並列アルゴリズムに適用した確率アルゴリズムの性能評価の試み

石原 鑑[†] 萩原兼一[††]
魚井宏高[†††] 首藤 勝[†††]

並列アルゴリズムにおいて最悪時の時間計算量を改善するために，乱数を導入する研究が進んでいる．しかし決定性アルゴリズムと確率アルゴリズムとは，必ずしも同一の規準で評価されていない．超立方体ネットワークのパケットルーティング問題においても，並列決定性オブリビアス (oblivious) アルゴリズムには最悪時の個別問題に対する解析がなされ，(Valiant-Brebner が開発した) 並列確率アルゴリズム (以下 VB 法) には平均的な解析がなされている．これより VB 法が“オーダ的に優れている”ことがわかるが，実際規模の具体問題で VB 法が優位であるかは解明されているとは言いがたい．そこで，超立方体の次元を大きくしていき，オブリビアスアルゴリズムの一つである LR 法が VB 法にどの程度まで優位であるかをシミュレーションにより実測値で評価した．その結果ある個別問題に関しては，10次元以上の超立方体において，確かに LR 法が VB 法より，問題を解くのに要するステップ数が多くなることが確認できた．本論文ではオブリビアスでない決定性ルーティングアルゴリズムを提案する．シミュレーションにより，この方法では大幅なステップ数の減少が見られ，15次元以下の問題規模ならどのような個別問題に対しても LR 法よりステップ数が多くなく，LR 法では最悪となる個別問題に関しても VB 法より性能が良いことがわかった．

Performance Evaluation of Randomized Algorithms Applied for Parallel Algorithms

Akira Ishihara,[†] Kenichi Hagihara,[††] Hirotaka Uoi[†††] and Masaru Sudo[†††]

Randomized algorithms, which make random choices in the course of their execution, have been studied to improve time complexity in the worst case. The packet routing problem for the n-dimensional hypercubic networks is considered. The Valiant-Brebner's randomized parallel algorithm finishes in $O(\log N)$ time with probability close to 1 and any deterministic oblivious parallel algorithm does in $O(\sqrt{N})$ time in the worst case permutation where N is the number of processors. However, it is not clear that the former is superior to the latter in practice. In this paper, these two algorithms and another deterministic one we propose are compared. It is shown that our algorithm works more efficiently than Valiant-Brebner's in the worst case permutation where n is less than or equal to fifteen and also always works more efficiently than the oblivious one. We conjecture that the former result is valid for any n.

1. はじめに

決定性の逐次アルゴリズムでは最悪時の時間計算量改善のために乱数を導入する研究が行われている．乱数導入の利点は，最悪時という概念をなくし，いかなる個別問題に対しても平均的に良い性能を示すアルゴリズムを実現できることにある．実行過程で乱数を用いて確率的な選択をするアルゴリズムを確率アルゴリズムと呼ぶ[1]．

逐次アルゴリズムほど多くはないが，並列アルゴリズムに関しても同様な試みが行われている．超立方体ネットワークのパケットルーティング問題[2]もその1つである．以降で議論するアルゴリズムはすべて並列アルゴリズムであるので，単にアルゴリズムと書くことにする．乱数を導入する際には，計算時間の改善の程度を客観的に評価する必要がある．しかし決定性アルゴリズムと確率アルゴリズムとは，必ずしも同一の規準で評価されていない[1]．たとえばパケットルーティング問題の場合，決定性オブリビアス (oblivious) アルゴリズムに関しては最悪時の個別問題に対する解析がなされ (命題1参照)，(Valiant-Brebner が開発した) 確率アルゴリズム (以下 VB 法) に関しては平均的な解析がなされている (命題2参照)．これによ

† 三菱電機(株)
Mitsubishi Electric Corporation
†† 奈良先端科学技術大学院大学情報科学研究科情報システム学専攻
Department of Information Systems, Graduate School of Information Science, Advanced Institute of Science and Technology, Nara
††† 大阪大学基礎工学部情報工学科
Department of Information and Computer Sciences, Faculty of Engineering Sciences, Osaka University

