非均質並列プロセッサ用プログラムの実行時間の下界

李 鼎超[†] 有田隆也[†]
石井直宏[†] 曽和将容[††]

プログラムを均質並列プロセッサで実行する場合，それに要する実行時間の下界として Fernandez の下界が知られているが，この下界は非均質並列プロセッサでプログラムを実行する際に必要な実行時間の下界として用いるのには適していない．本論文では，非均質並列プロセッサ用プログラムの実行時間の下界計算式を提案する．下界計算に当たっては，同一種類のタスクからなるタスク集合にタスク間の先行制約とプロセッサ制約を付加する．そしてそれらのタスク集合の実行時間の下界を求め，そのうちの最大値を非均質並列プロセッサでプログラムを実行した場合の実行時間の下界とする．この新しい下界は異なる種類のタスク間にまたがった影響を考慮して求められているので，Fernandez の下界より精度が高い．

A Lower Bound on Time of Programs for Heterogeneous Parallel Processors

DINGCHAO LI,[†] TAKAYA ARITA,[†] NAOHIRO ISHII[†] and MASAHIRO SOWA[††]

Fernandez's lower bound is known as a very tight one on the minimum time required to execute a program on a fixed number of homogeneous parallel processors. But it is not suitable for representing the lower bound on the minimum time of the program in the case that a fixed number of heterogeneous parallel processors is available. This paper presents a new formulation for the lower bound on time of a program executed by a fixed number of heterogeneous parallel processors. On the calculation of the lower bound, both dependence constraints and processor constraints are appended to the sets which consist of tasks of an identical kind, respectively, and then the lower bounds on time of these sets are calculated, the maximum of which is used as the lower bound on time to execute the program on heterogeneous parallel processors. The new lower bound is sharper than Fernandez's lower bound because the influence among tasks of different kind is considered in this formulation.

1. はじめに

計算機のコストパフォーマンスを向上させるために，非均質並列プロセッサが重要になりつつある．非均質並列プロセッサとは，並列プロセッサを構成する処理ユニットが実行するタスクの種類が，処理ユニットごとに異なるものである[1),2)]．このような並列プロセッサの能力を有効に引き出すためには，タスク間の先行制約に基づいたタスクグラフで表されたタスク集合（プログラム）を，処理ユニットに割り当てるスケジューリングアルゴリズムが必要となる．非均質並列プロセッサ用プログラムのヒューリスティックなスケジューリング方式として，機能別小並列プロセッサへのスケジューリング法[3)]がすでに提案されているが，スケジューリングの評価基準，すなわち，プログラムを非均質並列プロセッサで実行するのに要する実行時間の下界はまだ求められていない．

プログラムの実行時間を最小にするスケジューリング問題は，特殊な場合を除いて強 NP 困難である[4)]．そのため，並列プロセッサ上でプログラムをノンプリエンプティブに実行する際に必要な最小の実行時間の代わりに，プログラムの実行時間の下界がスケジューリングの評価基準として用いられている[9)~11)]．それに関しては，並列処理やオペレーションズリサーチなどの分野でいくつかの研究が報告されている[5)~7)]．そのうち Fernandez の下界[5)]は均質並列プロセッサ（能力の等しい処理ユニットからなるプロセッサ）用プログラムの実行時間の下界としてよく知られている．しかし，この下界は非均質並列プロセッサ用プログラムの実行時間の下界として用いるのには適していない．

† 名古屋工業大学電気情報工学科
Department of Electrical Engineering and Computer Science, Nagoya Institute of Technology
†† 電気通信大学
The University of Electro-Communications

なぜならば，非均質並列プロセッサの場合には，ある処理ユニットがアイドルになっていても，異なる種類のユニット用のタスクを実行するわけにはいかないので，タスクを実行する処理ユニットの台数に制限があるとき，その制限によるタスクの実行の影響が，従属関係のある他の種類のタスクに及ぶからである．Fernandez の下界は，このような異なる種類のタスク間にまたがった影響に基づく実行時間の増量を考慮していないので，非均質並列プロセッサにおけるプログラムの最小の実行時間からより大幅に下回る可能性がある．

本論文では，プログラムを非均質並列プロセッサで実行する場合の実行時間の下界計算式[8]を提案する．下界を求めるに当たっては，タスクグラフからタスクの種類別に，種類別タスクグラフを呼ぶサブグラフを作成する．種類別タスクグラフは，同一種類のタスクのみからなるグラフで，他の種類のタスクとの間にある先行制約による影響と，その種類のタスクを実行する処理ユニットの台数による影響が，タスクグラフのアーク上に遅延時間として付け加えられている．このようにして作られた種類別タスクグラフの実行時間の下界を求め，そのうちの最大値を非均質並列プロセッサでプログラムを実行した場合の実行時間の下界とする．

