The University of Osaka
Institutional Knowledge Archive

Title	故障を考慮した分散アルゴリズムとネットワーク・トポロジに関する研究
Author(s)	増澤, 利光
Citation	大阪大学, 1987, 博士論文
Version Type	VoR
URL	https://hdl.handle.net/11094/241
rights	
Note	

The University of Osaka Institutional Knowledge Archive : OUKA

https://ir.library.osaka-u.ac.jp/

The University of Osaka

故障を考慮した分散アルゴリズムと

ネットワーク・トポロジに関する研究

昭和６２年２月

増澤　利光

故障を考慮した分散アルゴリズムと

ネットワーク・トポロジに関する研究

昭和６２年２月

増澤　利光

内 容 梗 概

本論文は，筆者が大阪大学大学院基礎工学研究科（物理系専攻）の学生として，都倉研究室において行った研究のうち，故障を考慮した分散アルゴリズムとネットワーク・トポロジに関する研究をまとめたものである．

ある問題を解くために必要な情報が，ネットワークで結合されている複数台のプロセッサに分散している状況で，それらの情報を交換しながらその問題を解くアルゴリズムを分散アルゴリズムとよぶ．これまでに，多くの分散アルゴリズムが提案されているが，それらの多くは，プロセッサやリンクに故障があるとうまく動作しない．実際には，プロセッサやリンクが故障することは少なくないので，プロセッサやリンクが故障しているネットワークで，問題を解く分散アルゴリズムを考えることは重要である．

また，ある決められた数以下のプロセッサやリンクが故障しても任意の２つのプロセッサ間に通信路があるようなネットワークを構成することも重要である．このようなネットワークの構成法に関する問題の１つとして，連結性に関するグラフの拡大構成問題がある．

本論文の第１章では，分散アルゴリズム，及び，連結性に関するグラフの拡大構成問題の研究の現状，その工学上の意義，本研究の新しい諸結果について概説している．

第２章では，グラフ，ネットワーク，分散アルゴリズムに関する諸定義について述べている．

本論文の第３章～第５章は，プロセッサやリンクが故障している可能性のあるネットワークで問題を解く分散アルゴリズムについて考察している．特に，各プロセッサがネットワークに関して持っている情報（ネットワークのトポロジ，辺連結度，プロセッサ数など）や，プロセッサと通信の同期性が，問題を解く分散アルゴリズムの存在性に真に影響を与えることを示している．

第６章では，グラフの k-頂点連結拡大構成問題について議論し，有向木に対して，時間計算量がオーダ的に最適なアルゴリズムを示している．

第７章の結論では，本研究で得られた主な結果をまとめ，今後に残された問題について述べている．

関連発表論文

[1] 増澤，和田，萩原，都倉："グラフの拡大構成問題に関する一考察"，電子通信学会技術研究報告，CAS83-72 (1983-08).

[2] 増澤，萩原，和田，都倉："ｋ－頂点連結性に関する有向２進木の拡大構成問題"，電子通信学会論文誌(D)，J67-D，1，pp.77-84 (1984-01).

[3] 増澤，萩原，和田，都倉："頂点連結性に関する拡大構成問題(I)"，電子通信学会技術研究報告，CAS84-2 (1984-04).

[4] 増澤，萩原，都倉："無向グラフの点連結性に関する拡大構成問題について"，電子通信学会技術研究報告，AL84-11 (1984-06).

[5] 増澤，萩原，都倉："有向木の連結性に関する拡大構成問題について"，電子通信学会技術研究報告，AL84-27 (1984-09).

[6] T. Masuzawa, K. Hagihara, K. Wada and N. Tokura："The graph classes with optimal algorithms for k-connectivity augmentation problems"，Proceedings of International Conference on Computers, Systems and Signal Processing, Bangalore, India, pp.195-198 (December 1984).

[7] 増澤，萩原，和田，都倉："有向３進木のｋ－連結拡大構成問題"，電子通信学会論文誌(D)，J68-D，5，pp.1003-1010 (1985-05).

[8] 増澤，萩原，都倉："分散アルゴリズムの耐辺故障性とネットワークの辺連結度"，情報処理学会ソフトウェア基礎論研究会資料，SF14-3 (1985-10).

[9] 増澤，萩原，都倉："分散ネットワークにおけるプロセッサ故障診断について"，電子通信学会技術研究報告，COMP86-32 (1986-09).

[10] 増澤，萩原，都倉："辺故障を考慮したある分散アルゴリズム"，電子通信学会論文誌(D)，J69-D，10，pp.1394-1405 (1986-10).

[11] T. Masuzawa, K. Hagihara and N. Tokura："An optimal time algorithm for the k-vertex-connectivity unweighted augmentation problem for rooted directed trees"， (accepted for publication in Discrete Applied Mathematics).

[12] 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"(投稿中)．

故障を考慮した分散アルゴリズムと

ネットワーク・トポロジに関する研究

目　次

１．緒論 …… 1

２．諸定義 …… 8

３．非同期式ネットワークにおける生成木構成問題 …… ２６

３．１　故障プロセッサがあると生成木構成問題が解けないこと　２８

３．２　辺連結度既知アルゴリズム …… ２９

3.2.1　耐辺故障度「CON/2」のアルゴリズムの存在しないこと　３０

3.2.2　耐辺故障度「CON/2」-１のアルゴリズム CST …… ３１

3.2.3　アルゴリズム CST の正当性 …… ３８

3.2.4　アルゴリズム CST の評価 …… ４４

３．３　サイズ既知アルゴリズム …… ４９

３．４　隣接識別子既知アルゴリズム …… ５１

３．５　仮定の緩和 …… ５３

４．プロセッサ故障診断問題 …… ５５

４．１　Ｐ非同期式ネットワーク …… ５８

４．２　Ｌ非同期式ネットワーク …… ５９

４．３　ＰＬ同期式ネットワーク …… ６０

4.3.1　リンク (v,w) が故障していないとき …… ６０

4.3.2　リンク (v,w) が故障しているとき …… ６３

５．リンク故障診断問題 …… ７７

５．１　Ｌ非同期式ネットワーク …… ７９

５．２　ＰＬ同期式ネットワーク …… ８０

５．３　Ｌ同期式かつＰ非同期式ネットワーク …… ８３

６．k-頂点連結拡大構成問題 …… １０１
６．１　定義および記法 …… １０１
６．２　辺拡大に必要な付加辺数 …… １１４
６．３　超デイジーチェイン …… １１５
６．４　グラフの合成と頂点連結度 …… １２８
６．５　ｋ－進木のｋ－頂点連結拡大構成問題 …… １３４
6.5.1　アルゴリズム KDTB …… １３４
6.5.2　アルゴリズム KDTB の正当性 …… １４５
6.5.3　アルゴリズム KDTB の時間計算量 …… １５６
６．６　木のｋ－頂点連結拡大構成問題 …… １５９
6.6.1　アルゴリズム DT …… １５９
6.6.2　アルゴリズム DT の正当性と時間計算量 …… １６０
７．結論 …… １６２
謝辞 …… １６４
文献 …… １６５

1．緒　論

ある問題を解くために必要な情報が，ネットワークで結合されている複数台のプロセッサ（全てのプロセッサは対等であり，ホスト・プロセッサなるものはない）に分散している状況で，それらの情報を交換しながらその問題を解くアルゴリズムを分散アルゴリズムとよぶ．本論文で考えるネットワークは，ローカル・メモリを持つプロセッサを全２重リンク（双方向の通信が独立に行える）で相互結合したものである．プロセッサ間には共有メモリは存在せず，プロセッサ間の通信はリンクを介して，メッセージを送受信することで行う．

これまでに，いろいろな問題に対して，多くの分散アルゴリズムが提案されている(2),(5),(11),(12),(18)-(20),(29),(30),(32),(33)．例えば，最大値探索問題（初期状態で各プロセッサは相異なる値を持っており，全てのプロセッサがネットワーク全体での最大値を知る）や生成木構成問題（ネットワークをグラフとみなして，その生成木を１つ見つけ，各プロセッサは，自分に接合している辺（リンク）のうちで，生成木の辺になっている辺の集合を知る）などがある．分散アルゴリズムの評価は通常，通信量計算量 (communication complexity)，理想時間計算量（ideal execution time）やプロセッサの領域計算量を評価基準として行われる．また，ネットワークでは，プロセッサやリンクの増設や除去が行われることもあるので，分散アルゴリズムではなるべくネットワークに関する大域的な情報を利用しない方が望ましい．従って，各プロセッサで利用できるネットワークに関する大域的な情報（ネットワークのトポロジーやプロセッサ数など）がこれらの計算量にどのような影響を与えるかを明確にすることは理論的にも興味深い問題であり，これまでにいくつかの研究が行われている(20),(32),(33)．

実際のネットワークにおいて，プロセッサやリンクの故障が生じることが少なくない．しかし，これまでに提案された分散アルゴリズムの多くは，プロセッサ やリンクに故障があるとうまく動作しない．一方，ネットワークの

プロセッサやリンクが故障したときに，アグリーメント問題（故障していないプロセッサは全て，停止時に同じ値を出力値としてもつ）を解く分散アルゴリズムが存在するかどうかの研究も行なわれている(4),(6),(9),(31)．文献（9）は，非同期式ネットワークにおいて，１個のプロセッサが故障のために停止したとき，アグリーメント問題を解く分散アルゴリズムが存在しないことを示し，文献（4）では，ネットワークが k-頂点連結のとき，k/2 個以上のプロセッサでビザンチン故障（故障が生じるとプロセッサやリンクが誤動作する可能性があるという故障）が生じたときに，アグリーメント問題を解く分散アルゴリズムが存在しないことを示している．また，文献（6）は，プロセッサや通信に同期性を仮定したときに，それらの同期性が，プロセッサやリンクの故障にかかわらずアグリーメント問題を解くアルゴリズムの存在性にどのような影響を与えるかについて考察している．

本論文の第３章～第５章は，関連発表論文［８］～[10]，[12] として公表した分散アルゴリズムに関する研究をまとめたものである．ここでは，プロセッサやリンクが故障している可能性のあるネットワークで，問題を解く分散アルゴリズムについて考察する．特に，各プロセッサがネットワークに関して持っている情報（ネットワークのトポロジ，辺連結度，プロセッサ数など）がこれらの問題を解く分散アルゴリズムの存在性にどのような影響を与えるかを考察する．本論文では，故障が生じると機能が全く停止してしまうという故障（fail-stop）についてのみ考察し，ビザンチン故障は扱わない．つまり，故障したプロセッサは，一切動作しないし，故障したリンクを用いる通信は一切できないものとする．また，あるプロセッサに故障が生じたとき，その隣接プロセッサでも，その故障を直接的には検知できないものとする．同様に，あるリンクに故障が生じたとき，そのリンクに接合するプロセッサでも，その故障を直接的には検知できないものとする．

このような状況で問題を解く方法として，次の２つの方法が考えられる．

(1) プロセッサやリンクの故障を診断する分散アルゴリズムを提案する．そして，問題を解く前に，それを用いてネットワークに存在する故障を見つ

ける．そして，故障しているプロセッサやリンクを取り除いて得られる部分ネットワークを求め，その部分ネットワークに対し，既存の（故障を考慮していない）分散アルゴリズムを適用する．

(2)　どのプロセッサが故障しているか，あるいは，どのリンクが故障しているかを分らないままで，プロセッサやリンクの故障にかかわらず問題を解く分散アルゴリズムを新たに提案する．

