情報処理学会論文誌　コンピューティングシステム　Vol. 4　No. 1　111–146 (Feb. 2011)

計算資源の有効利用を目的としたタスククラスタリング

金光永煥[†1]　李吉憲[†1]　中里秀則[†2]
星合隆成[†2]　浦野義頼[†2]

限られた数の計算資源でタスクスケジューリングを行う際，少ないプロセッサ数でスケジュール長をできるだけ短縮する方針が重要である．タスククラスタリングによって得られたタスク集合（クラスタ）数がプロセッサ数よりも大きくなれば，従来手法では生成されたクラスタどうしを集約する方針がとられてきた．しかしながら，この手法により同時実行が可能なタスク数が減少し，タスククラスタリング後よりもスケジュール長が大きくなる可能性がある．そこで本論文では，事前にプロセッサ数を限定するために各クラスタサイズ（クラスタに属するタスク実行時間の和）の下限値を設け，この条件の下でスケジュール長を短縮することを目的としたクラスタサイズ δ_{opt} の算出手法を示す．そして，各クラスタサイズが δ_{opt} 以上となるようにタスクどうしを集約するタスククラスタリングを提案する．シミュレーションの結果，δ_{opt} というクラスタサイズを決めることにより，従来手法よりも少ないプロセッサ数でスケジュール長が短縮できるという結果が得られた．

A Task Clustering for Utilizing Computational Resources

Hidehiro Kanemitsu,[†1] Gilhyon Lee,[†1]
Hidenori Nakazato,[†2] Takashige Hoshiai[†2]
and Yoshiyori Urano[†2]

When we perform scheduling every task in a distributed environment where the number of processors is limited or unknown, it is important to decide a policy for suppressing both the required number of processors and its schedule length. If the number of processors (clusters) which has been derived by one of task clustering heuristics is too large with compared to the actual number of processors, a policy for merging several clusters to generate a larger one, has been adopted by conventional approaches. However, degree of task parallelism can be decreased and then schedule length by those approaches becomes larger than that of a task clustering. In this paper, we present a method for deriving the lower bound of each cluster size (δ_{opt}) in order to suppress the number of processors as well as to minimize schedule length. We also present a task clustering algorithm which performs clustering each task until every cluster size exceeds δ_{opt} for minimizing schedule length. Experimental results by simulation show that our proposed task clustering under δ_{opt} condition can suppress schedule length with the small number of clusters.

1. はじめに

プロセッサがネットワーク上で分散された並列処理環境において，タスク間の先行関係を考慮した様々なタスクスケジューリングが提案されてきた[1)]．互いに先行関係が定義されたタスク集合は，一般に有向非循環グラフ（DAG: Directed Acyclic Graph）のタスクグラフとして定義され，DAG に属する各タスクをスケジュールする問題は DAG スケジューリング問題として知られている．DAG スケジューリングは，さらにプロセッサ数があらかじめ与えられている問題と，そうでない問題に分類される．前者の問題に対し，従来は各タスクの優先度に応じてスケジュールする手法が多く用いられてきた．後者においては，タスククラスタリングがある[3),4),6),9),11)–15)]．タスククラスタリングでは，タスクどうしを集約してタスク間のデータ転送時間を局所化し，クラスタ（タスクの集合）を生成することによって各クラスタを各プロセッサへの割当て単位とする．タスククラスタリングによるスケジュール長が最小となるプロセッサ数を決定できる．しかしながら，タスククラスタリングではプロセッサ数を制限しないため，DAG によっては得られるクラスタ数が実際のプロセッサ数よりも大きくなる可能性がある．そこで従来，タスククラスタリングの後にクラスタどうしを集約する（本論文ではクラスタ集約と記す）ことにより，クラスタ数を制限するための発見的な手法が提案されている[7),8)]．文献 7) では，タスククラスタリングの後に各クラスタサイズ（クラスタに属するタスク実行時間の和）の均一化という基準でクラスタ数を減少させる手法を提案している．また文献 8) では，先行関係を持つクラスタどうしに着目しつつ，各クラスタサイズの均一化を目的とした Load Balancing（LB）と，データ転送時間の最小化を目的とした Communication Traffic Minimizing（CTM）について述べている．しかしながら，これらのクラスタ集約手法では同時実行が可能なタスク数が減少してしまい，スケジュール長がタスククラスタリング後よりも大きくなる可能性がある．特に CTM

†1 早稲田大学大学院国際情報通信研究科
Graduate School of Global Information and Telecommunication Studies, Waseda University
†2 早稲田大学国際情報通信研究センター
Global Information and Telecommunication Institute, Waseda University

の場合は CASS-II [11] によるタスククラスタリング後のスケジュール長よりも最大で 55%，LB の場合は最大で 19%増加している．すなわち，同時実行が可能なタスク数を考慮せずにクラスタ集約を行うとスケジュール長が短縮されない場合がある．さらに，最終的に得られるクラスタ数によってもスケジュール長が異なってしまうため，スケジュール長をできるだけ短縮させるときの適切なプロセッサ数が分からないという問題がある．

そこで本論文では，クラスタ集約は行わずに必要プロセッサ数を減少させ，かつスケジュール長を最小にすることを目的とした，各プロセッサへ割り当てる仕事量とプロセッサ数の決定手法を提案する．まず，タスククラスタリングとスケジューリングを行う前に，各クラスタサイズをどのようなサイズにすべきかを考察する．このとき，スケジュール長がとりうる値のうち，各タスクの実行開始時刻をできるだけ遅らせた場合の値（4.2 節で $sl_w(G^s_{cls})$ と定義する）を最小とするときのクラスタサイズを算出する．さらに算出されたクラスタサイズを満たしつつ，$sl_w(G^s_{cls})$ をできるだけ減少させるためのタスククラスタリングを提案し，その有効性をシミュレーションによって示す[⋆1]．

本論文の構成は次のとおりである．2 章では本提案で想定するシステムの定義および用途の定義を行い，3 章では従来手法の問題点と本論文の解決対象について述べる．4 章ではクラスタサイズの算出を行い，5 章では提案するアルゴリズムについて述べる．そして 6 章ではシミュレーションによる従来手法との比較結果について述べる．最後に 7 章では，まとめと今後の課題について述べる．

2. 準　　備

2.1 想定システム

まず，本論文で想定する環境について述べる．本論文で想定する環境は，能力（処理速度）が等しく，タスク実行とデータ送受信を独立して行うことができる，完全結合型のプロセッサ環境である．各プロセッサは実行プログラムおよび実行に必要なデータを保持するための局所メモリを持ち，データ送信にはメッセージパッシングモデルを用いるものとする．また，メッセージパッシングモデルにおいて，一定の通信速度で複数のデータを同時に送受信できるといった古典的なモデルを想定する[⋆2]．このような環境は，従来から分散メモリ型のマルチプロセッサシステム[17] で適用されてきた．

次に，実行モデルについて述べる．入力として与えられるアプリケーションを $G_{org} = (V, E)$ と表し，DAG（Directed Asyclic Graph）とする．ここで V をタスクの集合，E をタスク間の通信の集合とする．G_{org} は単一実行プログラムを指し，タスクは 1 つの分割できない処理単位（関数，サブルーチン）とする．一方，E に属する任意の通信は，1 つのタスクの処理結果をネットワーク上で転送しなければならないことを指す．また，各プロセッサの処理速度，プロセッサ間通信速度は均一であるとし，$w(n_i)$ を $n_i \in V$ のタスクサイズ，すなわち各プロセッサでの処理時間とする．$e_{i,j} = (n_i, n_j) \in E$ を n_i から n_j へのメッセージの通信，$c(e_{i,j})$ を $e_{i,j}$ の転送時間とする．タスクサイズ同様，ここでは $c(e_{i,j})$ を $e_{i,j}$ のデータサイズと呼ぶことにする．

n_j は n_i からのメッセージが到着しないと実行が開始できないものとする．また，n_i の直接先行タスク集合を $pred(n_i)$, 直接後続タスク集合を $suc(n_i)$ とし，$pred(n_i) = \emptyset$ であるタスクを START タスク，$suc(n_i) = \emptyset$ であるタスクを END タスクとする．

任意の n_i，$n_j \in V$ において n_i から n_j への経路が存在するとき $n_i \prec n_j$ と表す．本論文ではクラスタを 1 つまたは複数のタスク集合と定義し，特にクラスタ i を $cls(i)$ と表す．

2.2 タスククラスタリング

タスククラスタリングとは，複数のタスクを集約して 1 つのクラスタを生成し，データ転送時間の局所化によってスケジュール長（2.3 節で定義）を最小化しようとする手法である．本論文では，タスク集約を「2 クラスタ内の全タスクを，いずれか一方のクラスタへ含めるための手続き」と定義する．また，タスククラスタリングを「複数のタスク集約が行われ，ある終了条件を満たせば終了する手続き」と定義する．

タスククラスタリングの入力として与えられる DAG を $G^s_{cls} = (V_s, E_s, V^s_{cls})$ とし，s をタスク集約回数とする．V_s に属するタスク集合を $\{n^s_1, n^s_2, \ldots\}$ とし[⋆3]，E_s に属する辺集合を $\{\ldots, e^s_{k,l}, \ldots\}$ とする．すなわち G^s_{cls} は，G_{org} に対して s 回のタスク集約が行われた後の DAG である．初期状態では $s = 0$ とし，$V_0 \leftarrow V$，$E_0 \leftarrow E$ とする．さらに初期状態では $cls_0(1) = \{n^0_1\}$, $cls_0(2) = \{n^0_2\}, \ldots$ とし，$V^0_{cls} \leftarrow \{cls_0(1), cls_0(2), \ldots\}$ とする．

$cls_s(i)$ のタスクサイズの和を $w(cls_s(i))$ とし，本論文では $w(cls_s(i))$ を $cls_s(i)$ のクラスタサイズと呼ぶことにする．$cls_s(i)$ と $cls_s(k)$ とのタスク集約を $merge(cls_s(i), cls_s(k))$ と

⋆1 文献 5) において，我々はタスククラスタリングの方針のみをあげているが，本論文では詳細を述べる．

⋆2 本論文ではこの古典的モデルに従った DAG スケジューリングにおけるタスク集約上の問題点を明らかにし，その解決法を提案する．そのため，タスク間の複数のデータ送信をまとめて 1 度で行うモデル[22] や，データ受信側タスクでは 1 度に 1 つの通信データしか受信できないというモデル[23] は，本論文では想定しないものとする．

⋆3 タスク自体の情報（タスクサイズ，直接先行タスクの集合および直接後続タスクの集合）は，タスク集約回数 s によらず不変である．しかしながら，クラスタおよび辺の情報は，タスク集約によって変わりうるため s を付与している．そこで，表記の統一性から，各タスクに対してもクラスタおよび辺と同様，s を付与している．

表し，この処理手順を

$$\begin{aligned}&cls_{s+1}(i) \leftarrow cls_s(i) \cup cls_s(k);\\ &V_{cls}^{s+1} \leftarrow V_{cls}^{s} - \{cls_s(k)\},\ E_{s+1} \leftarrow E_s;\\ &c(e_{p,q}^{s+1}) \leftarrow 0 \text{ for } \forall n_p^{s+1},\ n_q^{s+1} \in cls_{s+1}(i),\ e_{p,q}^{s+1} \in E_{s+1};\\ &\text{return } cls_{s+1}(i);\end{aligned} \tag{1}$$

とする．そして 1 つのクラスタは 1 つのプロセッサへの割当て単位とする．

$cls_s(i)$ に属する任意の 2 タスクの間に先行関係が存在するとき，$cls_s(i)$ は linear であるという[6]．$cls_s(i)$ が linear であれば，$cls_s(i)$ 内の各タスクの実行順は一意に決定される[6]．

2.3　スケジュール長の定義

各タスクは，直接先行タスクからのデータをすべて受信すると実行開始できる可能性がある．まず，n_j^s の実行開始時刻（スケジュールされる時刻）を $t_s(n_j^s)$，実行完了時刻を $t_f(n_j^s)$ とし，$t_f(n_j^s)$ を以下に定義する．

$$t_f(n_j^s) = t_s(n_j^s) + w(n_j^s). \tag{2}$$

n_j^s の直接先行タスクからのデータが到着すると，n_j^s は即座に実行開始できる可能性がある．一方，n_j^s と同一クラスタに属するタスク n_m^s がある場合を考える．n_j^s の直接先行タスクからのデータがすべて到着しても，n_m^s が実行中であれば n_j^s の実行は n_m^s の実行完了まで待たされる．本論文では，任意のタスクに対して直接先行タスクからのデータがすべて受信完了となる時刻を Data Ready Time（DRT）[2),23)] と定義する．ここでタスク n_j^s の DRT を $t_{dr}(n_j^s)$，$n_j^s \in cls_s(k)$ とし，以下に定義する．

$$\begin{aligned}t_{dr}(n_j^s) &= \max_{n_i^s \in pred(n_j^s)} \left\{ t_f(n_i^s) + c(e_{i,j}^s) \right\}\\ &= \max \left\{ \max_{\substack{n_i^s \in pred(n_j^s),\\ n_i^s \in cls_s(k)}} \{t_f(n_i^s)\},\ \max_{\substack{n_i^s \in pred(n_j^s),\\ n_i^s \notin cls_s(k)}} \{t_f(n_i^s) + c(e_{i,j}^s)\} \right\}.\end{aligned} \tag{3}$$

式 (3) より，$t_{dr}(n_j^s)$ は n_j^s の直接先行タスク n_i^s のうちで，n_j^s と同一クラスタに属しているものの完了時刻とそうでないもののデータ到着時刻から決定される．前者は同一クラスタに属しているのでデータ転送時間は 0 である．一方，後者はデータ転送時間 $c(e_{i,j}^s)$ だけ要する．一方，同一クラスタ内の直接先行タスクはすべて実行完了しているが，他クラスタの直接先行タスクからのデータをまだ受信していない場合は，データ受信待ち時間が存在する．本論文では，クラスタ k に属する n_j^s がすべての直接先行タスク完了直後にスケジュールされる場合のデータ受信待ち時間を $I(n_j^s, k)$ とし，以下に定義する．

$$I(n_j^s, k) = \begin{cases} 0, \text{ if } \max\limits_{\substack{n_i^s \in pred(n_j^s),\\ n_i^s \in cls_s(k)}} \left\{ t_f(n_i^s) \right\} \geq \max\limits_{\substack{n_i^s \in pred(n_j^s),\\ n_i^s \notin cls_s(k)}} \left\{ t_f(n_i^s) + c(e_{i,j}^s) \right\}, \\ \max\limits_{\substack{n_i^s \in pred(n_j^s),\\ n_i^s \notin cls_s(k)}} \left\{ t_f(n_i^s) + c(e_{i,j}^s) \right\} - \max\limits_{\substack{n_i^s \in pred(n_j^s),\\ n_i^s \in cls_s(k)}} \left\{ t_f(n_i^s) \right\}, \text{ otherwise.} \end{cases} \tag{4}$$

$t_{dr}(n_j^s)$ が決まっても，同一クラスタ内に n_j^s と先行関係のないタスクがあると，実行順（スケジュール方針）によって n_j^s がスケジュールされる時刻が変動する．したがって $t_{dr}(n_j^s) \leq t_s(n_j^s)$ が成り立つ[2),23)]．すなわち各タスクの実行完了時刻もスケジュール方針によって変動する．本論文では G_{cls}^s のスケジュール長を $sl(G_{cls}^s)$ とし，以下に定義する．

$$sl(G_{cls}^s) = \max_{n_j^s \in cls_s(k), cls_s(k) \in V_{cls}^s} \{t_f(n_j^s)\}. \tag{5}$$

式 (5) 中，最初に実行される START タスクの実行開始時刻を 0 と仮定している．

2.4　クラスタ集約

タスククラスタリングによって生成されたクラスタ数が，実在するプロセッサ数以下である場合，各クラスタを各プロセッサへ割り当てればよい．しかしながら，クラスタ数が存在するプロセッサ数よりも大きくなれば，クラスタ集約[7),8)] といった，クラスタ数を減らすための手法が必要である．本論文では，クラスタ集約を「タスククラスタリングによって生成されたクラスタどうしを集約する手続き」と定義する．

図 1 に，タスククラスタリングとクラスタ集約の例を示す．初期状態の DAG のスケジュール長はクリティカルパス長，すなわち $n_1^0 \rightarrow n_3^0 \rightarrow n_5^0 \rightarrow n_7^0$ の経路長で 23 である．図 1 (b) は，タスククラスタリング後の DAG を示す．図 1 (b) では各クラスタが linear となっているため，各タスクのスケジューリングは必要ない．したがって図 1 (b) におけるスケジュール長は一意に決まり，20 である．また，クラスタ数を 2 へと減らす（たとえばプロセッサ数が 2 であるという理由で）場合は，図 1 (c)，(d)，(e) のようなクラスタ集約が必要である．図 1 (b) で生成された各クラスタについて，クラスタ 1（$cls_4(1)$）を基点として LB[8)] による集約基準でクラスタ 2（$cls_4(2)$）を集約すると，図 1 (c)，(d)，(e) のような n_2^5 と n_3^5，および n_2^5 と n_5^5 のような先行関係のないタスクどうしが存在することになる．その結果，図 1 (c) の $cls_5(1)$ のように $n_2^5 \rightarrow n_3^5 \rightarrow n_5^5$ の実行順ではスケジュール長が 23，図 1 (d) の $cls_5(1)$ のように $n_3^5 \rightarrow n_2^5 \rightarrow n_5^5$ の実行順では n_2^5 の実行開始時刻が増加することによって $e_{2,7}^5$ の n_7^5 における到着時刻が遅れ，スケジュール長が 24 となっている．さらに図 1 (e) では，n_2^5 を $cls_5(2)$ において最後にスケジュールすることにより，図 1 (c)，(d) よりもスケ

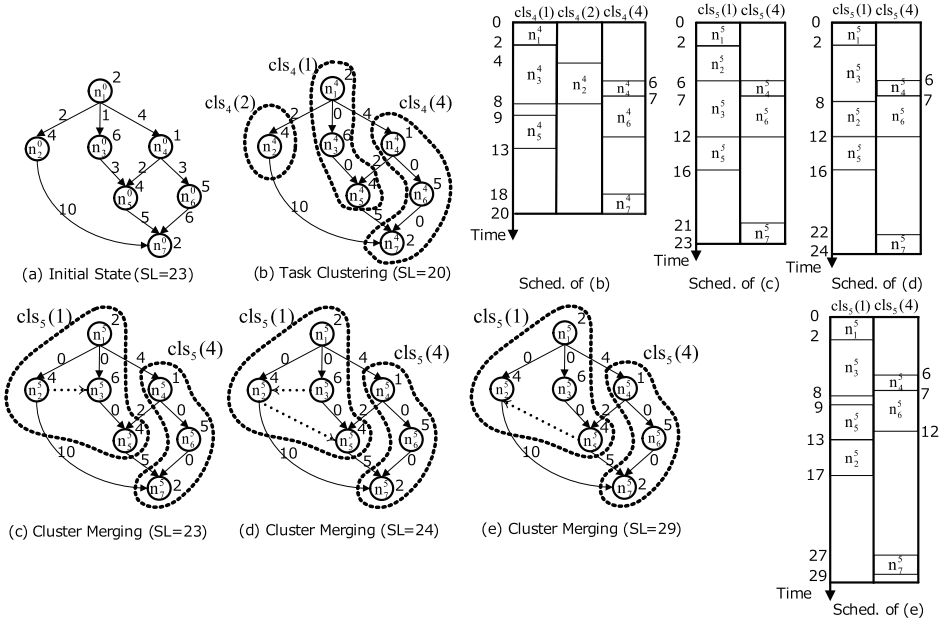


図 1　タスククラスタリングとクラスタ集約によるスケジュール長の決定（SL：スケジュール長）
Fig. 1　Derivation of schedule length by task clustering and cluster merging (SL: schedule length).

ジュール長が大きくなってしまう．これらの例から，タスククラスタリングの後にクラスタ集約を行うと同時実行が可能なタスク数の減少によりスケジュール長が大きくなる可能性がある．その一方で，図 1 (c)，(d)，(e) のように，同じクラスタ構造（各クラスタに属するタスク集合が同一）であっても，タスク実行順によってスケジュール長が変動することが分かる．

3. 問 題 定 義

3.1　従来手法とその問題点

従来のタスククラスタリング[3),4),6),9),11)–15)] では，主にスケジュール長を最小化できるときのクラスタ数の算出を目的としている．タスク集約基準としてはタスク集約によってスケジュール長が増加すれば集約せず，そうでなければ集約は受け入れられるという発見的アルゴリズム[4),9),11),12)] や，集約対象となるタスク組を特定して集約するもの[14)]，最適化手法に基づくもの[13),15)] がある．これらはいずれも生成されるクラスタ数に制限を設けておらず，DAG によっては得られるクラスタ数が膨大なものとなる場合がある．タスククラスタリングの目的がスケジュール長の最小化であれば，生成されたクラスタ数が多くなればなるほど，各プロセッサのスピードアップ率（= 1 プロセッサでのスケジュール長/生成されたクラスタによるスケジュール長）への寄与が小さいことにつながる．

クラスタ集約については，文献 7) では Cluster Merging（CM）を行う手法について述べている．CM ではプロセッサ数と同数のクラスタが生成されるまでクラスタサイズを均等化するという基準でクラスタ集約を行うが，タスク間の先行関係は無視している．その結果，互いに先行関係のないタスクどうしが 1 クラスタに属することになり，図 1 (c)，(d)，(e) のようにスケジュール長が大きくなる．また，文献 8) で述べられている Load Balancing（LB），Communication Traffic Minmizing（CTM）は，タスククラスタリングアルゴリズムである CASS-II[11)] の後に行われるクラスタ集約手法である．この 2 手法では，クラスタ間で少なくとも先行関係を持つタスクが存在するときに，クラスタどうしを集約する．しかしながら，LB および CTM はタスククラスタリング後の処理であるため，複数タスクが属するクラスタどうしを集約することになる．その結果，互いに先行関係のないタスクが 1 つのクラスタに属する場合があり，図 1 (c)，(d)，(e) のようにスケジュール長が大きくなる場合がある．そのため，DAG に対する実行プロセッサ数の決定を目的とした従来手法には，以下の 2 つの問題がある．

- タスククラスタリングにより，膨大な数のプロセッサが必要となる．
- クラスタ集約によるスケジュール長の増分を最小化する（またはスケジュール長を短縮させる）ための基準がない．

3.2　提 案 内 容

本論文では任意の DAG のスケジュール長を短縮し，かつ少ないプロセッサ数でスケジュールするために，以下の 2 つを述べる．

(1)　**スケジュール長が短縮しうるクラスタサイズの下限値 δ_{opt} の算出**
図 1 (c)，(d) により，クラスタ集約を行うと互いに先行関係のないタスクが同一クラスタに含まれスケジュール長が短縮しない場合があるため，少ないクラスタ数でできるだけスケジュール長を短縮させるための基準が必要である．本論文ではクラスタ数を制限するために各クラスタサイズの下限値 δ を設け，δ がどのような値であればスケジュール長を短縮できる可能性があるかを考察する．そして，スケジュール長が短縮しうるクラスタサイズの下限値 δ_{opt} を決定する．

(2)　**各クラスタサイズが δ_{opt} 以上となるまでタスク集約を行う，タスク集約方針**

δ_{opt} を決定しても，タスク集約方針（各クラスタに属するタスクの組合せ）およびタスク実行順によってもスケジュール長が変動する．本論文では，各クラスタサイズが δ_{opt} となるまでタスク集約を行いつつ，スケジュール長を短縮させることを目的としたタスククラスタリングについて述べる．

4. クラスタサイズの下限値算出

3.2 節 (1) の詳細について述べる．まず，G_{org} に対して S 回のタスク集約を行った状態 G^S_{cls} について，以下の条件を満たしているものとする．

$$w(cls_S(i)) \geq \delta \text{ for } \forall cls_S(i) \in V^S_{cls}. \tag{6}$$

式 (6) では，全クラスタサイズが δ 以上となっている状態である．また，S 未満のタスク集約回数では，少なくとも 1 つのクラスタサイズは δ 未満である状態とする．以降，2.2 節で用いた s の範囲を，$0 \leq s \leq S$ と仮定する．ここでは，δ がどのような値であればスケジュール長が短縮されうるのかを考察する．

4.1 クラスタサイズの下限値を決めるための方針

V^s_{cls} については，各クラスタ内のタスク実行順によってスケジュール長が変わりうる．また，スケジュール方針を決めたとしても各クラスタに属するタスクの組合せによってスケジュール長は変わりうる（いずれの問題も NP 困難[9] である）だけでなく，クラスタサイズの下限値が決まらないとクラスタは生成できない．すなわちタスククラスタリング前では各クラスタ内のタスクの組合せ，タスク実行順，各タスクにおけるデータ受信待ち時間（式 (4) で定義）は不明であるので，クラスタサイズの下限値を決定する段階ではスケジュール長が不明である．そのため，クラスタサイズの下限値を変動させることによってスケジュール長に影響を及ぼすような指標を決める必要がある．そこで各タスクについて，先行関係のないタスクを先に実行させることによってできるだけ遅くスケジュールし，かつ他クラスタからのデータ受信待ち時間（式 (4) で定義）を無視した場合の，START タスク開始から END タスク完了までの所要時間を $sl_w(G^s_{cls})$ とする（4.2 節で定義）．そして，$sl_w(G^S_{cls})$ の上限値が最小化されるときのクラスタサイズの下限値を δ_{opt}（4.7 節で定義）とする（S は式 (6) で定義）．さらに 4.8 節において，$sl_w(G^S_{cls})$ の増減がスケジュール長に対してどのような影響を及ぼすのかを示す．

4.2 $sl_w(G^s_{cls})$ の定義

表 1 に各種変数の定義を示す．まず，$cls_s(i)$ においてどのタスクが最も先に実行されるか（$top_s(i)$），$cls_s(i)$ においてどのタスクが他クラスタとのデータ通信を要するか（$in_s(i)$，$out_s(i)$），$cls_s(i)$ においてどのタスクが最後に実行されるか（$btm_s(i)$）を定義する．$top_s(i)$ を $cls_s(i)$ に属するタスクのうち，その直接先行タスクがすべて他のクラスタに属するものとする．すなわち $cls_s(i)$ で最も先に実行されるタスクは，$top_s(i)$ に属するタスクとなる．また，$in_s(i)$，$out_s(i)$ をそれぞれ少なくとも 1 つの直接先行タスクが他クラスタに属するタスク，少なくとも 1 つの直接後続タスクが他クラスタに属するタスクとする．そして $btm_s(i)$ をすべての直接後続タスクが他クラスタに属するようなタスクとする．

$desc(n^s_k, i)$ を，$cls_s(i)$ の中で n^s_k の後に必ず実行されるタスクの集合と n^s_k 自身の和集合とする．さらに $S(n^s_k, i)$ を，$cls_s(i)$ において n^s_k の実行順をできるだけ後にした場合の，n^s_k より先に実行可能なタスクサイズの和とする．

次に，各タスクが $cls_s(i)$ においてできるだけ後にスケジュールされる時刻 $tlevel(n^s_k)$ を定義する．$n^s_k \in top_s(i)$ のとき，直接先行タスクはすべて他クラスタに属しているため，こ

表 1　$sl_w(G^s_{cls})$ に関する定義（$n^s_k \in cls_s(i)$ とする）
Table 1　Parameter definition which is related to $sl_w(G^s_{cls})$ ($n^s_k \in cls_s(i)$).

