解　説

分散アルゴリズム†

萩原兼一††　増澤利光†††

1. はじめに

地理的に離れた多数の計算機を結合するネットワークが発展するにつれて，使用者がよくアクセスするデータを，その人がよく利用する計算機に記憶するというデータの分散化が進んでいる．これと対照的なものは，すべてのデータを一つの計算機（ホスト計算機と呼ぶ）が集中管理するものである．前者を分散方式，後者を集中方式と呼ぶことにする．分散方式の長所としては，次のことが考えられる．

① 記憶領域の負荷分散：各計算機は，それ自身が管理するデータを記憶するだけの記憶領域を備えておればよい．集中方式では，すべてのデータを記憶するための比較的多くの記憶領域をもつホスト計算機がネットワーク内で必要となる．

② 局所処理による効率向上：分散方式では，各計算機のもつデータだけで処理できる作業は，他の計算機とは独立に処理できるので，処理効率が向上する．

③ データ更新の迅速化：分散方式では，データの更新を引き起こす事象が発生した場合に，各計算機は自分自身のデータを更新すればよいのでデータ更新が迅速である．しかし，集中方式ではホスト計算機でデータ更新が行われるので，ネットワーク内でのメッセージ遅延によりデータを更新する事象の生起とホスト計算機でデータが実際に更新されるまでに時間差が生じる．したがって，データが実状を正確に反映していない期間が比較的長くなる．

④ 故障耐性：集中方式では，中心となる計算機の故障によりすべてのデータが使えなくなる．分散方式では，計算機の故障はその計算機が管理するデータが使えなくなるだけである．

† Distributed Algorithms by Kenichi HAGIHARA (Department of Information and Computer Sciences, Faculty of Engineering Science, Osaka University) and Toshimitsu MASUZAWA (Education Center for Information Processing, Osaka University).
†† 大阪大学基礎工学部情報工学科
††† 大阪大学情報処理教育センター

しかし，分散方式の欠点は，各計算機が自分自身のもつデータだけでは処理できない場合に，他の計算機と情報交換しながら処理する必要があり，そのためのアルゴリズムが難しくなる．集中方式ではホスト計算機がすべてのデータをもっているので，このようなことは生じない．このように，問題解決に必要なデータが複数計算機に分散してしまっている状況で，関係する計算機が必要に迫られてメッセージ通信で情報交換しながら問題（**分散問題**と呼ぶ）を解決するアルゴリズムを**分散アルゴリズム**と呼ぶ．分散アルゴリズムをもう少し広くとらえて，通信手段として共有メモリを用いるものを指すこともあるが，ここでは共有メモリはなく，通信手段はメッセージ通信のみとする．

ワークステーションなどの普及・データの分散化への社会の要請・通信技術の発展などにより，今後は分散環境の色彩が強くなり，分散アルゴリズムの重要性はますます増大すると思われる．そして，各種問題に対して多くの分散アルゴリズムが提案されるであろうが，本稿ではそれらの優劣をどのように議論すればよいかに関しての基礎的な事項の解説を試みる．そして，広範囲の問題を薄く説明するよりは，リーダ選択という基本的な問題をいろいろな観点から説明する．リーダ選択問題は，比較的静的に定義される問題であるが，停止判定問題[14]やスナップショット問題[13]のように比較的動的な対象に対して定義される問題もある．後者も興味深いが，紙面の都合上本解説からは省いた．解説という性格上，直観的説明が多いが，厳密な説明は文献 23) を参照していただきたい．

2. 分散ネットワーク・モデル

計算機の集合を $\{P_1, P_2, \cdots, P_n\}$ と表す．すべての計算機は対等である．すなわち，ホスト計算機のような特別の役割を果たす計算機はない．各計算機は適当規模の局所メモリをもつ．計算機間には共有メモリはなく，情報交換はメッセージを用いて行う．

P_i から P_j へ直接メッセージ送信可能なときに P_i

から P_j に**リンク**（$\langle P_i, P_j \rangle$ と表す）が存在すると考える．このリンクは物理的なものであってもよいし，論理的なものであってもよい．したがって，**計算機ネットワーク**（以降では単に**ネットワーク**という）は $\{P_1, P_2, \cdots, P_n\}$ を頂点集合とし，リンクを辺とする有向グラフと考えることができる．ただし，$\langle P_i, P_j \rangle$ が存在するときは $\langle P_j, P_i \rangle$ も存在するとする**双方向通信可能**である場合が多く，このときは無向グラフと考える（**図-1 参照**）．各計算機には一般に複数本のリンクが存在するが，計算機はそれらを区別できる．

リンク $\langle P_i, P_j \rangle$ を介して P_i から送信されたメッセージは，$\langle P_i, P_j \rangle$ に故障がないかぎり有限時間内に P_j に着信する．この有限時間（メッセージ遅延時間と呼ぶ）の上限値を仮定せず，またその時間自身が可変でもあるとする場合のネットワークを**非同期式ネットワーク**と呼ぶ．非同期式ネットワークは，ネットワーク全体で参照可能な“グローバル・クロック”が存在しない状況をモデル化している．これに対して，グローバル・クロックが存在し，メッセージ遅延時間比や計算機の実行速度比に上限があると仮定する場合のネットワークを**同期式ネットワーク**と呼ぶ．つまり，同期式ネットワークでは，グローバル・クロックの数クロックを抽象化した“ラウンド”というものを考え，すべての計算機は同期して各ラウンド内に次の①と②を行うことができる．

① 各計算機は，直前のラウンドに自分に対して送信されたすべてのメッセージを受信する．

② そのときの内部状態および受信したメッセージから，次の状態および送信するメッセージとその送信先を決定し，状態遷移とメッセージの送信を行う．

また，$\langle P_i, P_j \rangle$ を介して P_i から複数のメッセージが送信されるときに，その送信順のまま P_j に着信する場合を **FIFO 型ネットワーク**，その順が保証されない場合を**非 FIFO 型ネットワーク**と呼ぶ．

計算機 P_j の識別子を $id(P_j)$ と表す．たとえば，図-1 の P_5 の識別子 $id(P_5)$ は 2 である．$P_i \neq P_j$ のとき $id(P_i) \neq id(P_j)$ であることを仮定する場合（**連名ネットワーク**と呼ぶ）と，これを仮定しない場合（**匿名ネットワーク** (anonymous network) と呼ぶ）とがある．匿名ネットワークでの議論は，計算機の識別子がすべて等しい場合，あるいは識別子が存在しない場合という，極端な状況でなされることが多い[50]．

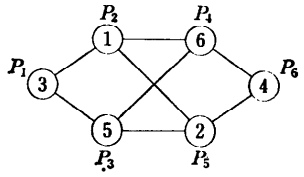


図-1 計算機ネットワーク

分散ネットワーク環境では，計算機あるいはリンクに故障が生じる可能性を考慮することも重要である．故障計算機や故障リンクの動作に一切仮定を置かない故障を**ビザンチン故障** (Byzantine fault) という．したがって，故障計算機がまったくでたらめのメッセージを送信したり，故障リンクでメッセージの内容が変更されることもある．一方，故障計算機は一切動作せず，故障リンクを用いて送信したメッセージは消滅する故障は**停止故障** (stopping fault) という．