り大きな問題規模で，決定性オブリビアスアルゴリズムにとっての最悪時で比較すると VB 法が“オーダ的に優れている”という結論がでているが，実際規模の具体的問題で VB 法が優位であるかは解明されているとは言いがたい．

そこで，この問題で超立方体の次元を大きくしていった場合，オブリビアスアルゴリズムの1つである **Left-to-right**（以下 LR 法）が VB 法にどの程度まで優位であるかをシミュレーションにより実測値で評価した．本論文では，確率アルゴリズムの実行により発生する乱数系列の最悪時のことは考慮していない．最悪時とはすべて，LR 法が問題を解くのに要するステップ数が命題1の下限値を越える個別問題のことを指す．乱数系列を変えて何度か確率アルゴリズムを実行してみたところ，ステップ数に与える影響は無視できるほどであった．

シミュレーションにより，最悪時に関しては**図1**のような結果を得た．これより 10 次元以上の超立方体（プロセッサ数は 1024 個以上）において，LR 法が VB 法よりステップ数が多くなる個別問題が存在することが確認できた．しかし確率アルゴリズムは，乱数発生の処理を必要とすること，その乱数により動作が決定的に決まらないことなどから，決定性アルゴリズムで処理できるものであればその方が良いと考え，最悪時でも VB 法より有効な決定性ルーティングアルゴリズムを提案する．ただし，これはオブリビアスアルゴリズムではない．

本論文で提案する決定性アルゴリズム **Greedy**（以下 GR 法）により大幅なステップ数の減少が見られ，15 次元までの超立方体なら，どのような個別問題に対しても LR 法よりステップ数が多くなく，最悪時にも VB 法より性能がよいことがわかった（図1参照）．GR 法の各ステップの処理は，少し LR 法より多いが，ステップ数自体の改善度が高いので，全体として評価した場合に，実際的なネットワーク規模と考えられる少なくとも 15 次元までの超立方体ネットワークではこの方法は LR 法や VB 法よりも効果的と思われる．

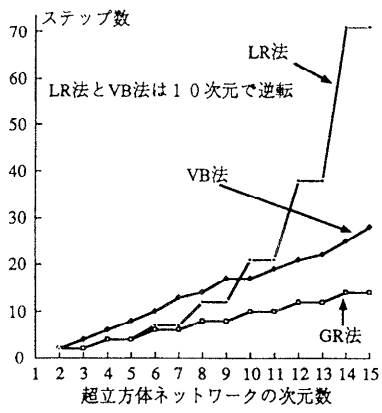


図 1 逆順置換に対する結果（ステップ数）
Fig. 1 The results in reverse permutation.

本論文では，すべて 15 次元までのネットワークで考えているが，これは決して 16 次元以上で否定的な結果が出ているわけではなく，計算時間，記憶容量の関係でシミュレーションがより困難であるため，手を触れなかったからである（図1参照）．

2. 諸定義およびルーティングモデル

n 次元超立方体ネットワークを $G_n=(P, E)$ と定義する．P はプロセッサ集合，E はプロセッサをつなぐリングの集合である．整数集合 $M_n=\{i \mid 0 \leq i \leq 2^n-1\}$ とする．P の各プロセッサは，$i \in M_n$ なる相異なる識別子を持つ．プロセッサ i のアドレス $ad(i)$ は i の2進表現である．プロセッサ $i, j \in P$ において，$ad(i)$, $ad(j)$ のハミング距離が1のときに限り，i, j 間にリンクがある．その異なるビットを第 k ビット（$1 \leq k \leq n$）とすると，このリンクをプロセッサ i およびプロセッサ j の k ビット目に対応するリンクと呼ぶ．各プロセッサは，同期して動作する．同期の単位時間をステップと呼ぶ．各プロセッサは同じプログラムを実行する．

本論文で対象とするルーティングモデルでは，プロセッサ間でパケットが送受信される際に，あるステップであるリンクに送信したパケットは，次のステップでそのリンクの先のプロセッサで読むことができ，各プロセッサは，そのプロセッサに接続する各リンクに対して1ステップの間に高々1つのパケットを送信できる．各リンクにはそのリンクを通じて送信するパケットを保持するための待ち行列が存在する．パケットルーティング問題を $Pr=(G, \pi)$ で定義する．G はネットワーク，π は M_n の全要素からなる置換である．本論文では G として G_n を考える．Pr の初期状態，最終状態は次のようになる．