Parameter	Definition
$top_s(i)$	$\{n^s_k \mid \forall n^s_l \in pred(n^s_k) \text{ s.t., } n^s_l \notin cls_s(i)\} \cup \text{START Tasks} \in cls_s(i)$.
$in_s(i)$	$\{n^s_k \mid \exists n^s_l \in pred(n^s_k) \text{ s.t., } n^s_l \notin cls_s(i)\} \cup \text{START Tasks} \in cls_s(i)$.
$out_s(i)$	$\{n^s_k \mid \exists n^s_l \in suc(n^s_k) \text{ s.t., } n^s_l \notin cls_s(i)\} \cup \text{END Tasks} \in cls_s(i)$.
$btm_s(i)$	$\{n^s_k \mid \forall n^s_l \in suc(n^s_k), \text{ s.t., } n^s_l \notin cls_s(i)\} \cup \text{END Tasks} \in cls_s(i)$.
$desc(n^s_k, i)$	$\{n^s_l \mid n^s_k \prec n^s_l,\ n^s_l \in cls_s(i)\} \cup \{n^s_k\}$
$S(n^s_k, i)$	$\sum_{n^s_l \in cls_s(i)} w(n^s_l) - \sum_{n^s_l \in desc(n^s_k, i)} w(n^s_l)$
$tlevel(n^s_k)$	$\begin{cases} \max\limits_{n^s_l \in pred(n^s_k)} \{tlevel(n^s_l) + w(n^s_l) + c(e_{l,k})\}, \text{ where } n^s_k \in top_s(i), \\ TL_s(i) + S(n^s_k, i), \text{ otherwise.} \end{cases}$
$TL_s(i)$	$\max\limits_{n^s_k \in top_s(i)} \{tlevel(n^s_k)\}$
$blevel(n^s_k)$	$\max\limits_{n^s_l \in suc(n^s_k)} \left\{w(n^s_k) + c(e^s_{k,l}) + blevel(n^s_l)\right\}$
$level(n^s_k)$	$tlevel(n^s_k) + blevel(n^s_k)$
$BL_s(i)$	$\max\limits_{n^s_k \in out_s(i)} \{S(n^s_k, i) + blevel(n^s_k)\}$
$LV_s(i)$	$TL_s(i) + BL_s(i) = \max\limits_{n^s_k \in cls_s(i)} \{level(n^s_k)\}$
$sl_w(G^s_{cls})$	$\max\limits_{cls_s(i) \in V^s_{cls}} \{LV_s(i)\}$

の場合の $tlevel(n_k^s)$ は直接先行タスクからすべてのデータが到着する時刻である．そして $TL_s(i)$ を $n_k^s \in top_s(i)$ における，$tlevel(n_k^s)$ の最大値とする．すなわちある $n_k^s \in top_s(i)$ について $TL_s(i) = tlevel(n_k^s)$ とすると，$top_s(i)$ に属するタスクのうちで n_k^s 以外のものは，n_k^s の実行完了までは実行されずに待たされるという状況を意味する．

次に，$n_k^s \notin top_s(i)$ である場合の $tlevel(n_k^s)$ について述べる．式 (4) より，各タスク n_k^s について，そのデータ受信待ち時間 $I(n_k^s, i)$（式 (4) で定義）はスケジュール方針（すなわち全タスクの実行順）によって決まる．すなわち，タスククラスタリング前では $I(n_k^s, i)$ は不明である．そこで，$top_s(i)$ に属さないタスク n_k^s においては $tlevel(n_k^s) = TL_s(i) + S(n_k^s, i)$ とする．$blevel(n_k^s)$ を，n_k^s から END タスクまでの経路長の最大値とする．すなわち $blevel(n_k^s)$ は，n_k^s から END タスクまでに実行されるタスクのうち，先行関係のあるもののみを実行させた場合の所要時間の最大値である．$BL_s(i)$ を，$cls(i)$ のあるタスクの実行開始時刻から END タスク実行完了までの所要時間，すなわち $S(n_k^s, i)$ と $blevel(n_k^s)$ との和の最大値とする．そして $LV_s(i)$ を $TL_s(i)$ と $BL_s(i)$ の和とする．$level(n_k^s) = tlevel(n_k^s) + blevel(n_k^s)$ とすれば，

$$\begin{aligned} LV_s(i) &= TL_s(i) + BL_s(i) = TL_s(i) + \max_{n_k^s \in out_s(i)} \{S(n_k^s, i) + blevel(n_k^s)\} \\ &= \max_{n_k^s \in out_s(i)} \{TL_s(i) + S(n_k^s, i) + blevel(n_k^s)\} \\ &= \max_{n_k^s \in cls_s(i)} \{level(n_k^s)\} \end{aligned} \tag{7}$$

である．

各クラスタ $cls_s(i) \in V_{cls}^s$ について，$LV_s(i)$ の最大値を $sl_w(G_{cls}^s)$ とする．どのクラスタのどのタスクを最後に実行すれば各クラスタの LV 値が最大値をとるかを決定することにより，$sl_w(G_{cls}^s)$ を算出できる．

例 4.2.1. 図 2 に，$sl_w(G_{cls}^5)$ 導出の例を示す．この図で示されている DAG は，図 1 (c), (d), (e) と同一である．この場合 $LV_5(1)$ は $level(n_2^5)$ であり，$LV_5(4)$ は $level(n_4^5)$ である．すなわち，$cls_5(1)$ では n_2^5 が最も後にスケジュールされる場合，スケジュール長が大きくなるということである．一方，$cls_5(4)$ は linear であるので $cls_5(4)$ 内のタスク実行順は一意である．さらに $LV_5(1) = 28$，$LV_5(4) = 20$ であるから，$sl_w(G_{cls}^5) = LV_5(1)$ となっている．データ受信待ち時間（式 (4) で定義）を無視すれば，$cls_5(1)$ の中で n_2^5 が後にスケジュールされるときが最大のスケジュール長となることがいえる．このことは図 1 (e) に該当する．図 1 (e) では $sl(G_{cls}^5) = 29$ であるのに対し，$sl_w(G_{cls}^5)$ では n_5 が n_4^5 からのデータ受信待ち時間（図 1 (e) で，$9 - 8 = 1$ 単位時間）だけ無視しているので，$sl_w(G_{cls}^5) = 28$ である．■

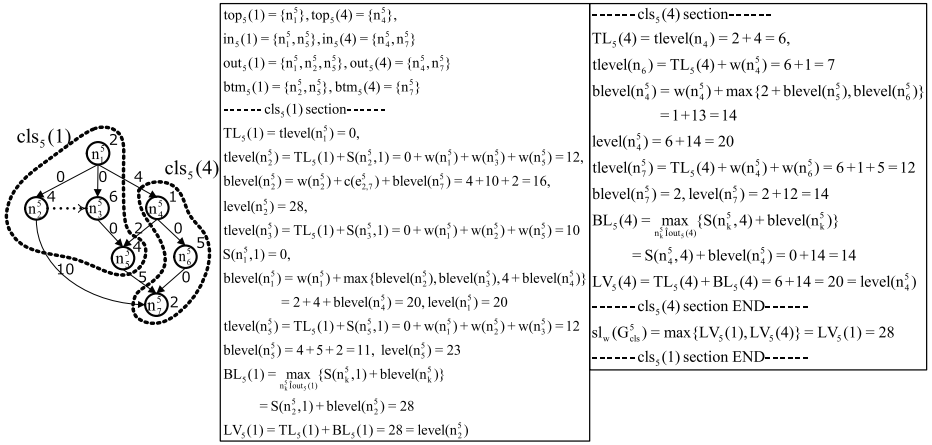


図 2　各種定義の例
Fig. 2　Example of each defined symbols.

4.3　$sl_w(G_{cls}^s)$ の解析のための準備

$sl_w(G_{cls}^S)$ は S 回のタスク集約後に決まる値のため（S は，式 (6) で定義），タスククラスタリング前では $sl_w(G_{cls}^S)$ の値は不明である．そこで，タスククラスタリング後のクラスタの構造（クラスタ内のタスク間の先行関係，タスク集合）に応じて $sl_w(G_{cls}^S)$ の上限値，すなわち $sl_w(G_{cls}^S) - sl_w(G_{org})$ の上限値がどのような値となるのかを調べる．この値が最小化されるときのクラスタサイズの下限値を δ_{opt} を求める．さらに $sl_w(G_{cls}^S)$ の値をどうすれば（短縮するのか，それとも大きくなるのか）スケジュール長が短縮されうるのかを明らかにすることにより，タスククラスタリングの目的を決める．そこで，まずは $sl_w(G_{cls}^S) - sl_w(G_{org})$ の上限値算出に用いる各種定義について述べる．

表 2 に，各種変数の定義を示す．まず，s（$s \leq S$）回のタスク集約の後，$sl_w(G_{cls}^s)$ を規定する実行経路に含まれるタスク集合を $seq_{max,s}$ とする（S は，式 (6) で定義）．すなわち $sl_w(G_{cls}^s) = LV_S(i) = level(n_k^S)$ を満たす n_k^S について，$tlevel(n_k^S)$ を決定するタスク集合と，$blevel(n_k^S)$ を決定する経路に属するタスク集合との和集合である（例 4.3.1 で，具体例をあげる）．さらに，$seq_{max,s}$ 内のタスク集合の中で先行関係にあるものはどの経路 p（タ

表 2　$sl_w(G^s_{cls})$ の解析に用いる定義（$0 \leq s \leq S$ とする）
Table 2　Parameter definitions which are used in analysis on $sl_w(G^s_{cls})$ $(0 \leq s \leq S)$.

Parameter	Definition
$seq_{max,s}$	Set of tasks by which $sl_w(G^s_{cls})$ is derived.
p	One path of G^0_{cls}, i.e., $\{n^0_0, n^0_1, n^0_2, \ldots, n^0_k\} \cup \{e^0_{0,1}, e^0_{1,2}, \ldots e^0_{k-1,k}\}$, by which a sequence $<n^0_0, n^0_1, n^0_2, \ldots, n^0_k>$ is constructed, where $e^0_{l-1,l} \in E_0$, n^0_0 is a START task and n^0_k is an END task.
p'	One subpath of G^0_{cls}, i.e., $\{n^0_0, n^0_1, n^0_2, \ldots, n^0_k\} \cup \{e^0_{0,1}, e^0_{1,2}, \ldots e^0_{k-1,k}\}$, by which a sequence $<n^0_0, n^0_1, n^0_2, \ldots, n^0_k>$ is constructed, where $e^0_{l-1,l} \in E_0$, n^0_0 is not a START task or n^0_k is not an END task.
$seq^{\prec}_{max,s}$	One path in which every task belongs to $seq_{max,s}$.
$seq^{\prec}_{max,s}(i)$	Set of subpaths in each of which every task in $cls(i)$ belongs to $seq^{\prec}_{max,s}$.
w_{max}	$\max\limits_{cls_S(i) \in V^S_{cls}} \{w(cls_S(i))\} - \delta$
ϕ	♯ of clusters in which at least one task belongs to $seq^{\prec}_{max,s}$.
cp	Critical path length of G^0_{cls}.
cp_w	$\max\limits_{p \in G^0_{cls}} \left\{ \sum\limits_{n^0_k \in p} w(n^0_k) \right\}$.
$g_{\min}{}^{2),6)}$	$\min\limits_{n^0_k \in V^0_{cls}} \left\{ \dfrac{\min\limits_{n^0_j \in pred(n^0_k)} \left\{ w(n^0_j) \right\}}{\max\limits_{n^0_j \in pred(n^0_k)} \left\{ c(e^0_{j,k}) \right\}}, \dfrac{\min\limits_{n^0_l \in suc(n^0_k)} \left\{ w(n^0_l) \right\}}{\max\limits_{n^0_l \in suc(n^0_k)} \left\{ c(e^0_{k,l}) \right\}} \right\}$.
$g_{\max}(n^s_k)$	$\max \left\{ \dfrac{\max\limits_{n^0_j \in pred(n^0_k)} \left\{ w(n^0_j) \right\}}{\min\limits_{n^0_j \in pred(n^0_k)} \left\{ c(e^0_{j,k}) \right\}}, \dfrac{\max\limits_{n^0_l \in suc(n^0_k)} \left\{ w(n^0_l) \right\}}{\min\limits_{n^0_l \in suc(n^0_k)} \left\{ c(e^0_{k,l}) \right\}} \right\}$.
Δsl_w	$sl_w(G^S_{cls}) - sl_w(G_{org}) = sl_w(G^S_{cls}) - cp$.
$\Delta sl_{w,up}$	An upper bound of Δsl_w.
$len(seq^{\prec}_{max,s}(i))$	$\sum\limits_{n^s_k \in seq^{\prec}_{max,s}(i)} w(n^s_k) + \sum\limits_{\substack{n^s_k, n^s_l \in seq^{\prec}_{max,s}(i), \\ e^0_{k,l} \in E_0}} c(e^0_{k,l})$.
$len(seq^{\prec}_{max,s})$	$\sum\limits_{n^s_k \in seq^{\prec}_{max,s}} w(n^s_k) + \sum\limits_{\substack{n^s_k, n^s_l \in seq^{\prec}_{max,s}, \\ e^0_{k,l} \in E_0}} c(e^0_{k,l})$.
ΔL_i	$\sum_{n^S_k \in cls_S(i) \cap seq_{max,S}} w(n^S_k) - len(seq^{\prec}_{max,S}(i))$.
ΔL_w	$sl_w(G^S_{cls}) - len(seq^{\prec}_{max,S})$.
$\Delta L_{w,up}$	An upper bound of ΔL_w.

スク集合と辺の集合で，表 2 で定義）か（$seq^{\prec}_{max,s}$）を定義する．$seq^{\prec}_{max,s}$ は 1 つの経路 p で，その中の要素 n^s_k，n^s_l および $e^s_{k,l}$ が，下記の条件を満たすものである．

$n^s_k \in seq_{max,s}$，$n^s_l \in seq_{max,s}$，$n^s_k \in pred(n^s_l)$.

$seq^{\prec}_{max,s}$ に属するのは互いに先行関係のあるタスクとこれらタスク間の辺であるので，$seq_{max,s}$ に属するタスクのうち，$seq^{\prec}_{max,s}$ は複数存在しうる（たとえば $seq_{max,s}$ を $\{n^s_1, n^s_2, n^s_3, n^s_4\}$ とすると，$\{n^s_1, n^s_2, n^s_4\} \cup \{e^s_{1,2}, e^s_{2,4}\}$ と $\{n^s_1, n^s_3, n^s_4\} \cup \{e^s_{1,3}, e^s_{3,4}\}$ といった 2 つの経路が存在すれば，これら 2 経路がそれぞれ $seq^{\prec}_{max,s}$ である）．

$seq^{\prec}_{max,s}(i)$ を $seq^{\prec}_{max,s}$ に属するタスクの中で，$cls_s(i)$ に属するものとそれらタスク間の辺との和集合とする．すなわち $seq^{\prec}_{max,s}(i)$ は，1 つの $seq^{\prec}_{max,s}$ における，以下の条件を満たす部分経路 p'（表 2 で定義）のうち，その中の要素 n^s_k，n^s_l が以下の条件を満たすものの和集合である（すなわち部分経路の和集合）．

$n^s_k \in seq^{\prec}_{max,s}$，$n^s_l \in seq^{\prec}_{max,s}$，$n^s_k \in pred(n^s_l)$，$n_k$, $n_l \in cls(i)$.

$seq^{\prec}_{max,s}(i)$ は部分経路の集合であり，$seq^{\prec}_{max,s}$ の部分集合でもある．そのため，1 つの $seq^{\prec}_{max,s}$ について，$seq^{\prec}_{max,s}(i)$ は 1 つのみ存在する．また，$seq^{\prec}_{max,s}$ が複数存在すれば，$seq^{\prec}_{max,s}(i)$ も複数存在する．G_{org}，すなわち G^0_{cls} のスケジュール長はクリティカルパス長であり，cp と表す．また，各経路 p においてタスクサイズの総和をそれぞれ算出し，その中の最大値を cp_w とする．$g_{max}(n^0_k)$ を n^0_k の粒度[6] の最大値，g_{min} を G^0_{cls} 内のタスク粒度の最小値[6] とする．

例 4.3.1. $seq_{max,s}$，$seq^{\prec}_{max,s}$ の意味を例をあげて述べる．図 2 において，$sl_w(G^5_{cls}) = level(n^5_2)$ である．$cls_5(1)$ 内で n^5_2 の前に実行され，かつ n^5_2 と先行関係にあるタスク集合は $\{n^5_1\}$ であるから，$seq^{\prec}_{max,5}(1) = \{n^5_1, n^5_2\} \cup \{e^5_{1,2}\}$ である．また，$blevel(n^5_2) = w(n^5_2) + c(e^5_{2,7}) + blevel(n^5_7)$，$seq^{\prec}_{max,5}(4) = \{n^5_7\}$ である．よって，

$seq^{\prec}_{max,5} = \{n^5_1, n^5_2, n^5_7\} \cup \{e^5_{1,2}, e^5_{2,7}\}$

である．

一方，$cls_5(1)$ 内で n^5_2 よりも先に実行できるのは $\{n^5_1, n^5_3, n^5_5\}$ である．そして，前述のように $blevel(n^5_2) = w(n^5_2) + c(e^5_{2,7}) + blevel(n^5_7)$ であるから，$seq_{max,5} = \{n^5_1, n^5_2, n^5_3, n^5_5, n^5_7\}$ である．■

例 4.3.2. 図 3 に，$seq_{max,s}$，$seq^{\prec}_{max,s}$ 算出のさらなる例を示す．(1) では $cls_s(i)$ および $cls_s(i+1)$ が linear であり，$sl_w(G^s_{cls}) = level(n^s_3)$ の場合，(2) では $cls_s(i)$ および $cls_s(i+1)$ がいずれも linear でなく，$sl_w(G^s_{cls}) = level(n^s_9)$ である場合のクラスタ構成を示す．また，破線矢印は $sl_w(G^s_{cls})$ を構成するタスクの実行順である．(1) では

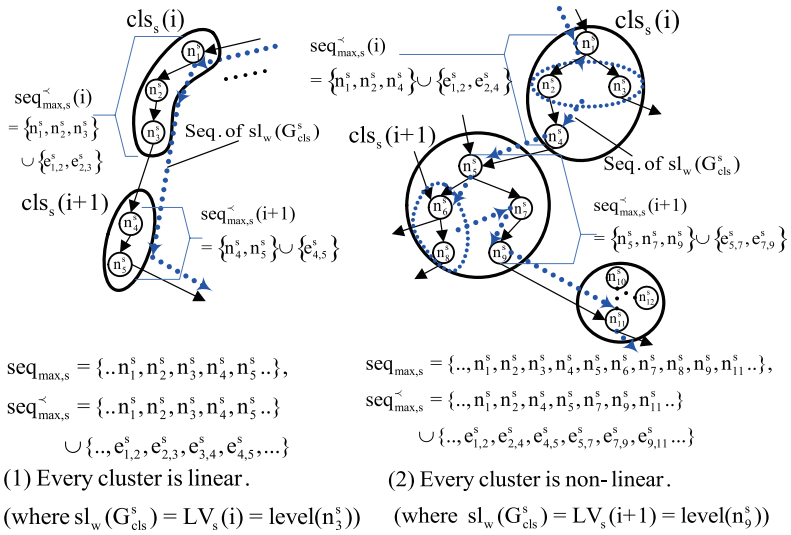


図 3　$sl_w(G^s_{cls})$ に関する定義の例
Fig. 3　Concept of upper bound of $sl_w(G^s_{cls})$.

$cls_s(i)$ および $cls_s(i+1)$ が linear のため，これら 2 つのクラスタ内タスクの実行順は一意である．その結果，$seq_{max,s}$ と $seq^{\prec}_{max,s}$ には，いずれも $\{n^s_1, n^s_2, n^s_3, n^s_4, n^s_5\}$ が属する．(2) では，破線矢印および破線枠内のタスクが $seq_{max,s}$ に属していることを示す．ここで $sl_w(G^s_{cls}) = level(n^s_9)$ とすると，$tlevel(n^s_9) = tlevel(n^s_5) + w(n^s_6) + w(n^s_8) + w(n^s_7)$，$blevel(n^s_9) = w(n^s_9) + c(e^s_{9,11}) + blevel(n^s_{11})$ である．n^s_4 は破線枠内にある n^s_2，n^s_3 の後に実行させることによってできるだけ n^s_4 の開始時刻を遅らせている（つまり $tlevel(n^s_4) = tlevel(n^s_1) + w(n^s_2) + w(n^s_3)$）．また，$n^s_1$，$n^s_2$，$n^s_4$ が先行関係にあるため，$seq^{\prec}_{max,s}(i)$ は $\{n^s_1, n^s_2, n^s_4\} \cup \{e^s_{1,2}, e^s_{2,4}\}$ である．$cls_s(i+1)$ において $seq_{max,s}$ 上にあり，かつ先行関係のあるものは n^s_5，n^s_7，n^s_9 であるので，$seq^{\prec}_{max,s}(i+1) = \{n^s_5, n^s_7, n^s_9\} \cup \{e^s_{5,7}, e^s_{7,9}\}$ である．

■

4.4　δ と $sl_w(G^S_{cls})$ の関係

2.1 節で述べたように，G^0_{cls} においては各クラスタに 1 タスクのみが属しているので，$G^0_{cls} = G_{org}$ のスケジュール長は cp である．さらに $G^0_{cls} = G_{org}$ では $\{n^0_k\} = cls_0(i) \in V^0_{cls}$ であるから，式 (7) より，

$$sl_w(G_{org}) = sl_w(G^0_{cls}) = \max_{cls_0(i) \in V^0_{cls}} \{LV_0(i)\}$$
$$= \max_{n^0_k \in V_0} \left\{level(n^0_k)\right\} = \max_{n^0_k \in V_0} \left\{tlevel(n^0_k) + blevel(n^0_k)\right\} \quad (8)$$

である．表 1 の $tlevel(n^S_k)$，$(n^S_k \in top_S(i))$ と $blevel(n^S_k)$ の定義により，式 (8) において

$$sl_w(G_{org}) = cp = sl(G_{org}) \quad (9)$$

が成り立つ．したがって，$sl_w(G^S_{cls})$ と $sl_w(G_{org})$ との差を Δsl_w とすると（表 2 で定義），Δsl_w は $sl_w(G^S_{cls})$ と cp との差と等しい（表 2 で定義）．しかしながら，4.3 節で述べたように各クラスタ内のタスク組はタスク集約方針によって決まるので，タスククラスタリング前では Δsl_w は算出できない．そこで，Δsl_w の上限値 $\Delta sl_{w,up}$（表 2 で定義）を算出する．

まず，$seq_{max,S}$ 内のタスクが属する 1 つのクラスタを $cls_S(i)$ とする．$cls_S(i), w(cls_S(i)) \geq \delta$ である $cls_S(i)$ が生成されることによって，$cls_S(i)$ の中で，$seq_{max,S}$ に属するが $seq^{\prec}_{max,S}(i)$ には属さないタスクのタスクサイズ和と，$seq^{\prec}_{max,S}(i)$ で局所化された通信のデータサイズ和との差を ΔL_i とする（表 2 で定義）．

すなわち ΔL_i は，$cls_S(i)$ において，$seq^{\prec}_{max,S}(i)$ 内のタスクと先行関係のないタスクのタスクサイズの和と，局所化された $seq^{\prec}_{max,S}(i)$ 内のデータサイズとの差によって得られる．そして，$seq^{\prec}_{max,S}(i)$ に属するタスクのタスクサイズ和と，それらタスク間の，G_{org} における通信のデータサイズ和との総和を $len(seq^{\prec}_{max,S}(i))$ とする．さらに，$seq^{\prec}_{max,S}$ に属するタスクのタスクサイズ和と，それらタスク間の，G^0_{cls} における通信のデータサイズ和との総和を $len(seq^{\prec}_{max,S})$ とする．4.3 節で述べたように，$seq^{\prec}_{max,S}$ は複数存在しうる．すなわち，$seq^{\prec}_{max,S}$ は G_{org} におけるクリティカルパスとは限らないので，

$$len(seq^{\prec}_{max,S}) \leq cp \quad (10)$$

が成り立つ．そして ΔL_w を $sl_w(G^S_{cls})$ と $len(seq^{\prec}_{max,S})$ との差（表 2 で定義）とすると，

$$\Delta sl_w = sl_w(G^S_{cls}) - cp \leq sl_w(G^S_{cls}) - len(seq^{\prec}_{max,S}) = \Delta L_w,$$
$$\Leftrightarrow \Delta sl_w \leq \Delta L_w \quad (11)$$

が成り立つ．ΔL_w の上限値を $\Delta L_{w,up}$（表 2 で定義）とすれば，$\Delta L_{w,up}$ は Δsl_w の上限値でもある．そこで，$\Delta L_{w,up}$ を求め，そして $\Delta sl_{w,up} = \Delta L_{w,up}$ とする．

次に，$\Delta L_{w,up}$ の算出方針について述べる．まず，ΔL_i の上限値を求める．この値を $seq_{max,S}$ 内のタスクが属するクラスタ数分だけ加算することにより，$\Delta L_{w,up}$ を求める方針とする．クラスタ内で先行関係のないタスク数が増加すればするほど，top_S 以外のタスクの $tlevel$ 値が大きくなりうる．その結果，$seq^{\prec}_{max,S}$ 内のタスクが属する各クラスタが linear である

か否かで ΔL_i の上限値が異なるものと考えられる．そのため，$seq^{\prec}_{max,S}$ 内のタスクが属する各クラスタのうち，linear であるクラスタ $cls_S(i)$ について ΔL_i の上限値と，linear でない各クラスタ $cls_S(j)$ について ΔL_j の上限値をそれぞれ求める．そして，$seq_{max,S}$ 内のタスクが属するクラスタ数のうち，linear であるクラスタ数とそうでないクラスタ数分だけ ΔL_i の上限値および ΔL_j の上限値を加算して，$\Delta L_{w,up}$ を求める．

例 4.4.1. 図 3 (1) において $s = S$ とすると，

$$\Delta L_i = w(n^s_1) + w(n^s_2) + w(n^s_3) - len(seq^{\prec}_{max,s}(i)) = -(c(e^s_{1,2}) + c(e^s_{2,3})),$$
$$\Delta L_{i+1} = w(n^s_4) + w(n^s_5) - len(seq^{\prec}_{max,s}(i+1)) = -c(e^s_{4,5}).$$

である．一方，図 3 (2) において $s = S$ とすると，

$$\Delta L_i = \sum_{k=1}^{4} w(n^s_k) - len(seq^{\prec}_{max,s}(i)) = w(n^s_3) - (c(e^s_{1,2}) + c(e^s_{2,4})),$$
$$\Delta L_{i+1} = \sum_{k=5}^{9} w(n^s_k) - len(seq^{\prec}_{max,s}(i+1)) = w(n^s_6) + w(n^s_8) - (c(e^s_{5,7}) + c(e^s_{7,9})).$$

である．■

4.5　ΔL_i の上限値の算出

まず，以下の (1)，(2) の場合に分けて，linear であるクラスタとそうでないクラスタにおける ΔL_i の上限値を求める．

(1)　$seq^{\prec}_{max,S}$ 内のタスクが属する 1 クラスタ $cls_S(i)$ が linear であるとき

$cls_S(i)$，$w(cls_S(i)) \geq \delta$ を生成することにより，$seq^{\prec}_{max,S}(i)$ に属するタスク間のデータ転送が局所化される．また，

$$\begin{aligned}
&\psi_i = seq^{\prec}_{max,S}(i) \cup \{n^S_{START(i+1)} \,|\, n^S_{END(i)} \in seq^{\prec}_{max,S}(i),\\
&n^S_{END(i)} \not\prec n^S_k \text{ for } \forall n^S_k \in seq^{\prec}_{max,S}(i),\\
&n^S_{START(i+1)} \in seq^{\prec}_{max,S}(i+1),\ n^S_{START(i+1)} \in suc(n^S_{END(i)}),\\
&e^S_{END(i),START(i+1)} \notin seq^{\prec}_{max,S}(i),\\
&e^S_{END(i),START(i+1)} \notin seq^{\prec}_{max,S}(i+1)\}
\end{aligned} \tag{12}$$

と定義する．すなわち式 (12) において，$n^S_{END(i)}$ は $seq^{\prec}_{max,S}(i)$ 内のタスクで最後に実行されるタスクである．また，$n^S_{END(i)}$ と $n^S_{START(i+1)}$ 間の通信 $e^S_{END(i),START(i+1)}$ は局所化されていないことを意味する（たとえば**図 4** の場合では，$n^s_{END(i)} = n^s_6$，$n^s_{START(i+1)} = n^s_9$ である）．

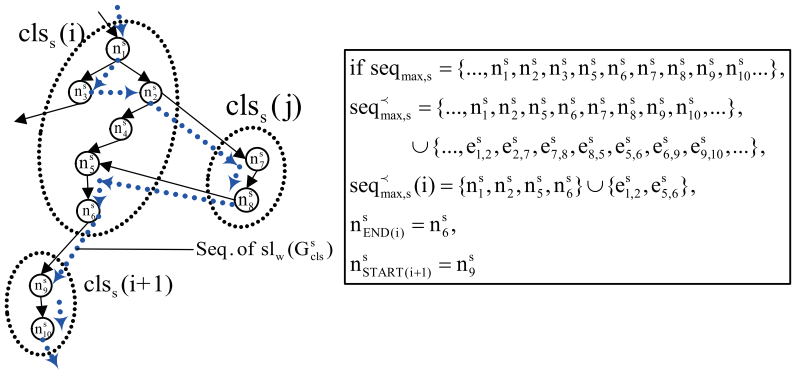


図 4　$n^s_{END(i)}$ と $n^s_{START(i)}$ の例
Fig. 4　An example of $n^s_{END(i)}$ and $n^s_{START(i)}$.