以上の前提のうち，分散アルゴリズムは次の前提のネットワークで議論されることが多い．この前提の成り立つネットワークを**標準ネットワーク・モデル**と呼び，本稿では特に断わらないかぎり標準ネットワーク・モデルを前提とする．

- リンクは双方向通信可能
- 非同期式ネットワーク
- FIFO 型ネットワーク
- 連名ネットワーク
- 故障なし

3. 分散問題

逐次アルゴリズムで解く問題は，入力と期待する出力との対応関係を与えることで定義される．分散問題は，入力に対応するものとしてその分散問題を解くための情報（**入力情報**と呼ぶ）がどのように分散しているかを，そして期待する出力に対応するものとして最終的に各計算機がどのような情報（**出力情報**と呼ぶ）を知るかを与えることで定義される．

入力情報のうち，ほとんどの分散問題に関しても共通に想定する情報として，各計算機 P の識別子 $id(P)$ および P のもつリンク数がある．これらの情報を**基本情報**と呼ぶことにする．ただし，基本情報を知っていることは，リンクで結合されている計算機の識別子を知っていることまで意味しない．たとえば，図-1 の P_5 のもつ基本情報は，$id(P_5)=2$ であること，および 3 本のリンクで他の計算機と結合されていることであるが，P_5 が結合されている計算機の識別子が 5, 1, 4 であることを知っているわけではない．さらに，他の計算機間のリンク情報，たとえば P_4 と P_6 間にリン

クが存在することなどはもちろん P_5 は知らない.

分散問題の例を次に示す.

[リーダ選択問題]

分散ネットワークでは固定したホスト計算機がないと考えているが, ネットワーク全体に係わるなんらかのことに関して合意を得るためには, その合意を得る間の暫定的にリーダの役割を果たす計算機を選択する必要が生じることが多い.

入力情報: 各計算機は基本情報を知っている.

出力情報: 各計算機はある計算機 P の識別子 $id(P)$ を知る. これは, 各計算機が $id(P)$ を識別子としてもつ計算機 P がリーダとして選ばれたと解釈する. たとえば, 図-1 で P_4 がリーダとして選ばれるならば, 各計算機は P_4 の識別子である 6 を知ることになる.

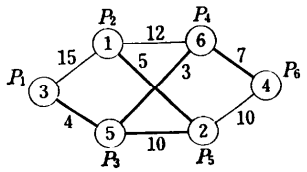
[最小費用生成木構成問題][21),24)]

各リンク ℓ に関して ℓ を介してメッセージを送るときの費用 (cost) が定義されているとする. ある計算機から他のすべての計算機に何かの情報を知らせる (放送と呼ぶ) ときにネットワークの適当な生成木 (spanning tree) を利用すると, メッセージの管理作業が簡単になる. 生成木のなかでも, リンクの費用の総和が最小となるような生成木 (最小費用生成木 (minimum cost spanning tree)) を構成したい.

入力情報: 各計算機 P は基本情報と, P の各リンクの費用を知っている. たとえば, 図-2 の P_5 は, 基本情報の他に, $\langle P_5, P_3\rangle$, $\langle P_5, P_2\rangle$, $\langle P_5, P_6\rangle$ のリンクの費用がそれぞれ 10, 5, 10 であることを知っている.

出力情報: ネットワークのある最小費用生成木を $MST=(\{P_1, P_2, \cdots, P_n\}, E)$ として, 各計算機 P は P のリンクのうちどれが E に属し, どれが属さないかを知る. ここで, MST に関する他の計算機間の情報を知ることまでは要求していない. たとえば, 図-2 の太線の部分が MST として採用されるならば, P_5 は $\langle P_5, P_3\rangle$, $\langle P_5, P_2\rangle$ が E に属し, $\langle P_5, P_6\rangle$ は属さないことを知る.

その他にも, センタ (メディアン) 探索問題[30)], 最大流問題[7)], 幅優先探索木構成問題[8),39)], 孤立頂点や橋を求める問題[40)], 最小費用生成木更新問題[41)]などの分散問題が研究されている.



リンクの横の数字はそのリンクの費用を表す.
太線は最小費用生成木を表す.

図-2 最小費用生成木

4. 分散アルゴリズム

4.1 分散アルゴリズムとは

分散アルゴリズムは, 一般に受信したメッセージに対してどのような "行動" をするかという約束事に関する記述である. この約束事はネットワークのすべての計算機に共通なものである. ここで行動とは, 計算機単独で局所計算を行い, 必要ならば接続する (いくつかの) リンクを介してメッセージを送信し, 新たなメッセージを待つ状態になることである. ある場合は, 以降メッセージの送受信や局所計算を行わない終了状態になることもできる. メッセージがどのリンクを介して受信したのかは知ることはできる. メッセージを送信する場合は, リンクごと異なるメッセージを送信してよい.

ただし, 分散アルゴリズムを開始するときだけ, メッセージを受信することなしに自発的にメッセージを送信する. 分散環境では各計算機が独立に各自の処理を行っているので, すべての計算機が自発的に分散アルゴリズムを開始するメッセージを送信すると考えるより, そのような計算機と, 自発的にメッセージを送信するより前に他の計算機が送信したメッセージを受信して分散問題の解決に参加する計算機とに分類して考えるほうが実状を反映している. 前者の計算機を**始動計算機**と呼び, 分散アルゴリズムではどの計算機が始動計算機になるか, また何台が始動計算機になるかを想定しないのが一般的である.

逐次アルゴリズムでは, 同じ入力データに関して計算の進行状況は唯一に定まる. しかし, 分散アルゴリズムでは, どの計算機が始動計算機になるかを想定できないこと, また非同期式ネットワークではメッセージ遅延時間にばらつきがあることから, ネットワーク全体における計算の進行状況はさまざまである.

分散問題 Q を分散アルゴリズム A が解くとは, A による任意の計算の進行状況において次の①および②を満たすことである.

① すべての計算機が終了状態になる.

② 終了状態になった時点で各計算機がもつ出力情報が Q の定義で期待されるものである. □

また, **Q を A が弱く解く**とは, 上の①および②のかわりに次の③を満たすことである.

