社団法人 情報処理学会　研究報告
IPSJ SIG Technical Report

2008－AL－120　(2)
2008／11／7

断続的故障を考慮したマルチプロセッサシステムの並列故障診断

伊東　桂，山田　敏規

埼玉大学　大学院理工学研究科　数理電子情報部門
〒 338-8570　埼玉県さいたま市桜区下大久保 255

E-mail: yamada@mail.saitama-u.ac.jp

要旨：高々s 個の断続的故障プロセッサを含めて $n/2$ 個未満の故障プロセッサが存在する n 個のプロセッサから成るマルチプロセッサシステムに対して，Fu と Beigel は $O(s^5)$ 検査ラウンドで動く適応的並列故障診断アルゴリズムを提案した．小文は，同様のシステムに対して $O(s^3)$ 検査ラウンドで動く適応的並列故障診断アルゴリズムを与える．さらに，任意の適応的並列故障診断アルゴリズムは $\Omega(s)$ 検査ラウンドを必要とすることを示す．

Parallel Diagnosis of Multiprocessor Systems in the Precense of Intermittent Faults

Kei ITOH and Toshinori YAMADA

Division of Mathematics, Electronics and Informatics
Graduate School of Science and Engineering, Saitama University
255 Shimo-Okubo, Sakura-ku, Saitama-shi, Saitama 338-8570, Japan

E-mail: yamada@mail.saitama-u.ac.jp

Abstract: For any multiprocessor system with n processors, which has less than $n/2$ faulty processors, including at most s intermittent ones, Fu and Beigel proposed an adaptive parallel fault diagnosis algorithm in $O(s^5)$ testing rounds. In this paper, we present an improved parallel diagnosis algorithm for the system in $O(s^3)$ testing rounds. Moreover, it is proved that any parallel diagnosis algorithm require $\Omega(s)$ testing rounds in order to diagnose the system.

1　はじめに

マルチプロセッサシステムのシステムレベル故障診断のためのグラフ理論的な枠組みは Preparata ら [20] によって提案され，PMC モデルとして知られており，PMC モデルに基づいたマルチプロセッサシステムの故障診断に関する研究がこれまでに数多く行われている．PMC モデルでは，各プロセッサは正常もしくは（常時）故障のどちらか一方の状態を取り，故障の最中にこの状態が変わることはないと仮定する．また，システム内の故障プロセッサの数は制限されていると仮定する．各プロセッサは，ネットワークを介して任意のプロセッサの検査を行ない，正常であるか故障しているかを判断する．ただし，正常プロセッサが行う検査は常に正しいが，故障プロセッサが行う検査は全く信頼できないものとする．マルチプロセッサシステムの故障診断とは検

査結果に基づきシステムの全ての故障プロセッサを同定することである．任意の検査結果からシステム内の全ての故障プロセッサが同定可能であるためには，故障プロセッサの数がマルチプロセッサシステム全体のプロセッサの数の半分未満でなければならないことが [20] で知られている．

PMC モデルにおけるマルチプロセッサシステムの故障診断は全ての検査が前もって決まっていなければならないという意味で非適応的である．一方，Nakajima [16] は，次に行う検査の箇所をそれまでに行われた検査の結果を元に決定することが可能な，マルチプロセッサシステムの適応的故障診断を提案した．適応的故障診断に関する研究はこれまで広く行われている [1–8,10–14,16–19,21]．とりわけ，Blecher [7] と Wu [21] は高々t 個 ($t < n/2$) の故障プロセッサを含む n 個のプロセッサから成るシステムに対して，故障診断を行うには高々$n+t-1$ 回の検査を行えば十分であることを示した．さらに Blecher [7] は故障診断のためには $n+t-1$ 回の検査が必要である場合が存在することも示した．

