS（Parallelized DF/IHS）である．PDF/IHS で k 台の PE（Processor Element）を用いて並列探索を行う場合，DF/IHS と同様の前処理後，PE_1（リーダ PE）は DF/IHS と同様に探索木の左から右に深さ優先探索を行い，他の PE_j $(j = 2, \ldots, k)$（スレーブ PE）はリーダ PE の探索経路上のノードを根ノードとする部分探索木を右から左へ探索する．この左右からの階層的深さ優先挟み撃ち探索により，スレーブ PE は割当て領域の探索を終了するかリーダ PE と出会うまでは独立に探索でき，かつ PE 間の負荷均等化が図れる．またリーダ PE が DF/IHS と同様の探索を行うため，減速異常は生じない[18]．なお，PDF/IHS でも探索木左端の CP/MISF 解が精度の良い探索上限値となるため，初期解が下界と一致しないことを確認した後にスレーブ PE が並列探索を開始する．

図 1 (c) は k 台の PE を使用した PDF/IHS の各 PE への探索領域割当て例である．各 PE への探索領域の割当ては，リーダ PE_1 が割当て待ち状態のスレーブ PE_j $(2 \le j \le k)$ に，PE_1 の探索経路上の深さの浅いノードを根とする部分探索木から順に割り当てる．この割当て領域の指定はその根ノードの深さをスレーブ PE に転送するだけで行えるため，探索領域割当てオーバヘッドを小さく抑えられる．

またこの階層的挟み撃ち探索では，同一の部分探索木を PE_1 と PE_j $(2 \le j \le k)$ が互いに反対方向に独自に深さ優先探索していくが，この PE どうしの出会いは PE_1 がブロードキャストする SP 値を用いて各スレーブ PE が独自に検出できる．出会いを検出したスレーブ PE は割当て待ち状態となり，まだ探索すべき部分木があれば再度探索領域が割り当てられ，PE 間の負荷均等化が可能である．図 1 (c) 右下の矢印は，PE_2 が最初の割当て領域の探索終了後，未割当て領域のうち最も浅いノード（深さ $k-1$）を根とする部分領域を再割当てされた例を示している．この並列探索は，探索すべき領域がなくなるか，どれかの PE が見つけた解が下界と一致する（すなわちそれが最適解であると判断できる）まで行われる．



(a) sameprob (b) samepred (c) layrprob (d) layrpred

図 2 タスクグラフ形状の例
Fig. 2 Examples of task graph shape.

4. 標準タスクグラフセット

従来，強 NP 困難な最適化問題である本スケジューリング問題に対するアルゴリズムの性能評価では，標準的に用いられているベンチマークや問題集などは存在しないため，評価者それぞれが作成したランダムタスクグラフ[6),8),16),17)]やガウス消去，コレスキー分解のようなアプリケーションを簡略化した単純なタスクグラフ[5),6),16)]が用いられてきた．しかし，一口にランダムといってもその論文で評価するアルゴリズムに都合の良いものであったり，単純化されたアプリケーションからのタスクグラフでは性能が良いとされていても他のタスクグラフでは有効でない場合があるなどの問題があった．また論文で使われたタスクグラフは他の研究者には手に入らず，追試あるいは自分自身のアルゴリズムと性能を比較しようとしても不可能であるという問題もあった．そこで本論文では，過去のマルチプロセッサスケジューリングアルゴリズムの論文[6),8),17)]などを調査し，できるだけ公平なタスクグラフセットを作成するためそれらのタスクグラフ生成方法を網羅的に取り入れた “標準タスクグラフセット” を作成し，タスクグラフだけでなくその生成方法や性質なども同時に公開することにより，研究者共通の一種のベンチマークとして用いることで公平に性能評価と比較を可能とすることを目指す．

本論文ではこの標準タスクグラフセットのうち，以下のように生成したランダムタスクグラフを用いる．まず，問題規模の変化が性能に与える影響を知るため，タスク数は 50，100，300，500，750，1000，1250，1500，1750，2000，2500，3000，3500，4000，5000 の 15 種類を用意している．なお，従来世界的に報告されている最適化アルゴリズムの性能評価の例では，DP を用いてプロセッサ数を 2 と制限してもタスク数 40 程度までしか最適解が得られていない[9)]こともあり，現時点では上記の範囲のタスクグラフを用意しているが，今後計算機能力などの進歩によりさらに大規模な問題が必要になれば，随時拡充していく予定である．

タスクグラフの形状（エッジの接続方法）は，以下の 4 種類を用意した．なお，以下ではタスク番号はタスクグラフの各ノードをトポロジカルソートした順序に従うものとし，タスク接続行列 A の各要素 $a(i,j)$（ただし $0 \le i \le n+1$，$0 \le j \le n+1$，n はタスク数）は，$a(i,j)=1$ のときタスク T_i はタスク T_j に先行する，すなわち T_i は T_j の開始前に実行が終了している必要があることを示し，$a(i,j)=0$ ならば T_i から T_j へのエッジは存在しないものとする．