③ ある時点においてネットワークで送信されるメッセージがなくなる定常状態になり，その時点で各計算機がもつ出力情報はQの定義で期待されるものである．ただし，定常状態になったことを各計算機が認識できなくてよい．□

分散アルゴリズムが利用できる情報は原則として入力情報だけである．たとえば，リーダ選択問題では基本情報だけである．したがって，リーダ選択問題を解く分散アルゴリズムは，どのような形状のネットワーク上でも走らすことができる．しかし，場合によっては，リング状ネットワーク（以降リングと略す）だけ，木状ネットワークだけ，あるいは完全グラフ状ネットワーク（以降完全ネットワークと略す）だけというように，分散アルゴリズムを走らせるネットワークに制限をつけて考えることがある（**5.3** 参照）．このようなある形状のネットワーク専用の分散アルゴリズムは，分散アルゴリズムの動作定義自体にこの形状情報が使われているので，想定していない形状のネットワークでは一般に正しく動作しない．

さらに，分散問題Qを解く場合に，Qの入力情報以外のなんらかの情報（**付加情報**と呼ぶ）を知っていると仮定することもある．たとえば，付加情報として次のものがある．

① リンクで結合されている計算機の識別子（**6.2** 参照）

② ネットワーク全体での計算機総数n（**6.2** 参照）

③ n の上限値

④ 過半数値（$n/2 \leqq m$ である値 m）（**6.2** 参照）

⑤ 方向感覚（**5.3** 参照）

⑥ 故障する可能性のある計算機総数の上限値 f_P（**6.2** 参照）

⑦ 故障する可能性のあるリンク総数の上限値 f_L（**6.2** 参照）

ネットワーク形状専用分散アルゴリズムや付加情報をもつ分散アルゴリズムでは，一般に次の **5.** で説明する評価基準に関して優位である．

4.2 分散アルゴリズムの例

ここでは，分散アルゴリズムの例として，完全ネットワークでリーダ選択問題を解く分散アルゴリズムを紹介する．

これまでに多くのリーダ選択アルゴリズムが提案されているが，それらのほとんどは，以下のようにして，始動計算機の中からリーダを選択する．

① 各始動計算機は，始動時にリーダ候補（以下では，単に**候補**とよぶ）になる．

② 各候補 P_i は，他の計算機を支配しようとする．このために，候補はまだ支配していない計算機の中の任意の一つにメッセージを送信する．このメッセージを受信した計算機が P_i に支配されるなら，P_i にメッセージを返信する．P_i はこのメッセージを受信すると，まだ支配していない計算機が存在するなら，その中の任意の一つにメッセージを送信する．

③ 他のすべての計算機を支配した候補がリーダとなる．

②で候補 P_i が送信したメッセージを受信した計算機（P_j とする）が候補でなく，P_i 以外の候補に支配されていなければ，P_j は P_i に支配される．しかし，P_j が候補であるか，あるいは，P_i 以外の候補にすでに支配されている場合，一つの候補だけがリーダになるようにするため，いずれかの候補が脱落しなければならない．簡単に思いつく方法として，たとえば，識別子の小さい候補が脱落するという方法がある．つまり，（a）P_j が候補のとき，（a1）$id(P_i) > id(P_j)$ なら，P_j は候補でなくなり，P_i に支配され，（a2）$id(P_i) < id(P_j)$ なら，P_j は P_i にメッセージを返信しない（この結果，P_i が他の計算機を支配するという作業が打ち切られ，P_i はこれ以上他の計算機を支配できないので，P_i がリーダになることはない）．また，（b）P_j が候補 $P_k(P_k \neq P_i)$ に支配されているとき，（b1）$id(P_i) > id(P_k)$ なら，P_j は P_i に支配され，P_k が候補でなくなり（このことを知らせるために，P_j は P_k にメッセージを送信する），（b2）$id(P_i) < id(P_k)$ なら，P_j は P_i にメッセージを返信しない．

この方法で，どれくらいのメッセージが送信されるだろうか．始動計算機の中で，識別子がm番目に大きい計算機を P_i とする．P_i は，最悪の場合，①始動計算機以外のすべての計算機，② P_i より識別子の小さいすべての始動計算機，および，③ P_i より識別子の大きい1個の始動計算機，の $n-m+1$ 個の計算機にメッセージを送信する．したがって，最悪の場合，全体で，

$$\sum_{i=1}^{k}(n-i+1) = O(n \cdot k) = O(n^2)^{\star}$$

☆ 関数 $f(n)$ に対して，ある正定数 n_0, c が存在し，$n \geqq n_0$ なるすべての n に対して $f(n) \leqq g(n)$ となるとき，
$f(n)=O(g(n))$
と表記する．$n \geqq n_0$ のときに $f(n) \geqq g(n)$ となるとき，
$f(n)=\Omega(g(n))$
と表記する．また，$f(n)=O(g(n))$ かつ $f(n)=\Omega(g(n))$ のとき，
$f(n)=\Theta(g(n))$
と表記する．

個のメッセージが送信される.

このメッセージ数は，これ以上少なくできないのであろうか．これまでに，完全ネットワークでリーダ選択問題を解くメッセージ数 $O(n \cdot \log n)$ のアルゴリズムが知られており，そのメッセージ数がオーダ的に最適であることも知られている[2),29)]．以下では，文献2) のアルゴリズムを，理解しやすさのために少し変更したもの（AG-アルゴリズムとよぶ）を紹介する．

- AG-アルゴリズム

前述のアルゴリズムでは，二つの候補のどちらが脱落するかを決定するのに，識別子の大小で決定した．そのため，識別子が m 番目に大きい計算機が $n-m$ 個の計算機を支配した後に脱落するという状況が生じる可能性があり，最悪の場合，$O(n^2)$ 個のメッセージを使用する．このメッセージ数を減らす一つの方法として考えられるのは，「多くの計算機を支配している候補ほど，候補から脱落しにくくする」ということである．AG-アルゴリズムでは，メッセージ数を少なくするために，識別子ではなく，支配している計算機数（**レベル**とよぶ）の大小によって，どちらの候補が脱落するかを決めている．つまり，（レベル，識別子）の2項組の辞書式順序☆ による比較を行い，小さいほうが脱落する．このレベルを用いる手法は，効率よくリーダ選択問題を解くアルゴリズムでよく利用される手法である．AG-アルゴリズムでは，支配している計算機数をレベルと定義するが，このレベルの定義などを変えることにより，異なる特長のアルゴリズムが得られる[2)]．

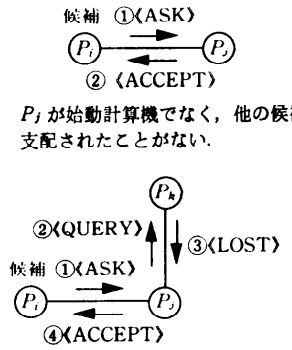
AG-アルゴリズムでは，候補 P_i は他の計算機（P_j とする）にメッセージ《ASK ($level$ (P_i), id (P_i))》を送信し，P_j を支配しようとする．ここで，$level(P_i)$ は，P_i のレベルを表す．

（a） P_j が始動計算機でなく，他の候補に支配されたこともない場合：P_j は P_i に支配され，このことを知らせるため，P_j は P_i に《ACCEPT》を返信する（**図-3**（a））．