マルチプロセッサシステムの適応的並列故障診断についてもこれまでに広く研究されている [1–3, 6, 11,14,17,19]．適応的並列故障診断の中では，各ラウンドにおいて各プロセッサは検査する側もしくは検査される側として高々1 つの検査にのみ携わることが出来る．Beigel ら [1] は高々t 個の故障プロセッサを含む n 個のプロセッサから成るシステムに対して，$t \leq \sqrt{n/3}$ であるならば 3 ラウンドで，$t \leq 0.03n$ であるならば 4 ラウンドで，$t < n/2$ であるならば 10 ラウンドで故障診断を行うことが出来ることを示した．また，故障診断を行うためには $t \geq 2$ であるならば 3 ラウンドを，$t \geq 2\sqrt{2n}$ であるならば 4 ラウンドを，$t \geq 0.49n$ であるならば 5 ラウンドを必要とすることも示した．

このように，PMC モデルは広く研究されているが，以下にあげる 2 つの現実的な問題に対応できないことも知られている：

- プロセッサが故障診断の最中に状態を変化させるかもしれない；
- 故障が断続的に起こるかもしれない．

この問題に対応するために，プロセッサの状態として正常，常時故障の他に断続的故障を導入することによって PMC モデルを拡張したモデルが考えられている [15, 22]．このモデルをここでは MM モデルと呼ぶ．MM モデルにおいて，断続的故障プロセッサが検査する側もしくは検査される側として関わる検査の結果は全く信用できないと仮定する．このとき，全ての（常時もしくは断続的）故障プロセッサを同定することが不可能である場合が存在することに注意されたい: なぜならば，故障診断の間，断続的故障プロセッサは正常プロセッサと全く同じ振舞いをするかもしれないからである．MM モデルでは，マルチプロセッサシステムの故障診断とは全ての常時故障プロセッサといくつかの断続的故障プロセッサを同定することである．Fu と Beigel [9] は，故障プロセッサの数が全体の半数未満であるという仮定の下で，高々s 個の断続的故障プロセッサを含むマルチプロセッサシステムの適応的並列故障診断を $O(s^5)$ ラウンドで行うことが出来ることを示した．

小文では，故障プロセッサの数が全体の半数未満であるという仮定の下で，高々s 個の断続的故障プロセッサを含むマルチプロセッサシステムの適応的並列故障診断を $O(s^3)$ ラウンドで行うことが出来ることを示す．また，高々s 個の断続的故障プロセッサを含むマルチプロセッサシステムの適応的並列故障診断を行うためには，少なくとも $\Omega(s)$ ラウンドが必要となることも示す．

2　検査ラウンド数の下界

小文を通して，P はマルチプロセッサシステムの n 個のプロセッサの集合とし，P は高々s 個の断続的故障プロセッサを含めて合計で $n/2$ 未満の故障プロセッサを含んでいると仮定する．

補題 1 P の故障診断を行うためには，P の各プロセッサは少なくとも $s+1$ 個のプロセッサによって検査されなければならない．

証明: マルチプロセッサシステムに対する全ての検査に対して，その結果が正常であったと仮定する．また，高々s 個のプロセッサにしか検査されなかったプロセッサ $p \in P$ が存在したと仮定する．$q_1, q_2, \ldots, q_l (l \leq s)$ を p を検査したプロセッサとする．このとき，少なくとも以下の 2 つの場合が考えられる：

- 全てのプロセッサが正常である；

- p は常時故障，$q_1, q_2, \ldots, q_l$ は断続的故障，それ以外のプロセッサは全て正常である．

したがって，p が正常であるか故障しているかを同定できず，P の故障診断を行うことが出来ない．よって，故障診断を行うことが出来ない．

□

適応的並列故障診断において各ラウンドで行うことが出来る検査の数は高々 $\lfloor n/2 \rfloor$ であるので，補題 1 より以下の定理を得る．

定理 1 P の故障診断を行うためには，少なくとも $2(s+1)$ ラウンドを必要とする．

3 マルチプロセッサシステムの適応的並列故障診断アルゴリズム

3.1 Fu と Beigel のアルゴリズム [9]

小文で提案する適応的並列故障診断アルゴリズムは Fu と Beigel のアルゴリズムに基づいている．そこでまず，Bu と Beigel のアルゴリズムについて述べ，それから提案アルゴリズムについて述べる．

3.1.1 準備

Fu と Beigel のアルゴリズムの記述に入る前に用語・表記を定義する．

P を n 個のプロセッサから成る集合とする．2 つのプロセッサ $p, q \in P$ がお互いに検査を行い，正常であると診断したとき，p と q はお互いに同意していると言う．お互いに同意しているプロセッサから成る集合を SN(supernode) と呼ぶ．以下の事実は簡単に確認できる．

