辞書式順極大3和問題に対するBSPモデル上のコスト最適な並列アルゴリズム

中島孝明† 藤原暁宏†

本論文では，並列化困難とされている P 完全問題の1つである辞書式順極大3和問題に対して，BSPモデル上でコスト最適な並列アルゴリズムを2つ提案する．入力サイズが n の場合，1つ目のアルゴリズムは，内部計算時間が $O(\frac{n^2}{p}+n\log n+np)$ 時間，通信計算量が $O(n(gp+\frac{L}{p}))$ であり，2つ目のアルゴリズムは，内部計算時間が $O(\frac{n^2}{p}+n\log n+nbp)$ 時間，通信計算量が $O(n(gbp+\frac{L}{bp}))$ である．ここで，p はプロセッサ数であり，g および L はBSPモデルで通信コストを表すパラメータである．これらのアルゴリズムは，それぞれ $p\leq\sqrt{\frac{n}{g}}$，$p\leq\sqrt{\frac{n}{gb}}$ の範囲において，漸近的にコスト最適なアルゴリズムとなる．また，これらのアルゴリズムをクラスタ並列処理環境に実装し，アルゴリズムが実用的なスピードアップを達成することを示す．

Cost Optimal Parallel Algorithms for the Lexicographically First Maximal 3 Sum Problem on the BSP Model

Takaaki Nakashima† and Akihiro Fujiwara†

The lexicographically first maximal 3 sum (LFM3S) problem is known as one of P-complete problems, which are hard to be parallelized. In this paper, we present two parallel algorithms for computing LFM3S on the BSP model. The first algorithm runs in $O(\frac{n^2}{p}+n\log n+np)$ computation time and $O(n(gp+\frac{L}{p}))$ communication time, where n is the input size, p is the number of processors, and g, L are parameters which represent communication cost on the BSP model. The second algorithm runs in $O(\frac{n^2}{p}+n\log n+nbp)$ computation time and $O(n(gbp+\frac{L}{bp}))$ communication time. The two algorithms are cost optimal for $p\leq\sqrt{\frac{n}{g}}$ and $p\leq\sqrt{\frac{n}{gb}}$, respectively. In addition, we implement these algorithms on a PC cluster, and show that these algorithms achieve practical speedup.

1. はじめに

並列計算量理論における基本的な計算量のクラスとしてクラス NC がある．クラス NC に属する問題とは，n を問題の入力のサイズとした場合，n の多項式個のプロセッサを用いて，n の対数多項式時間でその問題を解くアルゴリズムが存在する問題である．逐次的に多項式時間で解くことのできる問題のクラスをクラス P というが，このクラス P に含まれる問題は，その多くがクラス NC にも含まれていることが分かっている．しかし，クラス P に含まれる問題のいくつかは，多項式個のプロセッサを用いて対数多項式時間で解くアルゴリズムを持たないと考えられている．このような問題は一般に P 完全問題と呼ばれており，この P 完全問題に対しては，いくつかの効率的な確率的並列アルゴリズムは存在しているが，本質的に逐次性が高く並列化困難であると考えられている．

一方，実際の並列処理においてはプロセッサ数 p は問題のサイズ n に対して非常に小さいので，対数多項式時間の時間計算量というのはそれほど重要ではない．つまり，実用的な並列アルゴリズムに対しては，コスト最適性が最も重要な尺度となる．並列アルゴリズムのコストは，そのアルゴリズムの実行時間とプロセッサ数との積として定義される．このコストが同じ問題に対する最速の逐次アルゴリズムの計算量と漸近的に等しい場合，その並列アルゴリズムはコスト最適であると呼ぶ．言い換えると，コスト最適な並列アルゴリズムとは，最適なスピードアップを達成しているアルゴリズムである．

そこで，P 完全問題の並列化の方針の1つとして，多項式時間で実行できるコスト最適なアルゴリズムを

† 九州工業大学情報工学部
Faculty of Computer Science and Systems Engineering, Kyushu Institute of Technology

見つけることがあげられる．$\Omega(n^k)$ を P 完全問題 A に対する逐次アルゴリズムの下界とする．問題 A は P 完全問題であるので，問題 A には対数多項式時間の並列アルゴリズムは存在しないかもしれない．しかしながら，問題 A には n^ϵ $(0 < \epsilon < k)$ 個のプロセッサを用いて $O(n^{k-\epsilon})$ 時間で動作する並列アルゴリズムが存在する可能性は残されている．この並列アルゴリズムはコスト最適であるので，実際の並列処理において，適当な個数のプロセッサを用いて最適なスピードアップ☆を達成すると考えられる．したがって，本論文では P 完全問題に対して，多項式時間で動作するコスト最適な並列アルゴリズムを提案することを目的とする．