（b） P_j が他の候補に支配されたことがある場合：P_j を支配している，あるいは，最後に支配していた候補を P_k とする．P_j は P_k のレベルを問い合わせるために，P_k に《QUERY ($level(P_i)$, $id(P_i)$)》を送信する．($level(P_i)$, $id(P_i)$)>($level(P_k)$, $id(P_k)$) なら，P_j は P_i に支配され，P_k は候補から脱落する．このことを知らせるため，P_k は P_j に《LOST》を送信し，これを受信すると P_j は P_i に《ACCEPT》を送信する（**図-3**（b））．ただし，P_k は候補から脱落するだけで，まだ，P_i に支配されない．

（c） P_j が候補であるか，あるいは，候補から脱落したが，他の候補に支配されたことがない場合：($level(P_i)$, $id(P_i)$)>($level(P_j)$, $id(P_j)$) なら，P_j は P_i に支配され，P_j は候補でなくなる．このことを知らせるため，P_j は P_i に《ACCEPT》を返信する（**図-3**（c））．

そして，前述のアルゴリズムと同様，AG-アルゴリズムでも，他のすべての計算機を支配した候補（P_i とする）がリーダになり，P_i がリーダになったことと $id(P_i)$ を知らせるため，他のすべての計算機に《LEADER ($id(P_i)$)》を送信し，停止する．



（a） P_j が始動計算機でなく，他の候補に支配されたことがない．

（b） P_j が他の候補に支配されたことがある．

（c） P_j が候補，または，他の候補に支配されたことのない脱落した候補．

図-3 AG-アルゴリズムの動作

5. 分散アルゴリズムの評価尺度

5.1 評価基準

一つの分散問題を解く分散アルゴリズムは複数個存在する．これらの優劣を何で評価すればよいであろうか．逐次アルゴリズムの性能の良さは，処理時間が短いとか処理に用いる記憶領域量が少ないとかで判断される．前者は時間計算量，後者は領域計算量という概念を用いて議論され，これらは一般に入力データ量の関数として表される．

分散問題 Q を解く分散アルゴリズム A の性能の良さは，第1に Q を解き終わるまでに用いるすべてのメ

☆ 二つの2項組 (a_1, a_2), (b_1, b_2) に対して，$a_1 < b_1$，あるいは，$a_1 = b_1$ かつ $a_2 < b_2$ のとき $(a_1, a_2) < (b_1, b_2)$ とする．

ッセージ量がどの程度少ないかでまず判断される．一般に分散ネットワーク環境では，メッセージ量が多ければ，可能性として通信費用が高くなり，また処理時間も長くなるからである．メッセージ量は入力情報の具体値の違いにより異なる．さらに，Qの入力情報をある一つの具体値に固定しても，Aの計算状況の違いによって一般にメッセージ量が異なる．これは分散アルゴリズム特有の現象である．それらの最悪時あるいは平均時の評価で行う．

メッセージ量の測り方には2種類ある．ビット複雑度とメッセージ複雑度である．**ビット複雑度**は，メッセージのビット量の総和である．**メッセージ複雑度**は，1メッセージで伝達可能なビット量（**メッセージ長**）を定め，そのもとで必要なメッセージ数である．メッセージ長としては，入力情報を定数個伝達可能なビット量と定めることが一般的である．

第2に分散アルゴリズムが開始してから終了するまでの“処理時間”を評価したい．同期式ネットワークの場合は，この処理時間を分散アルゴリズムが終了するまでのラウンド数と定義してよい．

非同期式ネットワークの場合は，メッセージ遅延時間などの上限値を前提としていないので時間評価が難しい．しかし，一つの受信メッセージに対する局所計算に要する時間が，メッセージ遅延時間に比べてそう長くはない場合には，局所計算の時間をメッセージ遅延時間に含めて考えてよい．そして，時間評価をするときのみ，すべてのメッセージ遅延時間が単位時間である同期式ネットワークで分散アルゴリズムが動作しているとみなし，そのときのラウンド数で時間評価を行う．このような時間評価を**理想時間複雑度**と呼ぶ．メッセージ複雑度が小さいことは必ずしも理想時間複雑度が小さいことを意味しない．メッセージ総数が同じでも，独立に処理できる（因果関係のない）メッセージが少ないならば，理想時間複雑度は多くなる傾向にある．ただし，一つの受信メッセージに対する局所計算に要する時間が，メッセージ遅延時間に比べて無視できない場合は，計算機の局所計算時間も考慮した時間評価をする必要があるであろう．

第3は分散アルゴリズムの適用範囲の広さである．たとえば，適用可能なネットワークが広ければ広いほうがよい．完全ネットワークのときだけ正しく働く分散アルゴリズムよりは，それ以外のネットワークでも正しく働く分散アルゴリズムのほうが好ましい．FIFO型ネットワークでのみ走る分散アルゴリズムよりは，非FIFO型ネットワークでも走るもののほうがよい．また，付加情報を必要とする分散アルゴリズムよりは，必要としない分散アルゴリズムのほうが，適用範囲が広くなるので好ましい．

第4は各計算機の局所計算で必要な**作業用記憶領域量**である．メッセージ長と作業用記憶領域量を気にしないという立場をとれば，FIFO型ネットワークでのみ走る分散アルゴリズムを非FIFO型ネットワークでメッセージ複雑度を落とさずにシミュレートできる．

これらの評価は，ネットワークの計算機総数 n，ネットワークのリンク総数 e，始動計算機数 k などを用いた式で表されることが多い．

5.2 分散アルゴリズムの評価例

- メッセージ複雑度

4.2で紹介したAG-アルゴリズムのメッセージ複雑度を評価してみる．

リーダが他のすべての計算機に，自分がリーダになったことを知らせるために送信する《LEADER》は，$n-1$個である．また，候補 P_i が《ASK》を1個送信すると，このメッセージに対して，①《ASK》を受信した計算機 P_j が，P_j を支配している候補のレベルを問い合わせるためのメッセージ《QUERY》，②《QUERY》に対する返事《LOST》，③P_j が P_i に支配されることを知らせるためのメッセージ《ACCEPT》，の高々3個のメッセージが送信される．そこで，送信される《ASK》の総数を評価する．P_i を任意の始動計算機とする．アルゴリズム終了時に $level(P_i)=m$ が成り立つなら，P_i は高々 $m+1$ 個の《ASK》を送信している．また，アルゴリズム実行中に $level(P_i)\geqq m$ が成り立つ計算機は高々 n/m 個しかない．したがって，ネットワーク全体で送信される《ASK》の総数は，高々

$$\sum_{i=1}^{k}\left(\frac{n}{i}+1\right)=O(n\cdot\log k)=O(n\cdot\log n)$$

である．したがって，AG-アルゴリズムのメッセージ複雑度は $O(n\cdot\log n)$ である．また，このメッセージ複雑度はオーダ的に最適である[29]．

- 理想時間複雑度

4.2で紹介したAG-アルゴリズムの理想時間複雑度を評価してみる．《ASK》を送信した候補を P_i，この《ASK》を受信した計算機を P_j とする．P_j が P_i に支配されるなら，P_i が《ASK》送信後，遅くとも4ラウンド後に，P_i は P_j から《ACCEPT》を受信する．したがって，最終的にリーダになる計算機が始動