事実 1 SN は正常プロセッサと常時故障プロセッサを同時に含まない．

常時故障 [正常] プロセッサを含まない SN は正常である [故障している] と言われる．特に，正常 [常時故障] プロセッサのみを含む SN は真に正常である [真に故障している] と言われる．SN のサイズは SN に含まれるプロセッサの数である．

A, B を P の 2 つの非共有な部分集合とする．A の各プロセッサと B の各プロセッサがお互いに検査を行うことを A と B の間の完全検査と呼ぶ．A の各プロセッサが B の各プロセッサを検査することを A から B への完全検査と呼ぶ．$|A| = |B|$ であるとき，A と B の間に任意にマッチングをとり，このマッチングに沿って A のプロセッサと B のプロセッサがお互いに検査を行うことを A と B の間の並行検査と呼び，A のプロセッサがマッチングに沿って B のプロセッサを検査することを A から B への並行検査と呼ぶ．P の部分集合 A が与えられたとき，A のどの 2 つのプロセッサもお互いに検査を行うことを A 内の完全検査と呼ぶ．以下の 3 つの補題は簡単に確認できる [9]：

補題 2 非共有なプロセッサの集合 A, B が与えられたとき，A から B への完全検査を $\max\{|A|, |B|\}$ ラウンドで行うことが出来る．また，A と B の間の完全検査を $2\max\{|A|, |B|\}$ ラウンドで行うことが出来る．

補題 3 $|A| = |B|$ である非共有なプロセッサの集合 A, B が与えられたとき，A から B への並行検査と A と B の間の並行検査をそれぞれ 1 ラウンドと 2 ラウンドで行うことができる．

補題 4 プロセッサの集合 A 内の完全検査を高々 $2|A|$ ラウンドで行うことができる．

$\delta(H)$ は以下のように定義されるプロセッサの集合である：

- H がプロセッサであるならば，$\delta(H) = \{H\}$ である；
- H がプロセッサの集合であるならば，$\delta(H) = H$ である；
- H がプロセッサの集合の集まりであるならば，$\delta(H) = \bigcup_{A \in H} A$ である；
- H がこれらの複合によって構成されるならば，$\delta(H) = \bigcup_{A \in H} \delta(A)$ である；

s に関する以下の多項式はアルゴリズムの記述に使われる．

$$u(s) = 2^{\lceil \log(2s+1) \rceil}, \qquad x(s) = u(s)w(s)$$
$$y(s) = x(s) + u(s) + tu(s), \quad w(s) = 100s + 201$$
$$z(s) = \frac{1}{2}y(s) + s$$

3.1.2 概要

Fu と Beigel [9] によって提案されたアルゴリズムは以下の 9 つのステージから成る.

Stage 0: $n < 8z(s)(2s+1)$ であるならば P 内の完全検査を行い，故障と同定されたプロセッサのみを出力して終了する；

Stage 1: P の n 個のプロセッサを以下のように 3 つの集合 S, J, C へ分割する：

- S はサイズ $u(s)$ の SN から成る集合；
- J は各 $0 \leq i \leq \lceil \log(2s+1) \rceil - 1$ に対してサイズ 2^i の SN を高々1 個含む集合；
- C は互いに同意していないサイズ $2^i (0 \leq i \leq \lceil \log(2s+1) \rceil - 1)$ の SN の対から成る集合；

Stage 2: C の中から互いに同意しているプロセッサから成る鎖 (chain) を構成する；

Stage 3: $|S| \leq w(s)$ のとき，Stage 2 で構成した鎖の中から正常な鎖を見つけだし，この鎖を用いて他の全てのプロセッサを検査する.

Stage 4-8: $|S| > w(s)$ のとき，S の中から正常な SN をいくつか見つけだし，これらの SN を用いて他の全てのプロセッサを検査する.

本節の残りでアルゴリズムの Stage 0–3 を詳細に説明する. Stage 4–8 の詳細は [9] を参照されたい. Stage $i (0 \leq i \leq 8)$ での検査ラウンド数を $r_i(s)$ で表す.