以下，2章では，タスクグラフと一般化した非均質並列プロセッサのモデルについて，3章では，プリミティブな種類別タスクグラフの生成とプロセッサ制約の付加について，4章では，プログラムを非均質並列プロセッサで実行する際に要する時間の下界の計算式について述べる．5章では，具体例を示すと同時に本計算式の下界に関する評価を行う．

2. 問題の定義

2.1 タスクグラフ

プログラムの実行時間の下界計算では，タスクグラフとよばれる有限無サイクル有向グラフを扱う．タスクグラフ G はタスク T_i $(i=1,2,\cdots,n)$ を表すノードの集合 N と，タスク間の先行制約を表すアークの集合 A から構成される．ノード i からノード j へのアークがあるときには，ノード i の実行が終了しなければ j の実行が開始できないことを意味する．ここで，i のことを j の直接先行ノードと呼び，j のことを i の直接後継ノードと呼ぶ．また，タスクグラフはただ一つの入口ノードとただ一つの出口ノードをもっており，

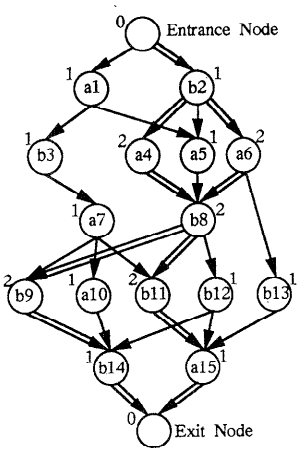


図 1 タスクグラフ
Fig. 1 A task graph.

入口ノードからすべてのノードに到達可能で，すべてのノードから出口ノードに到達可能であるものとする．これは，問題のグラフを与えられたとき，入口ノードとして一つのダミーノードを用意し，このノードから問題のグラフの直接先行ノードをもたないすべてのノードにアークを引き，また，同様に，出口ノードとして一つのダミーノードを用意し，問題のグラフの直接後継ノードをもたないすべてのノードからこのノードにアークを引くことによって，一般性を失うことなく実現できる．

図1に2種類の，15個のタスクからなるタスクグラフの例を示す．この図でノードはタスク，アークはタスク間の先行制約である．ノード内の a，b とその後ろにある数字はそれぞれ，タスク種類 a，タスク種類 b とタスク番号 i を表し，ノード外の数字はタスクの実行時間 t_i（ユニット時間）を表す．また2重線で示されているパスはクリティカルパスと呼ばれるタスクグラフ中の最長パスであり，その長さ t_{op} はプロセッサの処理ユニットが必要な数だけ使えるという前提のもとで，タスクグラフのタスク集合を並列に実行する際に必要な最小の実行時間を表す．

2.2 非均質並列プロセッサのモデル

タスクグラフで表されたプログラムは非均質並列プロセッサ上で実行されるとする．モデルとなる非均質並列プロセッサ P を次のように定義する．

1. プロセッサ P には，s 種類の処理ユニットがあり，同じ種類の処理ユニット PU_k $(1\leq k\leq s)$ が m_k 台ある．
2. 処理ユニット PU_k $(1\leq k\leq s)$ は種類 k のタスク

のみを実行でき，実行時に実行開始から終了まで割り込みなくタスクを実行できる．

本論文では，このような一般化した非均質並列プロセッサ $P(=P(s,[m_1,m_2,\cdots,m_s]))$ を前提として，タスクグラフ G で表されたプログラムを実行するのに要する実行時間の下界を求める．

3. 種類別タスクグラフ

3.1 プリミティブな種類別タスクグラフの生成

下界計算の準備として，タスクを実行する処理ユニットの台数に制限がないときの種類別タスクグラフを生成する．このグラフは，問題が本質的に持つ先行制約に基づいて各種類の処理ユニットが実行するタスクの実行可能な条件を陽に表現するものである．

$G=(N,A)$ をタスクグラフとする．まず，N を，入口ノード，出口ノードおよび種類 k のタスクを表すノードの集合 $N_k(1\leq k\leq s)$ に分割する（N_k にダミー入口/出口ノードがふくまれる理由は各タスクの実行時刻の定義を簡単にするためである）．そして，N_k のノード i からノード j へアークを付けるための必要十分条件を，タスクグラフ G に i から j へのパスがあって，しかもこのパス上のノードのうち i と j 以外に N_k のノードが含まれていないとする．次に，ノード i から j へのアークに，異なる種類のタスクに従属していることによって j の実行が遅延されなければならない時間 d_{ij} をラベルとして付け加える．この d_{ij} を次式で表し，ノード i と j の間の遅延時間と呼ぶ．

$$d_{ij}=\max_{w}\sum_{u\in\pi_w}t_u$$

ここで，π_w はタスクグラフ G 中，ノード i の直接後継ノードから j の直接先行ノードの間にあるパスのうち，種類 k のノードを含まないパス w 上にあるノードの集合である．$\Sigma t_u, u\in\pi_w$ は π_w に属するノードの実行時間の和である．