ΔL_i は

$$\Delta L_i = -\sum_{e^S_{m,n} \in seq^{\prec}_{\max,S}(i)} c(e^S_{m,n}) \tag{13}$$

となる．また，表 2 の $g_{max}(n^S_k)$ を式 (13) に適用すると，

$$\begin{aligned}
-\Delta L_i &= \sum_{e^S_{m,n} \in seq^{\prec}_{max,S}(i)} c(e^S_{m,n}) \geq \sum_{\substack{n^S_m, n^S_n \in seq^{\prec}_{max,S}(i),\\ n^S_n \in suc(n^S_m)}} \frac{w(n^S_n)}{g_{\max}(n^S_m)}\\
&= \sum_{\substack{n^S_k \in seq^{\prec}_{max,S}(i),\\ n^S_l \in \psi_i, n^S_l \in suc(n^S_k)}} \frac{w(n^S_l)}{g_{\max}(n^S_k)} - \frac{w(n^S_{START(i+1)})}{g_{\max}(n^S_{END(i)})}
\end{aligned} \tag{14}$$

となる（式 (14) の ψ_i は，式 (12) で定義）．したがって，以下が成り立つ．

$$\Delta L_i \leq \frac{w(n^S_{START(i+1)})}{g_{\max}(n^S_{END(i)})} - \sum_{\substack{n^S_k \in seq^{\prec}_{max,S}(i),\\ n^S_l \in \psi_i, n^S_l \in suc(n^S_k)}} \frac{w(n^S_l)}{g_{\max}(n^S_k)}. \tag{15}$$

(2)　$seq^{\prec}_{max,S}$ 内のタスクが属する 1 クラスタ $cls_S(i)$ が linear でないとき
ここでクラスタサイズの最大値を $\delta + w_{max}$ として（w_{max} は表 2 で定義），ΔL_i の上限値を算出する．$seq^{\prec}_{max,S}(i)$ に属するタスク以外がすべて先行関係のないタスクとすると，式 (15) の結果の導出と同様にして，以下が成り立つ．

$$\begin{aligned}\Delta L_i &\le (\delta + w_{\max}) - \sum_{n_k^S \in seq^{\prec}_{max,S}(i)} w(n_k^S) - \sum_{e_{p,q}^S \in seq^{\prec}_{max,S}(i)} c(e_{p,q}^S) \\ &\le (\delta + w_{\max}) - \sum_{n_k^S \in seq^{\prec}_{max,S}(i)} w(n_k^S) \\ &\quad + \frac{w(n^S_{START(i+1)})}{g_{\max}(n^S_{END(i)})} - \sum_{\substack{n_k^S \in seq^{\prec}_{max,S}(i),\\ n_l^S \in \psi_i, n_l^S \in suc(n_k^S)}} \frac{w(n_l^S)}{g_{\max}(n_k^S)}. \end{aligned} \tag{16}$$

4.6　$\Delta L_{w,up}$ の算出

次に $\Delta L_{w,up}$ を求める．そのためには，$seq_{max,S}$ 内のタスクが属するクラスタの個数を求める必要がある．表 2 の定義より，$seq^{\prec}_{max,S}$ は互いに先行関係のあるタスク集合，すなわち 1 つの経路に属するタスク集合である．そのため，まずは 1 経路上のクラスタ数の上限値を求める．$seq^{\prec}_{max,S}(i)$ に属するタスクサイズの和が δ 以上であることが任意の $cls_S(i)$ で成り立つ場合，$seq^{\prec}_{max,S}$ 内のタスクが属するクラスタ数を ϕ（表 2 で定義），ϕ の上限値を ϕ_{max} とすると，

$$\phi \le \left\lfloor \frac{cp_w}{\delta} \right\rfloor \le \frac{cp_w}{\delta} = \phi_{max} \tag{17}$$

とできる．ここで cp_w は 1 経路に属するタスクサイズ和の最大値（表 2 で定義）である．一方，任意の $seq^{\prec}_{max,S}(i)$ について，$seq^{\prec}_{max,S}(i)$ に属するタスクサイズの和が δ 未満である場合を考える．$seq_{max,S}$ に属するタスク（かつタスクサイズが δ 未満）がすべて別クラスタに属する場合，$seq_{max,S}$ には 1 経路内のタスクが属することになる．したがって，ϕ_{max} は 1 経路に属するタスク数の最大値となる．

(a) まず，$seq^{\prec}_{max,S}(i)$ に属するタスクのサイズ和が δ 以上であることが任意の $cls_S(i)$ で成り立つ場合を考える．
linear であるクラスタ数を $\phi - y$，linear でないクラスタ数を y とする（y は 0 以上の整数）と，ΔL_w は式 (15) で得られた上限値を $\phi - y$ 個分加算したものと，式 (16) でえられた上限値を y 個分加算したものとの総和で抑えられる．便宜上，ここで式 (15) を満たすクラスタの添え字を $i = 1, 2, \ldots, \phi - y$，式 (16) を満たすクラスタの添え字を $j = \phi - y + 1, \phi - y + 2, \ldots, \phi$ とすると，

$$\begin{aligned}\Delta L_w \le &\sum_{i=1}^{\phi-y} \left(\frac{w(n^S_{START(i+1)})}{g_{\max}(n^S_{END(i)})} - \sum_{\substack{n_k^S \in seq^{\prec}_{max,S}(i),\\ n_l^S \in \psi_i, n_l^S \in suc(n_k^S)}} \frac{w(n_l^S)}{g_{\max}(n_k^S)} \right) + y(\delta + w_{\max}) \\ &- y \sum_{n_k^S \in seq^{\prec}_{max,S}(j)} w(n_k^S) \\ &+ \sum_{j=\phi-y+1}^{\phi} \left(\frac{w(n^S_{START(j+1)})}{g_{\max}(n^S_{END(j)})} - \sum_{\substack{n_k^S \in seq^{\prec}_{max,S}(i),\\ n_l^S \in \psi_i, n_l^S \in suc(n_k^S)}} \frac{w(n_l^S)}{g_{\max}(n_k^S)} \right) \end{aligned} \tag{18}$$

である．式 (18) の右辺を整理すると，

$$\begin{aligned}\Delta L_w \le &\sum_{i=1}^{\phi} \frac{w(n^S_{START(i+1)})}{g_{\max}(n^S_{END(i)})} - \sum_{i=1}^{\phi} \sum_{\substack{n_k^S \in seq^{\prec}_{max,S}(i),\\ n_l^S \in \psi_i, n_l^S \in suc(n_k^S)}} \frac{w(n_l^S)}{g_{\max}(n_k^S)} + y(\delta + w_{\max}) \\ &- y \sum_{n_k^S \in seq^{\prec}_{max,S}(j)} w(n_k^S) \end{aligned} \tag{19}$$

である．式 (19) において

$$\sum_{i=1}^{\phi} \frac{w(n^S_{START(i+1)})}{g_{\max}(n^S_{END(i)})} \le \phi_{\max} \max_{n_k^S \in V_S} \left\{ \frac{\max_{n_l^S \in V_{cls}^S} \{w(n_l^S)\}}{g_{\max}(n_k^S)} \right\},$$

$$\min_{p \in G^0_{cls}} \left\{ \sum_{\substack{n_k^0, n_l^0 \in p,\\ n_l^S \in suc(n_k^S)}} \frac{w(n_l^S)}{g_{\max}(n_k^S)} \right\} \le \sum_{i=1}^{\phi} \sum_{\substack{n_k^S \in seq^{\prec}_{max,S}(i),\\ n_l^S \in \psi_i, n_l^S \in suc(n_k^S)}} \frac{w(n_l^S)}{g_{\max}(n_k^S)},$$

$$y \sum_{n_k^S \in seq^{\prec}_{max,S}(j)} w(n_k^S) \ge 0 \tag{20}$$

であるから，

$$\Delta L_w \leq \phi_{\max} \max_{n_k^S \in V_S} \left\{ \frac{\max_{n_l^S \in V_{cls}^S} \left\{ w(n_l^S) \right\}}{g_{\max}(n_k^S)} \right\} - \min_{p \in G_{cls}^0} \left\{ \sum_{\substack{n_k^0, n_l^0 \in p, \\ n_l^S \in suc(n_k^S)}} \frac{w(n_l^S)}{g_{\max}(n_k^S)} \right\} + y(\delta + w_{\max}) = \Delta L_{w,up} \tag{21}$$

である．よって式 (11) と式 (17) の ϕ_{max} の定義より，

$$\Delta sl_{w,up} = \frac{cp_w}{\delta} \max_{n_k^S \in V_S} \left\{ \frac{\max_{n_l^S \in V_{cls}^S} \left\{ w(n_l^S) \right\}}{g_{\max}(n_k^S)} \right\} - \min_{p \in G_{cls}^0} \left\{ \sum_{\substack{n_k^0, n_l^0 \in p, \\ n_l^S \in suc(n_k^S)}} \frac{w(n_l^S)}{g_{\max}(n_k^S)} \right\} + y(\delta + w_{\max}) \tag{22}$$

とする．式 (22) において，第 2 項は G_{cls}^0 の 1 経路上に存在するデータサイズの下限値の総和の最小値であるから，δ に依存しない．このことをふまえて式 (22) を δ について微分すると，

$$\delta = \sqrt{cp_w \max_{n_k^S \in V_S} \left\{ \frac{\max_{n_l^S \in V_{cls}^S} \{ w(n_l^S) \}}{g_{\max}(n_k^S) y} \right\}} = \sqrt{cp_w \max_{n_k^0 \in V_0} \left\{ \frac{\max_{n_l^0 \in V_{cls}^0} \{ w(n_l^0) \}}{g_{\max}(n_k^0) y} \right\}} \tag{23}$$

のときに $\Delta sl_{w,up}$ は極小値をとる．式 (23) より，$y > 0$ であれば，δ については $\Delta sl_{w,up}$ に極小値が存在する．しかしながら δ を固定させて w_{max} を増加させると，$\Delta sl_{w,up}$ も増加する．■

(b) 次に，少なくとも 1 つの $seq_{max,S}^{\prec}(i)$ に属するタスクサイズ和が δ 未満である場合を考える．$seq_{max,S}^{\prec}$ 上に存在するクラスタ数を ϕ，G_{cls}^0 の各経路 p に属するタスク数の最大値を T_{max} とすると，$\phi \leq T_{max}$ となる．T_{max} は G_{org} で決まり，δ に依存しない．したがってこの場合は

$$\phi \leq T_{max} = \phi_{max} \tag{24}$$

となる（ϕ は表 2 で定義，ϕ_{max} は ϕ の上限値）．この場合，式 (19) において

$$\sum_{i=1}^{\phi} \frac{w(n_{START(i+1)}^S)}{g_{\max}(n_{END(i)}^S)} \leq T_{max} \max_{n_k^S \in V_S} \left\{ \frac{\max_{n_l^S \in V_{cls}^S} \left\{ w(n_l^S) \right\}}{g_{\max}(n_k^S)} \right\},$$

$$\min_{p \in G_{cls}^0} \left\{ \sum_{\substack{n_k^0, n_l^0 \in p, \\ n_l^S \in suc(n_k^S)}} \frac{w(n_l^S)}{g_{\max}(n_k^S)} \right\} \leq \sum_{i=1}^{\phi} \sum_{\substack{n_k^S \in seq_{max,S}^{\prec}(i), \\ n_l^S \in \psi_i, n_l^S \in suc(n_k^S)}} \frac{w(n_l^S)}{g_{\max}(n_k^S)},$$

$$y \sum_{n_k^S \in seq_{max,S}^{\prec}(j)} w(n_k^S) \geq 0 \tag{25}$$

であるから，

$$\Delta sl_{w,up} = T_{max} \max_{n_k^S \in V_S} \left\{ \frac{\max_{n_l^S \in V_{cls}^S} \left\{ w(n_l^S) \right\}}{g_{\max}(n_k^S)} \right\} - \min_{p \in G_{cls}^0} \left\{ \sum_{\substack{n_k^0, n_l^0 \in p, \\ n_l^S \in suc(n_k^S)}} \frac{w(n_l^S)}{g_{\max}(n_k^S)} \right\} + y(\delta + w_{\max}) \tag{26}$$

である．すなわちこの場合は，δ の増加にともない，$\Delta sl_{w,up}$ が増加することになる．■

4.7 δ_{opt} の決定

以上の結果より，$seq_{max,S}$ 内のタスクが属するクラスタ $cls(i)$ について，任意の $seq_{max,S}^{\prec}(i)$ 内のタスクのサイズ和が δ 以上か否かで $\Delta sl_{w,up}$ の振舞いが異なることが分かった．4.6 節（a）のとき，$y > 0$ かつ w_{max} が固定であれば，$\Delta sl_{w,up}$ は極小値をとる．しかしながら 4.6 節（b）の場合では，δ の増加にともなって $\Delta sl_{w,up}$ も増加する．このことから，s 回のタスク集約後において $seq_{max,s}^{\prec}(i)$ 内のタスクと先行関係のないタスクを集約することは，$\Delta sl_{w,up}$ の増加につながりうる．

式 (23) において，$y = 0$ のときは $seq_{max,S}^{\prec}$ 内のタスクが属するクラスタがすべて linear の場合に相当し，$\Delta sl_{w,up}$ は単調減少である．しかしながら，少なくとも 1 クラスタが linear でなくなるのは δ がどの値であるときかは，タスク集約前では分からない．$y \neq 0$ の場合，$\Delta sl_{w,up}$ が最小となるのは，δ が式 (23) の右辺と一致するときである．式 (23) の右辺において y が増加すると linear でないクラスタが多くなり，$sl_w(G_{cls}^S)$ の上限値が増加してしまう．そのため，ここでは $y = 1$ として，できるだけ $sl_w(G_{cls}^S)$ の上限値が減少している場合

を考え，δ_{opt} の値を

$$\delta_{opt} = \sqrt{cp_w \max_{n_k^0 \in V_0} \left\{ \frac{\max_{n_l^0 \in V_{cls}^0} \{w(n_l^0)\}}{g_{\max}(n_k^0)} \right\}} \tag{27}$$

とする．

4.8　$sl_w(G_{cls}^S)$ とスケジュール長との関係

ここでは $sl_w(G_{cls}^S)$ の変動が $sl(G_{cls}^S)$ にどのように影響するかを考察する．まず，文献 2) で証明されている以下の 2 つの補題をあげる．

補題 1. $cp \leq (1 + \frac{1}{g_{min}})cp_w$.

補題 2. $cp_w \leq sl(G_{cls}^S)$.

次に $\Delta sl_{w,up}$ と $sl(G_{cls}^S)$ の関係について述べる．まず，式 (11) より

$$\Delta sl_w = sl_w(G_{cls}^S) - cp \leq \Delta sl_{w,up} \tag{28}$$

である．さらに補題 1 より，式 (28) は

$$sl_w(G_{cls}^S) - \left(1 + \frac{1}{g_{min}}\right) cp_w \leq sl_w(G_{cls}^S) - cp \leq \Delta sl_{w,up} \tag{29}$$

となる．さらに補題 2 より，

$$sl_w(G_{cls}^S) - \left(1 + \frac{1}{g_{min}}\right) sl(G_{cls}^S) \leq sl_w(G_{cls}^S) - \left(1 + \frac{1}{g_{min}}\right) cp_w \tag{30}$$

である．したがって

$$\begin{aligned} &sl_w(G_{cls}^S) - \left(1 + \frac{1}{g_{min}}\right) sl(G_{cls}^S) \leq \Delta sl_{w,up} \\ &\quad \Leftrightarrow sl(G_{cls}^S) \geq \frac{sl_w(G_{cls}^S) - \Delta sl_{w,up}}{1 + \frac{1}{g_{min}}} \end{aligned} \tag{31}$$

となる．式 (22) において，$\Delta sl_{w,up}$ は，δ によって変動する．そのため，δ_{opt} を決めれば式 (22) における $\Delta sl_{w,up}$ は固定である．式 (31) から，δ_{opt} を決めたうえで $\Delta sl_{w,up}$ を最小化する（または減少させる）ようにタスクを集約することは，スケジュール長の下限値を減少させることになるといえる．

4.2 節で定義したように，$sl_w(G_{cls}^s)$ は，各クラスタ $cls_s(i)$ について $n_k^s \in in_S(i)$, $n_k^s \notin top_s(i)$ であるようなタスク n_k^s について，$tlevel(n_k^s)$ の算出には n_k^s におけるデータ待ち時間は考慮していない．また，$sl_w(G_{cls}^S)$ の上限値を最小化させるためには各クラスタ内のタスクに先行関係が存在することが必要である．この場合の $sl_w(G_{cls}^S)$ は各タスクのデータ待ち時間なしで，かつ各タスクができるだけ早く実行開始させる状況のスケジュール長である．そのため，$sl_w(G_{cls}^S)$ を減少させることがデータ受信待ちのないという理想的なスケジュール長を短縮させることになるので，式 (31) の結果につながっている．

また，$seq_{max,S}^{\prec}$ に属する辺は局所化されたものとそうでないものが存在する．さらに，$seq_{max,S}^{\prec}$ に属する辺は，クリティカルパス上の辺とは限らない．したがって，以下が成り立つ．

$$-\max_{p \in G_{cls}^0} \left\{ \sum_{\substack{n_k^0, n_l^0 \in p, \\ n_k^0 \in pred(n_l^0)}} c(e_{k,l}^0) \right\} \leq sl_w(G_{cls}^S) - cp. \tag{32}$$

さらに，$sl(G_{cls}^S) \leq cp$ である場合に限り，以下が成り立つ．

$$\begin{aligned} &sl_w(G_{cls}^S) - cp \leq sl_w(G_{cls}^S) - sl(G_{cls}^S) \\ &\Leftrightarrow -\max_{p \in G_{cls}^0} \left\{ \sum_{\substack{n_k^0, n_l^0 \in p, \\ n_k^0 \in pred(n_l^0)}} c(e_{k,l}^0) \right\} \leq sl_w(G_{cls}^S) - sl(G_{cls}^S) \\ &\Leftrightarrow sl(G_{cls}^S) \leq sl_w(G_{cls}^S) + \max_{p \in G_{cls}^0} \left\{ \sum_{\substack{n_k^0, n_l^0 \in p, \\ n_k^0 \in pred(n_l^0)}} c(e_{k,l}^0) \right\}, \text{ if } sl(G_{cls}^S) \leq cp. \end{aligned} \tag{33}$$

式 (33) より，$sl_w(G_{cls}^S)$ がクリティカルパス長以下となるという制約を課すると，$sl_w(G_{cls}^S)$ を減少させることは $sl(G_{cls}^S)$ の上限値を抑えることにつながる．

5. タスククラスタリングアルゴリズム

5.1　アルゴリズムの要件

4 章では，$sl_w(G_{cls}^S)$ の上限値が最小化されるときのクラスタサイズの下限値 δ_{opt} を算出し，$sl_w(G_{cls}^S)$ を減少させるようなタスク集約を行うことによってスケジュール長がとりうる下限値を減少させることを示した．ゆえに実際にタスク集約を行うタスククラスタリングのためには，以下の方針が必要である．

INPUT: G_{org}
OUTPUT: G^S_{cls}
Define UEX_s as a set of clusters which satisfy eq. (34).;
Define RDY_s as a set of clusters which statisfy eq. (35).;
For each $n_k \in V$, let $n^0_k \leftarrow n_k$, $cls_0(k) = \{n^0_k\}$ and put $cls_0(k)$ into V^0_{cls}.
0. Derive δ_{opt} by eq. (27).
1. $E_0 \leftarrow E$, $UEX_0 \leftarrow V^0_{cls}$, $RDY_0 \leftarrow \{cls_0(k) \mid cls_0(k) = \{n^0_k\}, pred(n^0_k) = \emptyset\}$;
2. **WHILE** $UEX_s \neq \emptyset$ **DO**
3. $pivot_s \leftarrow getPivot(RDY_s)$; /* detailed in Sec. 5.5. */
4. $target_s \leftarrow getTarget(pivot_s)$; /* Detailed in Sec. 5.6.2. */
5. $RDY_{s+1} \leftarrow clustering(pivot_s, target_s)$; /* Detailed in Sec. 5.7. */
6. **ENDWHILE**
7. **RETURN** G^S_{cls};

図 5　タスククラスタリングの処理概要
Fig. 5　Whole procedures of our proposing task clustering.

(i) 各クラスタサイズが δ_{opt} 以上となるまでタスク集約を行う

各クラスタサイズが式 (27) で定義した δ_{opt} 以上となるようにタスク集約を行うことにより，少なくとも $sl_w(G^S_{cls})$ がとりうる上限値が最小化されるようなクラスタ集合が得られることになる．

(ii) $seq^{\prec}_{max,s}$ 内のタスクに対し，先行関係のあるタスクを集約する

4.6 節（a）で得られた結果より，$sl_w(G^S_{cls})$ の上限値を減少させるためには，$seq^{\prec}_{max,s}$ 内のタスクが属する任意のクラスタ $cls_s(i)$（$w(cls_s(i)) < \delta$）に対し，$seq^{\prec}_{max,s}(i)$ 内のタスクと先行関係にあるタスクを集約することが必要である．

(iii) タスク集約により，$sl_w(G^s_{cls})$ をできるだけ最小化する

4.8 節で得られた結果より，タスク集約によって $sl_w(G^S_{cls})$ をできるだけ減少させることができればスケジュール長がとりうる下限値も減少する．さらに（ii）の要件を満たすことによって $sl_w(G^S_{cls})$ の上限値が減少するが，$sl_w(G^S_{cls})$ の実際の増分（減少分）はタスク集約をしないと分からない．そのため，タスク集約においては，（ii）のように $seq^{\prec}_{max,s}$ 内のタスクが属するクラスタを可能な限り選択し，$sl_w(G^{s+1}_{cls})$ が最も減少されうるタスクを選択して集約する．

5.2　アルゴリズムの概要

図 5 に，アルゴリズムの概要を示す．まず，初期状態の DAG を G^0_{cls} へ代入する（$G^0_{cls} \leftarrow G_{org}$）．そして V^0_{cls} における各タスクはそれぞれ，1 つのタスクから構成されるクラスタに属しているものとする．すなわち $cls_0(k) \in V^0_{cls}$ に対し，$cls_0(k) = \{n^0_k\}$ とする（$cls_0(1) = \{n^0_1\}$, $cls_0(2) = \{n^0_2\}$, ...）．そして，5.1 節（i）を満たすためにはどのクラスタが δ_{opt} 以上か否かを特定しなければならない．そこで，以下の条件を満たすクラスタの集合を UEX_s 内で保持する．

$$\{cls_s(i) \mid w(cls_s(i)) < \delta_{opt}\}. \tag{34}$$

タスク集約前では，V^s_{cls} に属する各クラスタ（このときは各クラスタには 1 タスクのみが属している）は UEX_s に属するものとする．

アルゴリズムの主要な処理は図 5 行 2 から行 6 である．5.1 節（ii），（iii）より，タスク集約によってできるだけ $sl_w(G^{s+1}_{cls})$ を減少させる（$sl_w(G^s_{cls}) > sl_w(G^{s+1}_{cls})$）ために，$seq^{\prec}_{max,s}$ 内のタスクが属し，かつクラスタサイズが δ_{opt} 未満である 2 つのクラスタを選択する．そしてそれらを 1 つのクラスタに集約することによってクラスタサイズを大きくする．ここで，RDY_s というクラスタ集合（5.4 節で定義）からある優先度に従って選択された 1 クラスタを $pivot_s$ と定義する（図 5 行 3）．そして $pivot_s$ 内のタスクと先行関係を持つタスクが属するクラスタを $target_s$（5.6 節で定義）として選択する（図 5 行 4）．さらに，$pivot_s$ と $target_s$ を集約して 1 つのクラスタとする（図 5 行 5）．タスク集約後，$target_s$ 内の全タスクは $pivot_s$ であったクラスタに含まれることになるので，UEX_{s+1} および RDY_{s+1}（もし $target_s \in RDY_s$ であれば）には $target_s$ であったクラスタを含めない．図 5 行 5 ではさらに，$pivot_s$ と $target_s$ の集約後に $pivot_s$ であったクラスタのクラスタサイズが δ_{opt} 以上となれば，UEX_{s+1} および RDY_{s+1} から $pivot_s$ を除外する．また，以降のタスク集約で必要な $pivot_{s+1}$ 選択優先度の更新，および RDY_{s+1} へ新規にクラスタを追加する（図 5 行 5 の処理は，5.7 節で述べる）．全クラスタのサイズが δ_{opt} 以上となると（すなわち $UEX_S = \emptyset$ のとき），アルゴリズムは終了する．このアルゴリズムによって $sl_w(G^S_{cls})$ を減少させるための考慮点として，以下の点を決める必要がある．

(1)　計算量と性能を考慮した，タスク集約方針
(2)　$pivot_s$ の選択基準（図 5 行 3）
(3)　$target_s$ の選択基準（図 5 行 4）
(4)　$pivot_s$ 選択優先度の更新方法（図 5 行 5）

5.3　タスク集約方針

$seq_{max,s}$ に属するタスクと先行関係のあるタスクが集約されると，$sl_w(G^{s+1}_{cls})$ は最大で局所化されたタスク間の通信データサイズ分だけ減少する（$sl_w(G^s_{cls}) > sl_w(G^{s+1}_{cls})$）．また，この集約によって先行関係のないタスクどうしが 1 クラスタに含まれると，$sl_w(G^{s+1}_{cls})$ が増加する可能性がある（$sl_w(G^s_{cls}) < sl_w(G^{s+1}_{cls})$）．そのため，$seq_{max,s}$ 内のタスクが属するクラスタを $pivot_s$ としてタスク集約を行えば，$sl_w(G^{s+1}_{cls})$ が増減する可能性がある．各タスク集約の前（たとえば $s+1$ 回目のタスク集約前），$seq_{max,s}$ 内のタスクが属するクラス

タのうち，LV_s が最大のものを選択し，さらにタスク集約後の $sl_w(G_{cls}^{s+1})$ 値をできるだけ最小化できるようなクラスタを $target_s$ とする方針を考える．この場合，$seq_{max,s}$ に属するタスクを求めるためには各クラスタの LV_s 値の更新が必要であるが，次の問題がある．

(1)　計算量の問題

$s-1$ 回目の $pivot_{s-1}$ と $target_{s-1}$ との集約の結果，どのタスク集合が新たな $seq_{max,s}$ となるかは DAG 全体を走査しないと分からない．V_s に属する各タスクの $tlevel$ 値，$blevel$ 値の更新にはそれぞれ $|V_s|+|E_s|$ ステップかかる．さらに 1 クラスタ $cls_s(i)$ について，$TL_s(i)$，$BL_s(i)$ の算出にはそれぞれ $\log|top_s(i)|$ ステップ，$\log|out_s(i)|$ ステップかかる．したがって V_{cls}^s 全体に対して LV_s を更新するのに $|V_{cls}^s|(\log|top_s(i)|+\log|out_s(i)|)$ ステップかかる．そして，この処理が全クラスタサイズが δ_{opt} 以上となるまで繰り返される．その結果，1 度のタスク集約のみで UEX_s から $pivot_s$ が除外されたとしても，DAG 全体を走査して各クラスタの LV_s 値を更新するのにかかる計算量は $O_{all}=O(|V|(|V|+|E|)+|V|^2\log|V|)$ となる．一般に，タスククラスタリングにおいて計算量を減少させるための考慮点は，タスク集約優先度の更新処理である．従来のタスククラスタリングにおけるタスク集約優先度は主にスケジュール長である．このうち，全走査によりスケジュール長の更新に $O(|E|(|V|+|E|))$ の計算量を要するもの[9]，計算量を減少させるために走査範囲を一部のクラスタ集合に限定して $O(|E|\log|V|)$ または $O(|V|\log|V|)$ であるもの[4]，スケジュール長の更新は行わないもの[11] がある．全走査による LV_s 値更新処理の計算量 O_{all} は，全走査によるスケジュール長の更新処理の計算量 $O(|E|(|V|+|E|))$ よりも大きいので，計算量の点では大きな問題となりうるといえる．