第 1 の形状決定方法 “*sameprob*” は，以下のような確率 P で $a(i,j)$ の値を決定する方法である[17)]．

$$P[a(i,j)=1]=\rho \quad \text{for} \quad 1 \le i < j \le n$$
$$P[a(i,j)=0]=1-\rho \quad \text{for} \quad 1 \le i < j \le n$$
$$P[a(i,j)=0]=1 \quad \text{if} \quad i \ge j$$

ここで，ρ は T_i と T_j の間にエッジが存在する確率を表す．**図 2** (a) は，*sameprob*でタスク数 50，$\rho=0.1$ として生成したタスクグラフの例である．

第 2 の方法として，標準タスクグラフセットでは，各タスクの平均直接先行タスク数 pre を指定した “*samepred*” 法も採用しており，この方法では

$$\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} a(i,j) = pre \cdot n$$

となるようにエッジを生成する．図 2 (b) は *samepred* でタスク数 50，$pre=3$ として生成したタスクグラフの例である．

第 3 の形状決定方法としては，まずタスクグラフ中に作成する層（レイヤ）数を決定し，次に各層に独立なタスクをランダムに配置し，最後に *sameprob*のよ

うに等確率 ρ で層間にエッジを生成する “*layrprob*” を用いた[6]．ただし，この方式の確率 ρ は，同一層内のタスクへは依存が存在しない（独立なタスクが同一層内に配置される）ように，$L(i)$，$L(j)$ をそれぞれ T_i，T_j の配置された層番号とすると，

$$P[a(i,j)=1]=\rho \quad \text{for} \quad (1\le i<j\le n)\wedge(L(i)\neq L(j))$$
$$P[a(i,j)=0]=1-\rho \quad \text{for} \quad (1\le i<j\le n)\wedge(L(i)\neq L(j))$$
$$P[a(i,j)=0]=1 \quad \text{if} \quad (i\ge j)\vee(L(i)=L(j))$$

とする．図 2 (c) は *layrprob*でタスク数 50，層数 5，$\rho=0.2$ として生成したタスクグラフの例である．本論文では各層の幅（層内のタスク数）の平均値が 10 となるように層数を $n/10$ とし，実際の層内のタスク数は各タスクが何番目の層に配置されるかを 1 から $n/10$ の一様乱数で決めることで決定した．

最後に第 4 の方法として，*layrprob*と同様に層を作成し，*samepred*と同様に平均先行タスク数を指定してエッジを作成する “*layrpred*” と名付けた方法も用いた．図 2 (d) は *layrpred*でタスク数 50，層数 5，$pre=5$ として生成したタスクグラフの例である．

なお本論文では，これらの 4 種類の形状それぞれについて，各パラメータの値をそれぞれ *sameprob*および *layrprob*では $0.05\le\rho\le0.2$，*samepred*および *layrpred*では $1\le pre\le20$ の範囲内の一様乱数で決定している．

また各タスクの処理時間（processing time）は，一様乱数[8]，指数乱数[17]，正規乱数の 3 種類の乱数を用い，以下の 9 種類を用意した．すなわち，$[0,10)$ の範囲（0 以上 10 未満）の一様乱数を生成し，小数点以下を切り捨てて 1 を加えた 1〜10 [u.t.] の範囲の整数（以下 *unifproc* (A) と表記），同様にして生成した 1〜20 [u.t.] の整数（同 *unifproc* (B)），および *unifproc* (A)，(B) から一様乱数で確率 1/2 で選び混合したもの（*unifproc* (C)）と，平均値 5.0 の指数乱数を生成し，小数点以下を切り捨てて 1 を加えた整数（*expoproc* (A)），同様に平均値を 10.0 としたもの（*expoproc* (B)），およびそれらの確率 1/2 での混合（*expoproc* (C)），ならびに平均値 5.0，標準偏差 1 の正規乱数を生成し，小数点以下を切り捨てて 1 を加えた整数（*normproc* (A)），同様に平均値 10.0，標準偏差 3 としたもの（*normproc* (B)）とそれらの確率 1/2 での混合（*normproc* (C)）である．

以上の条件を組み合わせると，15 通りのタスク数，4 種類のタスクグラフ形状および 9 通りのタスク処理時間決定方法を用い，合計 540 通りの条件が生成できる．本論文では，この 540 通りの条件それぞれに 5 例ずつ，合計 2,700 例のタスクグラフを生成し，3 章で述べたアルゴリズムに適用して性能を評価する．