このようなラベル付きアークの集合を A_k としたとき，グラフ $G_k=(N_k,A_k)$，$k=1,2,\cdots,s$ を G のプリミティブな種類別タスクグラフと呼ぶ．

図2に，図1のプリミティブな種類別タスクグラフを示す．同図(a)は種類 a のノードに関するもの，(b)は種類 b のノードに関するものである．アークの近くにある数字は処理ユニットの台数に制限がない場合，そのアークの始点側のノードと終点側のノードとの間にある遅延時間である．例えば，同図(b)の $b2$ から $b8$ へのアークに図1のノード $a4$，$a5$，$a6$ の実行に必要な2ユニット時間が付加され，$b3$ から $b14$ へのアークに $a7$，$a10$ の実行に必要な2ユニット時間が付加されている．また，この図では2重線のパスは G_k のクリティカルパスを示す．ここでは，G_k のクリティカルパスは G_k の入口ノードから出口ノードまでのパスのうち，パス上にあるすべてのノードの実行時間とノード間の遅延時間の和が最大のパスであると定義する．その長さ t_{cp}^k は，処理ユニットが必要な数だけ使えるという前提のもとで，G_k のタスク集合の実行に要する最小の実行時間を表す．

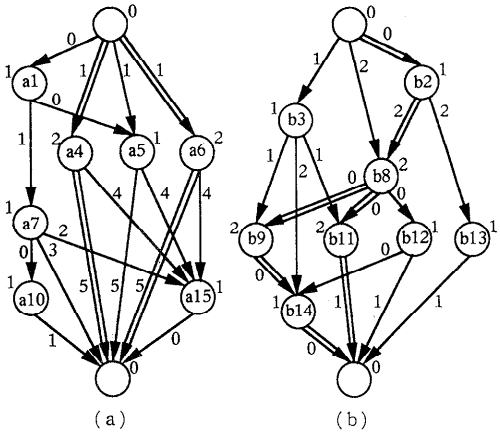


図 2 プリミティブな種類別タスクグラフ
Fig. 2 A primitive kind-partitioned task graph.

3.2 プロセッサ制約の付加

プリミティブな種類別タスクグラフは，同一種類のタスク間にある先行制約を表すと同時に異なる種類のタスク間にある先行制約によるタスクの実行遅延をも表している．しかし，非均質並列プロセッサにおけるタスクの実行は，異なる種類のタスク間の先行制約によって遅延される以外に，各種類の処理ユニットの台数に制限があることによっても遅延される．例えば，種類 a のタスクを実行する処理ユニットの台数に制限がない場合には，図1のノード $a4$，$a5$，$a6$ を2ユニット時間で並列に実行できるが，処理ユニットが2台の場合には3ユニット時間がかかる．したがって，処理ユニット台数の有限性による制約をプリミティブな種類別タスクグラフに付加する必要がある．このような制約を，ここではプロセッサ制約と呼ぶ．

G_k において，ノード i から j へのプロセッサ制約は，タスクグラフ G に i から j へのパスがあって，しかも i から j の間のプロセッサ制約による遅延時間がそれらの間の先行制約による遅延時間より大きいとき，次の操作のどちらかを行うことにより付加する．

1. i から j へのアークがない場合，i から j にアークを引き，そのアークにプロセッサ制約による遅延時間を付け加える．
2. i から j へのアークがある場合，そのアークに付けられた遅延時間を，プロセッサ制約による遅延時間に書きかえる．

プロセッサ制約による遅延時間 d'_{ij} を，次式によって求める．

$$d'_{ij}=\max_{1\le k\le s}\left\lceil\frac{1}{m_k}\sum_{u\in\phi^k}t_u\right\rceil$$

ここで，$\lceil\ \rceil$ は囲まれた式の値より大きい最小の正整数である．ϕ^k はタスクグラフ G 中，i の直接後継ノードから j の直接先行ノードへのすべてのパス上にある種類 k のノードの集合である．この式では，ϕ^k に属するノードの実行時間の和を処理ユニット台数 m_k で割った結果は，m_k 台の PU_k における ϕ^k の実行に要する時間の下界を表し，すべての ϕ^k，$k=1,2,\cdots,s$ に対して求めた下界のうちの最大値は，それらのノードの集合を非均質並列プロセッサ P で実行する際に必要な時間の下界を表す．