(2)　$sl_w(G_{cls}^s)$ の減少につながるとは限らないという問題

$seq_{max,s}$ 内のタスクが属するすべてのクラスタのサイズが δ_{opt} 以上のとき，他のクラスタのサイズが δ_{opt} 以上であるとは限らない．このような場合，5.1 節（i）を満たすために $seq_{max,s}$ に存在しないタスクが属するクラスタを $pivot_s$ とする必要がある．その結果，選択された $pivot_s$ と $target_s$ との集約によって $seq_{max,s+1}$ が更新され，$sl_w(G_{cls}^{s+1})$ が増加する（$sl_w(G_{cls}^s)<sl_w(G_{cls}^{s+1})$）可能性がある．すなわち，つねに $seq_{max,s}$ 内のタスクが属するクラスタを $pivot_s$ として選択しても，$sl_w(G_{cls}^{s+1})$ が減少する（$sl_w(G_{cls}^s)>sl_w(G_{cls}^{s+1})$）とは限らない．

そこで，$seq_{max,s}$ の特定および更新はせず，ある範囲内のクラスタ集合から $pivot_s$ を選択する．そして，タスク集約後は特定の範囲内のクラスタの LV_{s+1} の更新を行う方針とする．また，たとえ $seq_{max,s}$ 内のクラスタがすべて先行関係にあったとしても，上記 (2) の方針では，$seq_{max,s}$ 内のタスクと $seq_{max,s}$ 以外のタスクの集約によって先行関係のないタスクどうしが 1 クラスタに属する可能性がある．すなわちクラスタサイズが δ_{opt} 以上となるまでタスク集約を行うことにより，$sl_w(G_{cls}^{s+1})$ が増加しうる（$sl_w(G_{cls}^s)<sl_w(G_{cls}^{s+1})$）．そのため，アルゴリズム性能上の考慮点は，タスク集約ごとに $seq_{max,s}$ や $sl_w(G_{cls}^s)$ を決めないという条件の下，いかにして $pivot_s$，$target_s$ の選択を行うことによって $sl_w(G_{cls}^{s+1})$ を減少（$sl_w(G_{cls}^s)>sl_w(G_{cls}^{s+1})$）させる（または増分を最小にする）ことにつなげるかということである．

5.4　$pivot_s$ 選択範囲の定義とその影響

以上のことをふまえて，$pivot_s$ を，以下の条件を満たすクラスタ集合から選択するものとする．

$$\{cls_s(r)\,|\,cls_s(r)\in UEX_s,\ pred(n_{r'}^s)=\emptyset,\ cls_s(r)=\{n_{r'}^s\}\}\\ \cup\left\{cls_s(r)\,|\,cls_s(r)\in UEX_s,\ cls_s(q)\notin UEX_s,\ n_{q'}^s\in cls_s(q),\ n_{q'}^s\in pred(n_{r'}^s)\ \text{for}\ \forall n_{r'}^s\in top_s(r)\right\}. \tag{35}$$

RDY_s には，式 (35) の条件を満たすクラスタを保持するものとする．すなわち図 5 行 3 のときの RDY_s は，UEX_s に属するクラスタのうちで 1 つの START タスクのみを要素とするクラスタ集合か，またはその top_s タスクの任意の直接先行タスクが属するクラスタサイズが δ_{opt} 以上であるものの集合である．図 5 行 5 における RDY_s から RDY_{s+1} への更新（5.7 節で詳述）により，$s+1$ 回目のタスク集約前は，図 5 行 3 において RDY_{s+1} 内のすべてのクラスタサイズは δ_{opt} 未満である．すなわち図 5 行 5 の処理により，式 (35) で $s\leftarrow s+1$ とした場合のクラスタ集合のみが RDY_{s+1} に属するようにしている．

図 5 行 1 では全クラスタを UEX_s へ追加する．このときに RDY_s へ追加されるのは，START タスクを要素に持つクラスタ集合のみである（図 5 行 1 では，各クラスタの要素数は 1 である）．

次に，RDY_s に属するクラスタ集合から $pivot_s$ を選択する方針が，各クラスタの LV_{s+1} 値の増減にどう影響するかを述べる．まず，図 5 行 3 において RDY_s に属するクラスタ $cls_s(r)$ が $pivot_s$ として選択されたとする．そして，5.1 節（ii）の方針に従い，以下の集合に属する任意のクラスタ $cls_s(t)$ から 1 つを選択し，これを $target_s$ とする（$target_s$ の具体的な選択方針については 5.6 節で述べる）．

$$TGT_s(r)=\{cls_s(t)\,|\,n_{t'}\in suc(n_{r'}),\ cls_s(t)=\{n_{t'}\},\ cls_s(t)\in UEX_s,\ \exists n_{r'}\in out_s(r)\}. \tag{36}$$

その結果，次の性質がいえる．

性質 1. $cls_{s+1}(r)\leftarrow cls_s(r)\cup cls_s(t)$ としても，$TL_{s+1}(r)=TL_s(r)$ である．

証明. 付録 A.1 を参照．■

また，以下の性質 2 も，性質 1 と同様に示される．

性質 2. $cls_s(r)$，$cls_s(s)\in RDY_s$ とする．$pivot_s$ を $cls_s(r)$ とし，$TGT_s(r)$ に属するクラスタ $cls_s(t)$ を $target_s$ として集約しても，$TL_{s+1}(s)=TL_s(s)$ である．

証明. 付録 A.2 を参照．■

性質 1，2 より，RDY_s に属するクラスタ $cls_s(r)$ を $pivot_s$ とし，$TGT_s(r)$ に属するクラスタ $cls_s(t)$ を $target_s$ としてタスク集約を行っても，RDY_s 内のクラスタについて，TL_{s+1} の値は TL_s の値と等しい．そのため，このようなタスク集約により，少なくともすでに RDY_s に属しているクラスタと，δ_{opt} 以上のサイズであるクラスタの TL_{s+1} 値の更新は必要ない．RDY_s から $pivot_s$ を選択する方針は，時間計算量の点においては全走査による LV_{s+1} 値更新の場合よりも利点となる．

次に，RDY_s から $pivot_s$ を選択し，$target_s$ を式 (36) に基づいて選択することによって，$sl_w(G^{s+1}_{cls})$ に対してどう影響するのかについて述べる．図 5 行 1 により，最初はすべての START タスクが RDY_0 に属する．このときの $seq_{max,0}$ はクリティカルパスに属するタスクのため，START タスクのいずれかが $seq_{max,0}$ に属する．しかしながら，s 回のタスク集約後，$seq_{max,s}$ のタスクが，RDY_s 内のクラスタに属するとは限らない．そのため，$s+1$ 回のタスク集約後，$sl_w(G^{s+1}_{cls})-sl_w(G^s_{cls})$ の増減を明らかにするために，タスク集約によって $seq_{max,s+1}$ 内のタスクが属するクラスタの LV_{s+1} 値がどう変動するのかを述べる．ここで $seq_{max,s}$ 内のタスクが属するあるクラスタを $cls_s(i)$ とすると，以下の性質がいえる．

性質 3. $seq_{max,s}$ 内のタスクが属するクラスタ $cls_s(i)$ について，$cls_s(i)=\{n^s_{i'}\}$ とし，$cls_s(i)\notin RDY_s$ とする．また，RDY_s から選択されたクラスタ $cls_s(r)$ を $pivot_s$，$TGT_s(r)$ に属するクラスタ $cls_s(t)$ を $target_s$ とし，$pivot_s$，$target_s$ 内の任意のタスクは $seq_{max,s}$ に属していないものとする．ここで，$\Delta TL_{s+1}=TL_{s+1}(i)-TL_s(i)$，$\Delta BL_{s+1}=BL_{s+1}(i)-BL_s(i)$ とすると，以下が成り立つ．

(A). $n^s_{r'}\prec n^s_{i'}$ for $\exists n^s_{r'}\in cls_s(r)\cup cls_s(t)\Rightarrow \Delta TL_{s+1}(i)\geq 0,\ \Delta BL_{s+1}(i)=0.$

(B). $n^s_{i'}\prec n^s_{r'}$ for $\exists n^s_{r'}\in cls_s(r)\cup cls_s(t)\Rightarrow \Delta TL_{s+1}(i)=0,\ \Delta BL_{s+1}(i)=0.$

証明. 付録 A.3 を参照．■

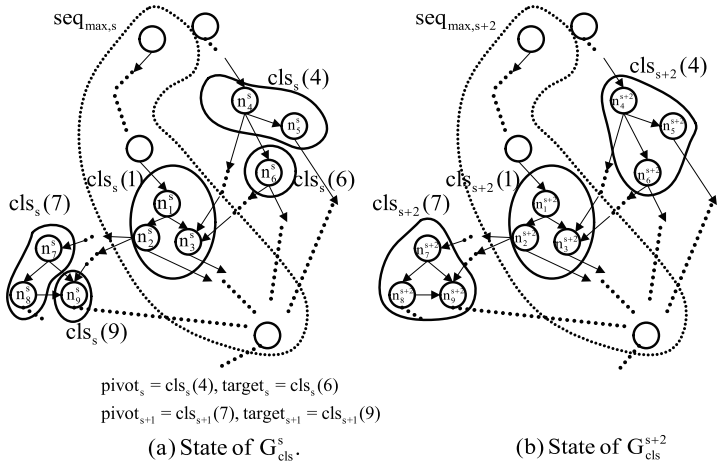


図 6　$pivot_s$ と $target_s$ 間のタスク集約による，他クラスタへの LV の影響

Fig. 6　Effect on LV_{s+1} value of every cluster after one task clustering of $pivot_s$ and $target_s$.

性質 4. $seq_{max,s}$ 内のタスクが属するクラスタ $cls_s(i)$ について，$|cls_s(i)|\geq 2$ とする．また，RDY_s で選択されたクラスタ $cls_s(r)$ を $pivot_s$，$TGT_s(r)$ に属するクラスタ $cls_s(t)$ を $target_s$ とし，$pivot_s$，$target_s$ 内の任意のタスクは $seq_{max,s}$ に属していないものとする．このとき $cls_{s+1}(r)\leftarrow cls_s(r)\cup cls_s(k)$ との集約による $TL_{s+1}(i)$，$BL_{s+1}(i)$ の増減をそれぞれ，$\Delta TL_{s+1}(i)$，$\Delta BL_{s+1}(i)$ とすると，$\exists n^s_{i'}\in cls_s(i)$ に対して以下が成り立つ．

(A). $n^s_{r'}\prec n^s_{i'}$ for $\exists n^s_{r'}\in cls_s(r)\cup cls_s(t)\Rightarrow \Delta TL_{s+1}(i)=0,\ \Delta BL_{s+1}(i)=0.$

(B). $n^s_{i'}\prec n^s_{r'}$ for $\exists n^s_{r'}\in cls_s(r)\cup cls_s(t)\Rightarrow \Delta TL_{s+1}(i)=0,\ \Delta BL_{s+1}(i)=0.$

証明. 付録 A.4 を参照．■

性質 3，4 より，$seq_{max,s}$ を特定せずに RDY_s から選択されたクラスタ $cls_s(r)$ を $pivot_s$，$TGT_s(r)$ から選択されたクラスタ $cls_s(t)$ を $target_s$ として集約すると，未集約（要素数が 1）であるクラスタの TL_{s+1} 値が増加する可能性がある（$TL_s<TL_{s+1}$）．一方，集約済み（要素数が 2 以上）であるクラスタの LV_{s+1} 値には影響はないことが分かる．また，$seq_{max,s}$ 内のタスクが属する任意のクラスタ $cls_s(i)$ について，$BL_{s+1}(i)$ への影響はない．

例 5.4.1. 図 6 に，性質 4 の例を示す．図中，破線は先行関係があることを示し，一方，破線枠では，その中のタスクが $seq_{max,s}$ に属していることを示す．$cls_s(1)$ 内のタスクが $seq_{max,s}$ に属しているものとし，$cls_s(4)$，$cls_s(6)$，$cls_s(7)$，$cls_s(9)$ はいずれも UEX_s に

属しているものとする．また，$pivot_s$ を $cls_s(4)$，$target_s$ を $cls_s(6)$ としたタスク集約と，$pivot_{s+1}$ を $cls_{s+1}(7)$，$target_{s+1}$ を $cls_{s+1}(9)$ とした 2 つのタスク集約が行われるものとする．そして，(b) ではこれら 2 回のタスク集約が行われた後の状態を示す．$cls_s(1)$ では (a) 以前にタスク集約が行われているので，$top_s(1)$，すなわち n_1^s の直接先行タスクが属するクラスタはサイズが δ_{opt} 以上のクラスタに属する．したがって，$cls_s(4)$ または $cls_s(6)$ 内の任意のタスクは n_1^s の直接先行タスクにはなりえず，図 6 (b) における $cls_{s+2}(4)$ 内の任意のタスクの *tlevel* 値の変動は $TL_{s+2}(1)$ には影響しない．一方，図 6 (a) における $cls_s(4)$ と $cls_s(6)$ とのタスク集約後，$cls_{s+1}(7)$ と $cls_{s+1}(9)$ との間のタスク集約により，(b) においては $BL_{s+2}(7) - BL_s(7) \neq 0$ である．しかしながら $cls_{s+2}(7)$ 内の任意のタスクの *blevel* 値は，s 回のタスク集約後の状態に比べて，データ局所化によって減少するが増加はしない．その結果，図 6 (b) における $cls_{s+2}(7)$ の任意のタスクは $seq_{max,s+2}$ に属さず，$BL_{s+2}(1) = BL_{s+1}(1) = BL_s(1)$ である．■

5.5 $pivot_s$ の選択

図 5 行 3 の $getPivot(RDY_s)$ について述べる．$seq_{max,s}$ 内のタスクが属するクラスタを $cls_s(i)$ とすると，$cls_s(i)$ は RDY_s に属していない可能性がある．もし $cls_s(i)$ が RDY_s に属していれば，その LV_s 値は RDY_s の中で最大値をとる．このような場合，$cls_s(i)$ を $pivot_s$ として選択してタスク集約後の $LV_{s+1}(i)$ を減少（$LV_s(i) > LV_{s+1}(i)$）させる（または増分を最小化する）ことが，$sl_w(G_{cls}^{s+1})$ を減少させる（$sl_w(G_{cls}^s) > sl_w(G_{cls}^{s+1})$）ことにつながる．しかしながら $cls_s(i)$ が RDY_s に属さない場合は，性質 3，4 より，RDY_s から選択されたクラスタを $pivot_s$，$TGT_s(r)$ から選ばれたクラスタを $target_s$ としたタスク集約により，$TL_{s+1}(i)$ が増加する可能性がある（$TL_s(i) < TL_{s+1}(i)$）．ここで，RDY_s に属するクラスタの中で LV_s 値が最大のクラスタを $cls_s(r)$ とし，$cls_s(r)$ を $pivot_s$ とする．そして $TGT_s(r)$ に属するクラスタ $cls_s(t)$ を $target_s$ として選択した場合，性質 3，4 より次のことがいえる．

性質 5. $cls_s(r) \in RDY_s$ とし，$seq_{max,s}$ 内のタスクが属するクラスタを $cls_s(i)$ とする．$cls_s(r)$ 内の任意のタスクが $seq_{max,s}$ に属さず，かつ $cls_s(r)$ を $pivot_s$，$TGT_s(r)$ から選択したクラスタを $target_s$ とする．$cls_s(r)$ が linear であり，かつこれらの集約後にも $cls_{s+1}(r)$ が linear であれば，性質 3，4 より $TL_{s+1}(i)$ は増加しない（$TL_s(i) \geq TL_{s+1}(i)$）．また，$BL_{s+1}(i)$ は不変（$BL_s(i) = BL_{s+1}(i)$）である．

性質 6. $cls_s(r) \in RDY_s$ とする．$cls_s(r)$ 内の少なくとも 1 タスクが $seq_{max,s}$ に属する場合，$cls_s(r)$ を $pivot_s$ としたタスク集約による $LV_{s+1}(r)$ の増減により，$sl_w(G_{cls}^{s+1})$ も増減する．

性質 5 が成立する理由は，タスク集約後の $cls_s(r)$ が linear であるから $TL_s(r)$，$BL_s(r)$ はそれぞれタスク集約前と後で不変であるためである．また，性質 6 により，$sl_w(G_{cls}^{s+1})$ を減少させる（$sl_w(G_{cls}^s) > sl_w(G_{cls}^{s+1})$）ためには，$RDY_s$ の中で LV_s 値が最大のクラスタを $pivot_s$ とすることが必要である．そのため，以下の条件を満たすクラスタ $cls_s(p)$ を $pivot_s$ とする．

$$cls_s(p) \in RDY_s,\ V_s(p) = \max_{cls_s(r) \in RDY_s} \{LV_s(r)\} \tag{37}$$

5.6 $target_s$ の選択

ここでは図 5 行 4 の $getTarget(pivot_s)$ の処理について述べる．

5.6.1 $target_s$ の選択条件

$pivot_s$ 内の任意のタスクが $seq_{max,s}$ に属していない場合，性質 5 より，$pivot_s$ 内の任意のタスクと先行関係のあるタスクのみをできるだけ集約することが必要である．一方，$pivot_s$ に $seq_{max,s}$ 内のタスクが属している場合，性質 1，2 より，タスク集約によって $pivot_s$ であったクラスタの BL_{s+1} 値を減少（$BL_s > BL_{s+1}$）させる（または増分を最小化する）ようなクラスタを $target_s$ とすることが必要である．また，性質 5 より，各クラスタの LV_{s+1} 値を減少（$LV_s > LV_{s+1}$）させる（または増分を最小化する）ためには，タスク集約後も $pivot_s$ であったクラスタが，引き続き linear であるような集約基準が必要である．そこで，まずは linear であるクラスタを $pivot_s$ としたときのタスク集約について，以下の性質を示す．

性質 7. 式 (37) で選択された $pivot_s$ を $cls_s(p)$ とする．また，あるタスク $n_{t'}^s$ に対し，$n_{t'}^s \in suc(n_{p'}^s)$，$n_{p'}^s \in btm_s(p)$，$cls_s(t) = \{n_{t'}^s\}$ とする．このとき $cls_s(p)$ が linear であれば，$cls_{s+1}(p) \leftarrow cls_s(p) \cup cls_s(t)$ としても $cls_{s+1}(p)$ は linear である．

証明. 付録 A.5 を参照．■

また，タスク集約によって $pivot_s$ であったクラスタの TL_{s+1} 値および BL_{s+1} 値が増加する場合（$TL_s < TL_{s+1}$，$BL_s < BL_{s+1}$）について，以下の性質を示す．

性質 8. 式 (37) で選択された $pivot_s$ を $cls_s(p)$ とする．また $out_s(p)$ の直接後続タスクが属する，サイズが δ_{opt} 未満のクラスタ $cls_s(t)$ を $target_s$ とする（$target_s \neq pivot_s$ とする）．このとき，$|cls_s(t)| \geq 2$ であれば，$\forall n_{p'}^s \in top_s(p)$，$\forall n_{t'}^s \in top_s(t)$ である $n_{p'}^s$，$n_{t'}^s$ に対して $n_{p'}^s \notin suc(n_{t'}^s)$ かつ $n_{p'}^s \notin pred(n_{t'}^s)$ である．

証明. 付録 A.6 を参照．■

性質 8 より，$pivot_s$ を $cls_s(p)$，要素数が 2 以上のクラスタを $target_s$ としたとき，タスク集約後に $top_{s+1}(p)$ の要素数が増加する．すなわち，これによって先行関係のないタスクどうしが存在することになる．その結果，$cls_{s+1}(p)$ 内のタスクのうち，$top_{s+1}(p)$ でないタスクの $tlevel$ 値だけでなく，$TL_{s+1}(p)$ 値と $BL_{s+1}(p)$ 値（すなわち $LV_{s+1}(p)$）も増加する可能性がある（$BL_s(p) < BL_{s+1}(p)$）．一方，性質 1 のように 1 タスクのみが属するクラスタを $target_s$ とすれば，タスク集約後も $pivot_s$ であったクラスタの TL_{s+1} 値は増加しない（$TL_s \geq TL_{s+1}$）．このことから，以下の条件を満たすクラスタを $target_s$ とすることが必要である．

条件 1．タスク集約後，$pivot_s$ であったクラスタの BL_{s+1} 値を減少させる（$BL_s > BL_{s+1}$）か，または増分を最小化するクラスタ（性質 1，2，8）

条件 2．$pivot_s$ 内の任意のタスクと先行関係のあるクラスタ（性質 5，7，8）

5.6.2　$target_s$ の選択処理

これらの条件をふまえて，$target_s$ 選択処理を述べる．図 5 行 4 の処理は，$pivot_s$ 内のタスクと先行関係（直接先行または直接後続関係）にあるタスクを探索し，それが属するクラスタのうち，$sl_w(G^{s+1}_{cls})$ を減少（$sl_w(G^s_{cls}) > sl_w(G^{s+1}_{cls})$）または増分を最小化する可能性のあるものを $target_s$ として選択する．$target_s$ の選択基準は，$pivot_s$ の状態（linear か linear でないか）および $pivot_s$ 内タスクの直接後続タスクの状態によって様々考えられる．**図 7** に，$target_s$ の選択処理の全パターンを示す．図 7 では $pivot_s$ を $cls_s(p)$ とし，$target_s$ 候補となるクラスタ集合を $TGT_c(cls_s(p))$ としている．そして $TGT_c(cls_s(p))$ の中からどのクラスタを $target_s$ として選択するかを示している．また，$cls_s(p)$ が linear または linear でない（non-linear）にかかわらず同様の $target_s$ 選択処理となる場合は，どちらか一方の場合のみを示している．図 5 行 4 では，図 7 (a)，(b)，(c)，(d)，(e) の順に $target_s$ を探索する．たとえば (a) に該当する $TGT_c(cls_s(p))$ がなければ (b) で $TGT_c(cls_s(p))$ があるか走査し，それでもなければ (c) . . . の順で走査する．該当する $TGT_c(cls_s(p))$ が存在する時点でその中から $target_s$ を選択し，図 5 行 4 は完了する．

(a) $pivot_s = \{cls_s(p)\}$ が linear でかつ，$btm_s(p) = \{n^s_{p'}\}$ の直接後続タスクのうち，未集約クラスタに属するものが存在する場合（図 7 (a)）．
$btm_s(p) = \{n^s_{p'}\}$ の直接後続タスクのうち，未集約クラスタに属するタスクを $n^s_{u'}$ とし，$n^s_{u'}$ が属するクラスタを $cls_s(u)$ とする（つまり $cls_s(u) = \{n^s_{u'}\}$）．そしてそのような $cls_s(u)$ の集合を $TGT_c(cls_s(p))$ とする．そして，以下の条件を満たすタスクが属するクラスタ $cls_s(t)$ を $target_s$ として，$TGT_c(cls_s(p))$ の中から選択する．

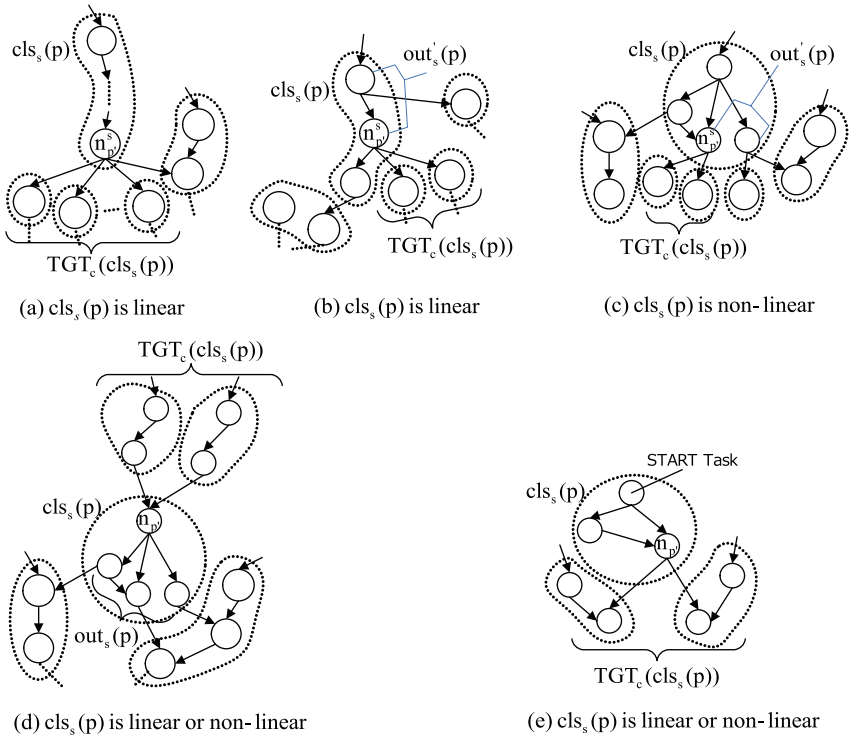


図 7　$target_s$ の選択パターン（$pivot_s = cls_s(p)$ とする）
Fig. 7　Pattern of selecting $target_s$ (where let $pivot_s = cls_s(p)$).

$$c(e^s_{p',t'}) + blevel(n^s_{t'}) = \max_{\substack{n_{u'} \in cls_s(u),\\ cls_s(u) \in TGT_s(cls_s(p))}} \left\{ c(e^s_{p',u'}) + blevel(n^s_{u'}) \right\},$$

$$\text{where } n^s_{t'} \in cls_s(t),\ cls_s(t) \in TGT_c(cls_s(p)),\ btm_s(p) = \{n^s_{p'}\},$$

$$TGT_c(cls_s(p)) = \{cls_s(u) \mid cls_s(u) = \{n^s_{u'}\},\ n^s_{u'} \in suc(n^s_{p'})\}. \quad (38)$$

その結果，タスク集約後，$cls_{s+1}(p)$ は linear であるから，$cls_{s+1}(p)$ 内の任意のタスク間で先行関係が保たれる．これは条件 2 を満たす．式 (38) に基づいて選ばれた $target_s$ との集約によって $-c(e^s_{p',u'}) \leq BL_{s+1}(p) - BL_s(p) \leq 0$ となる．すなわち，この集約によって $LV_{s+1}(p)$ は増加しない（$LV_s(p) \geq LV_{s+1}(p)$）．

(b) $pivot_s = \{cls_s(p)\}$ が linear でかつ，$btm_s(p) = \{n^s_{p'}\}$ の直接後続タスクがすべて，

要素数 2 以上のクラスタに属するとき（図 7 (b)）．

この場合，（a）の方針によって $btm_s(p)$ の直接後続タスクを走査して $target_s$ を選択すると，タスク集約後の $cls_{s+1}(p)$ は linear でなくなる可能性がある．そのため，ここでは条件 1，すなわちいかにして $BL_{s+1}(p)$ の増分を最小化するかを目標とする．表 1 の定義より，$BL_s(p)$ は $out_s(p)$ 内のタスクで決まる．そこで，まずは $out_s(p)$ に属し，さらに未集約クラスタ内のタスクを直接後続関係に持つタスクを n_k^s と定義する．そして，そのような n_k^s の集合を $out'_s(p)$ とすると，$out'_s(p)$ のうち，$S(n_k^s,p)+blevel(n_k^s)$ 値が最大であるタスクを選択する（これを $n_{p'}^s$ とする）．さらに，$n_{p'}^s$ の直接後続タスクが属するクラスタのうち，未集約（要素数 1）であるものの集合を $TGT_c(cls_s(p))$ とする．そして，以下の条件を満たすタスクが属するクラスタ $cls_s(t)$ を $target_s$ とする．

$$c(e^s_{p',t'})+blevel(n^s_{k'})=\max_{\substack{n^s_{u'}\in cls_s(u),\\ cls_s(u)\in TGT_c(cls_s(p))}}\left\{c(e^s_{p',u'})+blevel(n^s_{u'})\right\},$$

$$\text{where } n^s_{t'}\in cls_s(t),\ cls_s(t)\in TGT_c(cls_s(p)),\ n^s_{t'}\in suc(n^s_{p'}),$$

$$S(n^s_{p'},p)+blevel(n^s_{p'})=\max_{n^s_k\in out'_s(p)}\{S(n^s_{k'},p)+blevel(n^s_k)\},$$

$$out'_s(p)=\left\{n^s_k\,|\,n^s_k\in out_s(p),\ cls(q)=\{n^s_{q'}\}\text{ for }\exists n^s_{q'}\in suc(n^s_k),\ n^s_{q'}\in cls(q)\right\},$$

$$TGT_c(cls_s(p))=\left\{cls_s(u)\,|\,cls_s(u)=\{n^s_{u'}\},\ n^s_{u'}\in suc(n^s_{p'})\right\}. \tag{39}$$