してから，$O(n)$ 時間でアルゴリズムは終了する．一方，AG-アルゴリズムでは，多くの計算機を支配している候補が勝ち残っていくので，最終的にリーダとなる計算機が始動するのは，アルゴリズム実行開始時（最も早く始動計算機が始動したとき）から，高々 $O(n)$ 時間以内である．したがって，AG-アルゴリズムの理想時間複雑度は $O(n)$ である．

5.3 汎用性と計算量との関係について

ある問題を解く分散アルゴリズムを開発する場合，そのアルゴリズムが適用可能なネットワークを制限する（汎用性を低くする）ことにより，効率のよいアルゴリズムが得られることがある．このように，適用可能なネットワークに制限を加えたときの計算量を求めることは，興味深い問題であり，これまでに，表-1 のような結果が得られている．

表-1 対象ネットワークの違いによるメッセージ複雑度の違い

対象ネットワークの形状		リーダ選択問題 生成木構成問題	最小費用生成木 構成問題
任意のネットワーク		$\Theta(n\cdot\log n+e)$	
1方向リング		$\Theta(n\cdot\log n)$	
2方向リング	方向感覚なし	$\Theta(n\cdot\log n)$	
	方向感覚あり	$O(n\cdot\log n)$	
完全ネットワーク	方向感覚なし	$\Theta(n\cdot\log n)$	$\Theta(n^3)$
	方向感覚あり	$\Theta(n)$	$O(n^3)$
弦付きリング	各計算機の弦 $O(\log^{(m)}n)$ 本	$\Theta(n)$	$O(n\cdot\log n)$

- 完全ネットワーク

任意のネットワークでリーダ選択問題を解くメッセージ複雑度 $O(n\cdot\log n+e)$ のアルゴリズムが知られている[21),27)]．一方，メッセージをまったく送信しないリンクが存在すると，そのリンクがネットワークのある部分と他の部分を接続する唯一のリンクである場合に，リーダ選択問題が解けない（2個以上の計算機がリーダになってしまう場合がある）のは明らかである．したがって，$\Omega(e)$ はメッセージの複雑度の下界である．実際，任意のネットワークでのリーダ選択問題のメッセージ複雑度の下界は $\Omega(n\cdot\log n+e)$ である[46)]．メッセージ複雑度 $O(n\cdot\log n+e)$ の分散アルゴリズムを完全ネットワークに適用すると，メッセージ複雑度は $O(n^2)(=O(e))$ となる．しかし，前節の AG-アルゴリズムは，適用可能なネットワークを完全ネットワークに制限することにより，メッセージ複雑度 $O(n\cdot\log n)$ でリーダ選択問題を解いている．完全ネットワークで $\Omega(e)$ がメッセージ複雑度の下界にならないのは，メッセージを送信しないリンクが存在してもある候補がすべての隣接計算機を支配してからリーダになれば，ただ一つの候補だけがリーダになるからである．さらに，適用可能なネットワークを**方向感覚付き完全ネットワーク**（完全ネットワークの一つの有向ハミルトン閉路Hを考え，H上での距離によって，各計算機のリンクがラベル付けされていて，そのラベルを各計算機が利用できる．図-4 参照）に制限すると，メッセージ複雑度 $O(n)$ でリーダ選択問題を解くアルゴリズム（LMW-アルゴリズムと呼ぶ）が知られている[31)]．LMW-アルゴリズムでは，各候補は有向ハミルトン閉路 H に現れる順に他の計算機を支配していく．ある候補 P_i が他の候補 P_j を支配するとき，P_j が支配している計算機の範囲を，H 上での距離によって P_i に知らせ，P_j が支配していた計算機を P_i が支配するとみなす（図-5）．P_j が支配していた計算機に P_i がメッセージを送信することなく，それらを支配できるので，メッセージが節約でき，メッセージ

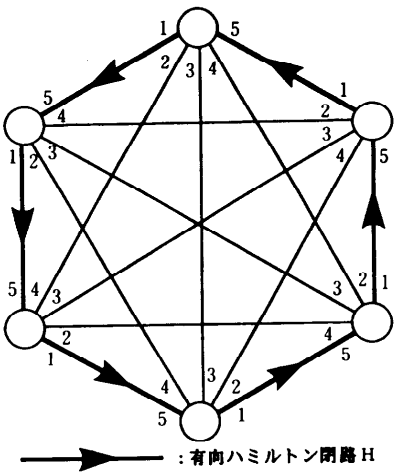


図-4 方向感覚付き完全ネットワーク

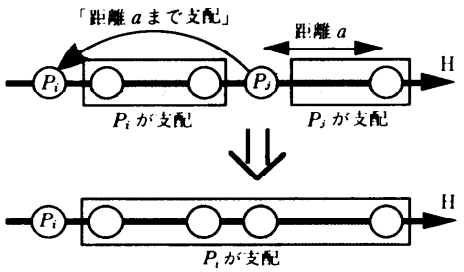


図-5 LMW-アルゴリズムでの支配領域の拡大

複雑度 $O(n)$ でリーダ選択問題を解ける．方向感覚がない場合，P_j が支配していた計算機に通じている P_i のリンクは，そのリンクからメッセージを受信して初めて知ることができるので，この LMW-アルゴリズムの手法は，方向感覚のない完全ネットワークでは利用できない．

● リ ン グ

リングでのリーダ選択問題のメッセージ複雑度の下界が $\Omega(n\cdot\log n)$ であることが知られている[11]．これは，任意のネットワークでリーダ選択問題を解くメッセージ複雑度 $O(n\cdot\log n+e)$ のアルゴリズムをリングに適用したときのメッセージ複雑度がオーダ的に最適であることを意味している（リングでは，$e=n$ が成り立つことに注意）．つまり，適用可能なネットワークをリングだけに制限しても，メッセージ複雑度のオーダは改善できない．リングを**方向感覚付きリング**（各計算機は自分の二つのリンクを局所的に右，左と識別しており，それがネットワーク全体で統一のとれているリング．**図-6** 参照）にするとメッセージ複雑度が改善できるかどうかは未解決の問題である．また，1方向通信可能なリンクで結合されたリング（**1方向リング**）でのリーダ選択問題に関する研究も行われている[16),38)]．

● 弦付きリング

方向感覚付き完全ネットワークや方向感覚付きリングを一般化したのが，**弦付きリング**である．弦付きリングは，方向感覚付きリングにいくつかの弦を規則正しく付加したものであり，計算機は自分の弦を(有向)リング上での距離によって識別できる（**図-7**）．弦の最も少ない弦付きリングは方向感覚付きリングであり，弦の最も多い弦付きリングは方向感覚付き完全ネットワークである．リーダ選択問題の方向感覚付き完全ネットワークでのメッセージ複雑度は $\Theta(n)$ であり，