この P 完全問題に対する並列アルゴリズムに関しては，これまでにいくつかの研究が行われている．代表的な P 完全問題である最大流問題については，理論的な並列計算モデルである PRAM モデル上で動作する $O(\frac{n^3 \log n}{p})$ 時間，p プロセッサ $(1 \leq p \leq n)$ の並列アルゴリズム[6]が提案され，実際の分散メモリ型並列コンピュータへの実装および実験[7]も行われている．しかしながら，この最大流問題に対しては $O(n^3)$ 時間の逐次アルゴリズムが知られているので，このアルゴリズムはコスト最適な並列アルゴリズムではなく，P 完全問題の並列化の目的を達成しているとはいえない．また，実験結果[7]においても，16 個のプロセッサを使用した場合に 2 倍程度のスピードアップしか達成しておらず，実用的な並列アルゴリズムであるとはいいがたい．

本論文では P 完全問題として，比較的単純な問題である辞書式順極大 3 和問題を取り上げる．辞書式順極大 3 和問題とは，整数集合 I の中から $a_i + b_i + c_i = 0$ を満たすすべての 3 項組の集合 $\{(a_0, b_0, c_0), (a_1, b_1, c_1), \cdots, (a_{m-1}, b_{m-1}, c_{m-1})\}$ を辞書式順に求める問題である．この辞書式順極大 3 和問題は，同じく P 完全問題として知られる辞書式順極大独立点集合問題からの帰着によってその P 完全性が示され[2]，また PRAM 上で，p $(1 \leq p \leq n)$ 個のプロセッサを用いて $O(\frac{n^2}{p} + n \log n)$ 時間で動作するコスト最適なアルゴリズム[2]も提案されている．

本論文では，この辞書式順極大 3 和問題に対して，Valiant によって提案され，実用的な並列計算モデルとして近年注目を集めている BSP モデル[8]上でコスト最適な並列アルゴリズムを 2 つ提案する．1 つ目のアルゴリズムの計算量は，内部計算時間が $O(\frac{n^2}{p} + n \log n + np)$ 時間，通信計算量が $O(n(gp + \frac{L}{p}))$ である（ここで，n は問題の入力サイズ，p はプロセッサ数，g，L は BSP モデル上で通信コストを表すパラメータである）．辞書式順極大 3 和問題に対する逐次アルゴリズムの時間計算量が $O(n^2)$ であることから，このアルゴリズムは $p \leq \sqrt{\frac{n}{g}}$ の範囲において，コスト最適なアルゴリズムとなっている．2 つ目のアルゴリズムは，1 つ目のアルゴリズムに対して，通信回数の軽減を目的としたアルゴリズムである．このアルゴリズムでは，1 つ目のアルゴリズムにおいて，分割して行われている b 個の処理および通信を，1 つのブロックとしてまとめて処理することにより，通信回数の削減を行っている．これにより，このアルゴリズムの計算量は，内部計算時間が $O(\frac{n^2}{p} + n \log n + nbp)$ 時間，通信計算量が $O(n(gbp + \frac{L}{bp}))$ であり，$p \leq \sqrt{\frac{n}{gb}}$ の範囲において，コスト最適なアルゴリズムとなる．

また本論文では，上記のアルゴリズムを，PVM[4]を用いた PC クラスタ上に実装し，アルゴリズムの実用的なスピードアップを検証する．実験結果より 1 つ目および 2 つ目のアルゴリズムは，ともに 8 台の PC を用いて 4 倍程度スピードアップを達成し，実用的な高速化が達成できることを示した．ただし，2 つ目のアルゴリズムにおける通信回数の削減は，内部計算量，および通信オーバヘッドの増加により，予想されたスピードアップの向上にはつながらなかった．

本論文は，以下のように構成される．まず，2 章では，P 完全性，BSP モデル，および，辞書式順極大 3 和問題の説明を簡単に行う．次に，3 章では，BSP モデルを用いて，辞書式順極大 3 和問題を解くコスト最適な並列アルゴリズムを示し，その計算量の検証を行う．4 章では，実験結果とその考察を行い，最後に 5 章でまとめを行う．

2. 準　　備

2.1 P 完全性の定義

本節では，本論文中で使用する P 完全の定義について簡単に説明する（詳細な定義については文献 5) 等を参照のこと）．問題の入力サイズ n の多項式時間の決定性逐次アルゴリズムが存在する問題を，クラス P に属する問題と呼ぶ．このクラス P は，効率の良い逐次アルゴリズムが存在するクラスとして知られている．同様に，効率の良い並列アルゴリズムが存在するクラスとして知られているのが，クラス NC である．クラス NC に属する問題とは，入力サイズの多項式個のプロセッサを使用した対数多項式時間の並列

☆ スピードアップとは，その問題を解く最速の逐次アルゴリズムの時間計算量を T_s，並列アルゴリズムの時間計算量を T_p としたとき，$S = \frac{T_s}{T_p}$ で表される値である．

アルゴリズムが存在する問題である．これらのクラスを用いて，P 完全性は次のように定義される．

定義 1（P 完全問題） 問題 Q が次の 2 つの条件を満たしている場合，問題 Q は P 完全であるという．

(1) $Q \in P$.