既知の分散アルゴリズムの多くは，非同期式ネットワークに対して提案されている．非同期式ネットワークとは，プロセッサの動作速度やリンクの伝播遅延に上限を設けないネットワークである．第４章，第５章で示すように，非同期式ネットワークでは，プロセッサやリンクの故障を診断する分散アルゴリズムは存在しない．そのため，非同期式ネットワークにおいては，上述の (1) の方法を用いることはできない．

第２章のグラフ，ネットワーク，分散アルゴリズムに関する諸定義に続き，第３章では，非同期式ネットワークで，プロセッサやリンクの故障にかかわらず生成木構成問題を解く分散アルゴリズムが存在するかどうかについて論じる．これは，上述の (2) の方法に関する議論である．ここで，対象として生成木構成問題を選んだのは，この問題が分散アルゴリズムを考えるときに，基本的であり，かつ，重要な問題であるからである[(12),(32)]．例えば，あるプロセッサが持っている情報をネットワークの全てのプロセッサに伝える (broadcast) 場合，生成木を利用すると，メッセージ数が減少し，しかも，同じメッセージの再送を防ぐための制御も不要となり，単純になる．また，最大値探索問題やネットワークの中心 (center) や中間点 (median) を求める問題等，生成木を利用すると比較的簡単に解ける問題が多い[(19)]．

まず，リンクに故障がなく，各プロセッサがネットワークの形状に関してどのような情報を持っていても，故障しているプロセッサが１個でも存在すると，生成木構成問題を解く分散アルゴリズムが存在しないことを示す．そして，故障しているプロセッサが存在しないときに，各プロセッサがネットワークに関して持っている情報と故障しているリンクの数が生成木構成問題

を解く分散アルゴリズムの存在性に真に影響を与えることを示す．第３章で得られる結果を表３．１にまとめておく．

第４章，第５章では，前述の (1) の方法に関する基本的な問題について考察している．

第４章では，任意の１つのプロセッサが自分の隣接プロセッサのうち任意の１つが故障しているかどうかを診断する問題（プロセッサ故障診断問題）を取り上げる．まず，プロセッサの動作速度の比に上限のないネットワーク（Ｐ非同期式ネットワークという）や，リンクの伝播遅延に上限のないネットワーク（Ｌ非同期式ネットワークという）では，故障診断されるプロセッサ以外のプロセッサにもリンクにも故障がなく，各プロセッサがネットワークの形状に関してどのような情報を持っていても，プロセッサ故障診断問題を解く分散アルゴリズムが存在しないことを示す．そして，第４章の後半では，プロセッサの動作速度の比と，リンクの伝播遅延の両方に上限のあるネットワーク（ＰＬ同期式ネットワークという）におけるプロセッサ故障診断問題について考察し，各プロセッサがネットワークに関して持っている情報やネットワーク全体でのプロセッサやリンクの故障状況（分布）がこの問題を解く分散アルゴリズムの存在性にどのような影響を与えるかを考察する．第４章で得られる結果を表４．１にまとめておく．

第５章では，任意の１つのプロセッサが自分に接合するリンクのうち任意の１つが故障しているかどうかを診断する問題（リンク故障診断問題）を取り上げ，第４章と同様の考察を行う．第５章で得られる結果を表５．１にまとめておく．

第３章～第５章では，プロセッサを相互結合したネットワークが既に構築されている状況で，故障したプロセッサやリンクが存在する可能性があるときに問題を解く分散アルゴリズムについて議論しているが，ネットワークを構築するときに，故障を考慮して，耐故障性のある（信頼性の高い）ネットワークを作ることも重要である．ネットワークの耐故障性の１つの尺度として，ネットワークをグラフとみなしたときの頂点連結度や辺連結度が用いら

れることが多い．これは，頂点（辺）連結度が，グラフの任意の２頂点間の頂点（辺）を共有しない通路の数を表すからである．つまり，k-頂点連結(k-辺連結）であるネットワークにおいては，どの k-1 個のプロセッサ（リンク）が故障しても，任意の２つの故障していないプロセッサ間に，故障のないプロセッサとリンクだけからなる通路が存在する．耐故障性のあるネットワークの構成法に関連する問題の１つとして，グラフの連結性に関する拡大構成問題という問題がある．これまでに，この問題に関して，多くの研究がなされている(7),(10),(14),(16),(17),(21)-(28),(34)-(37).

本論文の第６章は，関連発表論文［１］～［７］，[11] として公表したグラフの k-頂点連結拡大構成問題に関する研究をまとめたものである．グラフの k-頂点連結拡大構成問題（k-VCAP）とは，グラフ G=(V,E) と，重み関数 f：V × V → R（R は非負実数の集合）および正整数 k が与えられたとき，G が k-頂点連結になるように辺を付加するとき，付加辺の重みの和が最小となる辺集合の１つを求める問題である．また，特に，全ての辺の重みが等しいとき，重みなし k-頂点連結拡大構成問題（k-VCUAP）という．

これらの問題は，理論的に興味深いだけでなく，既存のネットワークの信頼性を，効率的に向上させるために，付加するリンクを決定する問題を形式化した問題であり，実際の応用も充分に期待できる．

以下では，グラフの連結性に関する拡大構成問題の研究の概説を行う．

１９７６年に，Eswaran，Tarjan は，グラフの連結性に関する拡大構成問題を提案し，無向グラフの２－頂点連結拡大構成問題，２－辺連結拡大構成問題や有向グラフの強連結拡大構成問題が，いずれも，ＮＰ－完全であり，辺の重みが一定の場合は，O(n+e) 時間で解けることを示した(7)．ここで，n，e はそれぞれ与えられたグラフの頂点数，辺数を表す．１９８１年に，Frederickson，Ja'ja は，無向木の２－頂点連結拡大構成問題，２－辺連結拡大構成問題およびサイクルのない有向グラフの強連結拡大構成問題が，いずれも，辺の重みを１か２の２値に制限してもＮＰ－完全であることを示した(10)．また，１９８１年に，渡辺，中村は，無向グラフの k-VCAP がＮＰ

一完全であり，k=3 のときに限り，k-VCUAP が $O(\{n+e\}^3)$ 時間で解けることを示した[35]．１９８３年に，上野，梶谷，和田は，辺の重みが一定の場合の k-辺連結拡大構成問題（k-ECUAP）に関して，グラフが無向木なら多項式時間で解けることを示し[34]，同年，梶谷，上野は基礎グラフ（underlying graph）が無向木である有向グラフの k-ECUAP に関して，$O(k \cdot n)$ 時間の解法を示した[17]．また，１９８４年に渡辺，中村は，任意の無向グラフの k-ECUAP が $O(k \cdot n^3 \cdot (e+k \cdot n) \cdot \max(k^2, n))$ 時間で解けることを示し[37]，同年，渡辺，中村，高橋は，特に k=3 の場合には $O(n^2 \cdot (n+e))$ 時間で解けることを示した[36]．

一方，グラフに付加しなければならない辺の数から，拡大構成問題の時間計算量の１つの下界が導出できる．そして，この下界にオーダ的に一致する最適な計算時間で拡大構成問題を解くアルゴリズムが存在するのは，いかなるグラフ族かを解明することは，興味深いことである．

増澤，和田，萩原，都倉は，１９８３年，１９８４年に，それぞれ，有向２進木，有向３進木の k-VCUAP の時間計算量が $\theta(k \cdot n)$ であることを示した[21],[23]．また，１９８４年に，増澤，萩原，都倉は，無向３－木（各頂点の次数が高々３の無向木），無向完全 k-木（各頂点の次数が１または k の無向木），基礎グラフが無向完全 k-木である有向グラフの k-VCUAP の時間計算量が，いずれも，$\theta(k \cdot n)$ であることを示し，基礎グラフが無向３－木である有向グラフの k-VCUAP の時間計算量が，k，n のうち，少なくとも一方が偶数のとき $\theta(k \cdot n)$ であること，及び，k，n がともに奇数のとき $\Omega(k \cdot n)$ であり，最適解より高々１つだけ多い辺を含む近似解が $O(k \cdot n)$ 時間で求まることを示した[24]．さらに，同年，増澤，萩原，都倉は，文献（21），(23) の結果を拡張し，任意の有向木の k-VCUAP の時間計算量が $\theta(k \cdot n)$ であることを示した[25]．

第６章では，有向木の k-頂点連結拡大構成問題について議論する．最初に，有向木を k-頂点連結にするために付加しなければならない辺の数の下界を求める．この付加辺数の下界から，時間計算量の１つの下界が得られる．

次に，k-頂点連結であり，かつ，k-正則（全ての頂点の入次数，出次数がともに k）なグラフの族として，k-B-グラフとよぶグラフの族を定義する．そして，任意の k-進木から k-B-グラフを辺付加によって構成するアルゴリズムを示し，これを用いて，k-進木の k-頂点連結拡大構成問題がオーダ的に最適時間で解けることを示す．また，このアルゴリズムを任意の有向木に適用する方法を示し，有向木の k-頂点連結拡大構成問題がオーダ的に最適時間で解けることを示す．

2. 諸定義

本章では，グラフ，ネットワーク，分散アルゴリズムに関する定義を行う．

単純な (simple) 有向グラフ $G=(V,E)$ は，空でない頂点集合 V と，有向辺（相異なる頂点 $u, v \in V$ の順序対で，これを $\langle u,v \rangle$ と表す）の集合 E からなる．従って，単純な有向グラフには，多重有向辺も自己ループ（$\langle v,v \rangle$ の型の有向辺）も存在しない．以下では，単純な有向グラフを単に有向グラフとよぶ．特に，$\langle u,v \rangle \in E$ なら，$\langle v,u \rangle \in E$ でもある有向グラフ $G=(V,E)$ を対称有向グラフとよぶ．

有向辺 $e=\langle u,v \rangle$ に対し，u, v をそれぞれ e の始点，終点とよぶ．有向グラフ $G=(V,E)$ の頂点 $v \in V$ に対し，v を始点とする有向辺の集合，v を終点とする有向辺の集合をそれぞれ $OE(v)$, $IE(v)$ と表す．つまり，

$$OE(v) = \{\langle v,w \rangle \in E \mid w \in V\},$$

$$IE(v) = \{\langle u,v \rangle \in E \mid u \in V\}$$

である．

単純な無向グラフ $G=(V,E)$ は，空でない頂点集合 V と，辺（相異なる頂点 $u, v \in V$ の順序のない対で，これを (u,v) と表す）の集合 E からなる．以下では，単純な無向グラフを単に無向グラフとよぶ．

無向グラフ $G=(V,E)$ の頂点 $v \in V$ の次数 $deg(v)$ を

$$deg(v) = |\{(u,v) \in E \mid u \in V\}|^{\dagger}$$

と定義する．

$G=(V,E)$ の互いに異なる頂点の系列 $\langle v_1,\ldots,v_m \rangle$ が，各 i $(1 \leq i \leq m-1)$ に対し，$(v_i,v_{i+1}) \in E$ を満たすとき，この系列を v_1-v_m 通路とよび，$m-1$ をその通路の長さという．特に，1頂点だけからなる系列も長さ0の通路とみなす．

$G=(V,E)$ と辺の集合 $E' \subseteq E$ に対し，G から E' の辺を除いた無向グラフを

---

† 集合 A に対し，$|A|$ は A の要素数を表す．

$G-E'=(V,E-E')$ と表す．

k を正整数とする．$G=(V,E)$ の $|E'|=k-1$ なる任意の辺集合 $E'\subseteq E$ に対して，$G-E'$ において任意の２頂点 u，$v\in V$ に対し，u-v 通路があるとき，G は k-辺連結であるという．特に，1-辺連結のことを，単に，連結ということがある．無向グラフの辺連結性に関して，次の命題が成立する