$\Delta BL_{s+1}(p)=BL_{s+1}(p)-BL_s(p)$ とすると，$e^s_{p',t'}$ の局所化により，$BL_s(p)=S(n^s_{p'},p)+blevel(n^s_{p'})$ である場合に $-c(e^{s+1}_{p',t'})\le\Delta BL_{s+1}(p)$ となる．一方，タスク集約後には $n^s_{t'}\in cls_s(p)$ であるから，$\Delta BL_{s+1}(p)\le w(n^s_{t'})$ である．したがって，タスク集約後の $LV_{s+1}(p)$ の増分は，$-c(e^s_{p',t'})$ から $w(n^s_{t'})$ の範囲以内となる．

（c）$pivot_s=\{cls_s(p)\}$ が linear でなく，$out_s(p)$ の直接後続タスクのうち，未集約クラスタに属するものがあるとき（図 7 (c)）．

$cls_s(p)$ は linear ではないので，タスク集約後も linear ではない．たとえ $btm_s(p)$ の直接後続タスクが属するクラスタの中で，未集約クラスタを $target_s$ としても，タスク集約後においては互いに先行関係のないタスク組が増加する可能性がある．したがって，この場合は条件 2 を満たさない．そこで，$out_s(p)$ の直接後続タスクを走査し，条件 1，すなわち $BL_{s+1}(p)$ の増分を最小化することを目標とする．そこで（b）同様，まずは $out_s(p)$ に属し，かつ未集約クラスタ内のタスクを直接後続関係に持つタスク n_k^s とし，n_k^s の集合を $out'_s(p)$ とする．そして $S(n_k^s,p)+blevel(n_k^s)$ 値が最大であるタスクを選択する（これを $n_{p'}^s$ とする）．さらに，$n_{p'}^s$ の直接後続タスクのうち，未集約クラスタを $cls_s(u)=\{n^s_{u'}\}$ とし，$cls_s(u)$ の集合を $TGT_c(cls_s(p))$ とする．そして $TGT_c(cls_s(p))$ の中から，以下の条件を満たすクラスタ $cls_s(t)$ を $target_s$ とする．

$$c(e^s_{p',t'})+blevel(n^s_{k'})=\max_{\substack{n^s_{u'}\in cls_s(u),\\ cls_s(u)\in TGT_c(cls_s(p))}}\left\{c(e^s_{p',u'})+blevel(n^s_{u'})\right\},$$

$$S(n^s_{p'},p)+blevel(n^s_{p'})=\max_{n^s_k\in out'_s(p)}\{S(n^s_{k'},p)+blevel(n^s_k)\},$$

$$out'_s(p)=\left\{n^s_k\,|\,n^s_k\in out_s(p),\ cls(q)=\{n^s_{q'}\}\text{ for }\exists n^s_{q'}\in suc(n^s_k),\ n^s_{q'}\in cls(q)\right\},$$

$$TGT_c(cls_s(p))=\left\{cls_s(u)\,|\,cls_s(u)=\{n^s_{u'}\},\ n^s_{u'}\in suc(n^s_{p'})\right\}. \tag{40}$$

その結果，$cls_{s+1}(p)$ には $n^{s+1}_{t'}$ が含まれるので，（b）同様，$LV_{s+1}(p)$ の増分は $-c(e^s_{p',t'})$ から $w(n^s_{u'})$ 以内である．

（d）$pivot_s=\{cls_s(p)\}$ が linear または linear でないかにかかわらず．$out_s(p)$ の直接後続タスクがすべて要素数 2 以上のクラスタに属するとき（図 7 (d)）．

この場合，$out_s(p)$ の直接後続タスクを走査しても要素数 1 のクラスタは存在しないので，$LV_{s+1}(p)$ を最小化することを目標とする．$cls_s(p)$ が linear であれば，$cls_{s+1}(p)$ も linear であることが望ましい．そのためには，$in_s(p)$ の直接先行タスクが属する，$cls_s(p)$ 以外のクラスタから $target_s$ を選択する必要がある．表 1 の定義より $in_s(p)$ には $top_s(p)$ に属するタスクとそうでないタスクが存在しうるので，まずは $in_s(p)$ のみに属するタスクと $top_s(p)$ 内のタスクのうち，どちらの直接先行タスクを走査すべきかを決める．$in_s(p)$ のうちで $top_s(p)$ に属さないタスクを n_k^s とし，n_k^s の直接先行タスクが属するクラスタを $target_s$ とした場合を考える．この場合，$target_s$ 内タスクのうちで $top_s(p)$ と先行関係のないタスクが存在する可能性がある．その結果，タスク集約後の $|top_{s+1}(p)|$ が増加し，$LV_{s+1}(p)$ が増加（$LV_s(p)<LV_{s+1}(p)$）してしまう（各クラスタにおいて top_s タスクは，同クラスタの他タスクよりも必ず先に実行されてしまい，必ず $BL_{s+1}(p)$ が増加（$BL_s(p)<BL_{s+1}(p)$）するため）．そのため，$cls_{s+1}(p)$ においてタスク間先行関係を保つためには，$top_s(p)$ の直接先行タスクが属するクラスタを $target_s$ とすることが必要である．

次に，$LV_{s+1}(p)$ を減少（$LV_s(p)>LV_{s+1}(p)$），または増分を最小化するための基準を述べる．$cls_{s+1}(p)$ が（a），（b），（c）いずれかの方針でタスク集約が行われたクラスタであれば，$|top_{s+1}(p)|=1$ である（$top_{s+1}(p)$ の要素数が増加する場合については，（e）で述べる）．その結果，タスク集約前において $|top_s(p)|=1$ であるとき，タスク集約後の $TL_{s+1}(p)$ は，$target_s$（$top_s(p)$ の直接先行タスクが属するクラスタ）の

TL_s 値となる（タスク集約後の $top_{s+1}(p)$ は，$target_s$ の top_s タスクとなるため）．そのため，この場合，いかにして $cls_{s+1}(p)$ の LV_{s+1} 値の増分を最小化するかを目標とする．タスク集約前の任意の $n^s_{p'} \in top_s(p)$ について，$n^s_{p'}$ の直接先行タスクを $n^s_{u'}$ とし，$n^s_{u'} \in cls_s(u)$ と定義する．$cls_s(u)$ の集合を $TGT_c(cls_s(p))$ とすると，$TGT_c(cls_s(p))$ のうちで LV_s 値が最大のクラスタ $cls_s(t)$ を $target_s$ とする．

$$
\begin{aligned}
&LV_s(t) = \max_{cls_s(u) \in TGT_c(cls_s(p))} \{LV_s(u)\},\\
&\text{where } n^s_{t'} \in pred(n_{p'}),\ n^s_{t'} \in cls_s(t),\ cls_s(t) \in TGT_c(cls_s(p)),\\
&TGT_c(cls_s(p)) = \{cls_s(u) \mid n^s_{u'} \in pred(n^s_{p'}),\ n^s_{u'} \in cls_s(u)\},\ n^s_{p'} \in top_s(p),\\
&TL_s(p) = tlevel(n^s_{p'}).
\end{aligned} \tag{41}
$$

この場合，タスク集約後の $LV_{s+1}(p)$ の増分は，タスク集約前における $target_s$ の TL_s 値と，タスク集約後の $BL_s(p)$ 内タスクの先行関係で決まる．

(e) $pivot_s = \{cls_s(p)\}$ が linear または linear でないかにかかわらず，$top_s(p)$ が START タスクのみであり，かつ $out_s(p)$ の直接後続タスクがすべて要素数 2 以上のクラスタに属するとき（図 7 (e)）．

$top_s(p)$ には START タスクしか属していないので，図 7 (a)，(b)，(c)，(d) の方針では $target_s$ を選択できない（該当する $target_s$ の候補が存在しない）．この場合は，$out_s(p)$ の直接後続タスクが属するクラスタを $target_s$ とする．そのような $target_s$ の要素数は 2 以上であるから，性質 8 より，$cls_s(p)$ と $target_s$ との集約により先行関係のないタスクどうしが $cls_{s+1}(p)$ に含まれる．一方，$top_s(p)$ には START タスクのみが属するので，$TL_s(p) = 0$ である．さらに性質 8 より，タスク集約によって $top_{s+1}(p)$ の要素数が増加する．その結果，タスク集約後の $TL_{s+1}(p)$ は $target_s$ の TL_s 値となり，$BL_{s+1}(p)$ は集約後の $cls_{s+1}(p)$ 内に属するタスク間の先行関係で決まる．そのため，この場合，タスク集約後の $BL_{s+1}(p)$ の増分を最小化することを目標とする．そこで以下の式 (42) のように，$out_s(p)$ に属するタスク $n^s_{k'}$ の中で $S(n^s_{k'}, p) + blevel(n^s_{k'})$ が最大のタスク（$n^s_{p'}$ とする）を特定する．そして，$n^s_{p'}$ の直接後続タスクのうちで $blevel(n^s_{p'})$ を支配しているタスク（$n^s_{t'}$ とする）が属するクラスタ $cls_s(t)$ を $target_s$ とする．

$$
\begin{aligned}
&c(e^s_{p',t'}) + blevel(n^s_{t'}) = \max_{\substack{n^s_{u'} \in cls_s(u),\\ cls_s(u) \in TGT_c(cls_s(p))}} \left\{c(e^s_{p',u'}) + blevel(n^s_{u'})\right\},\\
&\text{where } BL_s(p) = S(n^s_{p'}, p) + blevel(n^s_{p'}),\\
&n^s_{p'} \in out_s(p),\ n^s_{t'} \in cls_s(t),\ cls_s(t) \in TGT_c(cls_s(p)),\\
&TGT_c(cls_s(p)) = \{cls_s(u) \mid n^s_{t'} \in cls_s(u),\ n^s_{t'} \in suc(n^s_{p'})\}.
\end{aligned} \tag{42}
$$

これにより，先行関係のないタスク組による増分から，$c(e^s_{p',t'})$ だけ減算することによって $BL_{s+1}(p)$ の増分を最小化する方針とする．

5.7　タスク集約処理と集約優先度の更新

ここでは，図 5 行 5 の $clustering(pivot_s, target_s)$ の処理について述べる．**図 8** に，$clustering(pivot_s, target_s)$ の具体的な処理を示す．図 8 行 0 で定義したように，$pivot_s$ は式 (37) によって選択されたクラスタ $cls_s(p)$ を $pivot_s$ とし，一方，図 7 (a)，(b)，(c)，(d)，(e) いずれかの場合で選択された $target_s$ を $cls_s(t)$ とする．$clustering(pivot_s, target_s)$ はタスク集約処理，$pivot_s$ 選択優先度の更新処理，RDY_s 更新処理に分かれる．

まず，タスク集約処理を述べる．この処理は，図 8 行 1–4 に該当する．タスク集約によって $cls_{s+1}(p)$ のクラスタサイズが δ_{opt} 以上となれば，$cls_{s+1}(p)$ は RDY_{s+1}，UEX_{s+1} には属さない．一方，$cls_s(t)$ は $cls_{s+1}(p)$ に属することになるので，RDY_{s+1}，UEX_{s+1} には $cls_s(t)$ であったクラスタを含めない．これにより，サイズが δ_{opt} 以上のクラスタが RDY_{s+1} に属することはない．

次に $pivot_s$ 選択優先度の更新処理について述べる．この処理は図 8 行 5–25 に該当する．もし $target_s$ が図 7 (d) または (e) によって選択された場合，$TL_{s+1}(p)$ が変動しうる．そのため，行 5–7 において，$top_{s+1}(p)$ および $TL_{s+1}(p)$ の更新を行う．また，以降の RDY_{s+1} 更新処理（後述）によって $cls_{s+1}(p)$ が RDY_{s+1} に属する場合は $LV_{s+1}(p)$ を最新の値に保つ必要がある．$cls_{s+1}(p) \notin RDY_{s+1}$ だとしても，$out_{s+1}(p)$ の直接後続タスクが属するクラスタのうち，いずれかが新規に RDY_{s+1} へ追加される可能性がある．そのため，図 8 行 8 において $out_{s+1}(p)$，$btm_{s+1}(p)$，さらには図 8 行 10 で $out_{s+1}(p)$ 内のタスクに対して $tlevel$ 値の更新を行う．さらに $in_{s+1}(p)$ の直接先行タスクが属するクラスタのうち，RDY_{s+1} に属しているものが存在する可能性がある．そのため，これらのクラスタの BL_{s+1} 値も更新する必要がある．そこで図 8 行 17–19 でまずは $in_{s+1}(p)$ 内の $blevel$ 値更新を行い，そして行 23 において，そのようなクラスタ内タスクの $blevel$ 値を更新する．

最後に，RDY_{s+1} の更新処理について述べる．タスク集約後，$out_{s+1}(p)$ の直接後続タスクのクラスタのうち，式 (35) の条件にあてはまるようなクラスタが存在する可能性がある．このようなクラスタを特定するため，図 8 行 13 では，タスク集約後に $w(cls_{s+1}(p)) \geq \delta_{opt}$ となった場合に限り，$out_{s+1}(p)$ に属する各タスクの出辺に対して “checked” のラベルをつける．そして，RDY_{s+1} に入る可能性のある候補 $RDY_{candidate,s+1}$ へ追加する．その後，図 8 行 26–30 で $RDY_{candidate,s+1}$ 内のクラスタを走査して式 (35) を満たすかどうかの検

INPUT: $pivot_s, target_s$
OUTPUT: RDY_{s+1}
0. Let $pivot_s$ $cls_s(p)$ which has been obtained by eq. (37);
0. Let $target_s$ $cls_s(t)$ which has been obtained by Fig. 7 (a) or (b) or (c) or (d) or (e);
0. Let $RDY_{candidate,s+1} \leftarrow \emptyset$; /*$w(cls_{s+1}(p)) \geq \delta_{opt}$ である場合の，$out_{s+1}(p)$ の直接後続タスクが属するクラスタ集合 */
0. $RDY_{s+1} \leftarrow RDY_s$, $UEX_{s+1} \leftarrow UEX_s$ and $s \leftarrow s+1$ for every cluster in both RDY_{s+1} and UEX_{s+1}.;/*RDY_{s+1} と UEX_{s+1} それぞれに属するクラスタの状態 s を 1 増分させる*/
/******** タスク集約処理（*merge* 関数処理は，式 (1) で定義）********/
1. $cls_{s+1}(p) \leftarrow merge(cls_s(p), cls_s(t))$ and remove $cls_{s+1}(t)$ from RDY_{s+1} and UEX_{s+1} if exists;
2. **IF** $w(cls_{s+1}(p)) \geq \delta_{opt}$ **THEN**
3. Remove $cls_{s+1}(p)$ from RDY_{s+1} and UEX_{s+1};
4. **END IF**
/******** $pivot_s$ 選択優先度の更新処理 ********/
5. **IF** $cls_s(t)$ has been obtained by Fig. 7 (d) or (e) **THEN**
6. Update $top_{s+1}(p)$ and $TL_{s+1}(p)$; /*$TL_{s+1}(p)$ が変わりうる場合は，$TL_{s+1}(p)$ 値を更新*/
7. **END IF**
8. update $in_{s+1}(p)$, $out_{s+1}(p)$ and $btm_{s+1}(p)$; /*他クラスタとのデータ送受信を要するタスク集合を更新*/
9. **FOR EACH** $n^{s+1}_{p'} \in out_{s+1}(p)$ **DO**
10. update $S(n^{s+1}_{p'}, p)$ and $tlevel(n^{s+1}_{p'})$;/*各タスクの *tlevel* 値の更新*/
11. **IF** $w(cls_{s+1}(p)) \geq \delta_{opt}$ **THEN**
12. **FOR EACH** $n^{s+1}_{k'} \in suc(n^{s+1}_{p'}), n^{s+1}_{k'} \in cls_{s+1}(k) \neq cls_{s+1}(p)$ **DO**
/*$n^{s+1}_{p'}$ の直接後続タスクから，新規に RDY_{s+1} へ追加されうるクラスタを特定するための処理*/
13. mark "checked" on $e^{s+1}_{p',k'}$ and put $cls_{s+1}(k)$ into $RDY_{candidate,s+1}$;
14. **END FOR**
15. **END IF**
16. **END FOR**
17. **FOR EACH** $n^{s+1}_{p'} \in in_{s+1}(p)$ **DO**
18. update $blevel(n^{s+1}_{p'})$; /*行 20，および行 21-25 の処理のために必要*/
19. **END FOR**
20. Update $BL_{s+1}(p)$ by tracing $out_{s+1}(p)$; /*タスク集約後に $cls_{s+1}(p) \in RDY_{s+1}$ である場合のため*/
21. **FOR EACH** $n^{s+1}_{p'} \in in_{s+1}(p)$ **DO**
22. **FOR EACH** $n^{s+1}_{k'} \in pred(n^{s+1}_{p'}), n^{s+1}_{k'} \in cls_{s+1}(k) \neq cls_{s+1}(p), cls_{s+1}(k) \in RDY_{s+1}$ **DO**
23. update $blevel(n^{s+1}_{k'})$; /*$BL_{s+1}(k)$ 値更新のために必要*/
24. **END FOR**
25. **END FOR**
/******** RDY_{s+1} の更新処理 ********/
26. **FOR EACH** $cls_{s+1}(k) \in RDY_{candidate,s+1}$ **DO**
27. **IF** $\forall n^{s+1}_{q'} \in pred(n^{s+1}_{k'}), \forall n^{s+1}_{k'} \in top_{s+1}(k)$ *s.t.*, $e^{s+1}_{q',k'}$ is "checked" **THEN**
28. Update $LV_{s+1}(k)$ and put $cls_{s+1}(k)$ into RDY_{s+1};/*もし $cls_{s+1}(k)$ が RDY_{s+1} の条件を満たせば，RDY_{s+1} へ追加される*/
29. **END IF**
30. **END FOR**
31. **RETURN** RDY_{s+1};

図 8　タスク集約処理と，集約優先度の更新（図 5 行 5 の処理）
Fig. 8　Procedure of task clustering at Fig. 5 line 5.

査を行う．そして図 8 行 27 において，$top_{s+1}(k)$ に属するタスクについて，すべての入力辺が "checked" であれば，$cls_{s+1}(k)$ は RDY_{s+1} へ追加される（$top_{s+1}(k)$ の任意の直接先行タスクはすべて δ_{opt} 以上のクラスタに属するため）．また，以降の $pivot_{s+1}$ の選択のとき（図 5 行 3）に RDY_{s+1} に属する各クラスタの LV_{s+1} 値比較を行うので，図 8 行 28 で $LV_{s+1}(k)$ を更新しておく．これによって新規に RDY_{s+1} へ追加されるクラスタは，以降のタスク集約における $pivot_{s+1}$ 選択の候補となる．また，以下の性質よりアルゴリズム走行中はつねに $pivot_s$ が選択可能であるから，各タスク集約を繰り返すことによって $UEX_s = \emptyset$ となり，アルゴリズムは必ず終了する．

性質 9. アルゴリズム終了時以外，RDY_s にはつねに 1 以上のクラスタが存在する．

証明. 付録 A.7 を参照．■

5.3 節で述べたように，DAG 全体を走査すれば，s 回のタスク集約後，RDY_{s+1} に属するクラスタの LV_{s+1} 値は最新なものに保つことができる．しかしながら，計算量上の問題のため，ここでは新規に RDY_{s+1} に追加されるクラスタのみの LV_{s+1} を更新している．性質 2 より，RDY_{s+1} に属しているクラスタのうち，新規追加されたクラスタ以外，TL_{s+1} 値は不変（$TL_s = TL_{s+1}$）である．一方，それらのクラスタに属するタスクの *blevel* 値は減少する可能性がある．そのため，$pivot_s$ 選択対象となるクラスタについては，BL_s 値が最新ではない可能性がある（タスクの *blevel* 値減少により，実際にはそれより小さな値である可能性がある）．しかしながら性質 3，4 より，少なくとも $seq_{max,s}$ 内のタスクが属するクラスタの BL_s 値には影響しない．そのため，$pivot_s$ として選択されるクラスタ（すなわち RDY_s 内で LV_s 値が最大のクラスタ）内に $seq_{max,s}$ 内のタスクが属している場合，その LV_s 値は最新であるといえる．すなわち $sl_w(G^s_{cls})$ を直接減少させようとする点では，性能上の問題はない．

5.8　アルゴリズムの計算量

ここでは，図 5 全体の時間計算量について述べる．アルゴリズム全体として各タスクが参照される回数は，それが属するクラスタのサイズが δ_{opt} を満たすまでにタスク集約された回数である．その回数の上限値を $\varepsilon(G^0_{cls})$ とすると，

$$\varepsilon(G^0_{cls}) = \left\lceil \frac{\delta_{opt}}{\min\limits_{n^0_k \in V_0} \{w(n^0_k)\}} \right\rceil \tag{43}$$

であり，図 5 行 3–5 の各処理の計算量に対して $\varepsilon(G^0_{cls})$ だけの同一クラスタ参照回数分が乗算される．以下，図 5 の各処理の計算量を述べる．

- $pivot_s$ 選択処理の計算量（図 5 行 3）
  図 5 行 3 は，式 (37) において RDY_s から $pivot_s$ を選択する処理に該当する．ゆえにすでに RYD_s 内の各クラスタが LV_s 値の降順にソートされているものとすれば，1 度の $pivot_s$ 選択には 1 ステップで取得できる．したがって図 5 行 3 の計算量は，式 (43) より $O(\varepsilon(G^0_{cls})|V|)$ である．
- $target_s$ 選択処理の計算量（図 5 行 4）
  図 5 行 4 の処理は，図 7 (a)，(b)，(c)，(d)，(e) の順に $target_s$ を探索する処理に該当する．
  図 7 (a) では，$suc(n^s_{p'})$ から式 (38) に該当するタスク $n^s_{t'}$ を探索する．これには式 (38) における $TGT_c(cls_s(p))$ 内のクラスタ $cls(u) = \{n^s_{u'}\}$ について，$c(e^s_{p',u'}) + blevel(n^s_{u'})$ の降順でソートすればよい．したがって $|suc(n^s_{p'})| \log |suc(n^s_{p'})|$ ステップかかる．全体としては $O(\varepsilon(G^0_{cls})|E| \log |E|)$ である．
  図 7 (b) では，式 (39) において $out'_s(p)$ の特定を行う必要がある．この特定には $cls_s(p)$ 内のタスクおよびそれらのタスク間の辺を 1 度ずつ走査することが必要であり，ステップ数は $|cls_s(p)|$ と $cls_s(p)$ に属するタスク間の辺の数との総和である．そして $out'_s(p)$ に属するタスク $n^s_{k'}$ から $S(n^s_{k'}, p) + blevel(n^s_{k'})$ が最大のタスク（$n^s_{p'}$ とする）を選択するためには，マージソートによって $|out'_s(p)| \log |out'_s(p)|$ ステップかかる．その後，$suc(n^s_{p'})$ に属するタスク $n^s_{u'}$ のうちで，$c(e^s_{p',u'}) + blevel(n^s_{u'})$ が最大のタスク $n^s_{t'}$ を選択するのにマージソートにより $|suc(n^s_{p'})| \log |suc(n^s_{p'})|$ ステップかかる．図 7 (b) における $target_s$ 選択処理の中で支配的な処理は，$|V| \leq |E|$ であれば式 (39) の中の $n^s_{t'}$ 選択処理である．したがって $O(\varepsilon(G^0_{cls})(|E| \log |E|))$ である．図 7 (c) も同様である．図 7 (d) ではまず，式 (41) において $n^s_{p'}$ を選択する．$TL_s(p)$ が決まっていれば 1 ステップで $n^s_{p'}$ を選択できる．そして，$pred(n^s_{p'})$ 内の各タスクが属するクラスタの集合から，LV_s 値が最大であるものを選択する．これにはマージソートによって $|pred(n^s_{p'})| \log |pred(n^s_{p'})|$ ステップかかる．したがって全体としては $O(\varepsilon(G^0_{cls})(|E| \log |E|))$ の計算量である．
  そして図 7 (e) ではまず，式 (42) において $n^s_{p'}$ を選択する．$BL_s(p)$ が決まっていれば，1 ステップで選択できる．そして，$suc(n^s_{p'})$ に属するタスク $n^s_{u'}$ のうち，$c(e^s_{p',u'}) + blevel(n^s_{u'})$ が最大のものを選択する．これにはマージソートによって $|suc(n^s_p)| \log |suc(n^s_p)|$ ステップかかる．したがってこの場合の計算量は，$O(\varepsilon(G^0_{cls})(|E| \log |E|))$ である．よって図 5 行 4 の計算量は，$O(\varepsilon(G^0_{cls})(|E| \log |E|))$ である．
- タスク集約および集約優先度更新処理の計算量（図 5 行 5）
  図 5 行 5 は，図 8 の処理に該当する．図 8 行 2–4 では，RDY_s，UEX_s それぞれから $cls_s(p)$ を特定すればよいので全体として $O(\varepsilon(G^0_{cls}) \log |V|)$ である．
  図 8 行 5–7 ではまず，$top_s(p) \cup top_s(t)$ から $top_{s+1}(p)$ に属するタスクを選択する．これには $top_s(p) \cup top_s(t)$ 内の各タスクおよびその入力辺を 1 度ずつ走査すればよく，全体として $O(\varepsilon(G^0_{cls})(|V| + |E|))$ の計算量である．そして $top_{s+1}(p)$ 内のタスクおよびその入力辺を走査して $tlevel$ 値を更新する．したがって全体で $O(\varepsilon(G^0_{cls})(|V| + |E|))$ の計算量である．その後，$top_{s+1}(p)$ に属するタスクの $tlevel$ から，最大値を $TL_{s+1}(p)$ とする必要がある．これにはマージソートによって $|top_{s+1}(p)| \log |top_{s+1}(p)|$ ステップかかるので，全体としては $O(\varepsilon(G^0_{cls})|V| \log |V|)$ の計算量である．図 8 行 5–7 の計算量は，$O(\varepsilon(G^0_{cls})|V| \log |V|)$ である．
  図 8 行 8 ではタスク集約後の $cls_{s+1}(p)$ 内のタスクの入力辺，出力辺を走査すればよく，全体として $O(\varepsilon(G^0_{cls})|E|)$ である．
  図 8 行 10 の $out_{s+1}(p)$ 内の各タスク $n^{s+1}_{p'}$ についての $tlevel$ 値更新にはまず，$n^{s+1}_{p'}$ とは先行関係のないタスク集合を特定する必要がある．これには $out_{s+1}(p)$ 内の各タスクにつき，$cls_{s+1}(p)$ に属するタスクおよびタスク間の辺を走査すればよいので，全体として $O(\varepsilon(G^0_{cls})|V|(|V| + |E|))$ である．そして $out_{s+1}(p)$ に属する各タスク $n^{s+1}_{p'}$ の $tlevel$ 更新には，$cls_{s+1}(p)$ において $n^{s+1}_{p'}$ よりも先行するタスクのタスクサイズを加算すればよく，$|out_{s+1}(p)|$ ステップ要する．図 8 行 10 の計算量は $O(\varepsilon(G^0_{cls})|V|(|V| + |E|))$ の計算量である．
  図 8 行 12–14 では $out_s(p)$ 内の各タスクの出辺を走査すればよく，全体として $O(\varepsilon(G^0_{cls})|E|)$ の計算量である．
  図 8 行 18 では $cls_s(p)$ 内のタスクおよびタスク間の辺を 1 度走査し，そして各タスクの $bevel$ 値を更新すればよいので，$O(\varepsilon(G^0_{cls})(|V| + |E|))$ の計算量である．
  図 8 行 21–25 では $n^{s+1}_{p'} \in in_{s+1}(p)$ のタスク $n^{s+1}_{p'}$ について，$pred(n^{s+1}_{p'})$ に属するタスク $n^{s+1}_{k'}$ の $blevel$ を更新する．そのため，$in_{s+1}(p)$ 内のタスクとその入力辺，および $pred(n^{s+1}_{p'})$ に属するタスクとその出力辺を走査すればよいので $O(\varepsilon(G^0_{cls})(|V| + |E|))$ の計算量である．
  図 8 行 26–30 では $RDY_{candidate,s+1}$ 内のクラスタ $cls_{s+1}(k)$ それぞれに対し，$n^{s+1}_{k'} \in top_{s+1}(k)$ 内の各タスクについて $|pred(n^{s+1}_{k'})|$ ステップによる RDY_{s+1} に入るかどうかのチェックを行う．RDY_{s+1} 内のタスク間で LV_{s+1} の降順を維持していれ