一方，先行制約による遅延時間 d''_{ij} は次式に示すもので，タスクグラフ G における i の直接後継ノードから j の直接先行ノードまでの最長パスの長さと等しい．この d''_{ij} は，どのような非均質並列プロセッサを用いてもノード i の実行終了から j の実行開始まで必要となる最小の遅延時間を表す．

$$d''_{ij}=\max_{w}\sum_{u\in\phi_w}t_u$$

ここで，ϕ_w はタスクグラフ G 中，i の直接後継ノードから j の直接先行ノードの間にあるパス w 上にあるノードの集合である．

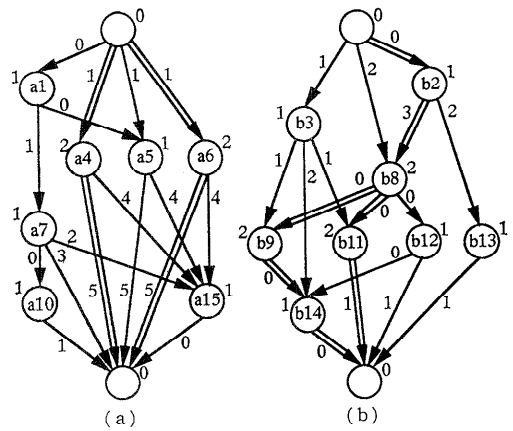


図 3 プロセッサ制約付き種類別タスクグラフ
Fig. 3 A kind-partitioned task graph with processor constraints.

図 3 は，非均質並列プロセッサが $P(2,[2,2])$ の場合，図 2 にプロセッサ制約を付加した結果である．この図でノード $b2$ と $b8$ の間にあるプロセッサ制約による遅延時間が 3 ユニット時間で，それらの間の先行制約による遅延時間（2 ユニット時間）より大きいので，$b2$ から $b8$ へのアークに付けられた遅延時間は 3 ユニット時間となっている．

4. プログラムの実行時間の下界

プロセッサ制約付き種類別タスクグラフをもとにして，タスクグラフ G のタスク集合の実行に要する時間の下界を求める．以下，各タスクの可能な実行開始時刻と終了時刻を定義し，それらの実行時間の下界計算式を導く．

4.1 タスクの実行時刻の定義

定義 1 グラフ G_k におけるノード i の最早開始時刻 $\tau_{es}(i)$ とは，i のすべての先行ノードが実行されてから，i の実行を開始できる最も早い時刻である．ノード i の最早終了時刻 $\tau_{ec}(i)$ とは，i の実行を終了できる最も早い時刻である．すなわち，

$$\tau_{es}(i)=\max_{w}\left\{\sum_{u\in\phi_w}t_u+\sum_{(u,v)\in\phi_w}d_{uv}\right\}$$

$$\tau_{ec}(i)=\tau_{es}(i)+t_i$$

ただし，ϕ_w，ψ_w はそれぞれ，G_k 中，ダミー入口ノードからノード i へ至るパス w 上にある i 以外のノードの集合とアークの集合である．(u,v) はノード u から v へのアークであり，d_{uv} は u と v の間の遅延時間である．

定義 2 グラフ G_k におけるノード i の最遅開始時刻 $\tau_{ls}(i)$ とは，t^k_{cp} の時間で G_k のタスク集合を実行終了することを遅らせることなく，i の実行を開始できる最も遅い時刻である．ノード i の最遅終了時刻 $\tau_{lc}(i)$ とは，i の実行を終了できる最も遅い時刻である．すなわち，

$$\tau_{ls}(i)=\min_{w}\left\{t^k_{cp}-\left(\sum_{u\in\hat{\phi}_w}t_u+\sum_{(u,v)\in\hat{\phi}_w}d_{uv}\right)\right\}$$

$$\tau_{lc}(i)=\tau_{ls}(i)+t_i$$

ただし，$\hat{\phi}_w$，$\hat{\psi}_w$ はそれぞれ，G_k のすべてのアークを逆向きにしたとき，ダミー出口ノードからノード i へ至るパス w 上にあるノードの集合とアークの集合である．

4.2 下界計算式

非均質並列プロセッサ P 上で，タスクグラフ G のタスク集合を実行する際に必要な時間の下界 $t_{\text{Lower Bound}}$

は次式によって計算できる．

$$t_{\text{Lower Bound}}=\max_{1\le k\le s}\{t^k_{\text{Lower Bound}}\}$$

$$t^k_{\text{Lower Bound}}=t^k_{cp}+\lceil q^k\rceil$$

ここで，$t^k_{\text{Lower Bound}}$ はグラフ G_k のタスク集合を処理ユニット PU_k 上で実行する場合の実行時間の下界である．q^k は PU_k が m_k 台しかないとき，G_k のタスク集合を実行するのに要する時間の t^k_{cp} からの最小増量である．