［命題２．１］[15] 無向グラフ $G=(V,E)$ が k-辺連結であるための必要十分条件は，$V_1\cap V_2=\phi$ † なる空でない任意の頂点集合 V_1 $(\subseteq V)$ と V_2 $(\subseteq V)$ に対し，V_1 の頂点と V_2 の頂点を結ぶ k 個の互いに辺を共有しない通路が存在することである． □

無向グラフ $G=(V,E)$ のある２頂点 $u,v\in V$ に対し，長さ２以上の u-v 通路と辺 (u,v) があるとき，閉路が存在するという．閉路のない連結な無向グラフを木という．根と呼ばれる１頂点を明示した木を根付き木という．u を根とする根付き木において，$\langle v_1(=u),\ldots,v_m\rangle$ が通路のとき，各 i $(1\leq i\leq m-1)$ に対し，v_i を v_{i+1} の親，v_{i+1} を v_i の子という．また，子のない頂点を葉とよぶ．

無向グラフ $G=(V,E)$ と $G'=(V',E')$ が $V'\subseteq V$ かつ $E'\subseteq E$ を満たすとき，G' を G の部分グラフという．特に，$V'=V$ のとき生成部分グラフという．また，特に，木である生成部分グラフを生成木という．

有向グラフ G の有向辺を（無向）辺で置換し，多重辺が生じたときはそれらを１本の辺で置換することにより得られる無向グラフを G の基礎グラフとよぶ．

［定義２．１］ 分散ネットワーク（以下，ネットワークとよぶ）は，２項組 $N=(P,L)$ で定義される．ここで，

---

† ϕ は空集合を表す．

1．P はプロセッサ（以下，PE と略記する）の集合．PE は互いに異なる識別子を持ち，PE v の識別子を id(v) と表す．また，|P| を N のサイズという．

2．L は P の相異なる要素の順序対（リンクという）の集合．リンクはいくつかのデータ（メッセージと呼ぶ）を記憶でき，$l=\langle u,v\rangle \in L$ のとき，u は l にメッセージを書き込め，v は l からメッセージを読みだせる．つまり，u から v へメッセージを送信できる． □

定義２．１より，ネットワークは各頂点が識別子を持つ有向グラフとみなせる．特に，対称有向グラフであるネットワークを全２重ネットワークとよぶ．これは，全２重リンク（同時に両方向に通信できるリンク）で PE 間が接続されたネットワークに対応する．本論文では，基礎グラフが連結である全２重ネットワークについてのみ考察するので，以下では，そのようなネットワークのことを単に，ネットワークという．また，簡単のため，その基礎グラフ（無向グラフ）を用いてネットワークを表し，無向グラフに対する用語や記法をネットワークに対しても用いる．

本論文では，次のことを仮定する．

［仮定２．１］ ネットワークには共有メモリは存在せず，PE 間の通信はメッセージ交換でのみ行われる． □

［仮定２．２］（通信の逐次性） リンクはサイズに制限のない FIFO (first-in first-out) キュー(1)とする． □

［定義２．２］ ネットワーク N=(P,L) の PE $v \in P$ は，入力ポートの集合 IP(v)={1,...,deg(v)} と出力ポートの集合 OP(v)={1,...,deg(v)} を持つ RAM (random access machine)(1) である．入力ポート，出力ポートはそれぞれ IE(v)，OE(v) のリンクと１対１に対応する．但し，各 a ($1 \leq a \leq deg(v)$)

に対し，v の入力ポート a と出力ポート a は同じ PE に接続するリンクに対応する．つまり，v の入力ポート a がリンク $\langle w,v\rangle$ に対応するならば，出力ポート a はリンク $\langle v,w\rangle$ に対応する．このとき，w を v の a-隣接 PE とよび，この対応関係（ポート対応とよぶ）を $\mu(v,a)=(v,w)$ と表す．特に，リンクの向きも表したいときは，$\mu^{i}(v,a)=\langle w,v\rangle$，$\mu^{o}(v,a)=\langle v,w\rangle$ と表す．

また，PE は，次の２つの状態を持つ．

１． 活性状態：プログラムを実行中の状態で，PE を起動すると活性状態になる．halt 命令を実行すると不活性状態に遷移する．

２． 不活性状態：プログラムを実行していない状態． □

定義２．２より，PE v が入力ポート a からメッセージを受信したとき，それを送信した PE にメッセージを返信したいときには，出力ポート a からメッセージを送信すればよいことになる．

本論文では分りやすさのために，PE のプログラムを示すときに，メモリなどの概念は直接用いず，ここでの目的に合わせて必要な道具立てをする．

［定義２．３］ PE $v\in P$ のプログラムでは，以下の通信命令を用いることができる．

１． 整数とメッセージの対の集合型の変数 M に対し，receive(M)：各 a ($1\leq a\leq deg(v)$) に対し，$\mu^{i}(v,a)$ から先頭のいくつかのメッセージが読みこまれ，a との対にして，M に記憶される．つまり，

$$M := \bigcup_{a=1}^{deg(v)} M_a$$

が実行される．ここで，

$$M_a = \begin{cases} \phi & \text{(receive(M) の実行で } \mu^i(v,a) \text{ からメッセージが読みこまれないとき)} \\ \{(a,m_1),\ldots,(a,m_k)\} & \end{cases}$$

(receive(M) の実行で $\mu^i(v,a)$ からメッセージ $m_1,\ldots,m_k$ が読みこまれたとき．但し，$\{(a,m_1),\ldots,(a,m_k)\}$ として得られるので，到着順序についての情報は利用できない)

とする．そして，読みこまれたメッセージはそのリンクの先頭から除去される．但し，PE は読みこむメッセージ数を指定できず，いくつのメッセージが読みこまれるかは前もって分らない．特に，リンク l が空でなくても，l からメッセージが読みこまれず，l の内容に変化が生じないことがある．これは，通信遅延があり，それが前もって分らないことを意味する．

２． 出力ポート a ($1 \leq a \leq deg(v)$) と変数 m に対し，send(a,m)：リンク $\mu^o(v,a)$ に最後の要素として m の内容を加える． □

［仮定２．３］(通信の無損失性) PE v が receive(M) を繰り返し行えば，任意の a ($1 \leq a \leq deg(v)$) に対し，$\mu^i(v,a)$ のメッセージはいつか必ず読みこまれる． □

［定義２．４］ ネットワーク N=(P,L) ($P=\{v_i \mid 1 \leq i \leq n\}$ とする) は，各 PE の識別子を表す $n \times 1$ 行列 $MD=(md_i)$ と，PE 間の隣接関係とポート対応を表す $n \times n$ 行列 $MA=(ma_{i,j})$ を用いて表せる．行列 MD，MA をそれぞれ識別子行列，ポート対応行列と呼び，次のように定義する．

・$MD=(md_i)$

各 i ($1 \leq i \leq n$) に対し，$md_i = id(v_i)$

・$MA=(ma_{i,j})$

各 i,j ($1 \leq i \leq n, 1 \leq j \leq n$) に対し，

$$ma_{i,j} = \begin{cases} a & (v_j \text{ が } v_i \text{ の a-隣接 PE } (1 \leq a \leq deg(v_i)) \text{ のとき}) \\ 0 & ((v_i,v_j) \notin L \text{ のとき}) \end{cases}$$ □

［仮定２．４］ 分散アルゴリズム（以下では，単にアルゴリズムという）は，ネットワークの各 PE が実行するプログラムによって表される．各 PE には，１つのプログラムが搭載されているが，本論文では，全ての PE のプログラムは同一であると仮定する．但し，PE には自分を識別するために，PE の識別子を持つ定数 ID が用意されており，プログラムで利用できる．ネットワークの形状に関する定数（形状定数とよぶ）で PE v が利用できるものとして，ID 以外に次の定数を考えることがある．

DEG ： v の次数．つまり，v の出力ポート数（入力ポート数も同じ）．

NLIST ： v の隣接 PE の識別子のリスト．NLIST[a] ($1 \leq a \leq deg(v)$) は，v の a-隣接 PE の識別子を表す．

SIZE ： ネットワークのサイズ（プロセッサ数）．

CON ： CON=k はネットワークが k-辺連結であることを意味する．

IDMATRIX ： ネットワークの識別子行列 MD．

ADMATRIX ： ネットワークのポート対応行列 MA． □

各 PE が初期状態で使える形状定数によって，解ける問題のクラスやアルゴリズムの効率は影響を受ける[20],[32],[33]．従って，初期状態で各 PE が使える形状定数を明示することは重要である．

［定義２．５］ 形状定数のうち，ID，DEG を基本情報という．初期状態で各 PE v が利用できる形状定数によって，アルゴリズムを次のように分類する．

基本情報アルゴリズム：基本情報だけを利用できる．

隣接識別子既知アルゴリズム：基本情報と NLIST だけを利用できる．†

サイズ既知アルゴリズム：基本情報と SIZE だけを利用できる．

---

† PE の次数は NLIST から分るので，NLIST が利用できれば，DEG が利用できるかどうかによる差は現れない．ここでは，簡単のため，DEG も利用できるものとする．トポロジ・アルゴリズムについても同様である．

辺連結度既知アルゴリズム：基本情報と CON だけを利用できる．

トポロジ・アルゴリズム：基本情報と IDMATRIX, ADMATRIX だけを利用できる． □

本論文では，PE やリンクに故障があるときのアルゴリズムについて考察する．但し，PE やリンクの故障に関して次のことを仮定する．

［仮定２．５］ PE v に故障があると，v は一切プログラムを実行しない．従って，故障している PE にメッセージを送信してもそのメッセージは受信されないし，故障している PE がメッセージを送信することもない．

PE が故障すると，その隣接 PE は故障が検知できるというモデルもありうるが，ここでは，そのような検知能力はないものとする． □

［仮定２．６］ リンク l に故障があると，l を使う送信ができなくなる．すなわち，l に対して send 命令を行うとメッセージは送信されたように見えるが，実際には l にメッセージは格納されない．従って，receive 命令を実行しても l からはメッセージを受信しない．但し，リンク $\langle u,v \rangle$ が故障していても，リンク $\langle v,u \rangle$ が故障しているとは限らない．

故障のあるリンクに send 命令や receive 命令を行うと，故障が検知できるというモデルもありうるが，ここでは，そのような検知能力はないものとする． □

以下では，故障している PE，リンクをそれぞれ故障 PE，故障リンクとよび，故障していない PE，リンクをそれぞれ健全 PE，健全リンクとよぶ．

リンク $\langle u,v \rangle$ または $\langle v,u \rangle$（あるいはその両方）の故障は，全２重リンク (u,v) の故障に対応する．そこで，$\langle u,v \rangle$ または $\langle v,u \rangle$（あるいはその両方）が故障しているとき，(u,v) が故障しているという．そして，ネットワークにおいて，故障している全２重リンクの総数を故障リンク数という．つまり，