3.1.3 Stage 0

補題 4 より P 内の完全検査は高々 $2n \leq 16z(s)(2s+1) = O(s^3)$ ラウンドで行うことが出来る. すなわち，$r_0(s) = O(s^3)$ である. また，P の過半数のプロセッサが正常であるので，各プロセッサは，半分以上のプロセッサから正常と診断されたならば正常であると，さもなければ故障していると同定することにより，断続的故障プロセッサ以外のプロセッサを正しく診断することが出来る. 以上をまとめると以下の補題を得る.

補題 5 $n < 8z(s)(2s+1)$ であるならば，Stage 0 は $O(s^3)$ ラウンドで全ての常時故障プロセッサといくつかの断続的故障プロセッサを出力する.

3.1.4 Stage 1

Stage 1 の詳細を図 1 に示す. ここで，2 つの SN A, B から成る対 (集合) を $\{A, B\}$ の代わりに $\langle A, B \rangle$ で表す.

```
各プロセッサをサイズ 1 の SN とする;
k := 1; C := ∅; J := ∅;
repeat
  if サイズ k の SN の数が奇数である
  then 任意の一つを J に加える;
  残りの SN で任意に対を作らせる;
  for 各 SN の対 ⟨A, B⟩ do
    A と B の間の完全検査を行う;
    if すべての検査結果が " 正常 " である
    then
      サイズ 2k の SN A ∪ B をつくる
    else C := C ∪ {⟨A, B⟩};
  k := 2k;
until k = u(s)
```

図 1: Stage 1

このとき，[9] において以下の補題が示されている:

補題 6 1. $r_1(s) \leq 2(u(s) - 1)$;

2. Stage 1 内の任意の SN は正常なプロセッサと常時故障しているプロセッサを同時に含まない；

3. $|\delta(J)| \leq u(s) - 1$;

3.1.5 Stage 2

Stage 2 の詳細を図 2 に示す. ここで，C の全ての対は

$$\langle A_{1,0}, A_{1,1} \rangle, \langle A_{2,0}, A_{2,1} \rangle, \ldots, \langle A_{k,0}, A_{k,1} \rangle$$

と表されている.

図 2 の Chains に含まれる集合 K は鎖と呼ばれる. 鎖 K は連続する SN $A_{i,a_i}, A_{i+1,a_{i+1}}, \ldots, A_{j,a_j} (i \leq j)$ の集合である. このとき，$\langle A_{i-1,0}, A_{i-1,1} \rangle$ と $\langle A_{j+1,0}, A_{j+1,1} \rangle$ をそれぞれ K の上対 (top pair) と下対 (bottom pair) と呼ぶ.

Stage 2 に関して以下の補題が示される [9]:

補題 7 1. $r_2(s) \leq 4(s+1)u(s)$;

2. 任意の鎖 K に対して，K は正常プロセッサと常時故障プロセッサを同時に含まない；

```
for l := 1 to s + 1 do
  for i := 1 to k do
    A_{i,0} と A_{i+l,1} の間の完全検査を行う;
    A_{i,1} と A_{i+l,0} の間の完全検査を行う;
C_0 := {A_{1,0}}; C_1 := {A_{1,1}};
i := 1; Chains := ∅ ;
repeat
  i := i + 1
  if ある j ∈ {0,1} に対して C_0 が A_{i,j} に同意する
  then
    そのような j を選ぶ
    C_0 := C_0 ∪ {A_{i,j}};
    if C_1 が A_{i,1-j} に同意する
    then
      C_1 := C_1 ∪ {A_{i,1-j}}
    else
      Chains := Chains ∪{C_1}
      C_1 := {A_{i,1-j}}
  else
    Chains := Chains ∪{C_0}
    if ある j ∈ {0,1} に対して
      C_1 が A_{i,j} に同意する
    then
      そのような j を選ぶ
      C_1 := C_1 ∪ {A_{i,j}}
      C_0 := {A_{i,1-j}}
    else
      Chains := Chains ∪{C_1}
      C_0 := {A_{i,0}}
      C_1 := {A_{i,1}}
until i = k
```