定理1 $T(\theta_1,\theta_2,i)$ は t^k_{cp} の時間で G_k のタスク集合の実行を終了するとき，整数区間 $[\theta_1,\theta_2]$ $(\subset[0,t^k_{cp}])$ でタスク T_i が実行されなければならない時間であるとする．G_k のタスク集合 N_k を，m_k 台の処理ユニット PU_k 上で実行するのに要する時間の t^k_{cp} からの最小増量 q^k は，以下となる．

$$q^k=\max_{[\theta_1,\theta_2]}\left[-(\theta_2-\theta_1)+\frac{1}{m_k}\sum_{i\in N_k}T(\theta_1,\theta_2,i)\right]$$

(証明) 整数区間 $[\theta_1,\theta_2]$ $(\subset[0,t^k_{cp}])$ において，m_k 台の PU_k が実行できるタスクの実行時間の和は $m_k(\theta_2-\theta_1)$ である．一方，この区間で実行されなければならないタスクの実行時間の和は $\sum T(\theta_1,\theta_2,i)$, $i\in N_k$ である．この両者の差を m_k で割った結果は，区間 $[\theta_1,\theta_2]$ におけるタスク集合 N_k の実行に必要な時間の t^k_{cp} からの最小増量を表す．したがって，$[0,t^k_{cp}]$ に属するすべての整数区間に対して求めた最小増量の最大値は，G_k のタスク集合の実行に必要な時間の t^k_{cp} からの最小増量である．

この証明からわかるように，q^k を求めるのに $t^k_{cp}(t^k_{cp}+1)/2$ 個の区間における最小増量の計算が必要となり，その計算量がおよそ $(t^k_{cp})^2$ 程度で増加する．

定理2 G_k のタスク集合の実行を t^k_{cp} の時間で終了するとき，タスク T_i が整数区間 $[\theta_1,\theta_2]$ $(\subset[0,t^k_{cp}])$ で実行されなければならない時間 $T(\theta_1,\theta_2,i)$ は，以下となる．

$$T(\theta_1,\theta_2,i)=\begin{cases}t_{\min} & \theta_1<\tau_{ec}(i)\ \text{and}\ \theta_2>\tau_{ls}(i)\\ 0 & \text{otherwise}\end{cases}$$

$$t_{\min}=\min[\tau_{ec}(i)-\theta_1,\theta_2-\tau_{ls}(i),\theta_2-\theta_1,t_i]$$

ただし，$\tau_{ec}(i)$ はタスク T_i の最早終了時刻であり，$\tau_{ls}(i)$ はタスク T_i の最遅開始時刻である．

(証明) 可能性をすべて調べることによって $T(\theta_1,\theta_2,i)$ の計算式が成り立つことを示す．

1. $\theta_1\ge\tau_{ec}(i)$ の場合：
   タスク T_i が区間 $[\tau_{es}(i),\tau_{ec}(i)]$ で実行できるので，区間 $[\theta_1,\theta_2]$ で実行されなくても，G_k のタスク集合の実行を遅らせない．ゆえに，$T(\theta_1,\theta_2,i)=0$ が成立する．
2. $\theta_2\le\tau_{ls}(i)$ の場合：
   タスク T_i が区間 $[\tau_{ls}(i),\tau_{lc}(i)]$ で実行できるので，$T(\theta_1,\theta_2,i)=0$ が成立する．
3. $\theta_1<\tau_{ec}(i)$，しかも $\theta_2>\tau_{ls}(i)$ の場合：
   タスク T_i が区間 $[\tau,\tau+t_i]$，$\tau_{es}(i)\le\tau\le\tau_{ls}(i)$ で実行されるとする．すると，区間 $[\theta_1,\theta_2]$ における T_i の実行時間 $t(\tau)$ は次のようになる．

$$t(\tau)=\begin{cases}0 & \tau+t_i\le\theta_1\\ 0 & \tau\ge\theta_2\\ \theta_2-\tau & \theta_1\le\tau<\theta_2\le\tau+t_i\\ \tau+t_i-\theta_1 & \tau\le\theta_1<\tau+t_i\le\theta_2\\ \theta_2-\theta_1 & \tau\le\theta_1<\theta_2\le\tau+t_i\\ t_i & \theta_1\le\tau<\tau+t_i\le\theta_2\end{cases}$$

タスク T_i は，グラフ内の他タスクには影響することなく $\tau_{es}(i)$ から $\tau_{ls}(i)$ の間の任意の時刻 τ に実行することができる．したがって，T_i が区間 $[\theta_1,\theta_2]$ で実行されなければならない時間は区間 $[\tau_{es}(i),\tau_{ls}(i)]$ における $t(\tau)$ の最小値である．上式では，$\theta_2-\tau$ は $\tau=\tau_{ls}(i)$ で最小値 $\theta_2-\tau_{ls}(i)$ になり，$\tau+t_i-\theta_1$ は $\tau=\tau_{es}(i)$ で最小値 $\tau_{ec}(i)-\theta_1$ になる．ゆえに，$\theta_1<\tau_{ec}(i)$，しかも $\theta_2>\tau_{ls}(i)$ の場合，区間 $[\tau_{es}(i),\tau_{ls}(i)]$ における $t(\tau)$ の最小値 $t_{\min}$ は