$\langle u,v\rangle$, $\langle v,u\rangle$ の両方のリンクが故障していても，１個の全２重リンクの故障と数える．

［定義２．６］ アルゴリズムでは，どの PE も同じプログラムを搭載している（仮定２．４参照）ので，同じ変数（定数）名を使っている．そこで，PE v の変数（定数）x を x_v と表すことがある．プログラムで用いられる変数のうち，入力変数，出力変数として着目するものの集合を IV，OV と表す．そして，ネットワーク N=(P,L) について，

$$IV_N = \{x_u \mid x \in IV,\ u \in P\},$$

$$OV_N = \{x_u \mid x \in OV,\ u \in P\}$$

とする．ここでは，特に入出力命令は考えず，初期状態では入力変数に入力値が設定されており，終了時には，出力変数が出力値を保持しているものとする．

アルゴリズム実行開始時の IV_N の値（$IV_N{}^\circ$ と表す）が満たすべき関係を表す述語（入力述語）を α_N，アルゴリズム実行終了時の OV_N の全変数の値と IV_N の初期値 $IV_N{}^\circ$ との間で成立すべき入出力対応を与える述語を β_N とする．α_N を満たす任意の $IV_N{}^\circ$ を IV_N の初期値とし，全てのリンクが空である状況で，全ての健全 PE がアルゴリズムＡで定められたプログラムの実行を開始するとする．各 PE がプログラムの実行を開始するタイミング，各 PE の動作速度，メッセージの伝播遅延などの違いにより，様々な実行経過が考えられる．どの実行経過においても，全ての健全 PE は有限個の命令の実行後 halt 命令を実行して停止し，全ての健全 PE が停止した時点の OV_N の全変数の値と $IV_N{}^\circ$ との間で β_N が満たされているなら，Ａは分散個別問題（以下では，単に，個別問題という）$\pi_N=(\alpha_N,\beta_N)$ を解くという．

ネットワークの任意の集合を $\mathcal{N}$ とする．個別問題の集合

$$\pi = \{\pi_N=(\alpha_N,\beta_N) \mid N \in \mathcal{N}\}$$

を $\mathcal{N}$ 上の問題という．アルゴリズム Ａが，全ての $N \in \mathcal{N}$ について π_N を解くとき，Ａは π を解くという． □

どのクラスのアルゴリズムも基本情報は初期状態で利用できるので，基本情報アルゴリズムで解ける問題は，他のクラスのアルゴリズムでも解ける．また，識別子行列とポート対応行列から，各 PE で形状定数 NLIST, SIZE, CON の値が求められるので，隣接識別子既知アルゴリズム，サイズ既知アルゴリズム，辺連結度既知アルゴリズムで解ける問題は，それぞれトポロジ・アルゴリズムでも解ける．

一方，PE やリンクに故障がないときは，全ての隣接 PE 間で識別子の交換を行えば，各 PE で NLIST と同じ内容の変数が作れる．各 PE のこの変数の値を全ての PE に知らせると，各 PE で識別子行列とポート対応行列が作れる．つまり，トポロジ・アルゴリズムで解ける問題は，使える時間，メモリ量やメッセージ数に制限がなければ，基本情報アルゴリズムでも解ける．従って，PE やリンクに故障がなければ，基本情報アルゴリズム，隣接識別子既知アルゴリズム，サイズ既知アルゴリズム，辺連結度既知アルゴリズム，トポロジ・アルゴリズムで解ける問題のクラスには差がない．

しかし，PE やリンクに故障があるときには，これらのアルゴリズムで解ける問題のクラスに差がある．本論文では，以下に定義する生成木構成問題，PE 故障診断問題，リンク故障診断問題について，これらのアルゴリズムで解ける問題のクラスに差があることを示す．但し，以下では，連結なネットワーク全ての集合を$\mathcal{N}$と表す．また，PE やリンクの故障について次のように仮定する．

［仮定２．７］ アルゴリズム実行中は PE やリンクの故障が新たに生じたり，故障 PE や故障リンクが復旧したりしない． □

［定義２．７］ 生成木構成問題 π^S

$\pi^S = \{\pi^S_N=(\alpha^S_N, \beta^S_N) \mid N=(P,L)\in\mathcal{N}\}$ は次のように定義される．

各 PE の入力変数：なし．

各 PE の出力変数：NEIGHBOR（ポートの集合型）．

α^{s}_{N} : 真.

β^{s}_{N} : Live(N)=(Pl,Ll) を次のように定義される無向グラフとする.

Pl = {v∈P|v は健全 PE}

Ll = {(u,v)∈L|u,v∈Pl かつ (u,v) は故障していない}

つまり, Live(N) は, N の健全 PE の集合を頂点集合とし, 両方向に直接通信できる PE 間だけに辺が存在するようにして作った無向グラフである. Live(N) の生成木の1つを T=(P',L') とする. u(∈P') の出力変数 $NEIGHBOR_u$ は u に接合する L' のリンクに対応する u のポートの集合を記憶する. □

[定義2.8] PE 故障診断問題 π^{p}

初期状態で入力変数 $tester_v$ の値が true である PE v が正確に1個だけ存在し (この PE を診断 PE という), v の入力変数 $target_v$ の初期状態での値が a のとき, v が自分の a-隣接 PE が故障しているかどうかを判定する問題である. つまり, $\pi^{p} = \{\pi^{p}_{N}=(\alpha^{p}_{N},\beta^{p}_{N})|N=(P,L)\in\mathcal{N}\}$ は, 次のように定義される.

各 PE の入力変数 : tester (論理型), target (整数型).

各 PE の出力変数 : result (論理型).

α^{p}_{N} : ある v∈P の入力変数 tester, target は,

$tester_v$ = true, かつ, $target_v$ = a (a は $1 \leq a \leq deg(v)$ なるある整数)

を満たす. また, 任意の u∈P-{v} に対し,

$tester_u$ = false

が, 成り立つ. つまり, α^{p}_{N} は,

$$\exists v\in P\ [tester_v=true \wedge 1\leq target_v\leq deg(v) \wedge \forall u\in P-\{v\}[tester_u=false]]$$

β^{p}_{N} : 診断 PE を v とし, 初期状態で $target_v=a$ $(1\leq a\leq deg(v))$ とする. また, v の a-隣接 PE を w とする. v が健全 PE ならば, v の出力変数 $result_v$ は, 次のように定まる.

$$result_v = \begin{cases} true & (w \text{ が故障しているとき}) \\ false & (w \text{ が故障していないとき}) \end{cases}$$ □

［定義２．９］ リンク故障診断問題 π^1

初期状態で入力変数 $tester_v$ の値が true である PE v が正確に１個だけ存在し（この PE を診断 PE という），v の入力変数 $target_v$ の初期状態での値が a のとき，v が自分のポート a で接続しているリンクが故障しているかどうかを判定する問題である．つまり，$\pi^1 = \{\pi^1_N=(\alpha^1_N, \beta^1_N) | N=(P, L) \in \mathcal{N}\}$ は，次のように定義される．

各 PE の入力変数：tester（論理型），target（整数型）．

各 PE の出力変数：result（論理型）．

α^1_N：ある $v \in P$ の入力変数 tester，target は，

$tester_v$ = true，かつ，$target_v$ = a（a は $1 \leq a \leq deg(v)$ なるある整数）

を満たす．また，任意の $u \in P-\{v\}$ に対し，

$tester_u$ = false

が，成り立つ．つまり，α^1_N は，

$$\exists v \in P\ [tester_v = true \wedge 1 \leq target_v \leq deg(v) \wedge \forall u \in P-\{v\}[tester_u = false]]$$

β^1_N：診断 PE を v とし，初期状態で $target_v = a$ $(1 \leq a \leq deg(v))$ とする．v が健全 PE ならば，v の出力変数 $result_v$ は，次のように定まる．

$$result_v = \begin{cases} true & (\mu(v,a) \text{ が故障しているとき}) \\ false & (\mu(v,a) \text{ が故障していないとき}) \end{cases}$$ □

［定義２．10］ $\mathcal{N}'$ をネットワークの任意の集合，π を $\mathcal{N}'$ 上の問題とし，$N=(P,L) \in \mathcal{N}'$ とする．また，P の分割を $P_1 (\subseteq P)$，$P_2 = P - P_1$ とする．π を解くある辺連結度既知（サイズ既知）アルゴリズムＡが存在して，P_1 の PE からのメッセージを P_2 の PE が受信せず，かつ P_2 の PE からのメッセージを P_1 の PE が受信せずにＡが π を解くことがある（実行経過が存在する）

とき，πは，N で P_1，P_2 に辺連結度分解可能（サイズ分解可能）であるという．また，任意の $N=(P,L)\in\mathcal{N}'$ の P の任意の分割 $P_1(\subseteq P,\ P_1\neq\phi)$，$P_2(=P-P_1\neq\phi)$ に対して，π が N で P_1，P_2 に辺連結度分解可能（サイズ分解可能）でないとき，π は辺連結度分解不可能（サイズ分解不可能）であるという． □

［例２．１］ 次のように定義される$\mathcal{N}$上のクリーク問題$\pi^c=\{\pi^c_N=(\alpha^c_N,\beta^c_N)\mid N=(P,L)\in\mathcal{N}\}$ は，辺連結度分解可能かつサイズ分解可能な問題である．

各 PE の入力変数：m（整数型）．

各 PE の出力変数：t（論理型）．

α^c_N：入力変数 m は，

任意の $u,v\in P$ について $m_u=m_v$

を満たす．

β^c_N：変数 m の初期値を m° とする．出力変数 t は，N が頂点数 m° の完全グラフ（相異なる任意の２頂点間に辺があるグラフ）を部分グラフとして持つ（m°-クリーク[15] を持つ）とき，t=true になり，その他の場合は，t=false となる．

π^c は，$N_1=(P_1,L\cap(P_1\times P_1))$，$N_2=(P_2,L\cap(P_2\times P_2))$ それぞれに m°-クリークが存在するような $N\in\mathcal{N}$ と P の分割 $P_1(\subseteq P)$，$P_2(=P-P_1)$ に対し，N で P_1，P_2 に辺連結度分解可能であり，かつサイズ分解可能である． □

$\mathcal{N}$上の最大値探索問題，k-辺連結（$k\geq 2$）のネットワークの集合上の生成木構成問題は共に辺連結度分解不可能な問題であり，かつサイズ分解不可能な問題である．

［定義２．11］ PE やリンクに故障のあるネットワーク N=(P,L) において，v_1-v_m 通路 $\langle v_1,v_2,\ldots,v_m\rangle$ が次の条件 (1)，(2) を満たすとき，この通路

は準健全であるという．

(1) 各 i ($1<i<m$) に対し，v_i は健全 PE である．

(2) 各 i ($1\leq i<m$) に対し，(v_i,v_{i+1}) は故障していない．つまり，$\langle v_i, v_{i+1}\rangle$ も $\langle v_{i+1},v_i\rangle$ も故障していない．

準健全な v_1-v_m 通路が，さらに次の条件 (3) を満たすとき，この通路は健全であるという．

(3) v_1 も v_m も健全 PE である． □

次に，アルゴリズム実行時に通信命令を実行した PE の順序を表すため，通信順序とよぶ系列を定義する．

［定義２．12］ 通信順序は PE の集合の有限または無限系列であり，アルゴリズム実行時の通信命令の実行順序を表す．つまり，通信順序 $S=(s_1,s_2,...)$ は，アルゴリズム実行時の最初の通信命令は s_1 に属する各 PE が同時に実行し，その次の通信命令は s_2 に属する各 PE が同時に実行した（以下同様）ことを表す．但し，通信命令の実行にかかる時間は考えない． □

１つの PE は２個以上の通信命令を同時に実行できないので，アルゴリズムを実行したとき（通信順序を $S=(s_1,s_2,...)$ とする），PE v が q 回の通信命令の実行後に停止したなら，$v\in s_i$ となる i ($1\leq i$) は正確に q 個存在する．

［例２．２］ あるアルゴリズムを $N=(P,L)$（但し，$P=\{v_1,v_2,v_3\}$）で実行したときの通信命令の実行の様子を図２．１に示す．このときの通信順序 S は次のようになる．

$S=(s_1=\{v_1,v_2\},s_2=\{v_2\},s_3=\{v_1,v_2,v_3\},s_4=\{v_2\},s_5=\{v_3\},s_6=\{v_1,v_3\},$
$s_7=\{v_3\},s_8=\{v_1,v_2\},s_9=\{v_1\})$ □