5. 標準タスクグラフセットを用いた性能評価

本章では，4 章で述べた標準タスクグラフセットを用いて CP，CP/MISF および DF/IHS，PDF/IHS の性能評価を行った結果について述べる．なお，以下では用語の混乱を避けるため，スケジューリング問題の中で使われるプロセッサを単に “プロセッサ” と表記し，PDF/IHS の並列探索に使用される主記憶共有型マルチプロセッサシステム Sun Ultra80 Model 4450（以下 Ultra80）のプロセッサを “PE” と表記する．

5.1 性能評価条件

本論文では Ultra80 上で 1PE だけを用いてヒューリスティックアルゴリズム CP，CP/MISF を標準タスクグラフセット中のランダムタスクグラフ 2,700 例に適用し，2，4，8，16 プロセッサで並列処理する際のスケジュール（合計 10,800 例）を求めた．ここで使用した Ultra80 は，450 MHz UltraSPARC IIs（一次キャッシュはデータ/命令ともに 16 KB，二次キャッシュ 1 MB）を 4 台，メインメモリ 1 GB を搭載した主記憶共有型マルチプロセッサシステムである．また最適化アルゴリズム DF/IHS（Ultra80 の PE を 1 台使用），PDF/IHS（PE を 4 台使用）を同じタスクグラフ 2,700 例に適用し，2，4，8，16 プロセッサで並列処理する際の最適（最短）スケジュールを求めた．ただし，DF/IHS および PDF/IHS では必ず最適解が求まるが，扱う問題が強 NP 困難で，PDF/IHS を使用しても最適解を得るのに数日から数カ月かかる可能性もあるので，探索上限時間を wall-clock time（プロセス起動後の経過時間）で 600 秒とし，その間の最適解求解数の推移についても調べた．

5.2 性能評価結果

表 1 に各アルゴリズムの最適解求解率と解の精度を示す．表 1 のように，CP は全問題の 68.22%，CP/MISF は 68.46%に最適解を得たが，600 秒以内に DF/IHS は 85.79%，4PE での PDF/IHS は 89.60%に最適解が得られている．ただし表 1 中では，下界または DF/IHS，PDF/IHS により得られた最適解と一致する解が得られた場合を “最適解が求まった” としており，探索上限時間内に最適解であると確認できていなくても得られている解が最適解である可能性があるため，各アルゴリズムの真の最適解求解率は表 1 より高い可能性がある．また表 1 のように，得られた解と下界との誤差率の平均値は CP，CP/MISF で

表 1 各アルゴリズムの最適解求解率と解の精度
Table 1 Optimal schedule ratio and precision of the results.

Algorithm	CP	CP/MISF	DF/IHS	PDF/IHS
最適解求解数	7368	7394	9265	9677
最適解求解率 [%]	68.22	68.46	85.79	89.60
下界との差の平均値 [u.t.]	3.82	3.82	3.35	3.31
下界との誤差率の平均 [%]	0.18	0.18	0.12	0.11

表 2 プロセッサ台数別最適解求解率 [%]
Table 2 Rate of optimal schedule for each number of processors.

プロセッサ台数	問題数	CP	CP/MISF	DF/IHS	PDF/IHS
2	2700	72.37	73.33	95.85	97.25
4	2700	52.88	52.96	80.62	85.33
8	2700	61.74	61.77	78.66	86.00
16	2700	85.88	85.77	88.00	89.81
Total	10800	7368	7394	9265	9677
%	100.00	68.22	68.46	85.79	89.60

0.18%，DF/IHS および PDF/IHS でそれぞれ 0.12%，0.11%となっており，これらのアルゴリズムの実用性が高く，有効であることが確認できる．

またスケジューリング問題の割当てプロセッサ台数別の最適解求解率を**表 2** に示す．表 2 のように，プロセッサ台数が 2〜8 ではヒューリスティックアルゴリズム CP，CP/MISF に対して最適化アルゴリズム DF/IHS，PDF/IHS の 10 分間での最適解求解率が大きく向上しているが，プロセッサ数が 16 の場合はヒューリスティックアルゴリズムでも 85%以上の問題に最適解が得られているため，DF/IHS，PDF/IHS を用いてもヒューリスティックより大幅に性能を向上するのが困難であることが分かる．また逆に，プロセッサ数が 2 の場合は PDF/IHS で 97.25%の問題に最適解が得られているなど，問題の並列度に比べプロセッサ数が少ない場合は DF/IHS，PDF/IHS による最適解求解率向上が著しい．この原因としては，プロセッサ数が少ないほどレディタスクのプロセッサへの割当ての組合せの数が少なく，DF/IHS および PDF/IHS の探索領域が狭くなることが考えられる．また，表 2 のように，CP，CP/MISF ヒューリスティックアルゴリズムではプロセッサ数が 2，16 の場合に比べて 4，8 の場合の最適解求解率が低下しているが，この理由については後で考察する．