$\min[\tau_{ec}(i)-\theta_1,\theta_2-\tau_{ls}(i),\theta_2-\theta_1,t_i]$

となる．

5. 評　価

タスクグラフ G のタスク集合を，m 台の処理ユニットからなる均質並列プロセッサでノンプリエンプティブに実行する際に必要な時間の下界を求めるためには，次に示す Fernandez の下界計算式が用いられる．

$$t_{\text{Fernandez}}=t_{cp}+\lceil q_F\rceil$$

$$q_F=\max_{[\theta_1,\theta_2]}\left\{-(\theta_2-\theta_1)+\frac{1}{m}|\mathcal{G}\cup\bar{\mathcal{G}}|\right\}$$

ここで，q_F は G のタスク集合を実行するのに要する時間の t_{cp} からの最小増量である．$\mathcal{G}\cap\bar{\mathcal{G}}$ は整数区間 $[\theta_1,\theta_2]$ の積バグ (bag) と呼ばれ，次の性質を持つ集合である[5]．

1. 整数区間 $[\theta_1,\theta_2]$ で実行されなければならないタスクの番号からなる．
2. タスク番号の繰り返しを許し，あるタスク番号の繰り返し回数は，このタスクが整数区間 $[\theta_1,\theta_2]$ で実行されなければならない時間と等しい．

$|\mathcal{G}\cap\bar{\mathcal{G}}|$ は積バグ $\mathcal{G}\cap\bar{\mathcal{G}}$ にあるタスク番号の数で

ある．

これを，非均質並列プロセッサP上でGのタスク集合を実行する際に必要な時間の下界の計算問題に対して適用する場合，その下界計算式は次のようになる．

$$t'_{\text{Fernandez}}=t_{cp}+\max_{1\le k\le s}\lceil q_F^k\rceil$$

$$q_F^k=\max_{[\theta_1,\theta_2]}\left\{-(\theta_2-\theta_1)+\frac{1}{m_k}|\mathcal{G}^k\cap\bar{\mathcal{G}}^k|\right\}$$

ここで，$\mathcal{G}^k\cap\bar{\mathcal{G}}^k$ は種類kのタスクの番号からなる整数区間 $[\theta_1,\theta_2]$ の積バグである．

補題1 本計算式の下界は，$t'_{\text{Fernandez}}$ より精度が高い．

(証明)

1. プリミティブな種類別タスクグラフの場合：

   3.1節の生成操作により，$t_{cp}^k=t_{cp}$，$k=1,2,\cdots,s$ である．したがって，4章の下界は

   $$t_{\text{Lower Bound}}=t_{cp}+\max_{1\le k\le s}\lceil q^k\rceil$$

   となる．次に，q^k と q_F^k について比較を行う．整数区間 $[\theta_1,\theta_2]$ において，タスク T_i が実行されなければならないとする．積バグの性質によって，この区間の積バグにあるタスク番号 i の繰り返し回数は $T(\theta_1,\theta_2,i)$ と等しい．ゆえに，$q^k=q_F^k$ となる．その結果，プロセッサ制約を付加しない場合，$t_{\text{Lower Bound}}=t'_{\text{Fernandez}}$ である．

2. プロセッサ制約付き種類別タスクグラフの場合：異なる種類のタスク間にまたがったタスクの実行影響はタスクグラフのクリティカルパスを長くする可能性がある．タスクグラフにプロセッサ制約を付加することにより，その影響によるクリティカルパス長からの増量を計算することができる．この増量を δ_{cp}^k とすると，$t_{cp}^k=t_{cp}+\delta_{cp}^k$，$k=1,2,\cdots,s$ である．したがって，4章の下界は

   $$t_{\text{Lower Bound}}=t_{cp}+\max_{1\le k\le s}\{\delta_{cp}^k+\lceil q^k\rceil\}$$

   となる．1.の場合の証明と同様に，$\delta_{cp}^k+\lceil q^k\rceil\ge\lceil q_F^k\rceil$ であることがわかる．その結果，プロセッサ制約を付加した場合，$t_{\text{Lower Bound}}\ge t'_{\text{Fernandez}}$ である．

以上により，補題が成り立つ．

次に，例として，図1のプログラムを非均質並列プロセッサ $P(2,[2,2])$ で実行する際に必要な時間の下界を計算する．

表1 図3の各タスクの可能な実行時刻
Table 1 The possible execution-time of each task in Fig. 3.