(2) すべての $S \in P$ に対して，S が Q に NC 帰着可能である． □

P 完全問題については，問題のサイズの多項式個のプロセッサを使用した対数多項式時間のアルゴリズムを得ることは困難であると考えられているので，本論文では多項式時間のコスト最適な並列アルゴリズムの提案を目指す．

2.2 BSP モデル

本論文では，実行時間を見積もるための並列計算モデルとして，近年注目されている Bulk Synchronous Parallel（BSP）モデル[8] を使用する．BSP モデルは Valiant によって提案された粗同期式並列計算モデルであり，次の 3 つの要素から構成されている．

- 局所メモリを持つ複数のプロセッサ
- プロセッサ間の 1 対 1 メッセージ通信を行う完全結合網
- プロセッサ間の同期を実現するための同期機構

この BSP モデルの特徴の 1 つにスーパーステップという概念の導入があげられる．スーパーステップとは BSP モデルにおける処理単位の 1 つであり，BSP モデル上での処理の実行は，スーパーステップの列としてとらえられる．スーパーステップの概念図を**図 1** に示す．

各スーパーステップでは，各プロセッサ内で，内部計算フェーズ，および通信フェーズが実行される．内部計算フェーズでは，他のプロセッサとの通信はいっさい行わず，各プロセッサ上のデータのみを使用した内部計算が行われる．一方通信フェーズでは，受信命令と送信命令以外の命令は実行することができないと仮定されている．この仮定により，各スーパーステップで受信したデータは，次のスーパーステップまで利用できないことになる．すべてのプロセッサがこれら 2 つのフェーズを終了すると，同期機構により自動的にバリア同期がとられ，スーパーステップが終了する．

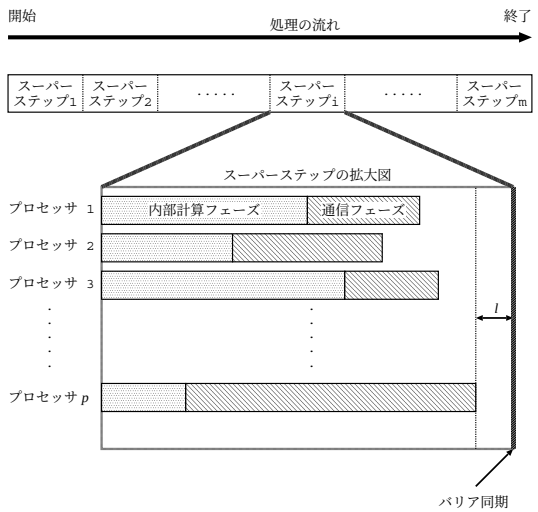

図 1 スーパーステップ
Fig. 1 Superstep.

BSP モデルでは，具体的なネットワーク構造やメッセージ配送の仕組みといったデータ通信のための各種要素を，以下の 2 つのパラメータを用いて抽象化している．

L： バリア同期の実行に要する時間

g： 1 語のデータ送信，もしくは受信に要する時間

つまり BSP モデルでは，L によって通信遅延を表し，g で送受信のオーバヘッドを表すことで，データ通信のコストを考慮している．ただし，g，L は要素数より非常に小さい $(g, L \ll n)$ と仮定する．

以上より，1 つのスーパーステップ内で各プロセッサが実行する内部計算フェーズのうち，最も長い内部計算フェーズの計算量を w，また通信フェーズの送信メッセージ数と受信メッセージ数の和の中で最大のものを h としたとき，そのスーパーステップの計算量は $O(w + gh + L)$ となる．この計算量をすべてのスーパーステップについて計算し，その和を求めると BSP モデルにおける処理全体の計算量となる．

2.3 辞書式順極大 3 和問題の定義

辞書式順極大 3 和問題の定義を行う前に，まず，極大 3 和問題の定義を行う．

定義 2（極大 3 和問題） I を n 要素の整数の集合とする．極大 3 和問題とは，以下の 3 つの条件を満たす 3 項組の集合 $M3S = \{(a_0, b_0, c_0), (a_1, b_1, c_1), \cdots, (a_{m-1}, b_{m-1}, c_{m-1})\}$ を求める問題である．

(1) 集合 $S = \{a_0, b_0, c_0, a_1, b_1, c_1, \cdots, a_{m-1}, b_{m-1}, c_{m-1}\}$ は I の部分集合である．

(2) 各 3 項組 (a_i, b_i, c_i) $(0 \le i \le m-1)$ について，$a_i + b_i + c_i = 0$ が成り立つ．

(3) $a' + b' + c' = 0$ を満たす $a', b', c' \in I - S$ は存在しない． □

定義より，この極大 3 和問題の解は一意には決定されないことが分かる．

次に定義を行う辞書式順極大 3 和問題は，極大 3 和問題に制限を設け唯一の解を持つように変更した問題である．$A = (a_0, a_1, \cdots, a_{m-1})$ と $B =$

$(b_0, b_1, \cdots, b_{m-1})$ を m 個の要素を持つ 2 つの列とする．ここで，$a_0 < b_0$，もしくは添え字 i 以下のすべての要素が等しく，かつ $a_i < b_i$ が成り立つような i $(1 \leq i \leq m-1)$ が存在するとき，A は B よりも辞書式順に小さいという．また，A が集合に含まれる他のすべての列より小さい場合には，A は列集合の中で辞書式順に最小であるという．