図 2: Stage 2

3. 任意の鎖 K に対して，K の全ての SN が真に正常であるならば，K の上対と下対はともに真に正常な SN を含まない．

3.1.6 Stage 3

Stage 3 の詳細を図 3 に示す．Stage 3 は $|S| \leq w(s)$ であるときに実行される．

補題 8 1. $r_3(s) \leq 64z(s)^2(2s+1)$;

2. K_i が正常な鎖であるならば，GOOD_i は常時故障しているプロセッサを一切含まず，全ての正常プロセッサを含む；

3. K_i が故障している鎖であるならば，GOOD_i は正常プロセッサを一切含まない．

```
Chains から SN の数が n/4z(s) 以上の鎖を全て
  見つけ，これを K_1, K_2, ..., K_m とする;
for i := 1 to m do
  GOOD_i := δ(K_i);
  δ(K_i) を
    |A_j| = 2s + 1(1 ≤ j ≤ r)
    |A_{r+1}| ≤ 2s
  となるように A_1, A_2, ..., A_{r+1} へ分割する;
  P − δ(K_i) を
    ||P_j| − |P_k|| ≤ 1(1 ≤ j, k ≤ r)
  となるように P_1, P_2, ..., P_r へ分割する;
  for j := 1 to r do
    P_j と A_j の間の完全検査を行う;
    for p ∈ P_j do
      if p が A_j の過半数のプロセッサと
        互いに同意する
      then p を GOOD_i に加える;
  if |GOOD_i| > n/2
  then P − GOOD_i を出力
```

図 3: Stage 3

3.1.7 Stage 4-8

Stage 4 以降は $|S| > w(s)$ であるときに実行される．詳細は [9] を参照されたい．[9] において Stages 4–8 に関して以下のことが示されている．

補題 9 1. $r_4(s) + r_5(s) + \cdots + r_8(s) = O(s^2)$;

2. Stage 8 の終了後，全ての常時故障プロセッサを同定することが出来る．

3.2 提案アルゴリズム

3.2.1 Fu と Beigel のアルゴリズムの問題点

補題 5–9 より，Fu と Beigel のアルゴリズムにおいてラウンド数が $O(s^5)$ であるのは Stage 3 のみである．Stage 3 において，鎖 $K_1, K_2, \ldots, K_m$ から正常な鎖を見つけるために，各 i に対して $\delta(K_i)$ のプロセッサを用いて $P - \delta(K_i)$ の全てのプロセッサの検査を行わせている．この部分で $O(s^5)$ ラウンドを要する．そこで，Fu と Beigel のアルゴリズムの全体の構成は変えずに，Stage 3 を改良することによってラウンド数を減らす．

まず適切に $R \subset P$ を選ぶ．次に各 i に対して $\delta(K_i)$ のプロセッサを用いて $R - \delta(K_i)$ の全てのプロセッサの検査を行うことにより，K_i が正常な鎖であるか否かを判断する．最後に，正常な鎖 K_k が見

つかったならば，$\delta(K_k)$ のプロセッサを用いて $P-(R\cup\delta(K_k))$ の全てのプロセッサを診断する．この改良によって，ラウンド数を $O(s^5)$ から $O(s^3)$ へ減らす．

3.2.2 改良版 Stage 1

Stage 3 の改良を行うために，Stage 1 で求められた SN の対の集合 C を以下のように分割する：

- $C_{\rm no}$ をどの $p\in A$ と $q\in B$ も互いに同意していない SN A,B の対 $\langle A,B\rangle$ の集合とする；
- $C_{\rm some}$ を互いに同意しているプロセッサ $p\in A,q\in B$ と互いに同意しないプロセッサ $p'\in A,q'\in B$ がともに存在する SN A,B の対 $\langle A,B\rangle$ の集合とする．

これを実装した Stage 1 の改良版を図 4 に示す．改良版 Stage 1 で行われるプロセッサ間の検査は図 1 の Stage 1 で行われるプロセッサ間の検査と全く同じである．したがって，補題 6 より以下が成り立つ．

