1999年度日本オペレーションズ・リサーチ学会
秋季研究発表会

線形ネットワークにおける時間複雑度を重視した分散ソーティングアルゴリズム

02201913　NTT コミュニケーション科学基礎研究所　佐々木 淳　SASAKI　Atsushi

1　はじめに

様々なアプリケーションにおいて，ソーティングは基本かつ重要な問題のひとつである．そのため，分散アルゴリズムの分野においてもソーティングアルゴリズムが研究されているが，その基本的な問題設定，および，評価基準が逐次・並列アルゴリズムとは異なる．そこで，本稿では，最も単純なネットワークである線形ネットワークを対象に，分散システムの仮定の下で，逐次・並列アルゴリズムにおける概念に準じたソーティング問題を考える．

2　モデルと問題の定義

本稿では，プロセス $P_1, P_2, \ldots, P_n$ を $P_i (1 \leq i < n)$ と P_{i+1} の間に双方向リンクを持つように結合した線形ネットワークを扱う．一般性を失うことなく，P_1 が左端となるように水平に置かれていると仮定する．このとき，各プロセスは隣接プロセスの存在と相対位置（左右）を認識できるが，両端のプロセス P_1, P_n を除き，自分の大域的な位置は認識できないとする．つまり，プロセス P_i は，P_{i-1} が左に，P_{i+1} が右に存在することは認識できるが，プロセスの番号 $(i-1, i, i+1)$ は認識できない．また，プロセス数 n も知らないとする．

このような仮定の下で，各プロセスが隣接プロセスとの通信を繰り返して問題を解くが，そのモデルとして，同期システムと非同期システムとがある [1]．一般に，分散アルゴリズムの評価には，時間複雑度とメッセージ複雑度の二つの基準が存在する．前者は同期システムではラウンド数，非同期システムではメッセージの最長鎖に属するメッセージ数で表され，後者はメッセージの総数で表される．

次に本稿で扱う分散ソーティング問題を定義する．まず，初期状態において，各プロセスはソーティングの対象となる値をひとつずつ持っている．そして，最終状態で P_i に i 番目に小さな値がくるように値を移動させるのが問題である．なお，各プロセスが初期状態で $k \geq 2$ 個の値を持っている場合に対しては，Hofstee ら [2] が nk 時間アルゴリズムを構築している．

3　アルゴリズムの検討

既存の分散ソーティングアルゴリズム [3] では，メッセージ複雑度のみで評価している．しかし，メッセージ複雑度のみを評価基準にしてしまうと，同時に通信できるメッセージを減らす方向でアルゴリズムが設計されるため，時間複雑度が大幅に悪化してしまう．本稿では，メッセージ複雑度よりも時間複雑度を重視したアルゴリズムの構築を行う．そこで，以下では同時に通信できるメッセージを最大限に利用して，時間複雑度の削減を検討する．最終的には非同期システムにおけるアルゴリズムを構築するが，説明を簡単にするために，本質的な部分は同期システムで説明する．

同期システムにおける本ソーティング問題は，線形アレイ上の並列ソーティング問題と非常に良く似ている．並列ソーティング問題では，奇偶交換ソートにより，最適な時間でソートすることができる．しかし，分散環境では各プロセスが自分の大域的な位置を知らないため，奇偶交換ソートをそのまま行うことができない．奇偶交換ソートを行う前に各プロセスの位置を判別することは可能だが，それに $n-1$ ラウンドを要するため，全体で $2n-1$ ラウンド要する．さらに，非同期環境においては起動プロセスが左右端プロセスに限定されてしまうという問題もある．

そこで，左右の隣接プロセスと同時に通信し，どのプロセスも起動から終了まで公平に動作する方法を考える．基本的なアイデアは，各プロセスにおいて初期値 u を v_L, v_R にコピーし，この二つの値を用いてネットワーク全体として奇偶交換ソートを行うことである．

4　アルゴリズム

本節ではアルゴリズムの本質的な部分のみを示す．実際にはプロセス数 n を求める処理も同時に行っているが，紙面の都合で省略する．なお，$x \in \boldsymbol{R}$ と $null$ に対し，$\max(x, null) = x$，$\min(x, null) = x$ と定義する．また，$null$ への送信は実際には実行されず，$null$ プロセスからは常に $null$ メッセージを受信しているとする．

1. 定数と変数の定義と初期値：

   t: 時刻，初期値：-1.

   P_L, P_R: 左 (右) 側のプロセス．左 (右) 端プロセスでは $null$.

   u: P_i が持つ初期値（定数）.

   v_L, v_R: P_i が保持している値，初期値：u.

   w_L, w_R: テンポラリ変数.