次に，標準タスクグラフセットにおいては，各タスクグラフの生成方法やパラメータなどの情報が同時に公開されているため，各タスクグラフ生成パラメータの違いがアルゴリズムの性能に与える影響を調べることが可能である．以下ではこの生成方法およびパラメータの違いによる性能の差について考察する．

まず**表 3** のタスク数別最適解求解数から分かるように，今回用いたタスクグラフでは，タスク数（問題規模）と最適解求解率の間には特に強い相関関係は認められず，従来解かれたことがなかったタスク数数千の超大規模問題の多くに対しても，DF/IHS，PDF/IHS は短時間で最適解を得られることが確認された．この世界最高レベルであるタスク数数千の大規模問題に対し，DF/IHS，PDF/IHS が 600 秒以内に最適解を与えることは，計算複雑度が C^n（C は定数，n はタスク数）といわれていることを考慮すると，これらのアルゴリズムの性能がいかに高いかを示している．

また，割当てプロセッサ台数 2，4，8，16 のそれぞれの場合のタスクグラフ形状と最適解求解率の関係を**図 3** (a)〜(d) にそれぞれ示す．図 3 のように，プロセッサ台数が 2，4 の場合は *sameprob*，8 の場合は *layrprob*形状タスクグラフの最適解求解率が低いが，プロセッサ台数が 16 の場合はどのアルゴリズムでも *sameprob*や *layrprob*のタスクグラフのほぼすべてに最適解が得られているなど，プロセッサ台数ごとに異なる傾向を示している．

この理由を考察するにあたり，生成パラメータ以外のタスクグラフの性質を表す指標として，タスクグラフの並列度 $para$ を

$$para = \frac{\text{タスク処理時間の総和}}{\text{タスクグラフの } CP\text{長}}$$

と定義する．この $para$ は，そのタスクグラフを無限個のプロセッサで並列処理するとどの程度のプロセッサが必要か，すなわちそのタスクグラフが持つ並列度

表 3 タスク数別最適解求解数
Table 3 Rate of optimal schedule for each graph size.

Tasks	問題数	CP	CP/MISF	DF/IHS	PDF/IHS
50	720	537	542	689	693
100	720	524	516	679	690
300	720	477	498	657	682
500	720	488	473	655	675
750	720	479	491	622	650
1000	720	476	472	606	634
1250	720	478	482	589	630
1500	720	490	483	602	636
1750	720	478	480	609	635
2000	720	486	472	583	626
2500	720	488	490	596	628
3000	720	487	497	596	630
3500	720	490	499	596	625
4000	720	495	496	590	623
5000	720	495	503	596	620
Total	10800	7368	7394	9265	9677
%	100.00	68.22	68.46	85.79	89.60

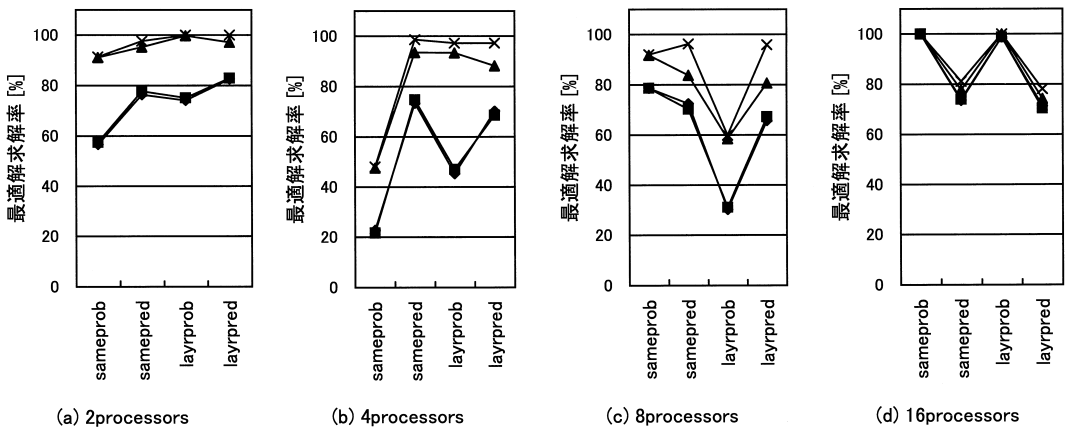


図 3 タスクグラフ形状別最適解求解率 [%]
Fig. 3 Rate of optimal schedule for each graph shape.