ば，$\log|RDY_{s+1}|$ ステップで RDY_{s+1} へのクラスタ追加を行う．全体としては $O(\varepsilon(G^0_{cls})|E|\log|V|)$ の計算量である．ゆえに，図 5 行 5 の処理の計算量は図 8 行 10 の処理が支配的であり，$O(\varepsilon(G^0_{cls})|V|(|V|+|E|))$ である．

以上より，提案手法の計算量は図 5 行 5 の処理が支配的であり，$O(\varepsilon(G^0_{cls})|V|(|V|+|E|))$ である．

6. 評　　価

ここでは，提案手法によって得られる各種性能について，シミュレーションを用いて従来手法と比較した．

6.1 評価内容

この評価の目的は，δ_{opt} に基づいて $sl_w(G^S_{cls})$ を減少させることがスケジュール長 $sl(G^S_{cls})$ の短縮につながるかどうかと，提案手法によって各プロセッサがどれだけ活用されるかを確かめることである（ここで S は，式 (6) を満たすものとする）．そこで，以下の項目について評価を行った．

(1)　各クラスタにおける，タスク構成の比較
(2)　CCR [2),23)] を変動させた場合の，$sl_w(G^S_{cls})$ と $sl(G^S_{cls})$ の比較（同一クラスタ数）
(3)　CCR を変動させた場合の，$sl(G^S_{cls})$ の比較（クラスタ数を変動）
(4)　異なるスケジュール方針によるスケジュール長の比較
(5)　δ_{opt} の適切性の評価
(6)　アルゴリズム走行時間の比較
(7)　特定アプリケーションの DAG を用いた，プロセッサ利用率（6.10 節で定義）の比較

DAG の特徴を表す指標としてよく採用されているのが CCR（Communication to Computation Ratio）[2),23)] である．CCR とはタスク間のデータサイズの総和のタスクサイズの総和に対する比，または平均データサイズと平均タスクサイズとの比で定義される．また，CCR 値の範囲を 0.1 から 10 までとすることによって様々な DAG の特徴が定義される [2),23)]．本実験においても，CCR に基づいて DAG を生成した．

提案手法では，各クラスタサイズが δ_{opt} 以上となるまでタスク集約を行い，少ないプロセッサ数でできるだけスケジュール長を最小化することを目的としている．まず (1) では，提案手法のタスク集約方針の目的である $sl_w(G^S_{cls})$ を減少させる要素についての比較を行う．この比較によって，なぜ $sl_w(G^S_{cls})$ が減少するのか（または増加するのか）を明らかにする．(2) では，提案手法によって決められたプロセッサ数で，$sl_w(G^S_{cls})$ と $sl(G^S_{cls})$ に関する従来手法との比較を行う．しかしながら，(2) で決められたプロセッサ数で，従来手法によるスケジュール長が最も短縮されているときのプロセッサ数かどうかは不明である．そのため，(3) では，提案手法で決まった $sl(G^S_{cls})$ およびプロセッサ数が，従来手法のどのプロセッサ数に相当するのかを調べる．すなわち，提案手法によって得られるプロセッサ数が，従来手法において $sl(G^S_{cls})$ と同等の値が得られる場合のプロセッサ数に比べてどの程度少ないのかを評価する．また，スケジュール長は，各タスクの実行順，すなわちスケジュール方針にも依存する．そこで (4) として，提案手法によるタスククラスタリングの後，複数のスケジュール方針を提供した場合のスケジュール長を比較する．これにより，提案手法によるタスククラスタリングとスケジュール方針との関係を明らかにする．(5) では，式 (27) で得られたクラスタサイズの下限値 δ_{opt} が，他の下限値の場合と比べてどの程度スケジュール長が抑えられうるのかを評価する．(6) では，5 章で述べたタスククラスタリングアルゴリズムと従来手法との走行時間に関する比較を行う．(7) では提案手法を特定アプリケーションに適用させた場合の性能を評価するために，ガウス消去法（GE）DAG [16)] と FFT DAG [18)] を入力として用いた場合，どの程度プロセッサが効率的に利用されるのかを比較する．

表 3　ランダム DAG 生成時の各値の設定方針
Table 3　Configuration policies for each values in a random DAG.

Parameter	Policy for Assigning Value
$w(n^0_k)$, $n^0_k \in V_0$	Max/Min Ratio = 100
$c(e^0_{k,l})$, $e^0_{k,l} \in E_0$	Max/Min Ratio = 100
Parallelism Factor (PF) [20),21)]	$\frac{\sqrt{\lvert V_0\rvert}}{\alpha}$, where α is selected randomly from 0.5, 1.0 and 2.0
Out Degree of each Task	Rondomly selected from 1 to 5 for each Task.
CCR [2),23)]	Each random DAG's CCR is within $[0.1, 10]$.

6.2 シミュレーション環境

6.1 節 (1) から (6) についてはシミュレーションの入力 DAG としてランダム DAG を生成し，(7) については GE DAG [16)]，FFT DAG [18)] を入力 DAG として生成した．

まず，ランダム DAG の生成方法について述べる．**表 3** に，ランダム DAG の生成方針を示す．ランダム DAG を生成するとき，各タスクのタスクサイズ，データサイズそれぞれの最大値，最小値の比を 100 とし，(1) では一様分布，(2) では一様分布と正規分布，(3) から (6) では一様分布の乱数に従って値を割り当てた．

DAG の並列度を決めるため，Parallelism Factor（PF）によって DAG の並列度 α を定義し，DAG の深さ（1 経路上の最大タスク数）を $\frac{\sqrt{|V_0|}}{\alpha}$ とした [20),21)]．ここで α の値を 0.5,

1.0，2.0[21] の中から選択される，一様分布に従った乱数によって DAG の深さを決めた．また，文献 21) のように，各タスクからの出辺の数を 1 から 5 までの整数からの一様分布に従った乱数で決めることにより，DAG を生成した．さらに，各タスクについて，その出次数を 1 から 5 までの一様分布に従った乱数を割り当てた．

タスクスケジューリングにおいて，特に通信回数の多いアプリケーション内のタスクをスケジュールする場合，いかにして通信遅延を最小化するかが重要となる．そこで，ランダム DAG 生成時では辺の数によらずにデータサイズを変動させる方針とする．ここでは CCR を平均データサイズと平均タスクサイズとの比とし，CCR が 0.1 から 10 以内に収まるように DAG を生成した．

シミュレーション環境を J2SE1.6_03 で作成した．また，シミュレーション実行環境については JRE1.6.0_03，OS は Windows XP SP3，CPU は Intel Core 2 Duo 2.66 GHz，メモリサイズは 2.0 GB である．

6.3 比較対象

6.1 節における比較対象を，1. タスククラスタリングの後にクラスタの集約を行うもの，2. クラスタの集約のみを行うもの，という 2 つの基準で選択した．このうち，1 は CASS-II[11] によるタスククラスタリングの後に Load Balancing（LB）[8] を行うもの（CASS-II + LB）[8]，DSC[4] によるタスククラスタリングの後に Cluster Merging（CM）を行うもの（DSC + CM）[7] があげられる．一方，2 としては提案手法によるタスククラスタリングと LB である．

なお，この節を通し，提案手法では S 回のタスク集約によって式 (6) において $\delta = \delta_{opt}$ の場合を満たすものとする．一方，6.1 節 (2)，(3)，(4)，(6)，(7) の実験では，従来手法では指定プロセッサ数となるまでタスク集約を行う．式 (1) より，1 度のタスク集約により，クラスタ数は 1 つ減少するので，提案手法において S 回のタスク集約後のクラスタ数が従来手法によって生成されたクラスタ数と等しければ，従来手法におけるタスク集約回数も S 回である．従来手法では，S 回のタスク集約後に全クラスタサイズが δ_{opt} 以上であるとは限らないが，ここでは各手法における S 回のタスク集約後の DAG を共通して G^S_{cls} と記す．

6.1 節 (2)，(3)，(5)，(7) における各手法適用後の $sl(G^S_{cls})$ の算出には，RCP（Ready Critical Path）[10] スケジューリングを用いた．

6.4 各クラスタ内におけるタスク構成の比較

6.4.1 タスク間先行関係の比較

5 章で述べたアルゴリズムの目的は，各クラスタサイズが δ_{opt} 以上としつつ，$sl_w(G^S_{cls})$ を最小化することである．このとき，5.1 節（ii）においてタスク集約方針としてできるだけタスク間の先行関係を保つことが重要であることを述べた．そのため，まずは生成された各クラスタにおいて，どの程度タスク間の先行関係が保たれているかを評価する．表 1 の定義より，$sl_w(G^s_{cls})$ は各クラスタ $cls_s(i)$ の $LV_s(i)$ の値から決まる．さらに，$cls_s(i)$ に属する各タスク n^s_k について，タスク集約によって n^s_k と先行関係のないタスクが $cls_{s+1}(i)$ に含まれると，$S(n^{s+1}_k, i)$ が増加する（$S(n^s_k, i) < S(n^{s+1}_k, i)$）．さらに，$tlevel(n^{s+1}_k)$ も増加する（$tlevel(n^s_k) < tlevel(n^{s+1}_k)$）．特に $out_{s+1}(i)$ 内のタスクの $tlevel$ 値が増加すると，他クラスタの TL_{s+1} 値も増加する（$TL_s < TL_{s+1}$）可能性がある．そこで，各クラスタ $cls_S(i)$ について $out_S(i)$ に属するタスクの前に最大何個のタスクが実行されうるかを比較する．そのために 6.2 節の方針に従ってランダム DAG を生成し，さらに以下の $N_{S,ave}$ によって従来手法と提案手法で比較した．

$$N_{S,ave} = \frac{1}{w_{average}} \sum_{cls_s(i) \in V^S_{cls}} \sum_{n^S_k \in out_s(i)} \frac{S(n^S_k, i)}{|out_s(i)|}. \tag{44}$$

$w_{average}$ はタスクサイズの平均値であり，$N_{S,ave}$ は，各クラスタ $cls_S(i)$ に属するタスク n^S_k について，$S(n^S_k, i)$ に含まれるタスク数の平均値である．この値が大きければ，1 クラスタに先行関係のないタスクが多く含まれてしまい，$out_S(i)$ に属するタスクの $tlevel$ 値，さらには $BL_S(i)$ の値が大きい可能性がある．

図 9 に，$N_{S,ave}$ の比較結果を示す．図 9 から分かるとおり，提案手法による $N_{S,ave}$ が，最も低い．そのため，提案手法では，従来手法よりも先行関係のあるタスクをできるだけ同一クラスタに集約しているといえる．

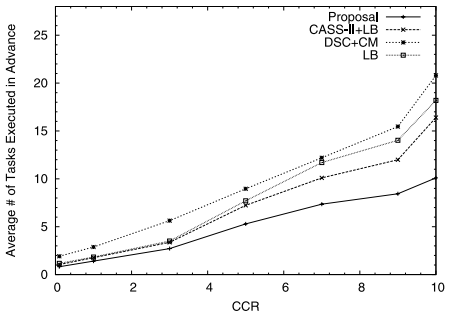


図 9　$N_{S,ave}$ の比較
Fig. 9　Comparison of $N_{S,ave}$.

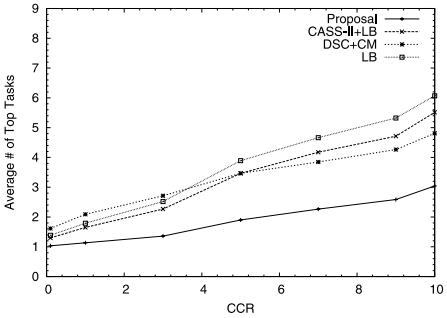


図 10　top_S の要素数の比較
Fig. 10　Comparison of $|top_S|$ for each cluster.

表 4　CCR 変動による，$sl_w(G^S_{cls})$ と $sl(G^S_{cls})$ の比較（各タスクサイズおよびデータサイズは一様分布の乱数に従う）
Table 4　Comparison of $sl_w(G^S_{cls})$ and $sl(G^S_{cls})$ with varying CCR (Each task size and data size is assigned according to random value in uniform distribution).

No.	$\|V\|$	CCR	$\|V^S_{cls}\|$	$sl_w(G^S_{cls})$ Ratio to A				$sl(G^S_{cls})$ Ratio to A			
				A. Proposal	B. CASS-II+LB	C. DSC+CM	D. LB	A. Proposal	B. CASS-II+LB	C. DSC+CM	D. LB
1	500	0.1	147	1.000	1.235	1.517	1.422	1.000	1.008	1.112	1.032
2	500	1.0	45	1.000	1.315	1.506	1.359	1.000	1.192	1.230	1.274
3	500	3.0	26	1.000	1.219	1.301	1.416	1.000	1.162	1.065	1.555
4	500	5.0	22	1.000	1.237	1.432	1.579	1.000	1.281	1.334	1.797
5	500	8.0	17	1.000	1.268	1.339	1.435	1.000	1.186	1.288	1.813
6	500	10.0	16	1.000	1.348	1.340	1.401	1.000	1.280	1.438	1.949
7	1000	0.1	271	1.000	1.316	1.439	1.506	1.000	1.021	1.114	1.042
8	1000	1.0	82	1.000	1.382	1.484	1.342	1.000	1.176	1.185	1.340
9	1000	3.0	48	1.000	1.222	1.442	1.429	1.000	1.219	1.326	1.614
10	1000	5.0	38	1.000	1.374	1.391	1.443	1.000	1.327	1.262	1.688
11	1000	8.0	31	1.000	1.237	1.288	1.378	1.000	1.248	1.350	1.796
12	1000	10.0	28	1.000	1.272	1.324	1.377	1.000	1.231	1.477	1.916

6.4.2　クラスタ内における，最早実行開始できるタスク数の比較

各クラスタ $cls_s(i)$ について，$top_s(i)$ に複数のタスクが属している場合，それらは互いに先行関係はない．そのため，タスク集約によって $|top_{s+1}(i)| > |top_s(i)|$ となれば，$top_{s+1}(i)$ 以外のタスクの $tlevel$ 値が増加しうる．このことが，$sl_w(G^S_{cls})$ の増分に影響を与えるものと考えられる．そこで，従来手法と提案手法それぞれで得られるクラスタについて，top_S に属するタスク数の比較を行った．**図 10** に，1 クラスタあたりの top_S に属するタスク数の比較結果を示す．図中，縦軸は各クラスタの top_S に属するタスク数の平均値を示している．この図から分かるとおり，CCR が低いときはいずれの手法も top_S に属するタスク数はほぼ 1 であるが，CCR が大きくなるに従って増加している．提案手法の場合，top_S に属するタスク数が増加する場合は，図 7 (e) の場合である．提案手法では，まずは図 7 (a), (b), (c), (d) 条件で $target_s$ を選択できるか検査し，それでも選択できなければ (e) によって $target_s$ 選択処理を行う．この方針でできるだけ top_{s+1} の要素数を増加させないようにしているので，各クラスタ $cls_{s+1}(i)$ について，$top_{s+1}(i)$ 以外のタスクの $tlevel$ を最小化しようとする．その結果，V^S_{cls} において各タスクの $tlevel$ 値はできるだけ最小となり，結果的に図 9 における $N_{S,ave}$ が，他手法の $N_{S,ave}$ よりも低くなっているものと考えられる．

6.5　CCR 変動による，$sl_w(G^S_{cls})$ と $sl(G^S_{cls})$ の比較

この実験では，固定のプロセッサ数という条件の下で，様々な CCR を持つ DAG における $sl_w(G^S_{cls})$ と $sl(G^S_{cls})$ の比較を行った．提案手法では各クラスタのサイズが δ_{opt} 以上となるようにタスク集約を行った後にクラスタ数，すなわちプロセッサ数が決まる．一方，他手法ではこのときに決定したプロセッサ数となるまでクラスタ集約処理を行う．これらの条件でランダム DAG を 100 個生成し，$sl_w(G^S_{cls})$ と $sl(G^S_{cls})$ の平均値によって比較を行った．CCR は，各タスクサイズ，データサイズによって決まる．様々な値の分布を考慮に入れるために，タスクサイズ，データサイズに対して一様分布に従った乱数を割り当てた場合，および正規分布に従った乱数を割り当てた場合に分けて評価を行った．

表 4 に，タスクサイズおよびデータサイズが一様分布である場合の比較結果を示す．タスク数 $|V|$ を 500 または 1000 とし，それぞれ CCR を変動させた．列「$|V^S_{cls}|$」の値は，δ_{opt} に従って提案手法によるタスク集約後に得られたクラスタ数を示す．列「$sl_w(G^S_{cls})$ Ratio to A」では，提案手法による $sl_w(G^S_{cls})$ に対する比を示す．列「$sl(G^S_{cls})$ Ratio to A」では，提案手法による $sl(G^S_{cls})$ に対する比を示す．表 4 より，提案手法による $sl_w(G^S_{cls})$ は総じて，他手法による $sl_w(G^S_{cls})$ よりも低くなっていることが分かる．また，No.1, 7（CCR = 0.1）のような CCR が極端に低い場合であっても提案手法による $sl(G^S_{cls})$ が低いが，他の CCR の場合よりも $sl(G^S_{cls})$ 間の差は小さい．CCR = 0.1 の場合では，他の CCR の場合よりも 1 度のデータ転送時間（データサイズ）の 1 タスクの実行時間（タスクサイズ）に対する割合が低く，かつクラスタ数が多い．そのため，1 クラスタ内のタスク数も少ないので，並列性（同時実行可能なタスク数）が他の CCR の場合よりも多いものと考えられる．よって，いずれの手法においても並列性が保たれており，かつデータ転送遅延の影響が小さいと考え

表 5　CCR 変動による，$sl_w(G^S_{cls})$ と $sl(G^S_{cls})$ の比較（各タスクサイズおよびデータサイズは正規分布の乱数に従う）
Table 5　Comparison of $sl_w(G^S_{cls})$ and $sl(G^S_{cls})$ with varying CCR (Each task size and data size is assigned according to random value in normal distribution).

No.	$\|V\|$	α	β	CCR	$\|V^S_{cls}\|$	$sl_w(G^S_{cls})$ Ratio to A				$sl(G^S_{cls})$ Ratio to A			
						A. Proposal	B. CASS-II+LB	C. DSC+CM	D. LB	A. Proposal	B. CASS-II+LB	C. DSC+CM	D. LB
1	1000	0.1	0.1	0.1	205	1.000	1.510	1.450	1.366	1.000	1.049	1.104	1.056
2	1000	0.1	0.1	3.0	39	1.000	1.197	1.351	1.348	1.000	1.163	1.183	1.571
3	1000	0.1	0.1	10.0	22	1.000	1.161	1.207	1.269	1.000	1.335	1.329	1.898
4	1000	0.1	0.9	0.1	276	1.000	1.285	1.451	1.333	1.000	1.020	1.090	1.024
5	1000	0.1	0.9	3.0	59	1.000	1.277	1.490	1.389	1.000	1.214	1.313	1.522
6	1000	0.1	0.9	10.0	34	1.000	1.415	1.412	1.475	1.000	1.398	1.519	1.966
7	1000	0.5	0.5	0.1	300	1.000	1.270	1.365	1.492	1.000	1.000	1.073	1.027
8	1000	0.5	0.5	3.0	56	1.000	1.256	1.567	1.599	1.000	1.187	1.240	1.632
9	1000	0.5	0.5	10.0	30	1.000	1.412	1.311	1.374	1.000	1.256	1.393	1.890
10	1000	0.9	0.1	0.1	251	1.000	1.484	1.375	1.383	1.000	1.019	1.086	1.048
11	1000	0.9	0.1	3.0	47	1.000	1.197	1.310	1.380	1.000	1.128	1.208	1.540
12	1000	0.9	0.1	10.0	25	1.000	1.294	1.333	1.440	1.000	1.299	1.512	2.100
13	1000	0.9	0.9	0.1	357	1.000	1.236	1.448	1.458	1.000	1.002	1.048	1.022
14	1000	0.9	0.9	3.0	61	1.000	1.278	1.494	1.561	1.000	1.213	1.212	1.727
15	1000	0.9	0.9	10.0	36	1.000	1.287	1.414	1.459	1.000	1.213	1.585	2.026

られる．いずれの手法においても他の CCR の場合よりは良いスケジュール長が得られているので，互いのスケジュール長が漸近しているものと考えられる．しかしながら，提案手法によって決定されたクラスタ数であれば，提案手法によって総じて $sl(G^S_{cls})$ が低くできることが分かった．

次に，タスクサイズおよびデータサイズが正規分布の乱数に従う場合の比較結果について述べる．実行アプリケーションによっては，これらの値の分布に様々な偏りがある場合が考えられる．そこで，まずはタスクサイズおよびデータサイズの最大値と最小値を決めた（表 3 に従い，タスクサイズおよびデータサイズの最大値と最小値の比は 100 とし，これらのサイズはあらかじめ設定した）．そしてタスクサイズおよびデータサイズの値の分布に偏りを持たせるために，以下のように平均値 μ_{task}，μ_{data} および標準偏差 σ_{task}，σ_{data} を決めた．

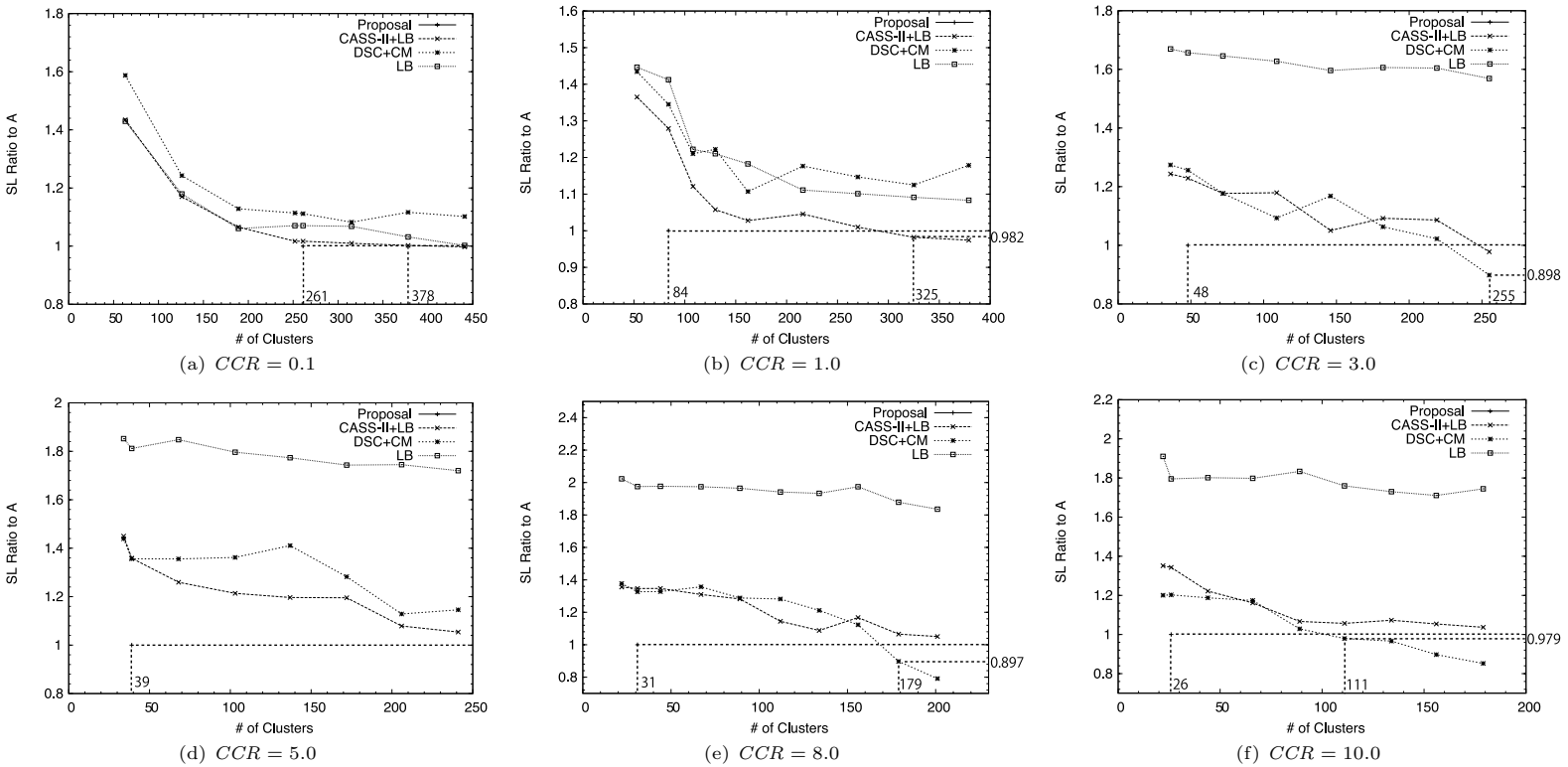
$$\mu_{task} = \min_{n^0_k \in V_0}\left\{w(n^0_k)\right\} + \left(\max_{n^0_k \in V_0}\left\{w(n^0_k)\right\} - \min_{n^0_k \in V_0}\left\{w(n^0_k)\right\}\right)\alpha,$$
$$\mu_{data} = \min_{e^0_{k,l} \in E_0}\left\{c(e^0_{k,l})\right\} + \left(\max_{e^0_{k,l} \in E_0}\left\{c(e^0_{k,l})\right\} - \min_{e^0_{k,l} \in E_0}\left\{c(e^0_{k,l})\right\}\right)\beta,$$

$$\sigma_{task} = \frac{1}{3}\max\left\{\mu_{task} - \min_{n^0_k \in V_0}\left\{w(n^0_k)\right\}, \max_{n^0_k \in V_0}\left\{w(n^0_k)\right\} - \mu_{task}\right\},$$
$$\sigma_{data} = \frac{1}{3}\max\left\{\mu_{data} - \min_{e^0_{k,l} \in E_0}\left\{c(e^0_{k,l})\right\}, \max_{e^0_{k,l} \in E_0}\left\{c(e^0_{k,l})\right\} - \mu_{data}\right\}. \quad (45)$$

そして，α，β それぞれの値を 0.1，0.5，0.9 とした場合の $sl_w(G^S_{cls})$ と $sl(G^S_{cls})$ を算出した．**表 5** に比較結果を示す．表中，提案手法による $sl_w(G^S_{cls})$ は総じて低い．一方，$sl(G^S_{cls})$ については，CCR が 0.1 である場合（No.1，4，7，10，13）よりも CCR が 1 以上のときの方が，A と B，C，D との差が大きい．これは表 4 における CCR = 0.1 の場合と同様，並列性が保たれることとデータ転送遅延の影響が少ないことが理由として考えられる．以上の結果より，タスクサイズおよびデータサイズの偏りのある DAG に対しても，CCR が 0.1 から 10 の範囲に収まる DAG であれば提案手法によって $sl(G^S_{cls})$ が低くなっているといえる．

6.6　必要プロセッサ数の比較

この実験では，提案手法によって得られたプロセッサ数が，従来手法における同等（またはそれ以下）の $sl(G^S_{cls})$ が得られるときのプロセッサ数と比較してどの程度低いのかを評価した．そこで，まずは提案手法を実行して必要プロセッサ数および $sl(G^S_{cls})$ を求める．



(a) $CCR = 0.1$　(b) $CCR = 1.0$　(c) $CCR = 3.0$

(d) $CCR = 5.0$　(e) $CCR = 8.0$　(f) $CCR = 10.0$

図 11　プロセッサ数変動による，スケジュール長の比較（$|V| = 1000$）
Fig. 11　Comparison of schedule length with changing the number of processors ($|V| = 1000$).