補題 10 改良版 Stage 1 は高々 $2(u(s)-1)$ ラウンドで実行される．

```
各プロセッサをサイズ 1 の SN とする;
k := 1; C_no := ∅; C_some := ∅; J := ∅;
repeat
  if サイズ k の SN の数が奇数である
  then 任意の一つを J に加える;
  残りの SN で任意に対を作らせる;
  SN の各対 ⟨A, B⟩ に対して
    A と B の間の完全検査を行う;
    if すべての検査結果が正常である
    then
      サイズ 2k の SN A∪B をつくる
    else
      if A と B の間に互いに同意する
        プロセッサの対が存在しない
      then C_no := C_no ∪ ⟨A, B⟩
      else C_some := C_some ∪ ⟨A, B⟩
  k := 2k;
until k = u(s)
C := C_no ∪ C_some
```

図 4: 改良版 Stage 1

補題 11 1. 任意の $\langle A,B\rangle\in C_{\rm no}$ に対して，A と B の少なくとも一方は故障している；

2. 任意の $\langle A,B\rangle\in C_{\rm some}$ に対して，以下の 2 つの命題の少なくとも 1 つが成り立つ：
   - A と B のどちらも故障している；
   - A と B の少なくとも一方は断続的に故障しているプロセッサを含む；
3. $\delta(S\cup J\cup C_{\rm some})$ の過半数のプロセッサが正常である；
4. $|S|\le w(s)$ であるならば，$|\delta(C_{\rm some})|\le y(s)+su(s)$ である．

証明: 1. A,B がともに故障していないと仮定する．このとき，いずれの SN にも少なくとも 1 個以上の正常プロセッサが含まれるので，A と B の間の完全検査の中に互いに同意しているプロセッサの対を含む．これは $C_{\rm no}$ の定義に矛盾する．

2. 以下の 2 つの事実から成り立つ：
   - A,B がともに真に正常な SN ならば，互いに同意しないプロセッサの対は存在しない
   - A,B が一方が真に正常で，もう一方が真に故障しているならば，互いに同意するプロセッサの対は存在しない

3. 1. より $C_{\rm no}$ の半分以上のプロセッサが (常時もしくは断続的) 故障している．したがって題意を満たす．

4. まず，$\delta(C_{\rm some})$ に含まれる (常時もしくは断続的) 故障プロセッサの数を見積もる．そのためには 3. より $\delta(S\cup J\cup C_{\rm some})$ に含まれる正常プロセッサの数を見積もれば十分である．$|S|\le w(s)$ より $|\delta(S)|\le u(s)w(s)=x(s)$ であり，補題 6 より $|\delta(J)|\le u(s)$ であり，2. より $\delta(C_{\rm some})$ に含まれる正常プロセッサの数は高々 $su(s)$ であるので，$\delta(C_{\rm some})$ に含まれる (常時もしくは断続的) 故障プロセッサの数は高々 $x(s)+u(s)+su(s)=y(s)$ である．したがって，$|\delta(C_{\rm some})|\le y(s)+su(s)$ である． □

3.2.3 改良版 Stage 3

改良版 Stage 3 は 2 つの Step で構成されている．

Step 1 では長さ $n/4z(s)$ 以上の鎖 $K_1,K_2,\ldots,K_m$ の中から正常な鎖 K_k を見つける．まず，3.2.1 で述べた R として $S\cup J\cup C_{\rm some}$

を設定し，各iに対して$\delta(K_i)$の任意の$2s+1$個のプロセッサをK'_iに保持する．次に，$\delta(K_i)$のプロセッサで$\delta(R)$に含まれているものをGOOD_iに保持する．さらに，K'_iと$\delta(R)-\delta(K_i)$の間で完全検査を行わせ，K'_iの過半数のプロセッサと互いに同意しているプロセッサ$p \in \delta(R)-\delta(K_i)$を$\mathrm{GOOD}_i$に加える．最後に，$|\mathrm{GOOD}_i| > |\delta(R)|/2$であるならば$K_i$を正常であると診断する．この手続きが正しいことは補題11の3.から示される．

Step 2では正しいと診断された鎖K_kを用いてC_{no}のプロセッサを診断する．K_kは高々s個のプロセッサを除く全てのプロセッサが正常であるので，各プロセッサ$p \in C_{\mathrm{no}}$は，K_kの$(2s+1)$個のプロセッサから診断されているならば正しい診断を行うことが出来る．