定義 3（辞書式順極大 3 和問題） I を n 要素の整数の集合とする．辞書式順極大 3 和問題とは，以下の 2 つの条件を満たす 3 項組の集合 $LFM3S = \{(a_0, b_0, c_0), (a_1, b_1, c_1), \cdots, (a_{m-1}, b_{m-1}, c_{m-1})\}$ を求める問題である．

(1) 集合 $LFM3S$ は集合 I に関する極大 3 和問題の解である．

(2) $s_i = \{a_i, b_i, c_i\}$ $(0 \leq i \leq m-1)$ とするとき，(a_i, b_i, c_i) は，集合 $I - (s_0 \cup s_1 \cup \cdots \cup s_{i-1})$ において，$a_i + b_i + c_i = 0$ を満たす辞書式順に最小の 3 項組である． □

辞書式順極大 3 和問題は，文献 3) のアルゴリズムを用いて，逐次的に $O(n^2)$ 時間で解くことができる．また，部分問題である 3 和問題について，ある計算モデル上で $\Omega(n^2)$ 時間の下界が示されている[1)] ので，辞書式順極大 3 和問題についても同様の下界が成り立つ．また，文献 2) により，問題の P 完全性が証明されている．

本論文では，簡単のため辞書式順極大 3 和問題の入力要素はすべて異なるものと仮定する．このような入力に対しても，文献 2) により問題の P 完全性は保持される．また，本論文では，すべてのプロセッサが同じ入力全体を保持するものと仮定する．

3. 辞書式極大 3 和問題を解くアルゴリズム

3.1 アルゴリズム 1

最初に，文献 3) で提案されている $O(n^2)$ 時間の逐次アルゴリズムを簡単に紹介する．

(1) n 個の要素からなる入力集合 I を昇順にソートする．ソート後の列を $S = \{s_0, s_1, \cdots, s_{n-1}\}$ とする．

(2) S の各要素 s_i $(0 \leq i \leq n-2)$ について，$lower = i+1, upper = n$ とし，$lower < upper$ の間以下のサブステップを繰り返す．

(2-1) $3sum = s_i + s_{lower} + s_{upper}$ を計算する．

(2-2) $3sum$ の値に応じて，以下の処理を行う．

$3sum < 0$ の場合： $lower = lower + 1$ とする．

$3sum > 0$ の場合： $upper = upper - 1$ とする．

$3sum = 0$ の場合： $(s_i, s_{lower}, s_{upper})$ を解として出力し，$s_i, s_{lower}, s_{upper}$ を S より削除する．また，$lower = upper$ とし，サブステップを終了する．

本アルゴリズムは，基本的には，この逐次アルゴリズムを元に並列化したアルゴリズムであり，各プロセッサが，それぞれが受け持った s_i に対して 3 和を同時に計算することで，並列処理を実現している．しかしながら，このアルゴリズムに基づいた単純な並列化には以下のような問題点がある．逐次的に解く場合には，ある s_i に対して 3 和が 0 になる最初の 3 項組を 1 つだけ求めればよかった．しかし，並列的に解く場合，あるプロセッサが求めている解が同時に求めている他の解の影響を受け，そのプロセッサが求めた最小の 3 項組は問題の解とはならない場合が考えられる．ここで，次のような例を考えてみる．

例 1 $S = \{-10, -9, -2, 0, 2, 9, 11, 12\}$ の辞書式順極大 3 和問題の解を，1 台，および 2 台のプロセッサを用いて求める．

1 台のプロセッサで解く場合 逐次アルゴリズムを実行すると，その解は $\{(-10, -2, 12), (-9, 0, 9)\}$ となる．

2 台のプロセッサで解く場合 プロセッサ P_1 の a 要素として -10 を，プロセッサ P_2 の a 要素として -9 を割り当て，それぞれのプロセッサにおいて逐次アルゴリズムを実行する．この場合に各プロセッサが求める 3 項組は，それぞれ

$P_1 : (-10, -2, 12)$, $P_2 : (-9, -2, 11)$

となる．しかし，これらの 3 項組は -2 が重複しているため，辞書式順に大きい $(-9, -2, 11)$ は解から削除される．つまり 2 台のプロセッサで解いた場合には，$\{(-10, -2, 12)\}$ が解となる． □

この例において，$(-9, 0, 9)$ を求めることができなかった原因は，-9 に対する 3 項組を 1 つしか求めていなかったことにある．このように並列的に解く場合に，ある s_i に対して 3 和が 0 になる最初の 3 項組を 1 つしか求めていないと，すべての解を求めることができない可能性がある．したがって，並列的に解く場合には，3 和が 0 になるような複数の 3 項組の集合を辞書式順に求める必要がある．

そこで，まず辞書式順極大 3 和問題の解の候補となる 3 項組の集合を，各要素 s_i に対して以下のように定義する．

定義 4 $S = \{s_0, s_1, \cdots, s_{n-1}\}$ を昇順ソート列とする．このとき，S の各要素 s_i について，3 項組の集

合 $3S_i$ を以下のように定義する．

$3S_i = \{(s_i, s_j, s_k) | i < j < k, s_i + s_j + s_k = 0\}$

また，集合 $3S_i$ に含まれるすべての要素を辞書式順にソートした 3 項組の列を，$L3S_i$ として定義する．

$$\begin{aligned} &L3S_i \\ &= (sum_0(i), sum_1(i), \cdots, sum_{m-1}(i)) \\ &= ((s_i, s_{j_0}, s_{k_0}), (s_i, s_{j_1}, s_{k_1}), \\ &\quad \cdots, (s_i, s_{j_{m-1}}, s_{k_{m-1}})) \end{aligned}$$ □