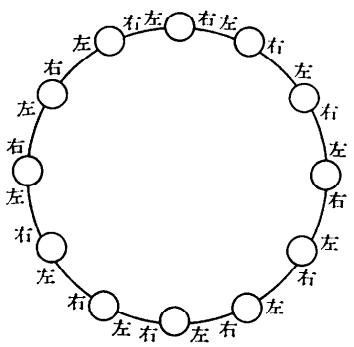


図-6 方向感覚付きリング

方向感覚付きリングでのメッセージ複雑度は $O(n\cdot\log n)$ であるので，方向感覚付きリングにどれくらいの弦を付加すれば，メッセージ複雑度が $O(n)$ に下がるかを解明することは興味深いことである．文献5)は，各計算機に $O(\log n)$ 本の弦をうまく持たせれば，リーダ選択問題がメッセージ複雑度 $O(n)$ で解けることを示している．この結果は，方向感覚付き完全ネットワークでリーダ選択問題を解くメッセージ複雑度 $O(n)$ のアルゴリズムの改良とみなせる．文献36)は，この結果をさらに改良し，任意の自然数の定数 m について，各計算機に $O(\log^{(m)}n)$ 本☆ の弦をうまく持たせれば，メッセージ複雑度を $O(n)$ にできることを示している．

これまでは，メッセージ複雑度の最悪時評価について述べてきたが，平均時評価[10),38)]や実験的評価[25)]なども行われている．また，どのような環境下（非同期式ネットワークでは，通信遅延などの違いにより，無限に多くの異なる環境が考えられる）においても，同じ環境下で動作させれば，同じ問題を解く他のアルゴリズムより遅く終了することはないという意味で最速なアルゴリズムのメッセージ複雑度に関する議論もある[37),51)]．

6. その他の分散ネットワーク・モデル

ここでは，標準ネットワーク・モデル以外のネットワーク・モデルで得られている主な結果を紹介する．

6.1 同期式ネットワーク（表-2）

● リ ン グ

5.3 で述べたように，非同期式リングでのリーダ選

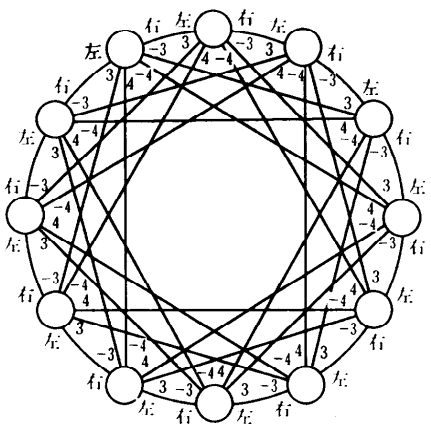


図-7 弦付きリング

☆ $\log^{(m)}n=\begin{cases}\log n & (m=1)\\ \log(\log^{(m-1)}n) & (m\geq 2)\end{cases}$

択問題のメッセージ複雑度は $\Theta(n\cdot\log n)$ である．文献 18) や 47) は，同期式リングでリーダ選択問題を解くメッセージ複雑度 $O(n)$ のアルゴリズムを示している．このアルゴリズムでは，計算機の識別子に関して，これまでのアルゴリズムのように大小比較するだけではなく，「識別子 id を含むメッセージは，2^{id} 単位時間に1回の割合でリングを進む」というように，識別子をメッセージ伝送の制御のために利用している．ただし，理想時間複雑度は $O(n\cdot 2^{id})$ である．文献 20) は，この理想時間複雑度を $O(n\cdot 2^n+id^2)$ に改良した．一方，識別子を大小比較のみに利用するならば，同期式リングでのリーダ選択問題のメッセージ複雑度の下界が $\Omega(n\cdot\log n)$ である[18]．

表-2 同期式ネットワークにおけるリーダ選択問題

対象ネットワークの形状		メッセージ複雑度	理想時間複雑度
2方向リング	識別子の操作制限なし	$\Theta(n)$	$O(n\cdot 2^n+id^2)$
	識別子の操作比較のみ	$\Theta(n\cdot\log n)$	$\Theta(n)$
完全ネットワーク		$\Theta(n\cdot\log n)$	$\Theta(\log n)$

- 完全ネットワーク

文献 2) は，同期式完全ネットワークにおけるリーダ選択問題について，識別子の利用法に制限を設けなくてもメッセージ複雑度の下界が $\Omega(n\cdot\log n)$ であることを示している．非同期式完全ネットワークでのメッセージ複雑度の上界が $O(n\cdot\log n)$ である（たとえば，AG-アルゴリズム）ので，完全ネットワークにおいては，同期式にしてもメッセージ複雑度のオーダが改善できないことが分かる．さらに，文献 2) は，同期式完全ネットワークでリーダ選択問題を解くメッセージ複雑度 $O(n\cdot\log n)$（オーダ的に最適）のアルゴリズムの理想時間複雑度の下界が $\Omega(\log n)$ であることを示し，メッセージ複雑度 $O(n\cdot\log n)$ 理想時間複雑度 $O(\log n)$ のアルゴリズムを示している．

- 非同期式ネットワークでのシミュレート

これまで述べたように，同期式ネットワークでは，非同期式ネットワークに比べ，問題を効率よく解ける可能性がある．同期式ネットワークを考える他の利点の一つとして，同期式ネットワークでは，分散アルゴリズムの開発や正当性の証明が容易であることがあげられる．このため，非同期式分散アルゴリズムの設計方法の一手法として，まず，同期式アルゴリズムを開発し，それを非同期式ネットワークで効率よくシミュレートするという方法が考えられる．文献 7) は，同期式アルゴリズムを非同期式ネットワークで効率よくシミュレートするための一般的手法を示している．シミュレートの効率は，ネットワークの形状にも依存するが，文献 28), 42), 48) は，シミュレートの効率に深く関係する“ネットワークの t-spanner”について議論している．

6.2 故障のあるネットワーク（表-3）

ここでは，停止故障についてのみ考える．非同期式ネットワークでは，故障と遅延を有限時間内で区別できないので，故障計算機，故障リンクは検出不能である[33),34)]．（ただし，文献 49) は，故障が検出できるという仮定のもとで，ネットワークの連結性判定問題について考察している．）このため，非同期式ネットワークでは故障耐性のあるアルゴリズムを開発することは容易ではない．実際，アルゴリズム実行中に故障する計算機が1個でもあると，リーダ選択問題が解けない[17]．したがって，すべての故障はアルゴリズム実行開始前に生じており，実行中には新たな故障は生じないと仮定することが多い（ただし，文献 12) は計算機が断続的に故障するリング，また，文献 1) はリンクが断続的に故障する完全ネットワークにおけるリーダ選択問題について考察している）．一方，同期式ネットワークでは，タイム・アウトの手法を使えば，故障を検出できるので，リーダ選択問題などを解ける．