を直感的に表すことができる．本論文で用いたランダムタスクグラフ 2,700 例の *para* の最大値は 441.8 で，全体の約 2/3 の 1,814 例が $1.5 \leq para < 20.5$ の範囲にあった．以下では特にこの $1.5 \leq para < 20.5$ の範囲内の問題について考えてみる．

まず，CP/MISF と PDF/IHS のタスクグラフ形状別最適解求解率を**図 4**，**図 5** にそれぞれ示す．なお以下の図においては $1.5 \leq para < 2.5$ のものを $para = 2$，$2.5 \leq para < 3.5$ のものを $para = 3$，$\cdots$ と表すものとし，図 4，図 5 はそれらの範囲内にある問題ごとの最適解求解率を表している．図 4，図 5 (a)〜(d) より，特に PDF/IHS では割当てプロセッサ台数とタスクグラフの *para* が近い場合に最適解求解率が落ちている，すなわち，プロセッサ数と *para* が近い問題については PDF/IHS を用いても最適解を得るのに非常に長時間を要する個別問題が存在することが分かる．なお図 4 と図 5 を比較すると，*para* がプロセッサ台数と近くない問題のほとんどについては，CP/MISF アルゴリズムで最適解が求まらなかった場合でも PDF/IHS アルゴリズムを用いればほぼ最適解を得られていることが分かる．

また**図 6** は，各形状で生成したタスクグラフの *para* ごとの問題数の分布である．図 6 のように，*samepred*，*layrpred*の問題の並列度は $1.5 \leq para < 20.5$ の範囲全体にほぼ均等に分布しているが，*sameprob*では $2.5 \leq para < 10.5$，*layrprob*では $5.5 \leq para < 15.5$ の問題が多くなっている．このため，*para* が 2 または 4 となる問題は *sameprob*に，*para* が 8 となる問題は

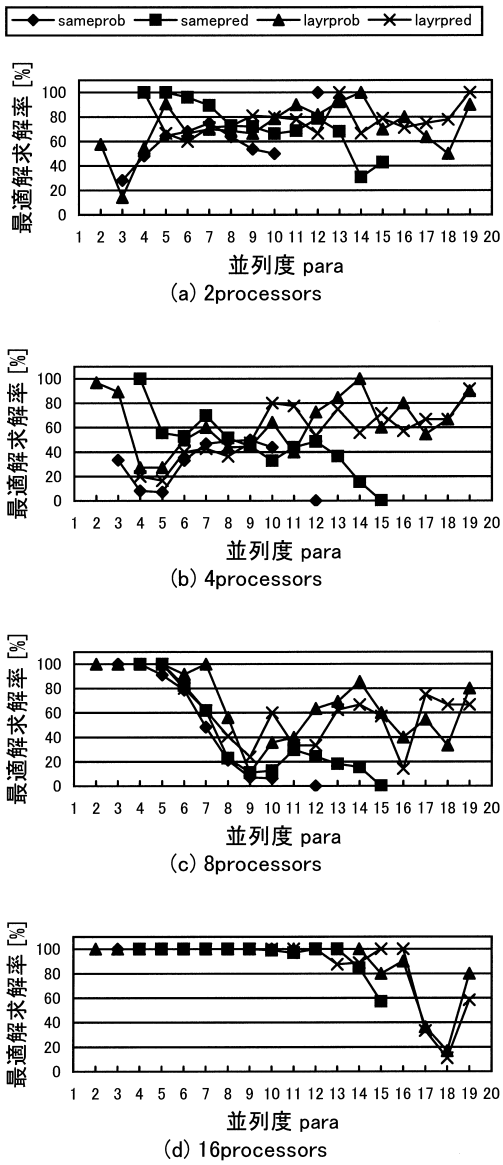


図 4 CP/MISF アルゴリズムのタスクグラフ形状別最適解求解率 [%]
Fig. 4 Rate of optimal schedule of CP/MISF for each graph shape.