$L3S_i$ の定義より，辞書式順極大 3 和問題の解は，集合 $L3S_0 \cup L3S_1 \cup \cdots \cup L3S_{n-1}$ の部分集合であることは明らかである．したがって，以下の手順により辞書式順極大 3 和問題を並列的に解くことができる．

(1) 入力の整数集合をソートする．

(2) 各プロセッサ P_i $(0 \le i \le p-1)$ に $L3S_i$ を割り当て，計算する．

(3) 各プロセッサの計算結果 $L3S_0, L3S_1, \cdots, L3S_{p-1}$ を 1 つのプロセッサに集め，辞書式順極大 3 和問題の解の条件を満たす 3 項組の集合を求める．

(4) (2) において，各プロセッサ P_i に $L3S_{p+i}$ を割り当て，以下同様に (2)，(3) を $\frac{n}{p}$ 回繰り返す．

しかしながら，上記のアルゴリズムでは，ステップ (2) における各 $L3S_i$ のサイズは最悪の場合 $O(n)$ であるので，ステップ (3) において計算結果の収集を行うプロセッサは，$O(np)$ のデータを受信する必要がある．ステップ (2)，(3) は $\frac{n}{p}$ 回繰り返されるので，全体では $O(n^2)$ 時間必要となり，処理全体のスピードアップは見込まれない．そこで，本アルゴリズムでは，以下の補題により，解の候補となる 3 項組の集合をサイズが $O(p)$ になるように限定する．これにより，収集を行うプロセッサは $O(p^2)$ のデータを受信すればよく，全体でも $O(np)$ 時間となる．

補題 1 $S = \{s_0, s_1, \cdots, s_{n-1}\}$ を昇順ソート列とする．また，S に対する $L3S_i$ の r 個の 3 項組による接頭部を $L3S_{i,r}$ とする．

$$L3S_{i,r} = (sum_0(i), sum_1(i), \cdots, sum_{r-1}(i))$$

このとき，S を入力とする辞書式順極大 3 和問題の解に，$sum = (s_i, s_j, s_k)$ が含まれているならば，$sum \in L3S_{i,2i-1}$ が成り立つ．

（証明）S を入力とする解集合に含まれる 3 項組のうち，$sum = (s_i, s_j, s_k)$ より辞書式順に小さい 3 項組の集合を，$LL_i = \{(s_{i_0}, s_{j_0}, s_{k_0}), (s_{i_1}, s_{j_1}, s_{k_1}), \cdots, (s_{i_{m'-1}}, s_{j_{m'-1}}, s_{k_{m'-1}})\}$ とする．このとき，問題の定義より，$sum = (s_i, s_j, s_k)$ は以下の 2 条件を満たす辞書式順に最小の 3 項組である．

(1) sum は集合 $L3S_i$ に含まれる．

(2) s_i，s_j，s_k は LL_i に含まれる 3 項組を構成する要素の集合 $\{s_{i_0}, s_{j_0}, s_{k_0}, s_{i_1}, s_{j_1}, s_{k_1}, \cdots, s_{i_{m'-1}}, s_{j_{m'-1}}, s_{k_{m'-1}}\}$ には含まれない．

そこで，まず，LL_i に含まれる 3 項組の個数 m' を検証する．S に含まれる s_i より小さい要素の数はたかだか $i-1$ であり，問題の定義より，sum より辞書式順に小さい 3 項組は，これらの要素を必ず 1 つずつ含んでいるので，$m' \le i-1$ である．

$L3S_i$ に含まれる 3 項組は，s_i より小さい要素を含むことはないので，LL_i に含まれる 3 項組を構成する要素で $L3S_i$ に含まれる要素の数はたかだか $2(i-1)$ である．また，定義より $L3S_i$ に含まれる 3 項組を構成する要素は，s_i を除いてすべて異なることが明らかである．したがって，$L3S_i$ の先頭から $2(i-1)+1$ 個目までの 3 項組の集合 $L3S_{i,2i-1}$ に，LL_i に含まれる 3 項組を構成する要素を 1 つも含まない 3 項組が必ず存在することになり，補題が成り立つ． □

上記の補題により，p 個のプロセッサを用いて計算を行う場合，各プロセッサ P_i は $L3S_{i,2p-1}$ を計算すればよいことになり，そのサイズは $2p-1 = O(p)$ となる．以下では簡単のため，$L3SP_i = L3S_{i,2p-1}$ とする．

以上より，BSP モデル上で辞書式順極大 3 和問題を解く p $(1 \le p \le n)$ プロセッサのアルゴリズムは，以下のようになる．

アルゴリズム 1

入力： n 要素の整数集合 I

(1) 各プロセッサ P_i $(0 \le i \le p-1)$ において，I を昇順にソートする．ソート後の列を $S = \{s_0, s_1, \cdots, s_{n-1}\}$ とする．