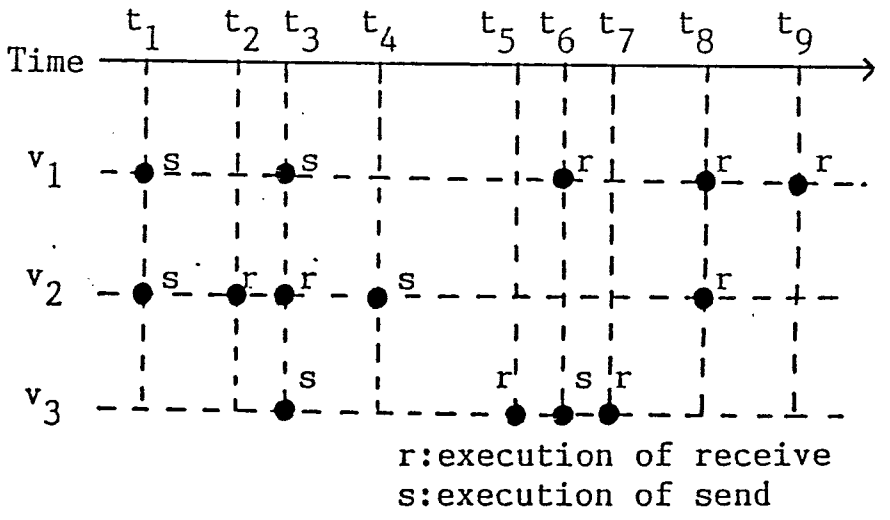


図２．１　$N=(\{v_1,v_2,v_3\},L)$ での入出力命令の実行例.

Fig. 2. 1　An example of execution in $N=(\{v_1,v_2,v_3\},L)$.

通信順序 $S=(s_1,s_2,...)$ に対し，S の部分系列 $(s_q,...,s_{q'})$ $(1\leq q\leq q')$ を $S[q,q']$ と表す．そして，$S'=(s'_1,s'_2,...)$ を通信順序（またはある通信順序の部分系列）とするとき，PE v に対し，$v\in s'_i$ $(1\leq i)$ なる i が q 個存在する（v が q 回以上通信命令を実行する）とき，v が S' に q 回現れるといい，S' に v が現れる回数を $\#(v,S')$ と表す．

［定義２．13］ ネットワーク $N=(P,L)$ に対し，ある定数Γ（$\Gamma\geq 1$）が存在して，N における任意のアルゴリズムの任意の実行の通信順序 $S=(s_1,s_2,...)$ が次の条件（PS）を満たすとき，N はＰ同期式であるといい，ΓをＰ同期定数とよぶ．

$$(PS)\ \forall u\ \forall q\ \forall q'[\ \exists v[\#(v,S[q,q'])\geq\Gamma]\ \wedge\ \exists q''[q'<q''\ \wedge\ u\in s_{q''}]\ \Rightarrow\ \#(u,S[q,q'])\geq 1]$$

また，ネットワーク N がＰ同期式でないとき，N はＰ非同期式であるという．□

つまり，Ｐ同期式ネットワーク（ΓをＰ同期定数とする）でアルゴリズムを実行したとき，任意のある PE v がΓ回通信命令を実行する間に，通信命令を実行しない PE は，それ以後も通信命令を実行しない．このことは，PE の実行時間は通信命令の実行回数に比例するという仮定のもとで，各 PE の動作速度の比の上界がΓであることを意味する．

［例２．３］ $N=(P,L)$ $(P=\{v_1,v_2,v_3\}$ とする）をＰ同期式ネットワーク（Ｐ同期定数は３）とする．N でどのようなアルゴリズムを実行しても，通信命令の実行順序は図２．１のようにはならない．なぜなら，例２．２で示した通信順序を S とするとき，v_3 は $S[5,7]$ に３回現れるが，v_2 は $S[5,7]$ に現れない．しかし，$v_2\in s_8$ であり，これは条件（PS）に反する．

しかし，図２．１で v_2 が時刻 t_8 の代りに時刻 t_7 に通信命令を実行するような実行（図２．２）は起こりうる．□

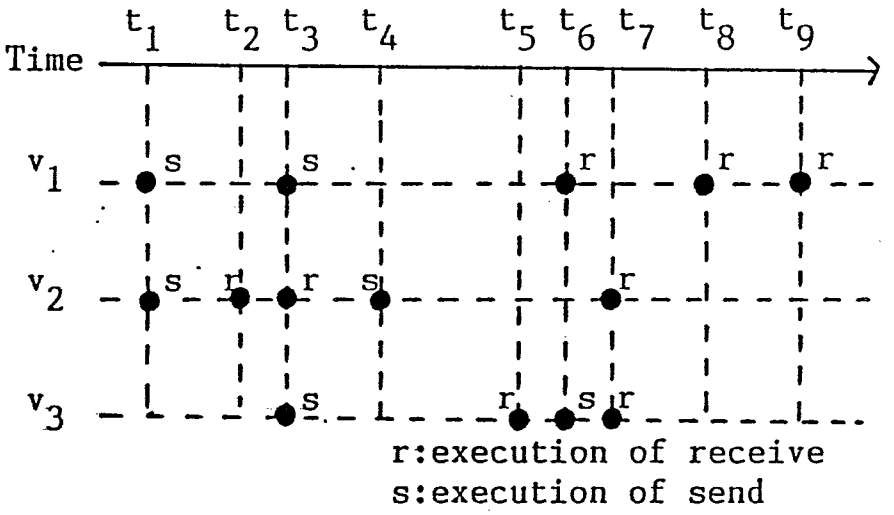


図２．２　N=({v_1,v_2,v_3},L) での入出力命令の実行例.

Fig. 2. 2　An example of execution in N=({v_1,v_2,v_3},L).

次に，L同期式ネットワークとL非同期式ネットワークを定義する．

［定義２．14］ ネットワーク $N=(P,L)$ に対し，ある定数Δ ($\Delta\geq 1$) が存在して，N における任意のアルゴリズムの任意の実行の通信順序 $S=(s_1,s_2,\ldots)$ が次の条件 (LS) を満たすとき，N はL同期式であるといい，ΔをL同期定数とよぶ．

(LS) $v\in s_q$ ($1\leq q$) で，このとき v は send 命令を実行し，隣接 PE w にメッセージ mes を送信したとする．そして，リンク $\langle v,w\rangle$ は故障していないとする．任意のある PE u が存在して，$\#(u,S[q+1,q'])\geq\Delta$ ($q+1\leq q'$) かつ $w\in s_q$，かつこのとき w が receive 命令を実行したなら，ある q'' ($q+1\leq q''\leq q'$) が存在して，$w\in s_{q''}$ かつこのとき w が receive 命令を実行し，メッセージ mes を受信している．

また，ネットワーク N がL同期式でないとき，N はL非同期式であるという． □

つまり，L同期式ネットワーク（ΔをL同期定数とする）でアルゴリズムを実行したとき，v が w に送信したメッセージは，任意のある PE がΔ回通信命令を実行してから w が receive 命令を実行すれば，遅くてもその receive 命令で受信される．このことは，PE の実行時間が通信命令の実行回数に比例するという仮定のもとで，メッセージがリンクを伝わる伝播遅延時間と PE の動作速度の比の上界がΔであることを意味する．

［例２．４］ $N=(P,L)$ (但し，$P=\{v_1,v_2,v_3\}$ とする) をL同期式ネットワーク（L同期定数は３）とする．あるアルゴリズムを N で実行したときの通信命令の実行順序が図２．１のようになったとし，例２．２で示した通信順序を S とする．また，時刻 t_4 に v_2 は send 命令で v_1 にメッセージ mes を送信し，リンク $\langle v_2,v_1\rangle$ は故障していないとする．v_3 は $S[5,8]$ に３回現れ，時刻 t_8 に v_1 は receive 命令を実行しているので，条件 (LS) よ

り，v_1 は時刻 t_6 か t_8 の receive 命令の実行で v_2 からのメッセージ mes を受信する． □

ネットワークが，Ｐ同期式かつＬ同期式のとき，ＰＬ同期式であるという．また，Ｐ非同期式かつＬ非同期式のとき，非同期式であるという．

Ｐ同期式やＬ同期式のネットワークで問題を解くとき，Ｐ同期定数やＬ同期定数を初期状態で使えるかどうかによって，問題を解くアルゴリズムが存在したり，存在しなかったりする．そこで，各 PE が利用できる定数として，次のものを考える．

［定義２．15］ 同期に関する定数（時間定数とよぶ）で，PE が利用できる定数として PSYNC，LSYNC があり，それぞれＰ同期定数，Ｌ同期定数を表す．□

但し，PSYNC，LSYNC について，次の仮定を置く．

［仮定２．８］ ネットワークがＰ同期式（Ｌ同期式）ならば，各 PE は初期状態で時間定数 PSYNC（LSYNC）を利用できる． □

３．非同期式ネットワークにおける生成木構成問題

本章では，PE やリンクが故障しているかもしれない非同期式ネットワークで，生成木構成問題 π^S を解くアルゴリズムについて考察する．表３．１に，本章の結果をまとめる．

但し，非同期式ネットワークの場合，１回の receive 命令の実行で受信できるメッセージは高々１個であると制限しても一般性は失われないので，本章では，次の仮定の下で議論する．

［仮定３．１］ １回の receive 命令で受信できるメッセージは高々１個である． □

表３．１　非同期式ネットワークでπ^Sを解く分散アルゴリズム

<table>
<tr><th>アルゴリズム
PE の故障</th><th>基本情報
アルゴリズム</th><th>サイズ既知
アルゴリズム</th><th>辺連結度既知
アルゴリズム</th><th>隣接識別子既知
アルゴリズム</th><th>トポロジ・
アルゴリズム</th></tr>
<tr><td>故障 PE が存在する
可能性があるとき</td><td colspan="5">故障 PE が高々１個で，故障リンクがなくても，存在しない．
（定理３．１）</td></tr>
<tr><td>故障 PE が
存在しないとき</td><td>故障リンク数が
高々１でも存在しない
（定理３．２）</td><td>SST
（故障リンク数に
かかわらず解く）
（定理３．７）</td><td>故障リンク数が
⌈CON/2⌉ 以上のとき
存在しない
（定理３．２）
故障リンク数が
⌈CON/2⌉-1 以下
のとき：
CST （定理３．３）</td><td colspan="2">NST
（故障リンク数にかかわらず解く）
（定理３．９）</td></tr>
</table>

３．１　故障プロセッサがあると生成木構成問題が解けないこと

次の定理より，PE が故障している可能性のある非同期式ネットワークでは，各 PE がネットワークの形状に関していかなる情報を持っていても生成木構成問題は解けない．

［定理３．１］ ネットワークが非同期式で，PE が故障している可能性があるとき，故障 PE が高々１個で，故障リンクが存在しなくても，生成木構成問題を解くトポロジ・アルゴリズムは存在しない．

（証明） リンクに故障がなく，故障 PE が高々１個のときに生成木構成問題 π^S を解くトポロジ・アルゴリズムＡが存在すると仮定する．

Ａは，PE w が PE v だけに隣接しているようなネットワーク N=(P,L) で，故障リンクがなく，故障 PE が w だけのときに，π^S を解く．このとき，w が故障しているので，(v,w) は Live(N) の生成木の辺でない．従って，Ａの停止時に $a \notin NEIGHBOR_v$ (a は v の (v,w) に対応するポートとする．つまり，w は v の a-隣接 PE とする）が成り立つ．また，w が故障しているので，Ａを実行したとき，v は w からメッセージを受信しない．