2. アルゴリズム (for P_i)：

   $t := t + 1$
   if $1 \leq t \leq n$ then
   　　$receive(v'_L, P_L)$
   　　$receive(v'_R, P_R)$
   　　$w_R := \min(v_R, v'_R)$
   　　$w_L := \max(v_L, v'_L)$
   　　$v_R := \max(w_L, w_R)$
   　　$v_L := \min(w_L, w_R)$
   $send(v_L, P_L)$
   $send(v_R, P_R)$

5 正当性の証明，複雑度解析

本アルゴリズムは要素数$2n$の奇偶交換ソートと同じ動作をするので，nラウンド後には$v_L = v_R$となる．よって，本アルゴリズムの正当性は自明で，時間複雑度はnラウンドである．一方，各ラウンドで$2(n-1)$個のメッセージが送信されるため，メッセージ複雑度は$2n(n-1)$となる．なお，メッセージ複雑度の下界は$\frac{n^2}{2}$である．

次に，本アルゴリズムをそのまま非同期システムに拡張した場合を考える．起動プロセスは選択できないので，起動メッセージが全プロセスに伝わるのに最大$n-1$時間要する．よって，両端のプロセスが本アルゴリズムの実行を開始するのは，最も遅いときで$n-1$時間後になる．従って，非同期システムにおける時間複雑度は$2n-1$時間となる．さらに，どの時点においても両隣からメッセージを受信しないと内部処理が行われない（両端のプロセスは常に一方から*null*を受信）ので，全体の整合性が保証される．また，メッセージ複雑度は同期システムと変わらず，$2n(n-1)$である．

6 単純なアルゴリズムとの比較

最も単純なアルゴリズムとして，すべての情報をあるひとつのプロセスに集め，そこでソートして配布するという処理が考えられるので，それとの比較を行う．ここでは非同期システムにおいて比較を行う．

最初にこの単純なアルゴリズムの複雑度を解析する．起動の通知，値の収集，ソート結果の配分にそれぞれ最悪$n-1$時間要するので，合計$3(n-1)$時間必要になる．一方，メッセージ複雑度は，情報収集・分配にそれぞれ$\frac{n^2-n}{2}$，さらに，起動の通知に$n-1$必要なので合計n^2-1となる．

本アルゴリズムは単純なアルゴリズムに比べ，メッセージ複雑度が約2倍になるものの，時間複雑度は約n時間短くて済む．また，起動プロセスが複数存在する場合，一ヶ所に集めて処理するにはどちらか一方を選択して情報を収集し直す必要があるが，本アルゴリズムは逆に起動プロセスが多いほど速くなるという利点がある．

さらに，耐故障性に関しても本アルゴリズムの方が優れている．一ヶ所に集めて処理する場合には，リンク・プロセスの故障により一切ソートされない範囲ができてしまう．一方，本アルゴリズムでは，リンク・プロセスが故障（$\neq$ ビザンティン故障）しても，それに接続するリンクを*null*としてしまえば，それを介した情報伝達はできなくなるものの，その両側でそれぞれソート列を作成することは可能である．

7 既存のアルゴリズム [3] との比較

既存のソーティングアルゴリズムは木を対象にしている．また，ソーティング問題の定義が本アルゴリズムとは異なり，uidの順に並べることを目的としているので，まずランキングをして，その後，配送を行っている．さらに，メッセージ複雑度を低く抑えることを第一にしているため，時間複雑度が大きくなっている．具体的には，メッセージ複雑度は$\frac{n^2}{2}+O(n)$に抑えられているものの，同時に起こる通信がほとんどないため，時間複雑度も$O(n^2)$になっている．これは，グラフ形状を線形ネットワークにした場合も変わりない．さらに，木の根を決める処理（木の構築）も必要になる．

単純に比較すると，本アルゴリズムは既存のアルゴリズムに比べメッセージ複雑度が4倍程度になるものの，時間複雑度は約n分の1になる．上記で述べたように，本アルゴリズムと既存のソーティングアルゴリズムとは仮定が異なるため，単純にどちらが良いかは判断できない．しかし，本アルゴリズムの仮定は既存のソーティングアルゴリズムの仮定の特殊化なので，本稿の仮定が満たされるような環境においては，既存のアルゴリズムよりも本アルゴリズムの方が適しているであろう．

なお，ファイルのソート等，大きな通信コストを要する場合には，一旦仮の値でソートして，その後，仮の値を持つプロセスにファイルを送信することになる．しかし，この送信も端から順にソートを行う線形ネットワークにおいては，仮の値が存在する方向が一意に分かるため，高々$n-1$時間で終了する．

8 おわりに

本稿では，既存の分散ソーティングアルゴリズムとは異なる視点で新たなアルゴリズムを示した．ソートすべき値の数を意図的に倍にすることで高速化できるという，興味深い結果が得られた．

本アルゴリズムの応用範囲は既存のものに比べかなり狭いが，メッセージ複雑度が4倍程度で抑えられているのに対し，時間複雑度が約$\frac{1}{n}$に大幅に改善できている．さらに，各プロセスが公平に動作するという特徴も持つ．各プロセスの持つ値の数が1を含む任意の数の場合への対応，メッセージ複雑度を下げること，グラフ形状を変えたときのアルゴリズムの構築，uid等を基準とした並べ替えへの対応などが今後の課題である．

参考文献

[1] 亀田恒彦，山下雅史：分散アルゴリズム，近代科学社 (1994).

[2] H. Peter Hofstee, Alain J. Martin, and Jan L.A. Van De Snepscheut: Distributed Sorting, *Science of Computer Programming*, Vol. 15, No. 2-3, pp. 119-133 (1990).

[3] Shmuel Zaks: Optimal Distributed Algorithms for Sorting and Ranking, *IEEE Transactions on Computers*, Vol. C-34, No. 4, pp. 376-379 (1985).