初期状態：各プロセッサ $i \in P$ に，G_n 上のプロセッサを目的地とするパケット v_i が1つ与えられる．パケットは〈出発地，目的地〉の2項組で表される．v_i の目的地となるプロセッサを $\pi(i)$，そのア

ドレスを $ad(\pi(i))$ とする．問題の仮定としてパケットには〈出発地，目的地〉のアドレス以外の付加的なルーティング情報は許さないことにする．

最終状態：各パケット v_i が目的地 $\pi(i)$ に送られた状態．

Gの各プロセッサがプログラム（アルゴリズムA）を実行し初期状態から最終状態に至れば，A が Pr を解くという．本論文では，Pr についてシミュレーションを行い，すべてのパケットが目的地に到達するまでのステップ数を計測する．

3. アルゴリズム

各プロセッサは1ステップに，前のステップで送られてきたすべてのパケット（高々 n 個）について，それらが目的地に到着したかどうか判定する．目的地に着いていない各パケットについては，次に通るべきリンクを決定し，そのリンクの待ち行列につなぐ．これをルーティングと呼ぶ．ステップ t でリンク Lの待ち行列に2個以上のパケットがつながれているとき，t において L は busy であると呼ぶ．今プロセッサ i を出発地とするパケット v_i の目的地が $ad(\pi(i))$ であるとする．パケット v_i があるステップでプロセッサ j にいるとする．j のアドレスを $ad(j)$ とする．ここではルーティング方法として次の2種類を考える．

left-to-right（以下 LR 法）：$ad(j)$ と $ad(\pi(i))$ を左から右へビットスキャンし，最初に異なるビットに対応するリンクを選択．

例えば，G_4 においてアドレス (1011) から出発したパケットが (0000) を目的地とするとき，経路は，(0011)，(0001)，(0000) となりこの順にプロセッサを移動する．

greedy（以下 GR 法）：$ad(j)$ と $ad(\pi(i))$ を左から右へビットスキャンし，異なるビットでリンクの待ち行列長0のものを探し，そのリンクを選択する．なければ，異なるビットの中からリンクの待ち行列長最短のものを選択．

例えば G_4 でプロセッサ2（アドレスは (0010)）があるステップで到着した3つのパケット v_0, v_1, v_2（それぞれアドレス (1011)，(1001)，(1110) のプロセッサを目的地とすると仮定する）をこの順に処理する場合を考える．プロセッサ2のリンクの待ち行列長は左から 0, 2, 0, 0 であるとする．このとき，LR 法ならどのパケットも左から1ビット目のリンクを選択する．GR 法では，v_0 は左から1ビット目のリンクを選択する．v_1 は1ビット目のリンクの待ち行列長が1になったので3ビット目を選択する．この時点で待ち行列長は，左から 1, 2, 1, 0 となっている．v_2 の異なるビットは1，2ビット目であるが，いずれも待ち行列長0でない．そこで待ち行列長を比較し短い方である1ビット目のリンクを選択する．

これらの二種の方法はともに，1つのパケットを処理する際にアドレス比較のために $O(n)$ 時間かかるので，あるステップで k 個のパケットが到着した場合に，そのステップの処理に $O(kn)$ 時間かかる．

次に各プロセッサは各リンクの待ち行列からパケットを1個ずつ先着順 FIFO (First-In-First-Out) に選択し，そのリンクを通して隣接するプロセッサにパケットを送る．ステップ t において，プロセッサ i のあるリンクにつながったパケットがそのステップで送信されないとき，そのパケットはステップ t においてプロセッサ i で待ち状態にあると呼ぶ．

各パケットの経路がその〈出発地，目的地〉だけで唯一に決まるようなアルゴリズムをオブリビアス (oblivious) アルゴリズムと呼ぶ[3]．LR 法でパケットのルーティングを行う決定性アルゴリズムはオブリビアスアルゴリズムである．また文献 2)の greedy アルゴリズムもそうである．一方，本論文で提案する GR 法はそうではない．