実行時刻	タスクの番号													
	a1	a4	a5	a6	a7	a10	a15	b2	b3	b8	b9	b11	b12	b13	b14
τ_{es}	0	1	1	1	3	4	7	0	1	4	6	6	6	3	8
τ_{ec}	1	3	2	3	4	5	8	1	2	6	8	8	7	4	9
τ_{ls}	1	1	2	1	4	6	7	0	4	4	6	6	7	7	8
τ_{lc}	2	3	3	3	5	7	8	1	5	6	8	8	8	8	9

表2 図3のタスク集合の実行時間の下界計算過程
Table 2 The calculation of the lower bound on time for the set of tasks in Fig. 3.

$[\theta_1,\theta_2]$	q_π^1	q_π^2	$[\theta_1,\theta_2]$	q_π^1	q_π^2	$[\theta_1,\theta_2]$	q_π^1	q_π^2	$[\theta_1,\theta_2]$	q_π^1	q_π^2	$[\theta_1,\theta_2]$	q_π^1	q_π^2
[0,1]	-0.5	-0.5	[0,2]	-0.5	-1.5	[0,3]	0	-2.5	[0,4]	-1	-3.5	[0,5]	-1.5	-3.5
[1,2]	0	-1	[1,3]	(0.5)	-2	[1,4]	-0.5	-3	[1,5]	-1	-3	[1,6]	-2	-3.5
[2,3]	0	-1	[2,4]	-1	-2	[2,5]	-1.5	-2.5	[2,6]	-2.5	-3	[2,7]	-3	-3
[3,4]	-1	-1	[3,5]	-1.5	-1.5	[3,6]	-2.5	-2	[3,7]	-3	-2	[3,8]	-3.5	-1
[4,5]	-1	-0.5	[4,6]	-2	-1	[4,7]	-2.5	-1	[4,8]	-3	-0.5	[4,9]	_	-1
[5,6]	-1	-0.5	[5,7]	-2	-0.5	[5,8]	-2.5	0	[5,9]	_	-0.5	[0,6]	-2.5	-4
[6,7]	-1	0	[6,8]	-1.5	(0.5)	[6,9]	_	0	[0,7]	-3	-4	[1,7]	-2.5	-3.5
[7,8]	-0.5	0	[7,9]	_	-0.5	[0,8]	-3.5	-3	[1,8]	-3	-2.5	[2,8]	-3.5	-2
[8,9]	_	-0.5	[0,9]	_	-3.5	[1,9]	_	-3	[2,9]	_	-2.5	[3,9]	_	-1.5

$q^k=\max_{[\theta_1,\theta_2]}\{q_\pi^k\}$, k=1,2

$t^1_{\text{LowerBound}}=t^1_{cp}+\lceil q^1\rceil=9$, $t^2_{\text{LowerBound}}=t^2_{cp}+\lceil q^2\rceil=10$ $\Longrightarrow$ $t_{\text{LowerBound}}=10$

表1は，図1から生成された図3の種類別タスクグラフにおける各タスクの可能な実行開始時刻と終了時刻である．**表2**は表1と4章の下界計算式を用いて，図3(a)のタスク集合の実行時間の下界と図3(b)のタスク集合の実行時間の下界を計算する過程である．図3(a)の t_{cp}^1 が8，図3(b)の t_{cp}^2 が9であるので，この表では区間 [0,8] に属するすべての整数区間における同図(a)の実行に要する時間の t_{cp}^1 からの最小増量 q_π^1 と，区間 [0,9] に属するすべての整数区間における同図(b)の実行に要する時間の t_{cp}^2 からの最小増量 q_π^2 を示している．q_π^k，$k=1,2$ は定理1により，以下となる．

$$q_\pi^k=\left[-(\theta_2-\theta_1)+\frac{1}{m_k}\sum_{i\in N_k}T(\theta_1,\theta_2,i)\right]$$

この表からわかるように，プロセッサ制約を付加している場合，図1のプログラムの実行時間の下界は10ユニット時間である．

また，同様に，図2(a)のタスク集合の実行時間の下界と図2(b)のタスク集合の実行時間の下界を計算できる．その結果，求めた下界は Fernandez の下界と同じ，9ユニット時間である．したがって，この例からも本計算式の下界が Fernandez の下界より精度がよいことがわかる．

6. おわりに

以上，これまで提案されていた，スケジューリングの評価基準となっている Fernandez の下界が非均質並列プロセッサ用プログラムの実行時間の下界として用いるのには適していないことを指摘し，非均質並列プロセッサ用プログラムの実行時間の下界計算式を提案した．本計算式は異なる種類のタスク間にまたがった影響を考慮するので，Fernandez の下界より高い精度の下界を得ることが明らかとなった．

本論文は共有レジスタを持つような非均質並列プロセッサを対象としているため，モデルは比較的シンプルとなっている．より，一般的なシステムを対象とするためには，たとえば，(1)データ通信時間を無視できないモデル，(2)処理ユニットにより同一タスクの実行時間が異なるような非均質性を持つモデル，などを考慮する必要があると考えられる．