その後，6.3 節であげた従来手法（B. CASS-II + LB，C. DSC，D. LB）において，必要プロセッサ数を様々な値（提案手法で得られたクラスタ数よりも小さい場合と大きい場合に分かれるように）に設定した．そして，B，C，D ではこの必要プロセッサ値となるまでクラスタ集約を行った．6.1 節の方針でランダム DAG を 100 個生成し，プロセッサ数および $sl(G^S_{cls})$ について，それぞれ平均値による比較を行った．ここでプロセッサ利用率を (スケジュール長)/(プロセッサ数) とする．そして提案手法によるプロセッサ利用率が，他手法に対してどの程度なのか（プロセッサ利用率の比）で比較する．

図 11 に，必要プロセッサ数と $sl(G^S_{cls})$ との比較結果を示す．図中，(a) から (f) になるに従って CCR の大きい DAG を入力として使用している．(a)–(f) 中，横軸はクラスタ数を示し，縦軸は B，C，D で得られた従来手法による $sl(G^S_{cls})$ の，提案手法による $sl(G^S_{cls})$ に対する比を示す．また，(a)–(f) 中，提案手法によって得られたクラスタ数はそれぞれ 261，84，48，39，31，26 である．(a) では，提案手法によって得られた $sl(G^S_{cls})$ は，B（CASS+LB）がプロセッサ数 378 のときとほぼ同等である．したがって提案手法の方が，B よりもプロセッサ利用率が $378/261 \fallingdotseq 1.45$ 倍良いということになる．一方，(b) ではプロセッサ数が 325 で B（CASS-II + LB）を実行したときに，提案手法で得られた $sl(G^S_{cls})$ の 0.982 倍となった．よって，プロセッサ利用率の比は $325 \times 0.982/84 \fallingdotseq 3.80$ である．(c) では，プロ

セッサ数が 255 のときに C を実行したとき，提案手法に対して 0.898 倍の $sl(G^S_{cls})$ である．したがって，プロセッサ利用率の比は，$255 \times 0.898/48 = 4.77$ である．(d) では，提案手法による $sl(G^S_{cls})$ よりも低い場合は観察されなかった．(e)，(f) ではそれぞれ，プロセッサ数が 179，111 のときに提案手法の $sl(G^S_{cls})$ よりも低くなっており，プロセッサ利用率の比は，(e) では約 5.18，(f) では約 4.18 である．以上の結果から，提案手法によって従来手法よりも少ないプロセッサ数でスケジュール長が低くなっているという結果が得られた．

6.7　異なるスケジュール方針によるスケジュール長の比較

6.3 節で述べたとおり，6.1 節 (2)，(3)，(5)，(7) における各比較では，RCP スケジューリングを用いてスケジュール長を算出している．タスク実行順によってもスケジュール長は変動しうるので，他のスケジュール方針を用いた場合，提案手法と従来手法におけるスケジュール長にどのような影響があるのかを評価した．従来のタスククラスタリングのうち，スケジュール長に基づいてタスク集約優先度を決めているものがある[4),9),12)]．これら従来のタスククラスタリングではタスク集約を行うごとにスケジュール長を計算する．このとき，他のクラスタに集約されていないタスクに対し，Down-Rank（START タスクから当該タスクまでの最大経路長）または Up-Rank（当該タスクから END タスクまでの最大経路長）を割り当てる[⋆1]．そして，Down-Rank または Up-Rank の降順にスケジュールする．ここでは，これら従来手法で用いられているスケジュール方針を評価対象とした．

ランダム DAG を表 3 に従って 100 個生成し，スケジュール長の平均値による比較を行った．**表 6** に，比較結果を示す．表 6 では，Down-Rank の降順および Up-Rank の降順に従ってそれぞれスケジュールした場合のスケジュール長の比較結果を示している．どの CCR の場合においても，提案手法の方が良いスケジュール長が得られている．しかしながら CCR = 0.1 のときは，互いにスケジュール長が漸近している．これは表 4 における CCR = 0.1 の場合と同様，並列性が保たれることとデータ転送遅延の影響が少ないことが理由として考えられる．以上の結果より，RCP 以外のスケジュール方針を用いた場合においても，提案手法によって従来手法よりも良いスケジュール長が得られることが分かった．

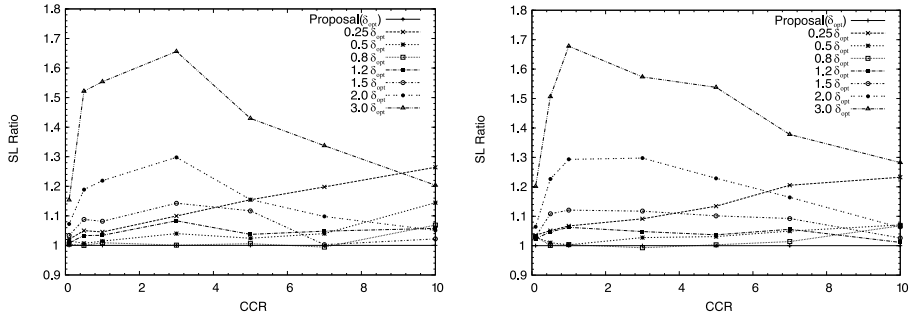
6.8　δ_{opt} の適切性の評価

式 (27) で得られた δ_{opt} はタスククラスタリング前に決められる値であり，近似的な最適値である．そのため，他のクラスタサイズの下限値の場合と比べて，δ_{opt} の場合に得られるスケジュール長がどの程度短縮されているか（または増加しているか）を評価した．比較対象となるクラスタサイズの下限値をそれぞれ $0.25\delta_{opt}$，$0.5\delta_{opt}$，$0.8\delta_{opt}$，$1.2\delta_{opt}$，$2\delta_{opt}$，$3\delta_{opt}$ とした．比較結果を，**図 12** に示す．図中，(a) は $|V| = 500$，(b) は $|V| = 1000$ の場合である．横軸は CCR，縦軸は提案手法（すなわちクラスタサイズの下限値が δ_{opt} の場

⋆1 文献によっては Down-Rank/Up-Rank を様々な名前で定義しており定訳はない．したがって本論文では便宜上，Down-Rank/Up-Rank と呼んでいる．

表 6　異なるスケジュール方針によるスケジュール長の比較
Table 6　Comparison of schedule length with two scheduling policies.

No.	CCR	Scheduling	$\lvert V^S_{cls}\rvert$	$sl(G^S_{cls})$ Ratio to A			
				A. Proposal	B. CASS-II+LB	C. DSC+CM	D. LB
1	0.1	Down-Rank [4)]	267	1.000	1.029	1.121	1.033
2	0.5		115	1.000	1.055	1.146	1.157
3	1		83	1.000	1.122	1.145	1.281
4	3		50	1.000	1.202	1.182	1.561
5	5		40	1.000	1.184	1.255	1.571
6	7		33	1.000	1.166	1.202	1.678
7	10		28	1.000	1.190	1.436	1.872
8	0.1	Up-Rank [9),12),21)]	277	1.000	1.059	1.112	1.068
9	0.5		119	1.000	1.152	1.192	1.205
10	1		85	1.000	1.205	1.239	1.388
11	3		50	1.000	1.295	1.527	1.678
12	5		39	1.000	1.391	1.555	1.870
13	7		33	1.000	1.429	1.422	1.926
14	10		28	1.000	1.411	1.391	1.932



(a) $|V| = 500$　　(b) $|V| = 1000$

図 12　δ_{opt} の適切性
Fig. 12　Optimality about δ_{opt}.

表 7　クラスタリング後における，non-linear なクラスタ数の内訳
Table 7　Breakout of ♯ of non-linear clusters after task clustering.

No.	$\|V\|$	CCR	♯ of non-lienar clusters on $sl_w(G^S_{cls})$/♯ of all clusters on $sl_w(G^S_{cls})$							
			$0.25\delta_{opt}$	$0.5\delta_{opt}$	$0.8\delta_{opt}$	δ_{opt}	$1.2\delta_{opt}$	$1.5\delta_{opt}$	$2.0\delta_{opt}$	$3.0\delta_{opt}$
1	500	0.1	5/30	4/23	3/19	3/16	3/14	3/11	2/7	2/5
2	500	0.5	3/26	3/17	2/13	2/11	1/9	2/7	2/5	1/3
3	500	1.0	3/20	3/14	2/11	2/8	2/6	1/5	1/3	1/2
4	500	3.0	3/15	3/12	1/8	1/6	2/5	2/4	1/3	1/1
5	500	5.0	2/13	2/10	2/7	2/5	1/4	1/3	1/2	1/1
6	500	7.0	2/11	2/8	2/6	2/4	2/4	1/2	1/2	1/1
7	500	10.0	3/10	2/7	1/5	1/4	1/3	1/2	1/1	1/1
8	1000	0.1	2/48	5/36	7/29	5/26	5/19	4/15	3/12	3/10
9	1000	0.5	5/36	6/31	4/25	3/22	2/14	2/12	2/9	2/8
10	1000	1.0	7/31	4/27	3/21	2/18	3/16	1/9	2/7	2/5
11	1000	3.0	5/26	3/22	2/18	2/15	2/13	3/8	2/5	1/3
12	1000	5.0	3/22	2/18	1/14	1/12	2/10	2/6	1/5	2/3
13	1000	7.0	3/18	3/15	2/11	2/10	2/8	2/5	1/4	1/2
14	1000	10.0	2/14	2/12	1/10	1/8	1/6	2/4	1/3	1/1

合）によって得られたスケジュール長を 1 とした場合の，他のクラスタサイズにおけるスケジュール長の比を示している．(a) において CCR = 7.0 の場合，$0.8\delta_{opt}$ の方が δ_{opt} の場合よりもスケジュール長がわずかに短縮されているが，それ以外ではほぼ同等かもしくは δ_{opt} の場合の方がスケジュール長は小さい．(b) においては，CCR = 3.0 のときに $0.8\delta_{opt}$ の場合の方が δ_{opt} の場合よりもわずかにスケジュール長が小さいが，それ以外ではほぼ同等かもしくは δ_{opt} の場合の方がスケジュール長が小さい．

さらに，**表 7** に，$sl_w(G^S_{cls})$ を構成するクラスタ集合のうちで，non-linear なクラスタ数の内訳を示す．式 (27) では，式 (23) において $y = 1$ とした場合の δ の値である．クラスタサイズの下限値が δ_{opt} 未満の場合は $y > 1$ であることに対応し，δ_{opt} より大きい場合は $y < 1$ であることに対応する．たとえば前者の場合，$0.25\delta_{opt}$，$0.5\delta_{opt}$ はそれぞれ式 (23) において $y = 16$，$y = 4$ と見なすことができる．表 7 において，δ_{opt} の場合では non-linear なクラスタ数が 1 よりも多くなっている．このように y の値とクラスタリング後の non-linear なクラスタ数は一致するとは限らない．

まず，クラスタサイズの下限値が δ_{opt} よりも大きな場合（すなわち $1.2\delta_{opt}$，$1.5\delta_{opt}$，$2.0\delta_{opt}$，$3.0\delta_{opt}$）について述べる．この場合では，図 12 の結果より，すべての CCR 値において δ_{opt} の場合よりもスケジュール長が大きい．CCR が増加すれば，non-linear なクラスタ内で互いに先行関係のないタスク数が，δ_{opt} における non-linear なクラスタ内の先行関係のないタスク数よりも多くなりうる．このことが，δ_{opt} の場合よりもスケジュール長が大きいことの要因として考えられる．図 12 (a) の CCR = 0.1，0.5，1.0，3.0 や図 12 (b) の CCR = 0.1，0.5，1 ではスケジュール長の比が増加しているのに対し，それ以上の CCR の場合ではスケジュール長の比が減少している．これは，non-linear なクラスタ内の先行関係のないタスク数が増加することによるスケジュール長へ影響が大きいことを示している．CCR が大きくなれば先行関係のないタスク数も多くなるが，クラスタ数の下限値が大きい分，データ転送の局所化によるスケジュール長への影響が大きくなっているものと考えられる．その結果，CCR が増加すればスケジュール長の比が減少している．一方，表 7 から，δ_{opt} の場合よりも $sl_w(G^S_{cls})$ 内のタスクが属するクラスタ数および non-linear なクラスタ数は，総じて δ_{opt} における non-linear なクラスタ数以下である．さらに図 12 から，CCR が小さな場合（CCR が 0.1，0.5）であってもスケジュール長は δ_{opt} の場合よりも大きい．このことから，たとえ CCR が小さくても，クラスタサイズの下限値が大きい分，並列度（同時実行可能なタスク数）は δ_{opt} の場合よりも小さくなり，結果としてスケジュール長が δ_{opt} の場合よりも大きくなっているものと考えられる．そのため，クラスタサイズの下限値が $1.2\delta_{opt}$，$1.5\delta_{opt}$，$2.0\delta_{opt}$，$3.0\delta_{opt}$ の場合（つまり $y < 1$）では，スケジュール長が δ_{opt} の場合よりも大きくなってしまうといえる．

次に，クラスタサイズの下限値が δ_{opt} よりも小さい場合（すなわち $0.25\delta_{opt}$，$0.5\delta_{opt}$，$0.8\delta_{opt}$）を考える．この場合はクラスタサイズの下限値が小さい分，データ転送は局所化されないが，δ_{opt} の場合よりも，non-linear なクラスタ内における，先行関係のないタスク数は少ないものと考えられる．そのため，CCR が増加すれば，局所化されなかったデータ転送遅延によるスケジュール長への影響が大きくなりうるものと考えられる．その結果，図 12 における $0.25\delta_{opt}$，$0.5\delta_{opt}$ では CCR の増加にともなってスケジュール長の比がおおむね増加傾向である．表 7 より，総じて δ_{opt} の場合よりも $sl_w(G^S_{cls})$ 内のタスクが属するクラスタ数および non-linear なクラスタ数は同等かそれ以上である．一方，図 12 より，$0.25\delta_{opt}$，$0.5\delta_{opt}$ の場合，CCR が 0.1，0.5 のような小さな場合であってもスケジュール長が δ_{opt} よりも大きい．これら 2 つの場合では δ_{opt} の場合よりも non-linear なクラスタ数が多い分，並列度（同時実行可能なタスク数）を保つことができていないことと，データ転送の局所化の効果が得られていないことが原因として考えられる．また，クラスタサイズの下限値が $0.8\delta_{opt}$ の場合では，図 12 (a) の CCR = 7.0 および (b) の CCR = 3.0 の場合を除いて δ_{opt} の場合よりもスケジュール長が大きい．特に CCR = 10.0 の場合では δ_{opt} よりも大幅にスケジュール長が大きくなっている．そのため，クラスタサイズの下限値が $0.8\delta_{opt}$ の場

合では，CCR が大きくなればデータ転送の局所化がさらに必要（クラスタサイズの下限値をさらに大きくすべき）ということがいえる．以上より，$y > 1$ の場合においては，ほとんどの CCR において δ_{opt} の場合よりもスケジュール長が大きくなってしまうといえる．

式 (22) で式 (23) を代入すると，式 (22) は y の単調増加となる．したがって，y が増加すれば $sl_w(G^S_{cls})$ の上限値も増加し，実際のクラスタリング後の $sl_w(G^S_{cls})$ が増大しうる．4.8 節での結果より，このことがスケジュール長の下限値および上限値の増大につながりうるものと考えられる．この性質と図 12，表 7 の結果から，式 (23) において $y = 1$ として δ_{opt} を決めることが，スケジュール長を最小化するための準最適なクラスタサイズといえる．

本論文の提案内容の 1 つは，タスククラスタリング前にクラスタサイズを決定するというものである．タスク集約を行うごとに DAG 内の各タスクサイズ，データサイズおよびタスク間の先行関係から次に生成するクラスタサイズを動的に決定すれば，より適切なスケジュール長が得られる可能性がある．しかしながら，これにはタスク集約ごとにクラスタサイズの下限値を算出するという追加の処理が必要である．動的なクラスタサイズの決定処理を行うためには計算量をいかにして抑えるかが問題であり，これは今後の課題である．

以上より，δ_{opt} によってつねにスケジュール長が最小となるわけではないが，δ_{opt} によっておおむね良好なスケジュール長が得られることが分かった．

6.9　アルゴリズム走行時間の比較

ここでは，提案手法が現実的な手法であるかを確かめるためにアルゴリズム走行時間を従来手法と比較した．提案手法の走行時間の計測対象は，図 5 行 0 から行 7 である．すなわち，δ_{opt} の算出や，初期状態の DAG に対する $sl_w(G^S_{cls})$ の算出も計測対象に含めた．従来手法の中で，CASS-II+LB と DSC+CM は事前処理としてクリティカルパス長を算出するので，このような事前処理も計測対象に含めた．すなわち，計測したアルゴリズム走行時間は，各手法ともに事前処理の開始から G^S_{cls} が得られるまでの所要時間である．また，クラスタ生成方針としては，6.5 節（同一クラスタ数で，CCR は変動）と同一とした．そして，表 3 の方針で 100 個生成されたランダム DAG（タスクサイズ，データサイズは一様分布の乱数）を用いて，走行時間の平均値による比較を行った．

表 8 に，走行時間の比較結果を示す．表中，$\varepsilon(G^0_{cls})$ は式 (43) の値であり，この値が大きければ同一クラスタ参照回数（各クラスタにおけるタスク集約回数であり，$pivot_s$ として選択される回数）の上限値が大きくなり，提案手法によるアルゴリズム走行時間も大きくなりうる．提案手法（表 8 の A）では，δ_{opt} が大きくなるとクラスタ数は減少する．そのため，$|V^S_{cls}|$ が小さくなるということは，タスク集約回数が多くなる（$\varepsilon(G^0_{cls})$ も大きくなる）ことに対応する．$|V|$ が 500，1000 のいずれの場合においてもクラスタ数が少なくなるほど走行時間は大きくなっている．また，CASS-II，DSC の計算量はそれぞれ $O(|E|+|V|\log|V|)$ [11]，$O((|V|+|E|)\log|V|)$ [4]，提案手法は 5.8 節で述べたように $O(\varepsilon(G^0_{cls})|V|(|V|+|E|))$ である．タスククラスタリングの計算量だけで比較すると，CASS-II が最も計算量は小さい．しかしながら CASS-II+LB の場合では，CASS-II の後に LB によるクラスタ集約処理を行っており，その結果提案手法よりも走行時間は大きくなっている．DSC も同様，クラスタ集約処理によって提案手法よりも走行時間は大きい．また，LB では，毎回のタスク集約処理時に DAG 全体を走査してクラスタを選択するため，走行時間は大きい．一方，提案手法では表 8 のいずれの場合においても他手法よりも走行時間は小さい．また，6.5 節の結果より，CCR が大きくなるにつれて提案手法による $sl(G^S_{cls})$ が，他手法による $sl(G^S_{cls})$ よりも低くできる．そのため，提案手法により，小さな走行時間で効率の良いスケジュール長が得られるといえる．

表 8　アルゴリズム走行時間の比較（各タスクサイズとデータサイズは一様分布の乱数に従う）
Table 8　Comparison of running time of each algorithm (Each task and data size is assigned according to randam value in uniform distribution).

No.	$\|V\|$	CCR	$\varepsilon(G^0_{cls})$	$\|V^S_{cls}\|$	Algorithm Running Time (ms)			
					A. Proposal	B. CASS-II+LB	C. DSC+CM	D. LB
1	500	0.1	122	147	10	17	29	29
2	500	1.0	420	45	19	33	31	38
3	500	3.0	723	26	24	29	33	38
4	500	5.0	924	22	27	29	38	36
5	500	8.0	1200	17	29	33	38	43
6	500	10.0	1287	16	29	34	38	44
7	1000	0.1	147	271	27	64	97	112
8	1000	1.0	460	82	43	88	102	140
9	1000	3.0	795	48	53	83	109	151
10	1000	5.0	1000	38	57	83	112	143
11	1000	8.0	1310	31	65	84	110	151
12	1000	10.0	1401	28	69	85	112	156

表 8 では，タスクサイズおよびデータサイズが一様分布の乱数に従った値のため，最終的に生成された各クラスタ $cls_S(i)$ について，$cls_S(i)$ を生成するまでに要するタスク集約回数はほぼ同一であると考えられる．そのため，$\varepsilon(G^0_{cls})$ の値がいくら大きくても実際にはこの値の回数に近づくことはほぼないものと考えられる．次に，同一クラスタ参照回数（各クラスタにおけるタスク集約回数であり，$pivot_s$ として選択される回数）に偏りのある場

表 9　アルゴリズム走行時間の比較（各タスクサイズとデータサイズは正規分布の乱数に従う）
Table 9　Comparison of running time of each algorithm (Each task and data size is assigned according to randam value in normal distribution).

No.	$\|V\|$	α	β	CCR	$\varepsilon(G^0_{cls})$	$\|V^S_{cls}\|$	Algorithm Running Time (ms)			
							A.	B.	C.	D.
1	500	0.1	0.9	0.1	87	104	23	32	31	39
2	500	0.1	0.9	3.0	488	35	31	54	39	55
3	500	0.1	0.9	10.0	876	14	54	86	67	47
4	500	0.9	0.1	0.1	121	114	25	38	39	43
5	500	0.9	0.1	3.0	664	22	34	41	44	49
6	500	0.9	0.1	10.0	1083	13	36	46	51	56
7	1000	0.1	0.9	0.1	103	267	39	94	101	134
8	1000	0.1	0.9	3.0	195	58	79	118	124	169
9	1000	0.1	0.9	10.0	328	36	110	162	187	172
10	1000	0.9	0.1	0.1	41	246	31	85	94	125
11	1000	0.9	0.1	3.0	226	49	71	93	109	156
12	1000	0.9	0.1	10.0	413	23	94	117	111	162

合の DAG について，アルゴリズム走行時間の比較を行った．この実験では，$|V| = 500$, 1000 とし，式 (45) に従ってタスクサイズおよびデータサイズは正規分布に従った乱数とした．**表 9** に，比較結果を示す．表 9 中，α および β は式 (45) 内の α および β であり，A, B, C, D は表 8 の A, B, C, D に対応する．表 9 において $\alpha = 0.1$ のとき（No.1, 2, 3, 7, 8, 9），タスクサイズは最小値に集中する．サイズの小さいタスクどうしが同一クラスタに属することになり，図 8 行 10 における $S(n^{s+1}_{p'}, p)$ 更新処理（提案手法の計算量支配的な処理）に時間を要するものと考えられる．そのため，提案手法では，同一 CCR である表 8 No.1, 3, 6, 7, 9, 12 よりも走行時間は大きくなっている．しかしながら表 8 B, C, D の手法は，提案手法よりも走行時間は大きい．以上より，タスクサイズおよびデータサイズに偏りのある場合においても，従来手法よりアルゴリズム走行時間は小さいことが分かった．

6.10　特定アプリケーションにおけるプロセッサ利用率の比較

CCR は一般に，タスク実行時間とデータ転送時間から算出される．すなわち，CCR は DAG の情報（タスクの仕事量とデータサイズ）および実行環境の情報（プロセッサの処理速度・プロセッサ間のデータ転送速度）によって決まる．そこでこの実験では，特定アプリケーションによる DAG において，提案手法によってプロセッサがどの程度有効利用されるのかを評価した．評価に用いる DAG には多くのタスクスケジューリングの評価に用いられている[20),21)] ガウス消去法[16)] と FFT[18)] を用いた．そしてプロセッサ利用率を以下の評価式の値[16)] とし，この値について他手法と比較した．

$$E(|V^S_{cls}|, Algorithm) = \frac{schedule\ length\ by\ 1\ processor}{|V^S_{cls}| \times (sl(G^S_{cls})\ by\ the\ Algorithm)}. \quad (46)$$

式 (46) の値が大きければ，各プロセッサが有効に利用されていることになる．

また，古典的モデルに従った従来のタスククラスタリング（またはクラスタ集約）との性能差を評価するため，この実験においても 2.1 節で述べたモデルに従うものとする．

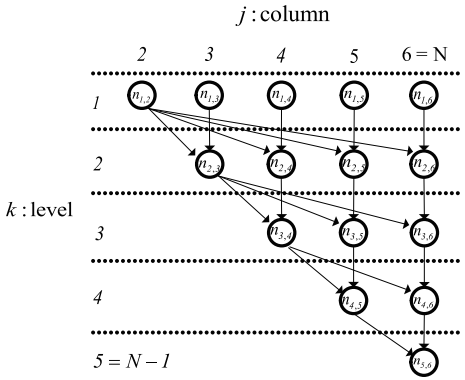


図 13　本実験で用いるガウス消去法の DAG の例（$N = 6$ の場合）
Fig. 13　An Example of Gaussian elimination DAG structure when $N = 6$.

6.10.1　ガウス消去法 DAG を用いたプロセッサ利用率の比較

ガウス消去法では，1 代入文を 1 タスクとした場合，3 重ループによって行列サイズ N に対して $O(N^3)$ で総タスク数が増加する．また，これにともなって辺の数も増加し，通信遅延が非常に大きくなってしまう．そこで，ピボット選択なしの kji ガウス消去法で，かつ上から第 2 層目である **for**$(j = k+1; j \le N; j{+}{+})$ のループ内処理をまとめて 1 タスクとした[16)]．すると，ガウス消去法の DAG 構造（$N = 6$ の場合）は**図 13** となる．図中，レベル k（$1 \le k \le N-1$），列 j（$k+1 \le j \le N$）のタスクを $n^0_{k,j}$ と示す．$n^0_{k,j}$ の処理に要する時間 $w(n^0_{k,j})$，および $n^0_{k,j}$ から $n^0_{k+1,m}$（$j+1 \le m \le N$）へのデータ送信時間を $c(e^0_{(k,j),(k+1,m)})$ とすると，それぞれ以下のように j, m によらない値となる[16)]．

$$w(n^0_{k,j}) = (2(N-k)+1)t_p,\ c(e^0_{(k,j),(k+1,m)}) = \beta + (N-k+1)t_c. \quad (47)$$

式 (47) 中，β はタスク間のデータ送信前に行われる，start-up 時間（プロセッサ間の手続きに必要なデータ転送時間[24)]）である．また，t_p は 1 算術演算にかかる時間，t_c は単位サ

イズあたりのデータ転送時間である．文献 16) では，理論上における speed-up 率が最も高い m 値（t_c と t_p との比に基づいた値で，文献 16) では m と定義）と，Intel iPSC/860 環境における speed-up 率の最も高いときの m 値が一致していることを示している．文献 17) では，式 (47) のように β を考慮してデータ転送時間が線形に振る舞うことを想定したモデルは，ある程度良い近似であると示している．そのため，本論文においても β，t_c，t_p を用いてタスク実行時間およびデータ転送時間を式 (47) に定義する．

β，t_p，t_c は環境に依存する値だが，一般に t_c は t_p よりも大きく[17),24)]，β による遅延もスケジュール長へ影響する[24)]．文献 16) では，文献 17) における Intel iPSC/860 のクラスタ環境を想定し，$t_c/t_p \fallingdotseq 18$，$\beta/t_c \fallingdotseq 42.5$ としている．また文献 17) における Ncube 6400 のクラスタ環境では，$t_c/t_p = 12$，$\beta/t_c \fallingdotseq 41.67$ である[⋆1]．本実験では Intel iPSC/860 および Ncube 6400 のクラスタ環境を想定し，t_c/t_p，β/t_c の値が，これら 2 種類のクラスタ環境を含むように設定した．一般に t_c，β は t_p よりも大きくなり，さらに通信の状態によっても変動しうるので，文献 17) におけるこれら 2 種類の環境よりも t_c/t_p，β/t_c が大きい場合も想定した．しかしながら式 (47) において $c(e^0_{(k,j),(k+1,m)})$ が $w(n^0_{k,j})$ に対して大きすぎる場合（すなわち β または t_c が，t_p に対して大きすぎる場合）は，スケジュール長が短縮できなくなるので（すなわち 1 プロセッサでの実行が最適），本実験では $t_p : t_c : \beta$ をそれぞれ 1 : 10 : 500，1 : 20 : 1000，1 : 20 : 2000，1 : 40 : 2000 とした．

6.5 節，6.6 節，6.9 節の結果より，「D. LB」は他の 3 手法（提案手法，CASS+LB，DSC+CM）よりも明らかに $sl(G^S_{cls})$ が大きく，かつ走行時間も大きい．そのため，この実験の比較対象を「A. 提案手法」,「B. CASS+LB」,「C. DSC+CM」とした．これら条件の下，提案手法で決定されるプロセッサ数で式 (46) の比較を行った．

表 10 に，$|V| = 1770$，3160 のときのプロセッサ利用率の比較結果を示す．列「$t_p : t_c : \beta$」は式 (47) で定義した各値の比である．いずれの場合においても，提案手法による $E(|V^S_{cls}|, Algorithm)$ が最も高いという結果が得られた．このことから，文献 17) における Intel iPSC/860 および Ncube 6400 のクラスタ相当の環境では，提案手法によって高いプロセッサ利用率が得られるものと考えられる．

⋆1 文献 16) において，t_p は文献 17) Table 3 の「REAL*4 *＋*＋*」値を採用している．また，β は文献 17) Table 6 の値を採用し，t_c については，文献 17) Table 6 における「1 byte あたりの送信時間」の 8 倍（つまり 8 バイトあたりのデータ送信時間」としている．そのため，Ncube 6400 についてもこの方針に従って t_p，t_c，β を決定した．

表 10　ガウス消去法 DAG における $E(|V^S_{cls}|, Algorithm)$ の比較
Table 10　Comparison of $E(|V^S_{cls}|, Algorithm)$ in Gaussian elimination DAG.