オブリビアスアルゴリズムが問題を解くのに要するステップ数の下限値がいくつか求められている[2),3)]．ここではより厳しい下限値を証明している文献2)の結果を示す．

[命題1] 任意の次数 d，プロセッサ数 N の任意のネットワーク上で置換ルーティング問題を解く任意の決定性オブリビアスアルゴリズムに対して少なくとも $\sqrt{N}/2d$ ステップかかるような置換が必ず存在する．

証明の方針を簡単に示す．任意のオブリビアスアルゴリズムで，パケットの経路は唯一に決まることから，ある1つのリンクを $\sqrt{N}/d$ 個のパケットが通るような置換が存在することが言える．1つのリンクは1ステップに，双方向にパケットを通信したとしても，たかだか2個のパケットしか通せないので，それらがすべて通過するのに少なくとも $\sqrt{N}/2d$ ステップかかる．例えば，ネットワークが G_n ならば，次数は n，プロセッサ数は 2^n であるから，少なくとも $\sqrt{2^n}/2n$ ステップかかることになる．次に示す逆順置換はこのような置換の1つである．

逆順置換：各パケット v_i に対し $ad(i)=(a_1, a_2, \cdots,$

a_n) とすると $ad(\pi(i))=(a_n, a_{n-1}, \cdots, a_1)$ である.

例えば G_{10} では，アドレス ($a_1, \cdots a_5$ 00000) のプロセッサから出発する 2^5 個のパケットの目的地はすべて5つの0から始まる．結果として 2^5 個のパケットすべてがアドレス (0000000000) のプロセッサを通ろうとする.

そこで，各プロセッサがランダムに中間目的地を選ぶことによりパケットを散らし，リンクが busy になるのをなるべく抑えようというのが，次に示す Valiant-Brebner の確率アルゴリズムである.

Valiant-Brebner の確率アルゴリズム

(以下 VB 法)[4]

フェイズ1：各プロセッサ i はランダムに中間目的地プロセッサ $s(i)$ を選び，LR 法で v_i をそこへ送る．$s(0), s(1), \cdots, s(2^n-1)$ は置換になっていなくてよい.

フェイズ2：$s(i)$ から $\pi(i)$ へ LR 法で v_i を送る.

VB 法に対して次の命題が成り立つ[4].

[**命題2**] 問題 $Pr=(G_n, \pi)$ において，VB 法は $1-1/2^n$ 以上の確率で $14n$ ステップ以下で最終状態に到達する.

4. シミュレーション

シミュレーションにより性能比較を行う場合，二つの問題点がある．まず，置換の場合の数が $2^n!$ 通りと非常に多いことである．次に，厳密に確率アルゴリズムの性能を把握するためには，非常に多くの乱数で実験する必要があることである．各パケットの中間目的地を選ぶときに，n ビットの乱数を発生し，パケット数は全部で 2^n であるから，全部で $n \cdot 2^n$ ビットの乱数を発生する必要がある．このとき試さねばならない乱数の場合の数は $2^{n \cdot 2^n}$ である．よってここでは，すべての置換，すべての乱数でシミュレーションを行うことはしない.

4.1 逆順置換，ビット反転置換に対するシミュレーション

逆順置換とビット反転置換という特徴的な置換に注目する．ビット反転置換とは，次の置換である．ビット反転置換はすべてのパケットが n 回プロセッサ間を移動しなければならないという置換である.

ビット反転置換：各パケット v_i に対して $ad(i)=(a_1, a_2, \cdots, a_n)$ とすると，$ad(\pi(i))=(\bar{a}_1, \bar{a}_2, \cdots, \bar{a}_n)$ である．ただし，$a_i=0$ のとき $\bar{a}_i=1$，$a_i=1$ のとき $\bar{a}_i=0$ である.

[**実験1**] $Pr=(G_n, \pi)$，$1 \leq n \leq 15$，π は逆順置換とビット反転置換，に対して，LR 法，GR 法，VB 法の3つのアルゴリズムのステップ数を比較する.