また，本論文では，非均質並列プロセッサにおけるタスク集合の実行時間に関する簡単な下界を，プロセッサ制約による遅延時間として用いた．より高い精度のプログラムの実行時間の下界を求めるためには，Fernandez の下界，あるいは本計算式の下界をこの遅延時間として使用できる．ただし，下界の計算に要する時間の計算量は大きくなる．今後，この点や本下界計算式中の最小増量 (q^k) の計算量などについて詳細な評価を行っていくつもりである．

謝辞 本研究を進めるにあたり，貴重なご意見をいただいた名工大の高木浩光氏に感謝いたします．

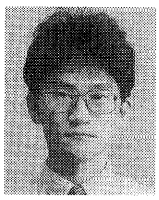
参考文献

1) McGeady, S.: The i960CA Superscalar Implementation of the 80960 Architecture, *Compcon Spring 90 digest of papers*, pp. 232-240, IEEE (1990).
2) 曽和将容，有田隆也，河村忠明，高木浩光：機能分割型プロセッサによる複数命令流実行方式，電子情報通信学会論文誌，Vol. J73-D-I, No. 3, pp. 280-285 (1990).
3) 李 鼎超，有田隆也，曽和将容：非均質並列プロセッサ用プログラムのスケジューリング法，電子情報通信学会論文誌，Vol. J75-D-I, No. 8, pp. 504-510 (1992).
4) Lenstra, K. K. and Kan, A.H.G.R.: Complexity of Scheduling under Precedence Constraints, *Oper. Res.*, Vol. 26, pp. 22-35 (1978).
5) Fernandez, E. B. and Bussell, B.: Bounds on the Number of Processors and Time for Multiprocessor Optimal Schedules, *IEEE Trans. on Comput.*, Vol. C 22, No. 8, pp. 745-751 (1973).
6) Hu, T. C.: Parallel Sequencing and Assembly Line Problems, *Oper. Res.*, Vol. 9, pp. 841-848 (1961).
7) Al-Mouhamed, M. A.: Lower Bound on the Number of Processors and Time for Scheduling Precedence Graphs with Communication Costs, *IEEE Trans. on SE*, Vol. 16, No. 12, pp. 1390-1401 (1990).
8) 李 鼎超，有田隆也，曽和将容：非均質並列プロセッサ用プログラムの実行時間の下限，電子情報通信学会技術研究報告，CPSY 92-10 (1992).
9) Coffman, E. G.: *Computer and Jobshop Scheduling Theory*, John Wiley & Sons (1976).
10) Kasahara, H. and Narita, S.: Practical Multiprocessor Scheduling Algorithms for Efficient Parallel Processing, *IEEE Trans. on Comput.*, Vol. C-33, No. 11, pp. 1023-1029 (1984).
11) Adam, T. L., Candy, K. M. and Dickson, J. R.: A Comparison of List Schedules for Parallel Processing Systems, *Commun. ACM*, Vol. 17, No. 12, pp. 685-690 (1974).

(平成4年9月14日受付)
(平成5年9月 8 日採録)

李 鼎超（正会員）

1962 年生．1983 年北京航空航天大学情報工学科卒業．1991 年福井大学大学院博士前期課程修了．現在，名古屋工業大学大学院博士後期課程在学中．並列処理システムに関する研究に従事．電子情報通信学会会員．

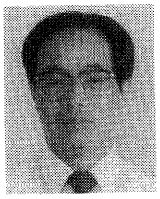


有田 隆也（正会員）

1960 年生．1983 年東京大学工学部計数工学科卒業．1988 年同大学大学院工学系博士課程修了．工学博士．同年名古屋工業大学工学部電気情報工学科助手．1993 年同学科講師，現在に至る．並列計算機アーキテクチャ，プログラミング方法論，人工生命に興味を持つ．IEEE，電子情報通信学会各会員．

石井 直宏（正会員）

昭和 38 年東北大学工学部電気卒業．昭和 43 年同大大学院博士課程電気および通信工学専攻修了．同年同大医学部助手．昭和 50 年名古屋工業大学工学部助教授を経て，現在，同大電気情報工学科教授．この間，しきい値論理，医用情報処理，および非線形処理の研究に従事．工学博士．



曽和 将容（正会員）

1974年名古屋大学大学院博士課程（電気電子専攻）修了．同年群馬大学工学部情報工学科助手．1976年助教授．1987年名古屋工業大学教授．1993年電気通信大学教授．この間，並列処理，計算機アーキテクチャ，特にデータフロー計算機，コントロールフロー計算機など次世代コンピュータの研究に従事．工学博士．IEEE, ACM, 電子情報通信学会各会員．