(2) $j = 0$ とおき，$j < n$ の間，以下のステップを繰り返す．

(2-1) 各 P_i $(0 \le i \le p-1)$ において，$L3SP_{i+j}$ を計算する．

(2-2) 各 P_i $(1 \le i \le p-1)$ は，3 項組の集合 $L3SP_{i+j}$ を P_0 に送信する．P_0 は，集合 $L3SP_j \cup L3SP_{1+j} \cup \cdots \cup L3SP_{(p-1)+j}$ を順に調べ，解となる 3 項組の集合を決定し，その集合をその他のすべてのプロセッサへ放送する．

(2-3) 各 P_i $(0 \le i \le p-1)$ において，$j = j+p$ とし，受け取った 3 項組の集合に含まれる要素を S より削除する．

上記のアルゴリズムより計算量については，以下の定理が成り立つ．

定理 1 アルゴリズム 1 は BSP モデル上で，内部計算時間 $O(\frac{n^2}{p}+n\log n+np)$，通信計算量 $O(n(gp+\frac{L}{p}))$ で辞書式順極大 3 和問題を解く．

（証明）アルゴリズムの各ステップの計算量の評価を行う．ステップ (1) は逐次ソートであるので，内部計算時間 $O(n\log n)$ で実行できる．ステップ (2) の計算量を評価するために，まず繰返し 1 回分の計算量を求める．ステップ (2-1) は前述の逐次アルゴリズム[3] を用いて，$O(n)$ 時間で計算できる．ステップ (2-2) では，ステップ (2-1) で求めた 3 項組の集合 $L3SP_{i+j}$ の送受信を行っているが，補題 1 より各プロセッサ P_i が送信する $L3SP_{i+j}$ のサイズはたかだか $O(p)$ であり，P_0 が受信する要素のサイズは $O(p^2)$ である．また，集めた 3 項組の集合から辞書式順極大 3 和問題の解となる 3 項組を決定する処理は，$O(p^2)$ 回の比較で容易に実現される．したがって，ステップ (2-2) は内部計算量 $O(p^2)$，通信計算量は $O(gp^2+L)$ となる．以上より，ステップ (2) の繰返し 1 回分は内部計算量 $O(n+p^2)$，通信計算量 $O(gp^2+L)$ となる．ステップ (2) の繰返し回数は $O(\frac{n}{p})$ であるので，ステップ (2) 全体では，内部計算量 $O(\frac{n^2}{p}+np)$，通信計算量 $O(n(gp+\frac{L}{p}))$ となる． □

定理 1 より，アルゴリズム 1 は $p\le\sqrt{\frac{n}{g}}$ のとき，内部計算時間，通信時間ともに $O(\frac{n^2}{p})$ となり，コスト最適であることが示される．

3.2 アルゴリズム 2

アルゴリズム 2 は，アルゴリズム 1 の通信回数を減らすことで，性能向上を目指したアルゴリズムである．具体的な改良点は以下のとおりである．アルゴリズム 1 では，各プロセッサ P_i に割り当てられる計算は $L3SP_i$ のみであったため，計算および通信を $O(\frac{n}{p})$ 回繰り返す必要があった．そこで，$L3SP_i$ を b 個ブロック化し，以下に定義する集合を各プロセッサに割り当てる単位とすることで，計算の繰返し回数を $O(\frac{n}{bp})$ 回におさえる．

定義 5 $S=\{s_0,s_1,\cdots,s_{n-1}\}$ を昇順ソート列とする．このとき，S の各要素 s_i について，3 項組の集合 $L3SB_i(b)$ を以下のように定義する．

$$L3SB_i(b) = L3SP_i\cup L3SP_{i+1}\cup \cdots\cup L3SP_{min(i+b-1,n-1)}$$ □

アルゴリズムの詳細を以下に示す．

アルゴリズム 2

アルゴリズム 1 のステップ (2) を，以下のように変更する．

(2) $j=0$ とおき，$j<n$ の間，以下のステップを繰り返す．

(2-1) 各 P_i $(0\le i\le p-1)$ において，$L3SB_{bi+j}(b)$ を計算する．

(2-2) P_i $(1\le i\le p-1)$ は，3 項組の集合 $L3SB_{bi+j}(b)$ を P_0 に送信する．P_0 は，集合 $L3SB_j(b)\cup L3SB_{b+j}(b)\cup\cdots\cup L3SB_{b(p-1)+j}(b)$ を順に調べ，解となる 3 項組の集合を決定し，その 3 項組の集合をその他すべてのプロセッサに放送する．

(2-3) 各プロセッサにおいて，$j=j+bp$ とし，受け取った 3 項組の集合に含まれる要素を S から削除する．

上記のアルゴリズムより計算量については，以下の定理が成り立つ．

定理 2 アルゴリズム 2 は BSP モデル上で，内部計算時間 $O(\frac{n^2}{p}+n\log n+nbp)$，通信計算量 $O(n(gbp+\frac{L}{bp}))$ で辞書式順極大 3 和問題を解く．