逆順置換に関して図1のような結果を得た．LR 法は $n<10$ 次元では VB 法に勝る．VB 法は n に線形な変化を示すが，LR 法は 10 次元あたりから，リンクが busy な状態が多発しステップ数が飛躍的に増大する．また本論文で提案した GR 法によって，大幅なステップ数の減少が見られた.

ビット反転置換に関して**図2**のような結果を得た．ビット反転置換はすべてのパケットが n 回プロセッサ間を移動しなければならないという置換であるが，LR 法により各パケットがうまく巡回してリンクが busy にならない．よって，LR 法でも n ステップで問題を解き終える．確率アルゴリズムはこのような置換に関しては，寄り道をするだけ不利である.

[**実験2**] $Pr=(G_n, \pi)$，$1 \leq n \leq 15$，π は逆順置換，に対して VB 法を乱数を変えて 300 回実行する.

実験2の結果，乱数によるステップ数の揺らぎは各次元でそれぞれ3ステップあり，その揺らぎによって決定性アルゴリズムとの優劣が変化することはなさそうである．以下のシミュレーションにおいても，VB 法の結果はランダムに選んだ1つの乱数による実行結果で代表することにする.

4.2 様々な置換に対するシミュレーション

ここでは，G_{12} に限定して，逆順，ビット反転置換以外の様々な置換に対するシミュレーションを行う．対象とするアルゴリズムは LR 法，GR 法，VB 法とする．置換をどのように選ぶかということが問題であるが置換を辞書式順序に番号付けし，その番号から置

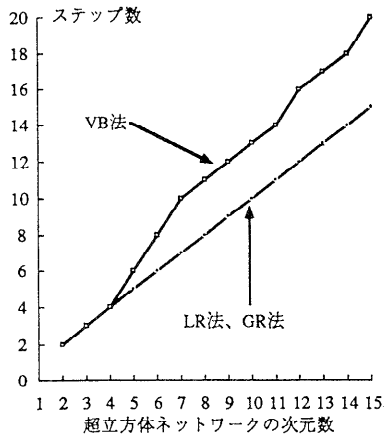


図2 ビット反転置換に対する結果（ステップ数）
Fig. 2 The results in bit-reverse permutation.

換を生成するアルゴリズムが知られている[5].

π_n は，整数集合 M_n のすべての要素からなる置換であるとする．また Π_n を起こり得るすべての π_n の集合とする．Π_n の要素数は $|\Pi_n|=2^n!$ である．このとき辞書式順に並べられた Π_n の各要素に順に1から $2^n!$ までの番号を付ける．$\pi_n \in \Pi_n$ に $j \in M_n$ が対応するとき，$rankp(\pi_n)=j, rankp^{-1}(j)=\pi_n$ とする．またこのとき，置換 π_n の置換番号が j であるという．Π_n の各要素を辞書式順に並べた置換の系列を P_n とする．文献5)では $rankp$, $rankp^{-1}$ のアルゴリズムが述べられている．

まず始めに，置換番号的に接近しているいくつかの置換に対するシミュレーションを行い，どのような傾向があるのかを調べてみる．次の2つの実験を行う．

[実験3] $Pr=(G_{12}, \pi_{12})$ に対して LR 法，GR 法，VB 法のステップ数を計測する．ここに π_{12} とは置換番号的に逆順置換に近い300個の置換のうちの任意の1個である．

[実験4] $Pr=(G_{12}, \pi_{12})$ に対して LR 法，GR 法，VB 法のステップ数を計測する．ここに π_{12} とは系列 P_{12} の4等分割点の置換に近い300個ずつの置換のうちの任意の1個である．

実験3の結果，決定性アルゴリズムでは逆順置換の周辺600個すべてで逆順置換に対するシミュレーションと同じステップ数を得た．

LR 法のステップ数＝38
GR 法のステップ数＝12
VB 法のステップ数＝20〜23

実験4の結果，やはり決定性アルゴリズムでは一定のステップ数を得た．

LR 法のステップ数＝12
GR 法のステップ数＝12
VB 法のステップ数＝15〜20

以上より，$Pr=(G_{12}, \pi)$ では，置換番号的に近い置換に対するシミュレーションは同じ傾向を示すことが推測できる．次に，置換系列 P_n をさらに細かく1200分割し，分割した系列から1つずつ置換を選びシミュレーションを行う（実験5）．