次に，故障 PE も故障リンクも存在しない N で，Ａを用いて π^S を解くことを考える．N が非同期式なので，w の動作が非常に遅い場合がある．このとき，w 以外の各 PE は，w だけが故障している N で π^S を解いたときと同じ動作（同じ時刻に同じ命令を実行し，しかも，それが receive 命令のときは同じポートから同じメッセージを受信する）をし，v の停止時に $a \notin NEIGHBOR_v$ が成り立つ場合がある．しかし，w は v だけに隣接しているので，(v,w) は Live(N) (=N) の任意の生成木の辺である．このことより，Ａが π^S を解くことに矛盾が生じる． □

３．２　辺連結度既知アルゴリズム

定理３．１より，ネットワークが非同期式で，PE が故障している可能性があるとき，生成木構成問題π^S を解くトポロジ・アルゴリズムは存在しない．また，定理３．１と同様にして，リンクの故障のためにネットワークが分断されると，π^S を解くトポロジ・アルゴリズムが存在しないことを示せる．そこで，本章の以下の部分では，次の２つの条件が成り立つときにπ^S が解けるかどうかを議論する．そして，本章の定理や補題では，特に断らないが，次の２つの条件が成立することを前提としている．

［条件３．１］ 故障 PE は存在しない． □

［条件３．２］ 任意の２つの PE u，v に対し，健全な u-v 通路が存在する． □

また，条件３．１，３．２を満たす範囲内で，何個のリンクの故障にかかわらずアルゴリズムが問題を解くかを表すために，アルゴリズムの耐辺故障度を次のように定義する．

［定義３．１］ πをネットワークの集合$\mathcal{N}'$ 上の問題とし，Ａをアルゴリズムとする．任意の $N \in \mathcal{N}'$ に対し，N が条件３．１，３．２を満たし，かつ，故障リンク数が h 以下ならば，Ａがπ_N を解くとき，Ａはπを耐辺故障度 h で解くという． □

任意の正整数 k に対し，k-辺連結なネットワーク全ての集合を$\mathcal{N}_k$ と表す．ここでは，まず，任意の正整数 k に対し，$\mathcal{N}_k$ 上の辺連結度分解不可能な問題を解く耐辺故障度「CON/2」の辺連結度既知（CON=k）のアルゴリズム

は存在しないことを示す．次に，$\mathcal{N}$で生成木構成問題（辺連結度分解不可能な問題の１つ）を解く耐辺故障度「CON/2⌉-1 の辺連結度既知のアルゴリズム CST を示す．このことから，これら２つの値はこれ以上改善できないことが分る．

３．２．１　耐辺故障度「CON/2⌉ のアルゴリズムの存在しないこと

［定理３．２］ 任意の正整数を k とし，πを $\mathcal{N}_k$ 上の辺連結度分解不可能な任意の問題とする．このとき，πを解く耐辺故障度「CON/2⌉ の辺連結度既知（CON=k）のアルゴリズムは存在しない．

（証明） 耐辺故障度「CON/2⌉ の辺連結度既知（CON=k）のアルゴリズムＡが存在すると仮定し，矛盾を導く．

ネットワーク N_1, N_2 を次のように定義する．

$N_1 = (P_1=\{u_i \mid 0\leq i\leq k+1\},\ L_1=\{(u_i,u_j) \mid 0\leq i<j\leq k+1\})$,

$N_2 = (P_2=\{v_i \mid 0\leq i\leq k+1\},\ L_2=\{(v_i,v_j) \mid 0\leq i<j\leq k+1\})$

つまり，N_1, N_2 はともに PE 数が k+2 で，任意の相異なる PE 間にリンクが存在するネットワークである．このとき，N_1, $N_2\in\mathcal{N}_k$である[15]．また，N_1, N_2 それぞれのリンクの部分集合 L_1', L_2' を次のように定める．

$L_1' = \{(u_{2i},u_{2i+1}) \mid 0\leq i\leq \lceil k/2\rceil-1\} \subseteq L_1$,

$L_2' = \{(v_{2i},v_{2i+1}) \mid 0\leq i\leq \lceil k/2\rceil-1\} \subseteq L_2$

仮定より，Ａは $\mathcal{N}_k$ でπを解く耐辺故障度「CON/2⌉ の辺連結度既知のアルゴリズムであり，各 j (j=1,2) に対して，N_j-L_j' は連結なので，L_j' のリンクだけが全て両方向とも故障している N_j (N_j^{er} と表し，入力変数の初期値を IV_j° とする)で個別問題π_{N_j} を解く．このとき，L_j' のリンクは全て両方向とも故障しているので，N_j の PE は，L_j' のリンクからはメッセージを受信しない．

次に，ネットワーク N=(P,L) を次のように定める.(図３．１)

$P = P_1 \cup P_2$

$L = (L_1 - L_1') \cup (L_2 - L_2') \cup L_{12}$

但し，$L_{12} = \{(u_i, v_i) \mid 0 \leq i \leq 2(\lceil k/2 \rceil - 1) + 1\}$ とする.

このとき，$N \in \mathcal{N}_k$ である(文献 (24) と同様の手法を用いて証明できる).

また，N のポート対応を次のように定める.

各 i $(0 \leq i \leq k+1)$ に対し，

$$\mu_N(u_i, a) = \begin{cases} \mu_{N'}(u_i, a) & (\mu_{N'}(u_i, a) \notin L_1' \text{ のとき}) \\ (u_i, v_i) & (\mu_{N'}(u_i, a) \in L_1' \text{ のとき}) \end{cases}$$

$$\mu_N(v_i, a) = \begin{cases} \mu_{N'}(v_i, a) & (\mu_{N'}(v_i, a) \notin L_2' \text{ のとき}) \\ (v_i, u_i) & (\mu_{N'}(v_i, a) \in L_2' \text{ のとき}) \end{cases}$$

どの辺も故障していない N (P_1, P_2 の入力変数の初期値はそれぞれ $IV_1{}^\circ$，$IV_2{}^\circ$ に等しいとする) に対してＡを実行したとする.

N はＬ非同期式なので，L_{12} の各リンクの伝播遅延時間が非常に大きい場合がある．このとき，各 PE が $N_1{}^{er}$, $N_2{}^{er}$ で π を解いたときと同じ動作(定理３．１の証明参照) を行い，L_{12} のリンクからメッセージを受信せずにＡが停止する場合がある．Ａは $\mathcal{N}_k$ で π を解く耐辺故障度「CON/2」のアルゴリズムなので，Ａは辺に故障のない N で個別問題 π_N を解いたことになるが，L_{12} のリンクからメッセージを受信していないので，P_1, P_2 の PE はそれぞれ P_2, P_1 の PE から，メッセージを受け取らずに π_N を解いたことになる．このことは，π が辺連結度分解不可能であることに矛盾する． □

３．２．２　耐辺故障度「CON/2」-1 のアルゴリズム CST

定理３．２で，任意の正整数 k に対し，$\mathcal{N}_k$ 上の辺連結度分解不可能な問題を解く耐辺故障度「CON/2」の辺連結度既知 (CON=k) のアルゴリズムは存在しないことを示した．この節では，$\mathcal{N}$ で生成木構成問題（辺連結度分解不

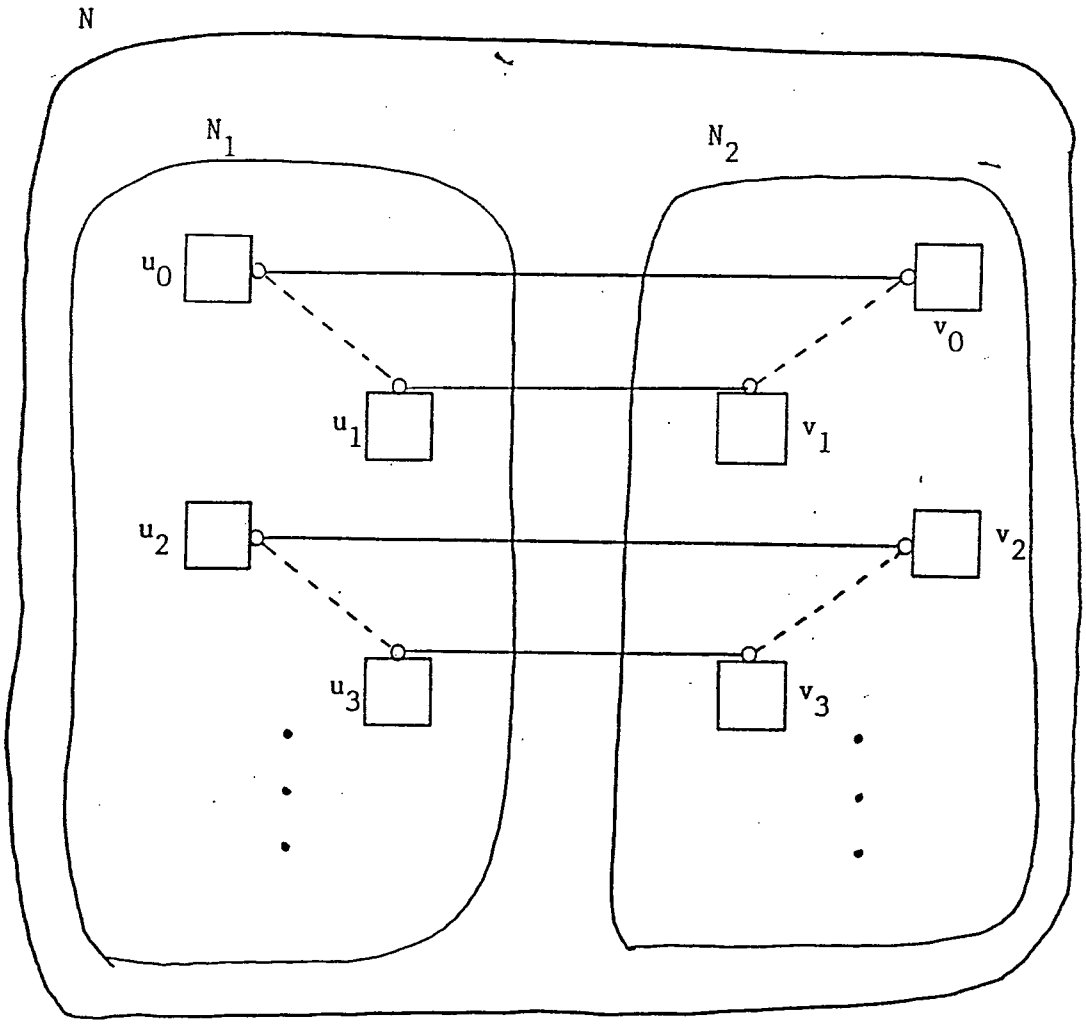


図３．１　N_1, N_2 から作られるネットワーク N.

Fig. 3. 1 The network N constructed from N_1 and N_2.