（証明）アルゴリズム 1 からの変更点である，ステップ (2) の計算量の評価を行う．ステップ (2-1) は前述の逐次アルゴリズム[3] により，$O(bn)$ 時間で実行できる．ステップ (2-2) では，ステップ (2-1) で求めた 3 項組の集合 $L3SB_{bi+j}(b)$ の送受信を行っているが，補題 1 より各プロセッサ P_i が送信する $L3SB_{bi+j}(b)$ のサイズはたかだか $O(b^2p)$ であり，P_0 が受信する要素のサイズは $O((bp)^2)$ である．また，集めた 3 項組の集合から辞書式順極大 3 和問題の解となる 3 項組を決定する処理は，$O((bp)^2)$ 回の比較で容易に実現される．したがって，ステップ (2-2) は内部計算量 $O((bp)^2)$，通信計算量は $O(g(bp)^2+L)$ となる．以上より，ステップ (2) の繰返し 1 回分は内部計算量 $O(bn+(bp)^2)$，通信計算量 $O(g(bp)^2+L)$ となる．ステップ (2) の繰返し回数は $O(\frac{n}{bp})$ なので，ステップ (2) 全体では，内部計算量 $O(\frac{n^2}{p}+nbp)$，通信計算量 $O(n(gbp+\frac{L}{bp}))$ となる． □

定理 2 より，アルゴリズム 2 は $p\le\sqrt{\frac{n}{gb}}$ のとき，内部計算時間，通信時間ともに $O(\frac{n^2}{p})$ となり，コスト最適であることが示される．

3.3 2 つのアルゴリズムの比較

今回提案した 2 つのアルゴリズムの計算量を**表 1** にまとめる．

アルゴリズム 2 は，アルゴリズム 1 に比べて繰返し回数が $\frac{1}{b}$ 倍になっているので，通信遅延 L の影響も $\frac{1}{b}$ 倍におさえられている．その一方で，送受信のオーバヘッド g や内部計算量の中に b 倍になっている部

表 1 アルゴリズム 1，アルゴリズム 2 の計算量
Table 1 Time complexities for algorithm 1 and algorithm 2.

	内部計算量	通信計算量
アルゴリズム 1	$O(\frac{n^2}{p} + n\log n + np)$	$O(n(gp + \frac{L}{p}))$
アルゴリズム 2	$O(\frac{n^2}{p} + n\log n + nbp)$	$O(n(gbp + \frac{L}{bp}))$

分がある．これは，処理をブロック化したことにより，各プロセッサが求める 3 項組の数が b 倍になったことに起因する．このように，処理のブロック化は長所と短所をあわせ持っており，スピードアップの向上を得るためには適当なブロックサイズを設定する必要があると考えられる．ここで，最適なブロックサイズが問題となるが，クラスタ並列処理には，各プロセッサはイーサネット等の比較的低速なネットワークによって接続されているため，通信遅延 L が内部計算量や通信オーバヘッド g に対して非常に大きいという特徴がある．したがって，ある程度大きいブロックサイズで処理した場合に，よいスピードアップが得られることが予測される．

4. 実験結果，および考察

本論文で提案した 2 つのアルゴリズムを PVM[4] を用いて PC クラスタ上に実装し，その性能評価を行った．PVM とは，複数の PC やワークステーションを 1 つの仮想並列計算機として利用するためのシステムソフトウェアである．今回の実験環境を**表 2** に示す．

上記の環境において 500,000 要素の整数集合に対して実験を行った．アルゴリズム 2 についてはブロック数 b を 10 から 60 まで変更し，スピードアップの変化を検証した．各アルゴリズムのスピードアップグラフを**図 2** および**図 3** に示す．なお，スピードアップの基準となる $p = 1$ の場合の実行時間は，文献 3) による逐次アルゴリズムの実行時間である．

図 2 および図 3 より，アルゴリズム 1，アルゴリズム 2 ともに，8 プロセッサ使用時に 4 倍程度のスピードアップが得られていることが分かる．これは，理想的なスピードアップではないが，実用的な範囲であるといえる．また，スピードアップがプロセッサ台数に比例して向上している点においても，実用的な並列アルゴリズムの条件を満たしているといえる．

実験結果を比較すると，すべてのプロセッサ数に対してアルゴリズム 2 よりもアルゴリズム 1 の方がよいスピードアップを達成している．アルゴリズム 2 では通信回数が削減されており，通信負荷の大きいクラスタ並列処理環境では，頻繁に通信を行う必要があるアルゴリズム 1 よりも高速であることが予想されたが，期待どおりの結果は得られていない．

これには，ブロック化により新たに生じた以下のような原因が考えられる．まず，アルゴリズム 2 は，アルゴリズム 1 に比べて通信回数は $\frac{1}{b}$ 倍になっているが，内部計算量や通信要素数において b 倍になっている処理が存在する．一般にクラスタ並列処理では，内部計算や送受信オーバヘッド g は，通信遅延である L より非常に小さいと考えられている．しかし，実験結果から考えると今回の環境においては，ほぼ同程度の負荷であったと考えられる．また，ブロック化により，負荷の割当てにある程度の不均衡が生じることになり，それによる各プロセッサの待ち時間が速度低下につながっているものと考えられる．

表 2 実験環境
Table 2 Experimental environment.