[実験5] $Pr=(G_{12}, \pi_{12})$ に対して LR 法，GR 法，VB 法のステップ数を計測する．ここに π_{12} とは置換集合

$$\left\{ rankp^{-1}(i) \mid i=\frac{|\Pi_n|}{1200}\cdot s,\ \text{ただし}\ s=1, 2, \cdots, 1200 \right\}$$

の任意の置換である．

実験5では次のような結果を得た．

LR 法のステップ数 T_o＝8〜12
GR 法のステップ数 T_g＝8〜12
VB 法のステップ数 T_r＝19〜25

決定性のアルゴリズムに関しては常に $T_g \leq T_o$ となるが $T_g < T_o$ となる置換の数は51個あった．実験3，4，5より，決定性アルゴリズムに不利な置換は非常にまれであるという感触を得た．

4.3 LR 法と GR 法の比較

パケットの動きに着目した比較

LR 法と GR 法の動作を比較，分析するために，アルゴリズム実行途中のパケットの動きに着目する．ステップ t のルーティング処理を受けるパケット数を N_t とする．ステップ t でプロセッサ間を移動したパケット数を M_t とする．ステップ t で各プロセッサでルーティング処理するパケット数の最大値を I_t，各プロセッサで送り出したパケット数の最大値を O_t，各プロセッサで待ち状態にあるパケット数の最大値を Q_t とする．

[実験6] $Pr=(G_{15}, \pi)$，π は逆順置換，に対して，LR 法，GR 法の N_t，M_t，I_t，O_t，Q_t の値を計測する．

N_t，M_t 値に関する結果を**図3**に示す．I_t，O_t，Q_t 値に関する結果については文献6)の表2，3を参照のこと．図3より，GR 法においては，N_t 値と M_t 値の差がほとんど0であり，パケットが待ち状態にならずに非常に有効に動いていることがわかる．GR 法では，待ち状態にあるパケットがほとんど存在しない．これに対して LR 法ではステップ17から33にかけ

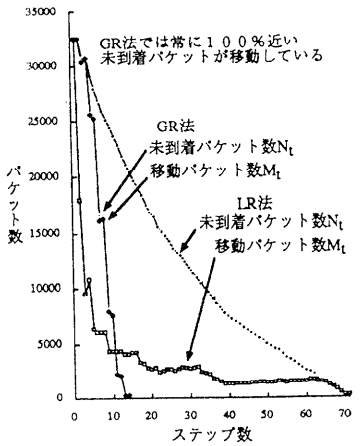


図3 LR 法，GR 法の N_t，M_t 値
Fig. 3 N_t, M_t values of LR, GR method.

て，31個のパケットが待ち状態になりボトルネックとなるプロセッサが256個あることがシミュレーションからわかった．

ステップ t 終了時に各プロセッサで待ち状態にあるパケット数の分布を D_t とする．

$$D_t=\{(p,q)\mid p,q \text{ は整数}, 1\leq p\leq Q_t, 1\leq q\leq 2^n\}$$

ここに任意の $(p,q)\in D_t$ は，ステップ t 終了時に p 個のパケットが待ち状態にあるようなプロセッサは全部で q 個存在することを意味する．

[実験7] $Pr=(G_{15},\pi)$，π は逆順置換，に対して，LR 法，GR 法の各ステップ t におけるパケット数の分布 D_t を計測する．

結果は文献6)の表4，5を参照のこと．なお D_t の要素のうち $q=0$ であるものは省略する．

実験7より，LR 法では一部の特別なプロセッサでパケットが待ち状態にあることがわかる．例えばステップ 10 においては，6個のパケットが待ち状態にあるプロセッサが1024個，15個のパケットが待ち状態にあるようなプロセッサが512個，24個のパケットが待ち状態にあるものが256個ある．文献6)の表5ではこのボトルネック的な状態が徐々に解消されていく様子がわかる．ステップ2から 63 までの各ステップでもっとも多くのパケットが待ち状態にある256個のプロセッサはそれらのステップを通して同じプロセッサである．またそれらのプロセッサで待ち状態にあるパケットはすべて各プロセッサで7ビット目に対応するリンクにのみつながっている．