No.	$\|V\|$	$t_p : t_c : \beta$	$\|V^S_{cls}\|$	$E(\|V^S_{cls}\|, Algorithm)$		
				A. Proposal	B. CASS-II+LB	C. DSC+CM
1	1770	1 : 10 : 500	9	0.6369	0.4215	0.5232
2	1770	1 : 20 : 1000	5	0.7015	0.5624	0.4790
3	1770	1 : 20 : 2000	4	0.6655	0.4269	0.3902
4	1770	1 : 40 : 2000	4	0.5812	0.5539	0.3152
5	3160	1 : 10 : 500	14	0.6097	0.4352	0.5348
6	3160	1 : 20 : 1000	11	0.5073	0.3390	0.3626
7	3160	1 : 20 : 2000	7	0.6187	0.4975	0.3652
8	3160	1 : 40 : 2000	7	0.5157	0.4393	0.2826

6.10.2　FFT DAG を用いたプロセッサ利用率の比較

FFT の DAG を用いた式 (46) の比較を行った．本実験では，文献 18)，19) に従った FFT DAG を用いた．この場合，行列サイズ N に対して $|V| = N \log N$ となる[18),19)]．各タスクはそれぞれ複素数の乗算と加算を行い，直接後続タスクへ結果を送る．文献 18) では実環境で FFT を実行した場合，タスクサイズおよびデータサイズについて $w(n^0_k) = c(e^0_{k,l})$，n^0_k，$n^0_l \in V_0$ と近似している．しかしながら，実行環境によってタスクの実行時間およびデータ転送時間は様々に想定される．本実験では文献 21) 同様，CCR を 0.1 から 10 まで変動させた FFT DAG をそれぞれ生成した．また，一様分布の乱数に従ったタスクサイズおよびデータサイズの最大値/最小値はそれぞれ 100 とし，DAG を 100 個生成した．そして，100 個それぞれに対して算出した $E(|V^S_{cls}|, Algorithm)$ の平均値で，比較を行った．

表 11 に，$|V| = 2048$，4608 の場合における比較結果を示す．$|V| = 2048$，4608 いずれの場合においても CCR が 0.1 のときは提案手法プロセッサ使用率は低いが，他手法よりは高い．また，$|V| = 2048$ においては CCR が 1 を境としてプロセッサ利用率が増加から減少に転じているが，CCR が 3 以降は増加となっている．一方，$|V| = 4608$ では CCR が 3 を境に増加から減少に転じているが，他手法よりは高い．以上の結果より，FFT DAG の CCR が 0.1 から 10 の範囲に収まる場合では，提案手法の方が他手法よりもプロセッサ利用率が高いといえる．

6.11　考　　察

以上の実験結果から，CCR が 0.1 から 10 に収まる DAG であれば，提案手法によって少ないプロセッサ数で効率の良いスケジュール長が得られることが分かった．まず，6.4 節の結果より，提案手法ではできるだけクラスタ内のタスク間先行関係を保っていることが分

表 11　FFT DAG における $E(|V_{cls}^S|, Algorithm)$ の比較
Table 11　Comparison of $E(|V_{cls}^S|, Algorithm)$ in FFT DAG.

No.	$\|V\|$	CCR	$\|V_{cls}^S\|$	$E(\|V_{cls}^S\|, Algorithm)$		
				A. Proposal	B. CASS-II+LB	C. DSC+CM
1	2048	0.1	275	0.2549	0.1053	0.1222
2	2048	1	129	0.3555	0.1558	0.1649
3	2048	3	79	0.2997	0.2439	0.2503
4	2048	5	62	0.3207	0.2911	0.2473
5	2048	8	44	0.3501	0.3030	0.2406
6	2048	10	32	0.3975	0.2866	0.2245
7	4608	0.1	789	0.2260	0.1026	0.1139
8	4608	1	276	0.2934	0.1450	0.2602
9	4608	3	189	0.3634	0.1746	0.1982
10	4608	5	138	0.3347	0.1462	0.1809
11	4608	8	97	0.3368	0.1654	0.2330
12	4608	10	93	0.3273	0.1513	0.1889

かった．その結果，6.5 節の結果から分かるとおり，$sl_w(G_{cls}^S)$ が減少し，さらに提案手法で決定されたプロセッサ数では，従来手法の $sl_w(G_{cls}^S)$ よりも低いことが分かった．すなわち式 (31) により，提案手法の方が従来手法よりもスケジュール長の下限値が減少できることになる．そのため，各手法間で同じスケジュール方針でタスク実行順を決めると，提案手法におけるスケジュール長のとりうる下限値が他手法よりも小さいことになる．このことが，スケジュール長の短縮につながっているものと考えられる．

次に，従来手法において $sl_w(G_{cls}^S)$ がなぜ提案手法よりも大きいかを述べる．CASS-II+LB，DSC+CM はいずれもタスククラスタリングの後にクラスタ集約を行う．このうち，CASS-II および DSC では，特定のスケジュール方針に従ってタスク実行順を決める．もしタスク集約によって先行関係のないタスクが含まれてスケジュール長が大きくなる場合，CASS-II および DSC ではタスク集約は行われない[4),11)]．その結果，1 タスクのみが属するクラスタが得られる可能性があり，必然的にクラスタ数は多くなる．そのため，タスク数の多い DAG に対しては，LB や CM といったクラスタ集約を行う必要がある．しかしながら，LB と CM では集約対象となるクラスタの選択基準はクラスタサイズの大小関係であり，集約後の各タスクの先行関係は考慮していない．その結果，CASS-II+LB，DSC+CM，LB ではともに図 10 のように各クラスタにおける top_s タスク数が増加してしまう．さらに図 9 により，互いに先行関係のないタスクが同一クラスタに含まれるといえるので，結果的に $sl_w(G_{cls}^S)$ の増加につながっているものと考えられる．

式 (27) で導出した δ_{opt} は，$sl_w(G_{cls}^S)$ がとりうる値のうち，その上限値が最小化される場合のクラスタサイズである．これは DAG 内のタスクサイズとデータサイズとの関係，および 1 経路のタスクサイズの和から決まる．そのため，タスクサイズに比べてデータサイズが大きいか，または小さいかに応じて変動する．さらに 1 経路のタスクサイズ和の最大値はスケジュール長 $sl(G_{cls}^S)$ の下限値でもある．この値が大きな DAG であれば δ_{opt} は大きくなり，できるだけデータ転送の局所化を行う必要があることを意味する．その結果，このクラスタサイズでタスク集約することが，$sl(G_{cls}^S)$ の下限値に近づけようとすることになる．提案手法では，クラスタサイズの下限値を δ_{opt} としたうえで 5 章で述べたタスク集約方針を行う．その結果，6.6 節での結果のように，どの CCR を持つ DAG においても従来手法よりも少ないプロセッサで同等の $sl(G_{cls}^S)$ が得られる．また 6.10 節での結果から，特定アプリケーション DAG を想定した場合においても提案手法によってプロセッサが有効利用できるといえる．これらのことから，提案手法によって，少ない計算資源でスケジュール長を短縮させることができるといえる．

提案手法の目的の 1 つはプロセッサ数の決定であるので，理想的にはプロセッサ数が不明（もしくは無数に存在する）な均一な分散環境において，タスクスケジューリングの前にプロセッサ数を決定させるときに有効な手法である．また，無数に存在する均一プロセッサ環境においてできるだけ少ないプロセッサ数でスケジュール長を最小化しようとする場合に適用可能な手法である．しかしながら，現実には無数にプロセッサが利用可能な環境は困難である．提案手法では，各プロセッサに割り当てられるクラスタサイズは，式 (27) で決めた δ_{opt} 以上である．タスククラスタリング前では，たかだか

$$\left\lfloor \frac{\sum_{k=1}^{|V|} w(n_k)}{\delta_{opt}} \right\rfloor \tag{48}$$

個のプロセッサが必要であることが分かる．そのため，2.1 節で述べた環境において，式 (48) の個数だけのプロセッサが利用可能であれば，提案手法が適用可能かつ有効であるといえる．

もし広域ネットワーク（たとえばインターネット）を想定した場合，プロセッサ処理速度，通信速度，通信遅延などが不均一である．このような環境で必要プロセッサ数を決めようとすると，プロセッサによって各タスクの実行時間も異なるため，クラスタサイズの下限値算出は困難となる．もし不均一環境でプロセッサ数が決めることができれば，グリッド環境などのような世界中の計算機を有効利用できることになる．このように提案手法を不均一環境に拡張することは非常に興味深い研究対象であるが，これは今後の課題である．

7. まとめと今後の課題

本論文では，少ない計算資源数でできるだけスケジュール長を短縮させるための，プロセッサ数の決定手法について述べた．提案手法では，スケジューリングを行う前にあらかじめ各プロセッサに割り当てる仕事量の下限値を決める．さらに，スケジュール長がとりうる値のうちで，各タスクをできるだけ後に実行させた場合の値 $sl_w(G^s_{cls})$ を定義した（s はタスク集約回数）．そして，S 回後のタスク集約後に各割当て単位の仕事量が一定サイズ以上となることを仮定した場合の，$sl_w(G^S_{cls})$ の上限値が最小化されるときのクラスタサイズ δ_{opt} を算出した．そして，$sl_w(G^S_{cls})$ を減少させることがスケジュール長の下限値を減少させることを示し，$sl_w(G^S_{cls})$ を減少させる（または増分を最小化するため）ことを目的としたタスククラスタリングを提案した．このタスククラスタリングにより，少ないプロセッサ数で良いスケジュール長を短時間に得られることをシミュレーションで示した．

本論文で想定した環境は，完全結合型の均一プロセッサ環境である．今後の課題は，実際の分散環境に適用させるためには各プロセッサの不均一性，プロセッサ間の通信パターンを考慮して各プロセッサへ割り当てる仕事量の決定手法を検討することである．

参 考 文 献

1) Kwok, Y.K. and Ahmad, I.: Static scheduling algorithms for allocating directed task graphs to multiprocessors, *ACM Computing Surveys*, Vol.31, No.4, pp.406–471 (1999).
2) Sinnen, O.: *Task Scheduling for Parallel Systems*, Wiley (2007).
3) Gerasoulis, A. and Yang, T.: A Comparison of Clustering Heuristics for Scheduling Directed Acyclic Graphs on Multiprocessors, *Journal of Parallel and Distributed Computing*, Vol.16, pp.276–291 (1992).
4) Yang, T. and Gerasoulis, A.: DSC: Scheduling Parallel Tasks on an Unbounded Number of Processors, *IEEE Trans. Parallel and Distributed Systems*, Vol.5, No.9, pp.951–967 (1994).
5) Kanemitsu, H., Lee, G., Nakazato, H., Hoshiai, T. and Urano, Y.: Static Task Cluster Size Determination in Homogeneous Distributed Systems, *Proc. 4th International Workshop on Automatic Performance Tuning* (*iWAPT2009*), pp.37–51 (2009).
6) Gerasoulis, A. and Yang, T.: On the Granularity and Clustering of Directed Acyclic Task Graphs, *IEEE Trans. Parallel And Distributed Systems*, Vol.4, No.6 (1993).
7) Yang, T. and Gerasoulis, A.: PYRROS: Static scheduling and code generation for message passing multiprocessors, *Proc. 6th ACM International Conference on Supercomputing*, pp.428–437 (1992).
8) Liou, J.C. and Palis, M.A.: A Comparison of General Approaches to Multiprocessor Scheduling, *Proc. 11th International Symposium on Parallel Processing*, pp.152–156 (1997).
9) Sarkar, V.: *Partitioning and Scheduling Parallel Programs for Execution on Multiprocessors*, Cambridge, MA: MIT Press (1989).
10) Yang, T. and Gerasoulis, A.: List scheduling with and without communication delays, *Parallel Computing*, pp.1321–1344 (1993).
11) Liou, J.C. and Palis, M.A.: An Efficient Task Clustering Heuristic for Scheduling DAGs on Multiprocessors, *Proc. 8th Symposium on Parallel and Distributed Processing* (Oct. 1996).
12) Wu, M.Y. and Gajski, D.D.: Hypertool: A programming aid for message-passing systems, *IEEE Trans. Parallel and Distributed Systems*, Vol.1, No.3, pp.330–343 (1990).
13) Zomaya, A.Y. and Chan, G.: Efficient clustering for parallel tasks execution in distributed systems, *Proc. 18th International Parallel and Distributed Processing Symposium* (*IPDPS'04*), pp.167–175 (2004).
14) Lepere, R. and Trystram, D.: A New Clustering Algorithm for Large Communication Delays, *Proc. 16th Parallel and Distributed Processing Symposium* (*IPDPS2002*), pp.68–73 (2002).
15) Pecero-Sanchez, J. and Trystram, D.: A new Genetic Convex Clustering algorithm for parallel time minimization with large communication delays, *Proc. International Conference Parallel Computing* (*ParCo'2005*), pp.709–716 (2005).
16) Amoura, A.K., Bampis, E. and Konig, J-C.: Scheduling Algorithms for Parallel Gaussian Elimination With Communication Costs, *IEEE Trans. Parallel and Distributed Systems*, Vol.9, No.7, pp.679–686 (1998).
17) Dunigan, T.H.: Performance of the Intel iPSC/860 and n-Cube 6400 Hypercubes, *Parallel Computing*, Vol.17, No.10 and 11, pp.1285–1302 (1991).
18) Chabridon, S. and Gelenbe, E.: Dependable parallel computing with agents based on a task graph model, *Proc. 3rd Euromicro Workshop on Parallel and Distributed Processing*, pp.350–357 (1995).
19) Li, Y., Zhao, L., et al.: A Performance Model for Fast Fourier Transform, *Proc. 2009 IEEE International Symposium on Parallel and Distributed Processing*, pp.1–11 (2009).
20) Daoud, M.I. and Kharma, N.: A high performance algorithm for static task scheduling in heterogeneous distributed computing systems, *Journal of Parallel and Distributed Computing*, Vol.68, No.4, pp.399–409 (2008).

21) Topcuoglu, H., Hariri, S.H. and Wu, H.Y.: Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing, *IEEE Trans. Parallel and Distributed Systems*, Vol.13, No.3, pp.260–274 (2002).
22) Fujimoto, N., Baba, T., Hashimoto, T. and Hagihara, K.: On Message Packaging in Task Scheduling for Distributed Memory Parallel Machines, *International Journal of Foundations of Computer Science*, Vol.12, No.3, pp.285–306 (2001).
23) Sinnen, O. and Sousa, L.A.: Communication Contention in Task Scheduling, *IEEE Trans. Parallel and Distributed Systems*, Vol.16, No.6, pp.503–515 (2005).
24) Bataineh, S.M.: Toward an analytical solution to task allocation, processor assignment and performance evaluation of network processors, *Journal of Parallel and Distributed Computing*, Vol.65, No.1, pp.29–47 (2005).

付　　録

A.1　性質 1 の証明

証明. $TL_{s+1}(r) - TL_s(r) \neq 0$ であるためには，$cls_s(t)$ に属するタスク $n^s_{t'}$ と $top_s(r)$ に属するタスク $n^s_{r'}$ に対し，$n^s_{t'} \prec n^s_{r'}$ でなければならない．そこで，次の 2 つの場合に分けて考える．

ケース 1：$n^s_{t'} \in pred(n^s_{r'})$ と仮定した場合

式 (35) より，$top_s(r)$ の任意の直接先行タスクは δ_{opt} 以上のサイズであるクラスタに属するが，$w(cls_s(t)) < \delta_{opt}$ であり矛盾．

ケース 2：$n^s_{t'} \notin pred(n^s_r)$，$n^s_{t'} \prec n^s_{r'}$ と仮定した場合

あるクラスタ $cls_s(l)$ について，$cls_s(l)$ 内に，$top_s(r)$ の直接先行タスクが存在すると仮定する．すると，RDY_s の定義より，$w(cls_s(l)) \geq \delta_{opt}$ である．$TL_{s+1}(r) - TL_s(r) \neq 0$ であるためには $n^s_{t'}$ が $top_s(l)$ 内のあるタスクの直接先行タスクか，または $n^s_{t'}$ が $top_s(l)$ のあるタスクよりも先行する必要がある．前者の場合，$cls_s(l)$ は UEX_s には属さず，$top_s(l)$ の任意の直接先行タスクが属するクラスタのサイズが δ_{opt} 以上である．したがってケース 1 と同様，$w(cls_s(t)) < \delta_{opt}$ であるので矛盾．さらに後者の場合も，ケース 2 を $n^s_{t'}$ が $top_s(l)$ の直接先行タスクとなるまで繰り返すことにより，矛盾が導かれる．ゆえに，題意は示された．■

A.2　性質 2 の証明

証明. $cls_s(r)$ のあるタスク $n^s_{r'}$ について，$n^s_{r'}$ が $top_s(s)$ のあるタスク $n^s_{s'}$ と $n^s_{r'} \prec n^s_{s'}$ であれば $TL_{s+1}(s) - TL_s(s) \neq 0$ がありうる．したがって，この仮定の矛盾を示せば十分である．そこで，次の 2 つの場合に分けて考える．

ケース 1：$n^s_{r'} \in pred(n^s_{s'})$ と仮定した場合

集約前は，$cls_s(r)$ も $cls_s(t)$ も UEX_s に属しているため，$cls_s(s) \in RDY_s$ という条件に矛盾する．

ケース 2：$n^s_{r'} \notin pred(n^s_{s'})$，$n^s_{r'} \prec n^s_{s'}$ と仮定した場合

あるクラスタ $cls_s(l)$ について，$cls_s(l)$ 内のタスクが，$top_s(s)$ 内のあるタスクの直接先行タスクであるとする．すると，$w(cls_s(l)) \geq \delta_{opt}$ である．$TL_{s+1}(s) - TL_s(s) \neq 0$ であるためには $n^s_{r'}$ が $top_s(l)$ の直接先行タスクか，または $n^s_{r'}$ が $top_s(l)$ のあるタスクよりも先行するはずである．前者の場合，$cls_s(l)$ は UEX_s には属さず，$top_s(l)$ の任意の直接先行タスクが属するクラスタのサイズが δ_{opt} 以上である．したがってケース 1 と同様，タスク集約前は $cls_s(r)$，$target_s$ ともにクラスタサイズが δ_{opt} 未満であるので矛盾．さらに後者の場合も，ケース 2 を $n^s_{r'}$ が $top_s(l)$ の直接先行タスクとなるまで繰り返すことにより，矛盾が導かれる．■

A.3　性質 3 の証明

証明. まず，(A) を示す．$cls_s(i)$ には 1 タスクのみが属しているので，$TL_s(i) = tlevel(n^s_{i'})$，$BL_s(i) = w(n^s_{i'}) + blevel(n^s_{i'})$ である．タスク集約により，$cls_s(r)$ と $cls_s(t)$ の中で先行関係のないタスクどうしが $cls_{s+1}(r)$ に含まれると $tlevel(n^{s+1}_{r'})$ が増加しうる（$tlevel(n^s_{r'}) < tlevel(n^{s+1}_{r'})$）ので，これによって $tlevel(n^{s+1}_{i'})$ 値も増加しうる（$tlevel(n^s_{i'}) < tlevel(n^{s+1}_{i'})$）．一方，タスク集約後の $cls_{s+1}(r)$ が linear のときは $cls_{s+1}(r)$ 内の全タスク間で先行関係が存在するので，$TL_{s+1}(i)$ は増加しない（$TL_s(i) \geq TL_{s+1}(i)$）．また，$n^s_{r'} \prec n^s_{i'}$ より，$BL_{s+1}(i)$ は明らかに不変（$BL_s(i) = BL_{s+1}(i)$）である．次に (B) を示す．$n^s_{i'} \prec n^s_{r'}$ なので，タスク集約によっても $tlevel(n^{s+1}_{i'})$ は変動しない．$cls_{s+1}(r)$ において，$cls_s(r)$ と $cls_s(t)$ の間で局所化されたデータサイズのみが，$cls_{s+1}(r)$ 内の各タスクの $blevel$ 値に影響する．したがって，$cls_{s+1}(r)$ 内の各タスクの $blevel$ 値は増加しない．また $cls_s(r)$，$cls_s(t)$ はともに $seq_{max,s}$ に属するタスクを持たないので，タスク集約後におけるこれら 2 クラスタ内のタスクの $blevel$ 値の変動は，$blevel(n^{s+1}_{i'})$ の増減に影響しない．したがって，題意は示された．■

A.4　性質 4 の証明

証明. $cls_s(i)$ の要素数が 2 以上であるから，$cls_s(i)$ は RDY_s に属するかまたはクラスタサイズが δ_{opt} 以上である．したがって，$top_s(i)$ の任意の直接先行タスクも δ_{opt} 以上のクラスタに属する．ここで，性質 2 から，(A)（B）いずれの場合も $\Delta TL_{s+1}(i) = 0$ がいえる．

次に（A）における $\Delta BL_{s+1}(i) = 0$ を示す．$cls_s(i)$ の要素数が 2 以上であるが，$n^s_{r'} \prec n^s_{i'}$

なので性質 3 同様，$\Delta BL_{s+1}(i) = 0$ である．最後に（B）における $\Delta BL_{s+1}(i) = 0$ を示す．性質 3（B）と同様，$cls_s(r)$，$cls_s(t)$ はともに $seq_{max,s}$ に属するタスクを持たないので，タスク集約後，これら 2 クラスタ内のタスクの *blevel* 値の変動は $cls_{s+1}(i)$ 内の各タスクの *blevel* 値に影響しない．■

A.5　性質 7 の証明

証明． タスク集約後に $cls_{s+1}(p)$ が linear でなくなる場合を考える．このような場合，$n^s_{t'} \not\prec n^s_k$ かつ $n^s_k \not\prec n^s_{t'}$，$n^s_k \in cls_s(p)$ となる n^s_k が存在していることになる．しかしながらタスク集約前は $cls_s(p)$ が linear であり，$cls_s(p)$ 内の任意のタスクが $n^s_{p'}$ に先行するので，$n^s_k \prec n^s_{p'}$ である．その結果 $n^s_k \prec n^s_{p'} \prec n^s_{t'}$，すなわち $n^s_k \prec n^s_{t'}$ となり，矛盾．■

A.6　性質 8 の証明

証明． $|cls_s(t)| \geq 2$ かつ $w(cls_s(t)) < \delta_{opt}$ より，$cls_s(t) \in RDY_s$ である．したがってこの場合は，RDY_s に属するクラスタどうしの集約となる．まず，$n^s_{p'} \in suc(n^s_{t'})$ と仮定する．しかしながら $cls_s(t) \in RDY_s$ であるから，$cls_s(p) \in RDY_s$ であることに反する．また，$n^s_{p'} \in pred(n^s_{t'})$ と仮定する．しかしながら $cls_s(p) \in RDY_s$ であるから，$cls_s(t) \in RDY_s$ であることに反する．ゆえに題意は示された．■

A.7　性質 9 の証明

証明． $RDY_s = \{\emptyset\}$ と仮定する．このとき V^s_{cls} 内の任意のクラスタ $cls_s(i)$ について，$top_s(i)$ の直接先行タスクのうち少なくとも 1 つは δ_{opt} 未満のクラスタに属していることになる．そこで，$top_s(i)$ の直接先行タスクが属するクラスタを $cls_s(h)$ とし，$w(cls_s(h)) < \delta_{opt}$，$cls_s(h) \notin RDY_s$ とする．すると，$cls_s(h)$ の要素数が 1 または 2 以上の場合が考えられる．

$|cls_s(h)| = 1$ の場合，これは全クラスタ内の要素数が 1 の場合に該当するので，$V^s_{cls} = V^0_{cls}$ である．しかしながら，図 5 行 1 で START タスクが属するクラスタが RDY_s に属するので矛盾．

$|cls_s(h)| \geq 2$ の場合，$cls_s(h)$ は少なくとも 1 度のタスク集約が行われたことに該当する．しかしながら式 (35) より，要素数が 2 以上のクラスタはクラスタサイズが δ_{opt} 以上となるまでは RDY_s に属するので，$cls_s(h) \notin RDY_s$ とはならない．したがって題意は示された．■

(平成 22 年 5 月 7 日受付)
(平成 22 年 9 月 6 日採録)

金光　永煥（学生会員）

昭和 54 年生．平成 14 年早稲田大学理工学部数理科学科卒業．同年（株）アルファシステムズ入社．平成 16 年早稲田大学大学院国際情報通信研究科入学．平成 18 年同研究科修士課程修了．同年同大学メディアネットワークセンター助手．現在，同大学大学院国際情報通信研究科博士後期課程在学中．分散処理について興味を持つ．IEEE，ACM，電子情報通信学会各会員．

李　　吉憲

昭和 49 年生．平成 19 年韓国漢陽大学大学院電子情報コンピュータ工学部情報工学専攻修士課程修了．同年早稲田大学大学院国際情報通信研究科入学．コンピュータネットワーク，P2P ネットワーク，ネットワークコーディング等を研究中．電子情報通信学会会員．

中里　秀則（正会員）

昭和 33 年生．昭和 57 年早稲田大学より工学士．同年沖電気工業（株）入社．平成 2 年，平成 5 年米国イリノイ大学アーバナ・シャンペーン校より M.S.，Ph.D. をそれぞれ取得．平成 5 年沖電気工業（株）復職．平成 12 年より早稲田大学教授．インターネット QoS，実時間システム等の研究に従事．ACM，IEEE，電子情報通信学会各会員．

星合　隆成

昭和 61 年 NTT 通信網第一研究所入所．工学博士（平成 6 年度）．平成 7〜9 年ベルコミュニケーション研究所（米国）客員研究員．現在，NTT ネットワークサービスシステム研究所主幹研究員．平成 10 年にブローカレスモデルを提唱して以来，その実現技術である意味情報ネットワークアーキテクチャSIONet（シオネット）を考案．平成 12 年以降は，ブローカレス型ポリシモデルやコミュニティコラボレーションの実現技術である COMNet（コムネット）の研究を推進するとともに，コミュニティを P2P 技術の融合，コミュニティ活性化のための活動や，P2P・ユビキタスコンピューティングの普及・啓蒙活動に深くかかわる．早稲田大学客員研究員．主な著書「ブローカレスモデルと SIONet」（電気通信協会/オーム社）．

浦野　義頼（フェロー）

昭和 40 年早稲田大学理工学部電気通信学科卒業．昭和 45 年同大学大学院理工学研究科博士課程修了（工学博士）．同年国際電信電話（株）入社．平成 5 年同社研究所長．平成 8 年早稲田大学理工総合研究センター教授．現在，同大学大学院国際情報通信研究科教授．主として e-ラーニング，e-ヘルスケア等インターネット応用等の研究や情報通信分野の人材育成にかかわる国際協力プロジェクト等に従事．本会，電子情報通信学会，人工知能学会等の理事を歴任．