表-3 故障のあるネットワークでのリーダ選択問題のメッセージ複雑度

対象ネットワークの形状		メッセージ複雑度	
任意のネットワーク		解けない	(付加情報：なし)
2方向リング		$O(n(\log n)^3+e)$	(付加情報：計算機総数)
		$O(n^2)$	(付加情報：隣接識別子)
完全ネットワーク	方向感覚なし	$\Theta(n\cdot\log k+k\cdot f_P)$	(高々 f_P 個の故障計算機)
		$\Theta(n\cdot\log k+n\cdot f_L)$	(高々 f_L 個の故障リンク)
	方向感覚あり	$\Theta(n+k\cdot f_P)$	(高々 f_P 個の故障計算機)
		$\Theta(n+k\cdot f_L+f_L\cdot\log f_L)$	(高々 f_L 個の故障リンク)

文献 15), 33), 34) は，問題が解けるためにはどの程度の同期性があればよいかについて議論している．

- 任意のネットワーク，リング

非同期式ネットワークでは，対象を任意のネットワークやリングにすると，故障計算機が1個だけで，その故障がアルゴリズム実行開始前に生じていると仮定しても，リーダ選択問題が解けない[9),22)]．そこで，計算機の停止を要求せずに，リーダ選択問題を弱く解くアルゴリズム[9)]や，ネットワークの計算機総数，過半数値や隣接計算機の識別子などの付加情報を利用してリーダ選択問題を解くアルゴリズム[3),9),22),32)]が知られている．

- 完全ネットワーク

文献 9) は，故障計算機のある（方向感覚がない）完全ネットワークにおけるリーダ選択問題について，故障計算機数の上限値 $f_P(f_P<n/2)$ を付加情報として利用するメッセージ複雑度 $\Theta(n\cdot\log k+k\cdot f_P)$ のアルゴリズムを示している．また，文献 35) は，方向感覚を利用できれば，メッセージ複雑度が $\Theta(n+k\cdot f_P)$ になることを示している．リンクが故障している完全ネットワークにおけるリーダ選択問題のメッセージ複雑度については，故障リンク数の上限値 $f_L(f_L<n-1)$ が利用できる場合，方向感覚がなければ $\Theta(n\cdot\log k+n\cdot f_L)$，方向感覚があれば $\Theta(n+k\cdot f_L+f_L\cdot\log f_L)$ である[35)]．したがって，故障が存在する場合にも，完全ネットワークの方向感覚を利用すれば，メッセージ複雑度のオーダを改善できる．

また，LAN（ローカル・エリア・ネットワーク）モデル上での故障を考慮した分散アルゴリズムに関する研究も行われている[44),45)]．

6.3 識別子のないネットワーク

文献 4) は，匿名リングでは，同期式，非同期式にかかわらず，リーダ選択問題が解けないことを示している．文献 19), 26) は，匿名リングでリーダ選択問題を解く確率的アルゴリズムを示している．文献 6) は，匿名リングで解ける問題の特徴付けを行っている．文献 50) は，リーダ選択問題が解ける匿名ネットワークのグラフ理論的な立場からの特徴付けを行っている．

7. あ と が き

分散アルゴリズムは，今後ますます研究の必要な分野であるが，特に次の課題が重要と思われる．

- 分散アルゴリズムの設計技法の確立
- 分散アルゴリズムの正しさの検証法の確立
- メッセージ複雑度や理想時間複雑度以外の分散アルゴリズムの評価基準の考案とそれによる評価
- 計算機やリンクの故障を考慮した分散アルゴリズムの設計

最後に，分散アルゴリズムに興味をもたれた方のために，分散アルゴリズムの論文が比較的多く発表される会議および論文誌を紹介しておく．

（会　議）

- Annual ACM Symposium on Principles of Distributed Computing
- International Workshop on Distributed Algorithms
- *IEEE Distributed Computing Systems*

（論 文 誌）

- *Distributed Computing*
- *Journal of Parallel and Distributed Computing*

謝辞　本解説執筆の機会を与えていただいた編集委員会の委員各位に深謝いたします．日頃ご指導いただく大阪大学都倉信樹教授および首藤勝教授に感謝いたします．また，貴重なご意見をいただいた査読者の方に感謝いたします．本解説は一部昭和 63 年度マツダ研究助成金，平成元年度稲盛財団研究助成金，平成元年度文部省科学研究費補助金一般研究（C）（課題番号 01580030），一般研究（B）（課題番号 01460150），平成2年度重点領域研究（1）（課題番号 02249102），総合研究（A）（課題番号 02302047）および奨励研究（A）（課題番号 02750279）の補助を受けている．

参 考 文 献

1) Abu-Amara, H.H.: Fault-Tolerant Distributed Algorithm for Election in Complete Networks, Proc. of 2nd International Workshop on Distributed Algorithms (LNCS 312), pp. 355-373 (1987).
2) Afek, Y. and Gafni, E.: Time and Message Bound for Election in Synchronous and Asynchronous Complete Networks, Proc. of 4th PODC, pp. 186-195 (1985).
3) Afek, Y. and Saks, M.: Detecting Global Termination Conditions in the Face of Uncertainty, Proc. of 6th PODC, pp. 109-124 (1987).
4) Angluin, D.: Local and Global Properties in Networks of Processors, Proc. of 12th STOC, pp. 82-93 (1980).