改良版Stage 3について以下の補題が成り立つ．

補題 12 1. $|S| \leq |w(s)|$であるとき，改良版Stage 3は$O(s^3)$ラウンドで実行される；

2. $|S| \leq |w(s)|$であるとき，改良版Stage 3終了時にGOOD_kはPの全ての正常プロセッサを含み，常時故障プロセッサを一切含まない．

証明: 1. $|K'_i| = 2s+1$であり，補題6の3.と補題11の4.より$|S| \leq w(s)$であるとき$|\delta(R)| = O(s^2)$であるので，補題2より，K_{in}と$\delta(R)$の間の完全検査は$O((2s+1)m) = O((2s+1)z(s)) = O(s^3)$ラウンドで行うことが出来る．$K'_1, K'_2, \ldots, K'_m$の間の並行検査は$O(m) = O(s^2)$ラウンドで行うことが出来る．$r = \Theta(n/(4(2s+1)z(s))) = \Theta(n/s^3)$であるので，各$j$に対して$|P_j| = O(s^3)$である．各$j$に対して$|A_j| = 2s+1$であるので，補題2より$A_j$から$P_j$への完全検査は$O(s^3)$ラウンドで行うことが出来る．以上より，$|S| \leq w(s)$であるとき，改良版Stage 3は$O(s^3)$ラウンドで行うことが出来る．

2. $n/4z(s)$個以上の正常プロセッサから成る鎖K_kが存在することは[9]によって示されている．補題11の3.より$\delta(R) = \delta(S \cup J \cup C_{\mathrm{some}})$の過半数のプロセッサが正常であるので，$K_i$が正常であるための必要十分条件は$K'_i$の$s+1$個以上のプロセッサが$\delta(R)$の過半数のプロセッサと互いに同意することである．$K'_k$の$s+1$個以上のプロセッサは$\delta(R)$の過半数のプロセッサと互いに同意しているので，$K_k$は正常である．すなわち，$K_k$は高々$s$個のプロセッサを除く全てのプロセッサが正常である．したがって，各プロセッサ$p \in C_{\mathrm{no}}$は，K_kの$(2s+1)$個のプロセッサから検査を行い，少なくとも$(s+1)$個のプロセッサから正常であると診断されたならばpは正常もしくは断続的故障である．よって，GOOD_kはPの全ての正常プロセッサを含み，常時故障プロセッサを一切含まない． □

```
Chainsからプロセッサの数がn/4z(s)以上の鎖を全
て見つけ，これをK_1, K_2, ..., K_m とする;
[Step 1]
K_in := ∅; R := S ∪ J ∪ C_some;
for i := 1 to m do
  δ(K_i) から 2s+1 個のプロセッサを任意に選び，
    K'_i に保持する;
  K_in := K_in ∪ K'_i;
  GOOD_i := δ(K_i) ∩ δ(C_some)
K_in と R − K_in の間で完全検査を行う
K'_1, K'_2, ..., K'_m の任意の 2 つの集合の間で
  並行検査を行う;
for i := 1 to m do
  K'_i の過半数のプロセッサとお互いに同意している
    p ∈ δ(R) − K_i を GOOD_i に追加する;
  for j := 1 to m do
    if K'_i と K'_j(i ≠ j) の間の並行検査で
      過半数のプロセッサ対が互いに同意している
    then GOOD_i := GOOD_i ∪ δ(K_i);
  if |GOOD_i| > |P − C_no|/2
  then k := i とし，for ループを抜ける;

[step 2]
δ(K_k) を
  |A_j| = 2s + 1(1 ≤ j ≤ r), |A_{r+1}| ≤ 2s
  となるように A_1, A_2, ..., A_{r+1} へ分割する;
δ(C_no) − δ(K_k) を
  ||P_j| − |P_l|| ≤ 1(1 ≤ j, l ≤ r)
  となるように P_1, P_2, ..., P_r へ分割する;
for j := 1 to r do
  A_j から P_j への完全検査を行う;
  for p ∈ P_j do
    if p が A_j の過半数のプロセッサから
      正常と診断される
    then p を GOOD_k に加える;
P − GOOD_k を出力
```

図 5: 改良版 Stage 3

補題5,7,9,10,12より以下の定理を得る．

定理 2 提案アルゴリズムは$O(s^3)$ラウンドでPの全ての常時故障プロセッサといくつかの断続的故障プロセッサを出力する．

参考文献

[1] R. Beigel, W. Hurwood, and N. Kahale. Fault diagnosis in a flash. In *Proc. 36th FOCS*, pp. 571–580, 1995.