CPU	メモリ	OS
PentiumII 233 MHz	64 MB	Solaris 2.6
ネットワーク	コンパイラ	PVM
100BASE-TX	gcc 2.8.1	Version 3.4.1

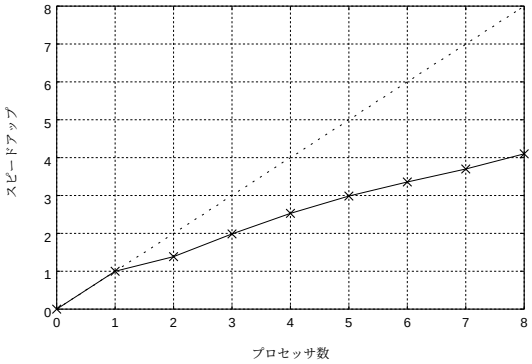

図 2 アルゴリズム 1 のスピードアップグラフ
Fig. 2 Speedup for algorithm 1.

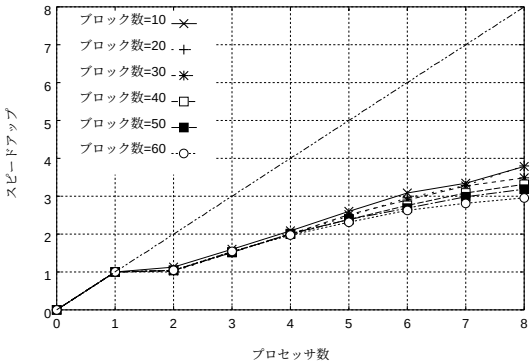

図 3 アルゴリズム 2 のスピードアップグラフ
Fig. 3 Speedup for algorithm 2.

5. ま と め

本論文では，辞書式順極大 3 和問題を解く 2 つの並列アルゴリズムを示した．アルゴリズム 1 は，BSP モデル上で内部計算時間が $O(\frac{n^2}{p} + n\log n + np)$ 時間，通信計算量が $O(n(gp + \frac{L}{p}))$ のアルゴリズムであり，各プロセッサが求める 3 項組の数を制限することで，実際のクラスタ環境でも実用的なスピードアップを達成することが示された．アルゴリズム 2 は，アルゴリズム 1 の通信回数が減少するように改良したアルゴリズムで，内部計算時間が $O(\frac{n^2}{p} + n\log n + nbp)$ 時間，通信計算量が $O(n(gbp + \frac{L}{bp}))$ である．しかし，改良による内部計算量の増加の影響等から，期待どおりのスピードアップは得られなかった．

今後の課題としては，アルゴリズム 2 において負荷の均等な割当てを行い，実際の並列処理環境でのさらなるスピードアップの達成を目指すこと等が考えられる．また，その他の P 完全問題（線形計画法，最大流問題）への応用が考えられる．

謝辞 本研究の一部は，文部省科学研究費補助金（特定研究 (B)(2)10205218）による．

参 考 文 献

1) Erickson, J. and Seidel, R.: Better lower bounds on detecting affine and spherical degeneracies, *34th Annual IEEE Symposium on Foundations of Computer Science* (*FOCS '93*), pp.528–536 (1993).
2) Fujiwara, A., Inoue, M. and Masuzawa, T.: Parallelizability of some P-complete problems, *Proc. Workshop on Advances in Parallel and Distributed Computational Models*, Lecture Notes in Computer Science, Vol.1800, pp.116–122 (2000).
3) Gajentaan, A. and Overmars, M.H.: On a class of $O(n^2)$ problems in computational geometry, *Computational Geometry*, Vol.5, pp.165–185 (1995).
4) Geist, A., et al.: *PVM*: *Parallel Virtual Machine A User's Guide and Tutorial for Networked Parallel Computing*, MIT Press (1994).
5) 宮野 悟：並列アルゴリズム：理論と設計，近代科学社 (1993).
6) Shiloach, Y. and Vishkin, U.: An $O(n^2 \log n)$ Parallel MAX-FLOW Algorithm, *Journal of Algorithm*, Vol.3, pp.128–146 (1982).
7) Träff, J.: Distributed, Synchronized Implementation of an Algorithm for the Maximum Flow Problem, *Proc. International Conference on Parallel Processing*, Vol.III, pp.110–114 (1994).
8) Valiant, L.: A bridging model for parallel computation, *Comm. ACM*, Vol.33, No.8, pp.103–111 (1990).

(平成 12 年 8 月 18 日受付)
(平成 12 年 12 月 1 日採録)

中島 孝明

平成 10 年九州工業大学情報工学部電子情報学科卒業．平成 12 年同大学大学院博士前期課程修了．現在同大学院博士後期課程在学中．並列アルゴリズムの研究に従事．電子情報通信学会会員．

藤原 暁宏（正会員）

平成 5 年大阪大学基礎工学部情報工学科卒業．平成 9 年奈良先端科学技術大学大学院博士後期課程修了．同年九州工業大学情報工学部講師を経て，平成 12 年同大学情報工学部助教授．現在に至る．並列分散アルゴリズム，クラスタ処理に関する研究に従事．工学博士．IEEE，電子情報通信学会各会員．