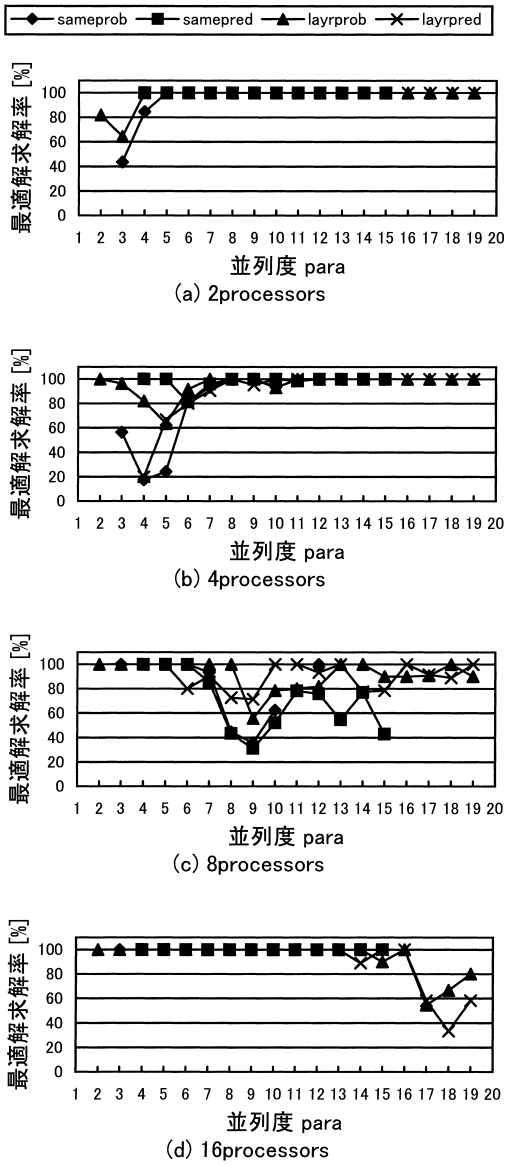


図 5 PDF/IHS アルゴリズムのタスクグラフ形状別最適解求解率 [%]
Fig. 5 Rate of optimal schedule of PDF/IHS for each graph shape.

*layrprob*に，そして *para* が 16 となるのはそれら以外の *samepred*，*layrpred*にそれぞれ多くなるため，図 3 のように，プロセッサ台数に近いタスクグラフを生成する割合の高い形状のタスクグラフの最適解求解率が低下したものと考えられる．さらに，図 4，図 5 のように同一の並列度 *para* を持つタスクグラフごとの最適解求解率を比較すると，タスクグラフ形状ごとの最適解求解率の差はあまり大きくないことから，形状によって最適解求解率が左右されるのではなく，タスクグラフの持つ並列度のみに影響されると考えられる．

表 4 に示すタスク処理時間ごとの最適解求解率について考えると，特にヒューリスティックアルゴリズムで最適解求解率が低い問題は *normproc*のものに集中している．これは，*normproc*のタスク処理時間は，他の乱数を用いたタスク処理時間決定手法に比べて平均値近辺の値をとることが多く分散が小さいため，CP 長が等しいレディタスクが多く，CP，CP/MISF ヒューリスティックでは適切なタスク選択が行えない

こと，また各タスク処理時間生成方法の（A），（B），（C）では 2 種類の平均値を持つ乱数値の混合である（C）の最適解求解率が最も高く，次いで平均値が 5 である（A），そして平均値が 10 である（B）の最適解求解率が最も低いことから，ガントチャート中に生じるタスク処理時間に比較して小さいプロセッサの空き時間を埋められるようなタスクが少なく，3.2 節で述べた下界が有効でないことなどが考えられる．

最後に，本論文では問題の強 NP 困難性から最適化アルゴリズムの探索上限時間を 600 秒と制限したが，この探索上限時間内での最適解求解率の推移について検討する．

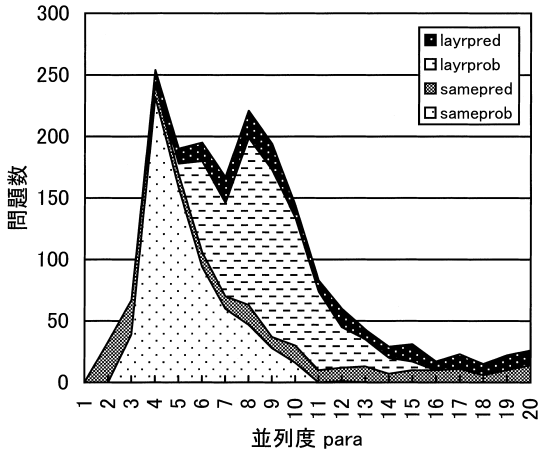


図 6 $1.5 \leq para < 20.5$ の各形状別問題数の分布
Fig. 6 Distribution of number of taskgraphs for each graph shape (where $1.5 \leq para < 20.5$).

表 5 に最適化アルゴリズムの最適解求解率の推移を示す．表 5 のように，DF/IHS，PDF/IHS はタスクグラフファイル入力後初期解の求解も含めて 10 秒以内に，それぞれ 71.57%，74.70%の問題に最適解を得ているほか，60 秒までにそれぞれ最適解求解率が約 10 ポイントずつ向上し，600 秒までにさらに約 5 ポイントずつ向上している．このことから，およそ 1 分程度の探索でも 8 割以上の問題に最適解が得られていることが分かる．

なお，公開されている標準タスクグラフセットを用いてアルゴリズムの性能評価を行うことにより，使用したタスクグラフそのものが公開されているために再現性を確保できるだけでなく，本論文の性能評価結果のように，種々のタスクグラフ生成手法から多数のランダムタスクグラフを生成することで各生成手法ごとにアルゴリズムの性能への影響を調べられるほか，タスクグラフの性質も含めて公開されていることから最適解が得られにくい問題の性質を調べることも可能となるなど，標準タスクグラフセットがこれらのヒューリスティックおよび最適化アルゴリズムの性能評価に有用であることが確かめられた．

6. ま と め

本論文では，強 NP 困難な最適化問題である実行時間最小マルチプロセッサスケジューリング問題に対するアルゴリズムの性能評価において，“標準タスクグラフセット” を用いて評価を行う手法を提案するとともに，それを用いてヒューリスティックアルゴリズム CP，

表 4 タスク処理時間別最適解求解数
Table 4 Rate of optimal schedule for each task processing time.