[2] R. Beigel, S.R. Kosaraju, and G.F. Sullivan. Locating faults in a constant number of testing rounds. In *Proc. 1st SPAA*, pp. 189–198, 1989.

[3] R. Beigel, G. Margulis, and D.A. Spielman. Fault diagnosis in a small constant number of parallel rounds. In *Proc. 5th SPAA*, pp. 21–29, 1993.

[4] R.P. Bianchini, Jr., and R. Buskens. An adaptive distributed system-level diagnosis algorithm and its implementation. In *Proc. 21st International Symposium on Fault Tolerant Computing*, pp. 222–229, 1991.

[5] R.P. Bianchini, Jr., K. Goodwin, and D.S. Nydick. Practical application and implementation of distributed system-level diagnosis theory. In *Proc. 20th International Symposium on Fault Tolerant Computing*, pp. 332–339, 1984.

[6] A. Björklund. Optimal adaptive fault diagnosis of hypercubes. *Lecture Notes in Computer Science*, 1851:527–534, 2000.

[7] P. M. Blecher. On a logical problem. *Discrete Mathematics*, 43:107–110, 1983.

[8] Chao Feng, Laxmi N. Bhuyan, and Fabrizio Lombardi. Adaptive system-level diagnosis for hypercube multiprocessors. *IEEE Transactions on Computers*, 45:1157–1170, 1996.

[9] B. Fu and Richard Beigel. Diagnosis in the presence of intermittent faults. *Lecture Notes in Computer Sciences*, 3341:427–441, 2004.

[10] S.L. Hakimi and K. Nakajima. On adaptive system diagnosis. *IEEE Transactions on Computers*, c-33:234–240, 1984.

[11] S.L. Hakimi, M. Otsuka, and E.F. Schmeichel. A parallel fault identification algorithm. *Journal of Algorithms*, 11:231–241, 1990.

[12] S.L. Hakimi and E.F. Schmeichel. An adaptive algorithm for system level diagnosis. *Journal of Algorithms*, 5:524–530, 1984.

[13] E. Kranakis, Andrzej Pelc, and A. Spatharis. Optimal adaptive fault diagnosis for simple multiprocessor systems. *Networks*, 34:206–214, 1999.

[14] Evangelos Kranakis and Andrzej Pelc. Better adaptive diagnosis of hypercubes. *IEEE Transactions on Computers*, 49(10):1013–1020, 2000.

[15] S. Mallela and G. M. Masson. Diagnosable systems for intermittent faults. *IEEE Transactions on Computers*, C-27:360–366, 1978.

[16] K. Nakajima. A new approach to system diagnosis. In *Proc. 19th Ann. Allerton Conf. Commun. Contr. and Computing*, pp. 697–706, 1981.

[17] Kumiko Nomura, Toshinori Yamada, and Shuichi Ueno. On adaptive fault diagnosis for multiprocessor systems. *Lecture Notes in Computer Science*, 2223:86–98, 2001.

[18] A. Okashita, T. Araki, and Y. Shibata. Adaptive diagnosis of butterflies with optimal number of tests. *Technical Report of IEICE*, 101(488, COMP2001-70):55–62, 2001.

[19] A. Okashita, T. Araki, and Y. Shibata. Adaptive diagnosis of butterflies networks. *IPSJ SIG Notes*, 2001(79,2001-AL-79);29–36, 2001.

[20] F. P. Preparata, G. Metze, and R. T. Chien. On the connection assignment problem of diagnosable systems. *IEEE Transactions on Electronic Computers*, 16;848–854, 1967.

[21] Pei Yuan Wu. Partial solution to problem 81-6. *Journal of Algorithms*, 3:379–380, 1982.

[22] C. L. Yang and G. M. Masson. A generalization of hybrid fault diagnosability. *IEEE Transactions on Computers*, C-36:1369–1374, 1987.