可能な問題の１つ）π^S を解く耐辺故障度 ⌈CON/2⌉-1 の辺連結度既知のアルゴリズム CST を示す．アルゴリズムの分りやすさのために，まず，リンクが故障しているなら，必ず両方向とも故障しているという仮定（仮定３．２）の下でπ^S を解くアルゴリズム CST を示す．CST を簡単に改良すれば，片方向だけが故障しているリンクが存在するネットワークで，π^S を解くアルゴリズムが得られる．(３．５節参照)

［仮定３．２］ N=(P,L) の任意の (u,v)∈L に対し，〈u,v〉 が故障していれば，〈v,u〉 も故障しているものとする． □

CST は，生成木として，識別子が最大の PE を根とする根付き生成木を求める．

１． 起動されて活性状態になった PE は receive 命令を実行する．このとき，メッセージを受信しないか，または，自分より小さい識別子を持つメッセージ MERGE を受信した場合は，自分が王になり，自分の識別子を持つ MERGE を各ポートに送信し，隣接プロセッサを自分の家来にしようとする．また，自分の識別子より大きい識別子を持つ MERGE を受信した場合は２．を実行する．†

２． MERGE を受信した PE は，MERGE の持つ識別子 p と自分の王（王の場合は自分自身）の識別子 k_id を比較し，p>k_id の場合は，p の家来になり，MERGE を送信してきた PE を自分の新たな（根付き木での）親とする．

---

† 文献 (12)，(32) 等のアルゴリズムでは，最初は全ての PE は asleep であり，１個以上の PE が自発的に awake になりアルゴリズムの実行を開始し，他の PE はメッセージを受信すると awake になりアルゴリズムの実行を開始するというモデルを用いている．CST の起動時にメッセージを受信しない PE は自発的に awake になる場合に対応し，MERGE を受信する PE は他からのメッセージによって awake になる場合に対応する．

p≦k_id のときには，その MERGE を無視する．

３． 家来も自分の王の識別子を持つ MERGE を親以外の全ての隣接 PE に送り，それらを自分の王の家来にしようとする．

４． 王は自分の王国（王とその家来からなる）と王国以外を結んでいる可能性があるか，故障している可能性がある辺の総数を数えておき，その数が CON-1 以下になったときアルゴリズムを終了する．

［アルゴリズム CST］

(known-Connectivity algorithm for constructing a Spanning Tree)

CST は，生成木として根付き木を求め，各 PE は変数 par に親に通じるポートを，また変数 SON に子に通じるポートの集合を求める．各 PE が実行するアルゴリズムを表３．２に示す．つまり，起動された PE は，まず，receive 命令を実行し，受信メッセージによって，(0,0) または (1,0) を実行する．表３．２から決まるメッセージに対する一連の処理（表３．２の１つの枠に対応）をメッセージに対する処理という．そして，(0,0) か (1,0) の一連のプログラムの実行を終えてから，再び receive 命令を実行し，受信メッセージと変数 st の内容から決まるメッセージに対する処理を行う．それ以後は同様のことを halt 命令を実行するまで繰り返す．表３．２に現れる変数等の型を以下に示す．

```
type process=0..maxint;{PE の識別子}
     port=1..maxdegree;{PE のポート}
     status=(king,pending,subject);
```

```
const ID=PE の識別子(非負整数);
      DEG=PE の次数;
      CON=k のときネットワークは k-辺連結;
      R={1,...,DEG};

var par:port; {親に通じるポート}
    SON:set of port; {子に通じるポートの集合}
    st:status; {下の説明参照}
    k_id:process; {自分の王の識別子}
    x:port;
    p,q:process;
    n_rep,m:integer; {下の説明参照}
```

変数 st と n_rep について説明を加える.

st：活性状態の PE の状況を示し， st の値は king（自分が王である), pending（自分の王が替わる．自分が家来になることによって新たな王が保持している変数 n_rep（後述）の値が更新されるのを待っている), subject（家来である）のいずれかである.

n_rep ：王が，自分の王国（王の識別子と同じ k_id を持つ PE 全てからなる）の PE の入力ポートの総数（次数の総和）から王の識別子を持つメッセージ MERGE（後述）または MERGED（後述）を受信したことのあるポートの数を引いた数を記憶する．これらのメッセージを受信したポートは，同じ王国の PE に通じている．従って，n_rep の値は王国と王国以外を結んでいる可能性または故障している可能性のある辺の数を表す．ネットワークが CON-辺連結であることより，n_rep の値が CON-1 以下になったとき，全プロセッサがこの王国に属したことを意味する（命題２．１）ので，王は生成木が完

表3．2　アルゴリズム CST

x∈R から受信したメッセージ	起動時	st		
		king	pending	subject
なし	new_king (0,0)			
《MERGE,p》	p<ID → (1,0) new_king p>ID → subjugated	p>ID → (1,1) subjugated p=ID → dec_n_rep	p>k_id → (1,2) subjugated p=k_id → s(par,《REP,k_id》)	p>k_id → (1,3) subjugated p=k_id → s(par,《REP,k_id》)
《MERGED, p,q,m》		p=ID → (2,1) SON:=SON∪{x} n_rep:=n_rep-1+m s(x,《ACK,ID,q》)		p=k_id → (2,3) SON:=SON∪{x} s(par,《NEW,p,q,m》)
《NEW,p,q,m》		p=ID → (3,1) n_rep:=n_rep-1+m s(x,《ACK,ID,q》)		p=k_id → (3,3) s(par,《NEW,p,q,m》)
《ACK,p,q》			p=k_id∧q=ID → (4,2) st:=subject s(R-{par}, 《MERGE,k_id》)	p=k_id → (4,3) s(SON,《ACK,p,q》)
《REP,p》		p=ID → (5,1) dec_n_rep		p=k_id → (5,3) s(par, 《REP,p》)
《HALT》				s(SON,《HALT》) (6,3) halt

(注1) 表中に現れる手続き new_king, subjugated, dec_n_rep は，次の通りである．

```
proc new_king;
   k_id:=ID
   SON:={}
   n_rep:=DEG
   st:=king
   s(R,《MERGE,ID》)

proc subjugated;
   k_id:=p
   par:=x
   SON:={}
   st:=pending
   s(x,《MERGED,p,ID,DEG-1》)

proc dec_n_rep;
   n_rep:=n_rep-1
   n_rep<CON →
      s(SON,《HALT》)
      halt
```

(注2) A → B は条件 A が成立するときに B を実行することを意味する．

(注3) s(x,M) (x はポート，M はメッセージ)は send(x,M) の略記である．また，s(X,M) (X はポートの集合，M はメッセージ) は，X の各ポートに対し，任意の順で M を送信することを表す．

(注4) 空白の欄および表中に現れない条件の場合は，プロセッサは何もしない (no operation)．

成したことが分る.

次に，CST で用いられるメッセージの説明を行う．以下では，p，q は共に PE の識別子を，m は非負整数を表すとする．《，》で囲まれたのが１つのメッセージである．メッセージを送信，受信する PE をそれぞれ送り手，受け手とよぶ.

《MERGE,p》：王 p またはその家来によって，送信される．「p の王国に加われ.」を意味する.

《MERGED,p,q,m》：《MERGE,p》を受信した q がその MERGE の送り手(親) に送信する．「p の家来になる．q が子として持ちえるのは，高々 m=deg(q)-1 個† である.」を意味する.

《NEW,p,q,m》：《MERGED,p,q,m》または《NEW,p,q,m》の受け手が親に送信する．「q が p の王国に加わるというメッセージを受信した．q の子の数は高々 m である.」を意味する.

《ACK,p,q》：王 p によって発送され，以下，伝達されていく．「q を p の新しい家来として認める.」を意味する.

《REP,p》：《MERGE,p》の受け手が，既に p の家来になっていた場合に，親に送信する．「p の王国の PE に通じるポートを，新たに発見した.」を意味する.

---

† q は《MERGE,p》の送り手以外の全隣接 PE を生成木における自分の子とする可能性がある.

《HALT》：生成木が求まったので，作業を終えることを知らせる． □

3.2.3 アルゴリズム CST の正当性

以下では，$\mathcal{N}_k$ の任意のネットワーク N=(P,L) で CST (CON=k とする) を実行した場合を考える．但し，$P=\{v_1,v_2,...,v_n\}$ (PE の識別子を v_1, $v_2,...,v_n$ とする) とし，$v_1<v_2<...<v_n$ とする．また，実時刻 (以下では，単に時刻という) を考える．時刻には全順序関係があり，時刻 t' が時刻 t 以前のとき，$t'\leq t$ とかく．また，以下の記法を用いる．

a_i^t ：時刻 t における $v_i(1\leq i\leq n)$ の変数 a の値．

$\nu(i,e)$ ：e は v_i と他の PE を結んでいるリンクで，$\nu(i,e)$ はリンク e に対応する v_i のポート．

j $(1\leq j\leq n)$ と時刻 t に対し，

$S^t(j) = \{v_i\in P \mid \exists t'(\leq t)[k_id_i^{t'}=v_j \wedge st_i^{t'}=subject]\}$

$P^t(j) = \{v_i\in P \mid \exists t'(\leq t)[k_id_i^{t'}=v_j \wedge st_i^{t'}=pending]\}$

$M^t(j) = \{v_i\in P \mid \exists t'(\leq t)$[時刻 t' に《MERGED,$v_j$,$v_i$,deg($v_i$)-1》または《NEW,$v_j$,$v_i$,deg($v_i$)-1》を v_j が受信した (receive 命令で読みこんだ)]}

$KS^t(j) = S^t(j) \cup \{v_j\}$

$KM^t(j) = M^t(j) \cup \{v_j\}$

つまり，$S^t(j)$は，時刻 t 以前に $k_id=v_j$ かつ st=subject となったことのある PE の集合である．

《MERGED,v_j,v_i,m》は，$k_id=v_j$ かつ st=pending の PE v_i によって送信される (手続き subjugated)．また，《NEW,v_j,v_i,m》は，《MERGED,v_j,v_i,m》の受け手から v_j まで順々に par へ伝えられる．従って，

$M^t(j) \subseteq P^t(j)$

が成立する．また，v_i は k_id=v_j のときに，《ACK,v_j,v_i》を受信して st:= subject を行う((4,2))．そして，《ACK,v_j,v_i》は，《MERGED,v_j,v_i,m》または《NEW,v_j,v_i,m》を受信した v_j によって送信され，v_i まで子から子へと伝えられる．従って，

$S^t(j) \subseteq M^t(j)$

が成立する．

PE の集合 $P'(\subseteq P)$，$P''(\subseteq P)$ に対し，P' に属する PE と P" に属する PE を結ぶ（全２重）リンクの集合を LB(P',P") と表す．つまり，

$LB(P',P'') = \{(u,v) \in L \mid u \in P',\ v \in P''\}$

とする．このとき，$P' \cap P'' \neq \phi$ でも構わない．

先に説明したように，変数 n_rep は，王国と王国以外を結んでいる可能性または故障している可能性のある辺の数を表す変数であり，次の補題が成立する．

［補題３．１］ st_j^t=king であり，時刻 t において v_j が時刻 t までに受信した全てのメッセージに対する処理を終えているならば，

$n_rep_j^t \geq |LB(KM^t(j), P-KM^t(j))|$

が成り立つ．

（証明） st:=king は (0,0) または (1,0) の new_king でのみ実行されるので，起動時に v_j は new_king で，n_rep を deg(v_j) に初期設定したはずである．また，王 v_j が《MERGED,v_j,q,m》か《NEW,v_j,q,m》を受信すると，m(=deg(q)-1) から１だけ減じた値を n_rep に加える((2,1)，(3,1))．これは，q が v_j の王国に新たに加わったので，v_j の王国外に通じる可能性があるリンクとして，q に接合するリンクのうち，(q,q') (q' は q の親) 以外の m 個のリンクが加わり，それまで王国外に通じる可能性があると考えられていた (q,q') が王国内のリンクであることが分ったためである．従って，