処理時間	問題数	CP	CP/MISF	DF/IHS	PDF/IHS
unifproc（A）	1800	993	999	1107	1112
unifproc（B）	1800	856	850	1072	1083
unifproc（C）	1800	940	931	1101	1103
expoproc（A）	1800	1058	1058	1122	1122
expoproc（B）	1800	985	985	1100	1100
expoproc（C）	1800	1024	1025	1098	1099
normproc（A）	1800	520	524	851	1018
normproc（B）	1800	462	462	917	1010
normproc（C）	1800	530	560	897	1030
Total	10800	7368	7394	9265	9677
%	100.00	68.22	68.46	85.79	89.60

表 5 最適化アルゴリズムの最適解求解率 [%] の推移
Table 5 A change of optimal schedule ratio of optimization algorithms [%].

Time[s]	10	30	60	120	300	600
DF/IHS	71.57	77.68	81.22	83.70	85.50	85.78
PDF/IHS	74.70	81.26	84.88	87.37	89.27	89.57

CP/MISFと，逐次最適化アルゴリズムDF/IHS，並列最適化アルゴリズムPDF/IHSの性能を評価した結果について述べた．提案した標準タスクグラフセットのうちタスク数50から5,000までの従来世界で試みられたことのない超大規模問題を含む2,700例を用いての性能評価を行った結果，CP，CP/MISFの各ヒューリスティックアルゴリズムでもそれぞれ68.22%，68.46%の問題に最適解を得られたほか，Ultra80上で600秒の上限時間内に1PEを用いたDF/IHSが85.79%，4PEを用いた場合のPDF/IHSが89.60%の問題に最適解を与えることが確かめられた．また最適解を得られなかった場合でも全問題に平均誤差0.11%以下の解を与えるなど，これらのアルゴリズムが強NP困難な最適化問題に対して非常に強力なものであることも確かめられた．標準タスクグラフセットは，従来のスケジューリングアルゴリズムに関する論文などで用いられたランダムタスクグラフ生成手法を網羅的に用いて作成することで特定のアルゴリズムが有利になることを排除し，評価に使うタスクグラフを公開することで従来困難であった任意の研究者による同一問題を用いた性能評価・比較を可能としているだけでなく，タスクグラフとともに公開されているタスクグラフ生成方法やパラメータなどの情報から最適解が得られにくい問題の性質を調べることも可能である．このため本論文で評価したスケジューリングアルゴリズムについては，問題規模（タスク数）やタスク間のエッジ接続方法には性能はあまり左右されないが，正規分布に近いタスク処理時間の分布を持つ問題，タスクグラフの並列度と割当てプロセッサ台数が近い場合は，他の場合と比較してこれらのアルゴリズムで最適解を得るのが困難であることが確認されるなど，標準タスクグラフセットがこれらのヒューリスティックおよび最適化アルゴリズムの性能評価に有用であることが確かめられた．

なお，本論文で提案した標準タスクグラフセットおよび得られた最適解の情報は，Webサイト
http://www.kasahara.elec.waseda.ac.jp/schedule/
にて公開されており，今後このタスクグラフを用いることにより，多くの研究者が同一条件の下でマルチプロセッサスケジューリングアルゴリズムのより公平な評価が可能になり，マルチプロセッサスケジューリングアルゴリズムの研究推進に寄与できると考えている．

今後の課題としては，世界の標準タスクグラフセットユーザから寄せられている情報なども含め，実アプリケーションタスクグラフ，データ転送オーバヘッドを考慮したタスクグラフなどの追加による標準タスクグラフセットのいっそうの充実や，それらを用いたスケジューリングアルゴリズムの評価などがあげられる．

謝辞 本研究の一部は経済産業省（NEDO）ミレニアムプロジェクトIT21アドバンスト並列化コンパイラ，ならびに早稲田大学特定課題研究助成費2000A–168の一環として行われた．