5) Attiya, H., van Leeuwen, J., Santoro, N. and Zaks, S.: Efficient Elections in Chordal Ring Networks, Algorithmica, Vol. 4, pp. 437-446 (1989).
6) Attiya, C., Snir, M. and Warmuth, M.: Computing on an Anonymous Ring, Proc. of 4th PODC, pp. 196-203 (1985).
7) Awerbuch, B.: Complexity of Network Synchronization, JACM, Vol. 32, No. 4, pp. 804-823 (1985).
8) Awerbuch, B. and Gallager, R. G.: Distributed BFS Algorithms, Proc. of 26th FOCS, pp. 250-256 (1985).
9) Bar-Yehuda, R., Kutten, S., Wolfstahl, Y. and Zaks, S.: Making Distributed Spanning Tree Algorithms Fault-Resilent, Proc. of STACS 87 (LNCS 247), pp. 432-444 (1987).
10) Bodlaender, H. L. and van Leeuwen, J.: New Upperbounds for Decentralized Extrema-finding in a Ring of Processors, Proc. of STACS 86, pp. 119-129 (1986).
11) Burns, J. E.: A Formal Model for Message Passing Systems, TR-91, Computer Science Department, Indiana University (1980).
12) Chan, M. Y. and Chin, F. Y. L.: Optimal Resilient Ring Election Algorithms, Proc. of 2nd International Workshop on Distributed Algorithms (LNCS 312) (1987).
13) Chandy, K. M. and Lamport, L.: Distributed Snapshots: Determining Global States of Distributed Systems, ACM Trans. Computer Systems, Vol. 3, No. 1, pp. 63-75 (1985).
14) Dijkstra, E. W. and Scholten, C. S.: Termination Detection for Diffusing Computations, Inf. Process. Lett., Vol. 11, No. 1, pp. 1-4 (1980).
15) Dolev, D., Dwork, C. and Stockmeyer, L.: On the Minimal Synchronism Needed for Distributed Consensus JACM, Vol. 34, No. 1, pp. 77-97 (1987).
16) Dolev, D., Klawe, M. and Rodeh, M.: An $O(n \log n)$ Unidirectional Distributed Algorithm for Extrema Finding in a Circle, Journal of Algorithms, Vol. 3, pp. 245-260 (1982).
17) Fischer, M., Lynch, N. A. and Paterson, M. S.: Impossibility of Distributed Consensus with One Faulty Processes, JACM, Vol. 32, No. 2, pp. 374-382 (1985).
18) Frederickson, G. N. and Lynch, N. A.: The Impact of Synchronous Communication on the Problem of Electing a Leader in a Ring, Proc. of 16th STOC, pp. 493-503 (1984).
19) Frederickson, G. N. and Santoro, N.: Breaking Symmetry in Synchronous Networks, VLSI Algorithms and Architectures (LNCS 227), pp. 26-33 (1986).
20) Gafni, E.: Improvements in the Time Complexity of Two Message-Optimal Election Algorithms, Proc. of 4th PODC, pp. 175-185 (1985).
21) Gallager, R. G., Humblet, P. A. and Spira, P. M.: A Distributed Algorithm for Minimum-Weight Spanning Trees, ACM TOPLAS, Vol. 5, No. 1, pp. 66-77 (1983).
22) Goldreich, O. and Shrira, L.: Electing a Leader in a Ring with Link Failures, Acta Informatica, Vol. 24, pp. 79-91 (1987).
23) 萩原兼一:(招待講演)分散アルゴリズムの複雑度について, The Logic Programming Conference '87, pp. 11-20 (1987).
24) 萩原兼一:分散アルゴリズム入門, コンピュータ・サイエンス bit, No. 256, pp. 4-26, 共立出版 (1988).
25) 池川, 山下, 阿江:リング・ネットワークにおけるリーダー選挙アルゴリズムの実験的評価, 信学技報, COMP 89-21 (1989).
26) Itai, A. and Redeh, M.: Symmetry Breaking in Distributive Networks, Proc. of 22nd FOCS, pp. 150-158 (1981).
27) Johansen, K. E., Jorgensen, U. L., Nielsen, S. H., Nielsen, S. E. and Skyum, S.: A Distributed Spanning Tree Algorithm, Proc. of 2nd International Workshop on Distributed Algorithms (LNCS 312), pp. 1-12 (1987).
28) 梶谷, 宮野, 上野, Thulasiraman, T.: 分散ネットワークの最適シンクロナイザーについて, 信学技報, IN88-103 (1988).
29) Korach, E., Moran, S. and Zaks, S.: Tight Lower and Upper Bounds for Some Distributed Algorithms for a Complete Network of Processors, Proc. 3rd PODC, pp. 199-207 (1984).
30) Korach, E., Rotem, D. and Santoro, N.: Distributed Algorithm for Finding Centers and Medians in Networks, ACM Trans. Prog. Lang. Syst., Vol. 6, No. 3, pp. 380-401 (1984).
31) Loui, M. C., Matsushita, T. A. and West, D. B.: Election in a Complete Network with a Sense of Direction, Inf. Process. Lett., Vol. 22, No. 4, pp. 185-187 (1986).
32) 増澤, 萩原, 都倉:辺故障を考慮したある分散アルゴリズム, 信学論 (D), Vol. J69-D, No. 10, pp. 1394-1405 (1986).
33) 増澤, 萩原, 都倉:プロセッサ故障診断のための分散アルゴリズム, 信学論 (D), Vol. J70-D, No. 6, pp. 1092-1103 (1987).
34) 増澤, 萩原, 都倉:リンク故障診断のための分散アルゴリズム, 信学論 (D), Vol. J71-D No. 12, pp. 2648-2658 (1988).

35) Masuzawa, T., Nishikawa, N., Hagihara, K. and Tokura, N.: Optimal Fault-Tolerant Distributed Algorithms for Election in Complete Networks with a Global Sense of Direction, Proc. of 3rd International Workshop on Distributed Algorithms (LNCS 392), pp. 171-182 (1989).
36) 中野，増澤，都倉：コーダルリング上のリーダ選択分散アルゴリズム，信学技報，COMP 89-115 (1990).
37) 大戸，茨木：グラフ構築およびリーダ選出問題における時間最小分散アルゴリズムについて，信学論 (D I), Vol. J72-D-I, No. 10, pp. 726-733 (1989).
38) Pachl, J., Korach, E. and Rotem, D.: Lower Bounds for Distributed Maximum Finding Algorithms, JACM, Vol. 31, pp. 905-918 (1984).
39) 朴，増澤，萩原，都倉：幅優先生成木構成問題の効率のよい分散アルゴリズム，信学論 (D), Vol. J71-D No. 7, pp. 1176-1188 (1988).
40) 朴，増澤，都倉：トポロジ変化時の重み最小生成木を再構成する分散アルゴリズムについて，情報処理学会アルゴリズム研究会資料，AL13-2 (1990).
41) 朴，増澤，萩原，都倉：ネットワークの連結性関連問題を解く効率のよい分散アルゴリズム，信学論 (D I), Vol. J72-D-I, No. 5, pp. 343-356 (1989).
42) Peleg, D. and Schaffer, A. A.: Graph Spanners, Journal of Graph Theory, Vol. 13, No. 1, pp. 99-116 (1989).
43) Peleg, D. and Ullman, J. D.: An Optimal Synchronizer for Hypercube, Proc. of 6th PODC, pp. 77-85 (1987).
44) 齋藤，辻野，都倉：高速 LAN モデルにおける生成木維持問題，信学技報，COMP 89-22 (1989).
45) 齋藤，都志，辻野，都倉：高速 LAN モデルとその上での更新アルゴリズムについて，信学論 (D I), Vol. J72-D-I, No. 2, pp. 108-116 (1989).
46) Santoro, N.: On the Message Complexity of Distributed Problems, International Journal of Computer and Information Sciences, Vol. 13, No. 3, pp. 131-147 (1984).
47) Vitanyi, P. M. B.: Distributed Elections in an Archimedean Ring of Processors, Proc. of 16th STOC, pp. 542-547 (1984).
48) 和田，川口，藤嶋：有向，無向グラフに対する最適な t-spanner の構成について，情報処理学会第39回全国大会論文集，3L-8 (1989).
49) 和田，守谷，川口，森下：故障発生時における計算機網の連結性を判定する分散アルゴリズム，情報処理学会アルゴリズム研究会資料(発表予定).
50) Yamashita, M. and Kameda, T.: Computing on an Anonymous Network, Proc. 7th PODC, pp. 117-130 (1988).
51) 山下，茨木：時間最小アルゴリズムのメッセージ複雑さ，信学技報，COMP 88-97 (1989).

(平成2年1月10日受付)