内部処理に着目した比較

逆順置換に対して，GR 法はステップ数を減らすという点で非常に有効であることがわかった．本論文ではあくまでプロセッサ内部の処理時間単位に対して，プロセッサ間の通信時間すなわち，パケットがプロセッサ間を移動するのに要する時間が非常に大きいという前提を取る．従ってステップ数での比較を第一とする．しかしここではこの仮定をいったん取り去り，LR 法，GR 法が問題を解き終えるまでに，どれだけの内部処理を行っているかを比較する．1つの内部処理は各アルゴリズムの比較，代入などの基本操作とする．

1つの内部処理を行うのに要する時間を1内部処理時間と呼び，それは一定であるとする．各プロセッサ i にステップ t において到達したパケットの集合を V_i^t とする．各プロセッサ i のステップ t におけるルーティング処理時間を rs_i^t とする．$v_j\in V_i^t$ がステップ t において受ける内部処理の和を ms_j とする．このとき

$$rs_i^t=\sum_{v_j\in V_i^t} ms_j$$

である．

アルゴリズムがステップ t に要したルーティング処理時間を rs^t とする．アルゴリズムが問題を解き終えるまでにルーティング処理に要した時間を rs とする．アルゴリズムが問題を解き終えるまでに T ステップかかったとする．このとき

$$rs^t=\max_i\{rs_i^t\} \quad \text{また} \quad rs=\sum_{t=1}^{T} rs^t$$

である．

[実験8] $Pr=(G_{15},\pi)$，π は逆順置換，に対して，LR 法，GR 法の rs，各 rs^t の値を計測する．

シミュレーションの結果 rs 値は次のようになった．

LR 法の内部処理時間の総和 $rs=4662$

GR 法の内部処理時間の総和 $rs=2872$

rs^t 値に関する結果は文献6)の表2，3を参照のこと．このように内部処理時間の総和 rs で比較しても LR 法は GR 法の約1.6倍かかっている．また各ステップ t の rs^t で比較しても各ステップで GR 法は LR 法のたかだか3倍であるため，ステップ数を抑えたことが全体に反映している．

本論文では，パケットに出発地および目的地アドレス以外の付加的なルーティング情報を許さないという立場を取った．付加的情報を許すならば，ただ一度出発地と目的地アドレスのビットスキャンを行うことによって，出発地から目的地までの経路を決め，その情報をパケットに保持することで，各ステップの内部処理は抑えることができる．しかし，その場合でもステップ数は変わらない．

5. おわりに

- 逆順置換に対しては，命題1，2が示すとおり，LR 法は $n>10$ で VB 法に劣ることがわかった．しかしここで提案する GR 法では，大幅なステップ数の減少が見られた．
- 1ステップで行っている内部処理にまで着目して LR 法と GR 法の二つの方法の比較を行ったところ，GR 法の方が1ステップの内部処理は増えているものの内部処理の和は少なく，1ステップの内部処理が増えてもステップ数を減らすことが有効であることがわかった．

- 逆順置換以外の様々な置換で G_{12} でシミュレーションを行った．その結果，置換番号的に接近している置換に対するシミュレーションのステップ数は非常に似た傾向を示すことがわかった．そこで2分探索的にとびとびに置換を選びシミュレーションを行ったところ，逆順置換のように決定性オブリビアスアルゴリズムにとって不利な置換は見つけられなかった．
- 様々な置換に対してシミュレーションを行ったがVB法には，“有利，不利な”置換というものは存在しないということが実測値で確認できた．

以上より，$1 \leq n \leq 15$ の Pr では，逆順置換のような一部の，決定性オブリビアスアルゴリズムに不利な置換のために確率アルゴリズムを導入せずとも，内部処理を多少増やしてでもステップ数を抑える，ここで提案する GR 法が優位であるという見通しを得た．