参考文献

1) Coffman, E.G.: *Computer and Job-shop Scheduling Theory*, John Wiley & Sons (1976).
2) Kasahara, H. and Narita, S.: Practical Multiprocessor Scheduling Algorithms for Efficient Parallel Processing, *IEEE Trans. Comput.*, Vol.C-33, No.11, pp.1023–1029 (1984).
3) Kasahara, H., Honda, H. and Narita, S.: Parallel Processing of Near Fine Grain Tasks Using Static Scheduling on OSCAR, *Proc. IEEE ACM Supercomputing '90*, pp.856–864 (1990).
4) Garey, M.R. and Johnson, D.S.: *Computers and Intractability; A Guide to the Theory of NP-Completeness*, San Francisco, CA: Freeman (1979).
5) Yang, T. and Gerasoulis, A.: A Fast Static Scheduling Algolithm for DAGs on an Unbounded Number of Processors, *Proc. Supercomputing '91*, pp.633–642 (1991).
6) Yang, T. and Gerasoulis, A.: DSC: Scheduling Parallel Tasks on an Unbounded Number of Processors, *IEEE Trans. Parallel and Distrivuted Systems*, Vol.5, No.9, pp.951–967 (1994).
7) Darbha, S. and Agrawal, D.P.: Optimal Scheduling Algorithm for Distributed-Memory Machines, *IEEE Trans. Parallel and Distributed Systems*, Vol.9, No.1, pp.87–95 (1998).
8) Adam, T.L., Chandy, K.M. and Dickson, J.R.: A Comparison of List Schedules for Parallel Processing Systems, *Comm. ACM*, Vol.17, No.12, pp.685–690 (1974).
9) Ramamoorthy, C.V., Chandy, K.M. and Gonzalez, M.J.: Optimal Scheduling Strategies in a Multiprocessor System, *IEEE Trans. Comput.*, Vol.C-21, No.2, pp.137–146 (1972).
10) Fernández, E.B. and Bussell, B.: Bounds on the Number of Processors and Time for Multiprocessor Optimal Schedules, *IEEE Trans. Comput.*, Vol.C-22, No.8, pp.745–751 (1973).
11) 笠原博徳，伊藤　敦，田中久充，伊藤敬介：実行時間最小マルチプロセッサスケジューリング問題に対する並列最適化アルゴリズム，信学論，Vol.J74-D-I, No.11, pp.755–764 (1991).
12) Hwang, J.J., Chow, Y.C., Anger, F.D. and Lee, C.Y.: Scheduling Precedence Graphs in Systems with Interprocessor Communication

Times, *SIAM J. Comput.*, Vol.18, No.2, pp.244–257 (1989).
13) El-Rewini, H. and Lewis, T.G.: Scheduling Parallel Program Tasks onto Arbitrary Target Machines, *J. Parallel and Distributed Computing*, Vol.9, No.2, pp.138–153 (1990).
14) 藤原和典，白鳥健介，鈴木 真，笠原博徳：データプレロードおよびポストストアを考慮したマルチプロセッサスケジューリングアルゴリズム，信学論，Vol.J75-D-I, No.8, pp.495–503 (1992).
15) Yang, T. and Gerasoulis, A.: List scheduling with and without communication delays, *Parallel Computing*, Vol.19, No.12, pp.1321–1344 (1993).
16) Kwok, Y.K. and Ahmad, I.: Dynamic Critical-Path Scheduling: An Effective Technique for Allocating Task Graphs to Multiprocessors, *IEEE Trans. Parallel and Distributed Systems*, Vol.7, No.5, pp.506–521 (1996).
17) Almeida, V.A.F., Vasconcelos, I.M.M., Árabe, J.N.C. and Menascé, D.A.: Using Random Task Graphs to Investigate the Potential Benefits of Heterogeneity in Parallel Systems, *Proc. Supercomputing '92*, pp.683–691 (1992).
18) Li, G.J. and Wah, B.W.: Coping with anomalies in parallel branch-and-bound algorithms, *IEEE Trans. Comput.*, Vol.C-35, No.6, pp.568–573 (1986).

(平成 13 年 9 月 10 日受付)
(平成 14 年 2 月 13 日採録)

飛田 高雄（正会員）

昭和 48 年生．平成 8 年早稲田大学理工学部電気工学科卒業．平成 10 年同大学大学院理工学研究科電気工学専攻修士課程修了．平成 11 年同大学理工学部助手．平成 13 年同大学大学院理工学研究科電気工学専攻博士後期課程単位取得退学．平成 14 年早稲田大学理工学総合研究センター助手，現在に至る．マルチプロセッサスケジューリングアルゴリズムに関する研究に従事．

笠原 博徳（正会員）

昭和 32 年生．昭和 55 年早稲田大学理工学部電気工学科卒業．昭和 60 年同大学大学院博士課程修了．工学博士．昭和 58 年同大学同学部助手．昭和 60 年学術振興会特別研究員．昭和 61 年早稲田大学理工学部電気工学科専任講師．昭和 63 年同助教授．平成 9 年同大学理工学部電気電子情報工学科教授，現在に至る．平成元年〜2 年イリノイ大学 Center for Supercomputing Research & Development 客員研究員．昭和 62 年 IFAC World Congress 第 1 回 Young Author Prize．平成 9 年度情報処理学会坂井記念特別賞受賞．著書「並列処理技術」（コロナ社）．電子情報通信学会，IEEE 等の会員．