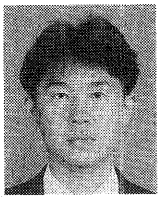
$n>15$ ではシミュレーションの結果（ステップ数）はどのようなものになるのか，また $n \leq 15$ でも逆順置換のようにオブリビアスアルゴリズムに不利な置換はどれほど存在するのかといった疑問が残っている．しかし，これらのことをシミュレーションで確かめようとする膨大な計算時間であるいは記憶領域が必要となる．本研究グループでは，このために並列計算機を使用して，現在実験中である．

謝辞 日頃から細部に渡って御協力いただく，首藤研究室の皆様に感謝いたします．本研究は，平成3年度文部省科学研究費補助金一般研究(C) (03680030)，重点領域研究 (1) (03235102)，奨励研究 (A) (03780256)，総合研究(A) (02302047)，および平成4年度重点領域研究「超並列記述系・処理系に関する研究」(04235104) の補助を受けている．

参 考 文 献

1) Raghavan, P.: Lecture Notes on Randomized Algorithms, IBM Research Report RC 15240 (1990).
2) Leighton, F. T.: *Introduction to Parallel Algorithms and Architertures: arrays・trees・hypercubes*, Morgan Kaufmann (1992).
3) Borodin, A. and Hopcroft, J. E.: Routing, Merging, and Sorting on Parallel Models of Computation, *Journal of Computer and System Sciences*, Vol. 30, pp. 130-145 (1985).
4) Valiant, L. G. and Brebner, G. J.: Universal Schemes for Parallel Computation, *Proceedings of the Thirteenth Annual ACM Symposium on Theory of Computing*, pp. 263-277, Milwaukee, Wisconsin (1981).
5) Akl, S. G.: *The Design and Analysis of Parallel Algorithms*, Prentice Hall (1989).
6) 石原，萩原，魚井，首藤：並列アルゴリズムに適用した確率アルゴリズムの性能評価の試み—超立方体パケット交換ネットワークのルーティング問題の場合—，並列処理シンポジウム JSPP '92, pp. 93-100 (1992).

(平成4年9月14日受付)
(平成5年1月18日採録)



石原 鑑（正会員）

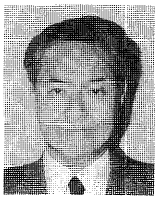
昭和 42 年生．平成3年大阪大学基礎工学部情報工学科卒業．平成5年同大学院基礎工学研究科修士課程修了．現在三菱電機(株)勤務．確率アルゴリズム，分散アルゴリズム，並列アルゴリズムに関する研究に興味を持つ．

萩原 兼一（正会員）

昭和 27 年生．昭和 54 年大阪大学大学院基礎工学研究科博士課程修了．工学博士．同年阪大基礎工助手．同助教授を経て，現在奈良先端科学技術大学院大学情報科学研究科教授．平成4年から5年にかけメリーランド大学にて文部省在外研究員．並列アルゴリズム，分散アルゴリズム，並列プログラム作成支援環境などの研究に興味を持つ．本学会元編集委員(地方)．電子情報通信学会コンピューテーション研究会元幹事．日本ソフトウェア科学会編集委員．

魚井 宏高（正会員）

昭和 37 年生．昭和 59 年大阪大学基礎工学部情報工学科卒業．昭和61年同大学院基礎工学研究科修士課程修了．昭和 63 年より同大同学部同学科助手．オブジェクト指向言語，ユーザインタフェースなどの研究に従事．電子情報通信学会，日本ソフトウェア科学会各会員．



首藤 勝（正会員）

昭和 9 年生．昭和 32 年大阪大学工学部通信工学科卒業．同年三菱電機(株)に入社，主に研究部門でソフトウェアの研究開発を担当．制御用言語の開発と国際標準化活動，通産省スーパーコンピュータおよび第5世代コンピュータの両プロジェクトなどに参画した．平成元年より大阪大学基礎工学部情報工学科教授（ソフトウェア構成学講座担当）．工学博士．昭和 53-54 年および 56-57 年本会理事．ソフトウェア科学会，電子情報通信学会各会員．