初期設定から時刻 t までに v_j の n_rep に加えられた値の和 INC_j^t は，

$$INC_j^t = \sum_{v \in M^t(j)} (deg(v)-2) \tag{3.1}$$

一方，(1,1) と (5,1) の dec_n_rep で，王 v_j は《MERGE,v_j》か《REP,v_j》を受信すると，n_rep を１だけ減じる．

ところで，REP の送信は (1,2)，(1,3)，(5,3) で行なわれるが，(5,3) は REP の中継なので，(1,2) または (1,3) で送信される REP（送信した PE を発送元という）について考える．《REP,v_j》は，《MERGE,v_j》の受け手が，k_id=v_j かつ，st=pending か subject の場合に送信する．これらの PE u は，《REP,v_j》を送信するより前に，k_idを v_j に更新したとき((1,0)，(1,1)，(1,2)，(1,3) の subjugated)，《MERGED,v_j,u,deg(u)-1》を送信している．REP も MERGED も，同じ通路を通って v_j まで伝わるので，仮定２．２（通信の逐次性）より，時刻 t までに v_j が受信した《REP,v_j》の発送元は $M^t(j)$ に属する．

従って，時刻 t までに (1,1) か (5,1) の dec_n_rep で，王 v_j の n_rep から引かれた値の和を，DEC_j^t とすると，DEC_j^t は，$KM^t(j)$ の PE が，時刻 t までに受信した《MERGE,v_j》の数以下である．

一方，《MERGE,v_j》の送信は k_id=v_j である king または subject の PE によってのみ行われる ((0,0)，(1,0)，(4,2))ので，時刻 t までに《MERGE,v_j》を送信する PE は $KS^t(j)$ に属する．また，$M^t(j)$ の PE は《MERGE,v_j》を初めて受信して k_id が更新されるときは((1,0)，(1,1),(1,2)，(1,3))，《REP,v_j》を送信しないし，$S^t(j)$ の PE は par に対しては《MERGE,v_j》を送信しない((4,2))．従って，

$$DEC_j^t \leq 2|LB(KS^t(j),KS^t(j))| + |LB(KS^t(j),M^t(j)-S^t(j))| - |S^t(j)| - |M^t(j)| \tag{3.2}$$

が成り立つ．

また，$M^t(j)$ の PE は，時刻 tまでに，《MERGE,v_j》を受信しているので，$KS^t(j)$ の１個以上の PE とリンクで結ばれている．従って，

$$|M^t(j)| - |S^t(j)| \leqq |LB(KS^t(j), M^t(j)-S^t(j))| \tag{3.3}$$

が成り立つ．式 (3.2)，(3.3) より，

$$DEC_j{}^t \leqq 2|LB(KM^t(j), KM^t(j))| - 2|M^t(j)| \tag{3.4}$$

$$(\because\ |LB(KS^t(j), KS^t(j))| + |LB(KS^t(j), M^t(j)-S^t(j))| \leqq |LB(KM^t(j), KM^t(j))|)$$

が成り立つ．

v_j の n_rep の初期値が $deg(v_j)$ であること及び式 (3.1)，(3.4) より，$n_rep_j{}^t$ に関して次式が成立する．

$$\begin{aligned} n_rep_j{}^t &= deg(v_j) + INC_j{}^t - DEC_j{}^t \\ &\geqq \sum_{v \in KM^t(j)} deg(v) - 2|M^t(j)| - (2|LB(KM^t(j), KM^t(j))| - 2|M^t(j)|) \\ &= |LB(KM^t(j), P-KM^t(j))| \end{aligned}$$

$$(\because\ \sum_{v \in KM^t(j)} deg(v) = 2|LB(KM^t(j), KM^t(j))| + |LB(KM^t(j), P-KM^t(j))|)$$

□

［補題３．２］ (1,1) または (5,1) の dec_n_rep で halt 命令を実行するのは v_n (識別子が最大の PE) だけである．

(証明) v_i $(1 \leqq i < n)$ が，dec_n_rep で halt 命令を実行したと仮定する．halt 命令を実行したとき，v_i において st=king かつ n_rep<k が成立したはずである．一方，$v_n > v_i$ より，任意の t について，$v_n \in P-KM^t(i)$ である．従って，$v_i \in KM^t(i)$，N が k-辺連結であること，および命題２．１より，

$$|LB(KM^t(i), P-KM^t(i))| \geqq k$$

である．このことと補題３．１より，

$$n_rep_i{}^t \geqq k$$

となる．これは，仮定に矛盾する． □

［補題３．３］ 時刻 t において，v_n が時刻 t までに受信した全てのメッセージに対する処理を終えているならば，$KM^t(n)$ は以下の性質を持つ．

(a) $\forall v_i(\in M^t(n)),\ \forall v_j(\in KM^t(n))$

$$[par_i{}^t = \nu(i,(v_i,v_j)) \Leftrightarrow SON_j{}^t \ni \nu(j,(v_i,v_j))]$$

(b) グラフ

$$G^t(n) = (KM^t(n), L^t(n) = \{(v_i,v_j) \in M^t(n) \times KM^t(n) \mid par_i{}^t = \nu(i,(v_i,v_j))\})$$

は木である．

（証明）(a) v_i が《MERGE,v_n》を初めて受信したとき，それを受信したポートが $\nu(i,(v_i,v_j))$ ならば，par:= $\nu(i,(v_i,v_j))$ を実行し，MERGED を新たな par に送信する ((1,0), (1,1), (1,2), (1,3) の subjugated)．v_j の k_id は v_n なので，この MERGED を受信したポート $\nu(j,(v_i,v_j))$ を SON に加える ((2,1), (2,3))．k_id が一度 v_n になるとそれ以後に par の値の更新も SON の要素の削除も行われない．そして，j=n のときは，v_j がこの MERGED を受信した時刻に，また，j≠n のときは，この MERGED の内容が NEW によって v_n に伝わった時刻に，v_i は $M^t(n)$ の要素になる．従って，(a) が成立する．

(b) $v_i \in M^t(n)$ かつ $par_i{}^t = \nu(i,(v_i,v_j))$ とする．《MERGE,v_n》を最初に par から受信したことから，v_j は時刻 t 以前に《MERGE,v_n》を送信したので，$v_j \in KS^t(n) \subseteq KM^t(n)$ である．従って，$G^t(n)$ の定義より，

$$|L^t(n)| = |KM^t(n)| - 1$$

である．また，v_n 以外の PE は《MERGE,v_n》を受信後に，《MERGE,v_n》を送信したのだから，$M^t(n)$ の PE から par を順にたどっていくと v_n に達する．$|L^t(n)| = |KM^t(n)| - 1$ かつ任意の PE から v_n への通路があることより $G^t(n)$ は木である[15]． □

［定理３．３］ 辺連結度既知アルゴリズム CST は任意のネットワーク N∈$\mathcal{N}$ で耐辺故障度 ⌈CON/2⌉-1 で生成木構成問題を解く．

（証明） 補題３．２より，(1,1) または (5,1) で halt 命令を実行するの

は v_n（識別子が最大の PE）だけである．v_n が (1,1) または (5,1) の処理を終える時刻を t とする．$KM^t(n)=P$ が成立することが，補題３．２と同様に示せる．また，補題３．３より，変数 par_i^t, SON_i^t $(1 \leq i \leq n)$ はそれぞれ N の生成木における v_i の親，子に通じるポート（の集合）を保持している．一方，$v_j \in M^t(n)$ $(1 \leq j < n)$ より，v_n は t 以前に《MERGED,v_n,v_j,m》か《NEW,v_n,v_j,m》を受信し，生成木のリンクを用いて《ACK,v_n,v_j》を送信している．ACK も HALT も生成木のリンクを用いて送信されるので，v_j は《HALT》より前に《ACK,v_n,v_j》を受信し（通信の逐次性），(4,2) を実行するので，《HALT》を受信するときは，st=subject が成立する．従って，v_j は生成木の子に《HALT》を送信し，halt 命令を実行する((6,3))．以上のことから，v_n が (1,1) または (5,1) で halt 命令を実行すれば，CST は生成木を求めて停止する．

つぎに，v_n が (1,1) または (5,1) で halt 命令を実行することを示す．ここで，故障リンク数を l $(\leq \lceil k/2 \rceil -1)$ とする．k_id=n かつ st=subject が成立する PE は，par 以外の全ポートに《MERGE,v_n》を送信する．N がk-辺連結であることと命題２．１より，時刻 t' において $KS^{t'}(n) \neq P$ なら，$|LB(KS^{t'}(n), P-KS^{t'}(n))| \geq k > l$ なので，$P-KS^{t'}(n)$ のうち少なくとも１つは，いつかは《MERGE,v_n》を受信する(通信の無損失性)．従って，$KS^{t''}(n)=P$ となる時刻 t" が存在し，いつかは全ての PE が par 以外の全ポートに《MERGE,v_n》を送信する．故障辺数が l なので，決して受信されない《MERGE,v_n》は $2l$ $(\leq 2(\lceil k/2 \rceil -1) \leq k-1)$ 個である．従って，いつかは，受信されない《MERGE,v_n》は k-1 個以下になり，v_n の n_rep が k-1 以下になることを補題３．１の証明と同様に示せる．従って，v_n が (1,1) または (5,1) で halt 命令を実行し，CST は生成木を求めて停止する．□

（注１） 故障 PE が存在するとき，または，故障リンク数が $\lceil CON/2 \rceil$ 以上のときは，CST は停止しない．

（注２） 故障 PE が存在せず，故障リンク数が ⌈CON/2⌉-1 以下なら，条件３．２は必ず満たされる．

（注３） CON がネットワークの辺連結度に関する正しい情報を保持しておらず，CON の値が，ネットワークの辺連結度よりも大きい場合は，故障リンク数が ⌈CON/2⌉-1 以下であっても，CST は生成木を求める前に停止してしまうことがある．また，故障リンク数が ⌈CON/2⌉ 以上のときは，停止しないことも，生成木を求める前に停止してしまうこともある．

3.2.4 アルゴリズム CST の評価

ここでは，アルゴリズム CST を評価する．評価基準として，PE の領域計算量とネットワーク全体での通信量計算量，理想時間計算量を用いる．また，これらの評価においては，各 PE の識別子に関して，次の仮定を設ける．

［仮定３．３］ 各 PE の識別子は，O(log n) ビットであるとする．但し，n はネットワークのサイズを表す． □

【各 PE の領域計算量】

［定理３．４］ アルゴリズム CST を n_k の任意のネットワークで実行したときの各 PE の領域計算量はO(log n+d) ビットである．但し，n はネットワークのサイズ，d は PE の次数を表す．

（証明） プログラムに現れる変数・定数とその型およびそのサイズを表３．３に示す． □

表３．３　アルゴリズム CST で用いる変数と定数．

変数	型	サイズ
par, x	port 型	O(log d)
SON	port の集合型	O(d)
st	数え上げ型	O(1)
k_id, p, q, ID	process 型	O(log n)
CON	整数型(最大 n)	O(log n)
n_rep	整数型(最大 n^2)	O(log n)
m	整数型(最大 d)	O(log d)
DEG	整数型(d)	O(log d)

【通信量計算量】

ここでは，通信量計算量として，send 命令の実行数を数える．

［定理３．５］ アルゴリズム CST を $\mathcal{N}_k$ の任意のネットワークで実行したときの通信量計算量は $O(n^2 \cdot e)$ である．但し，n，e はそれぞれネットワークのサイズ，リンク数（無向グラフの辺数）を表す．

（証明） CST が停止した時刻を t' とする．このとき，

$$S^{t'}(i) \subseteq P^{t'}(i) \subseteq \{v_1, \ldots, v_{i-1}\}$$

となり，

$$|S^{t'}(i)| \leq |P^{t'}(i)| \leq i-1$$

である．

MERGE メッセージ：各 i ($1 \leq i \leq n$) について，v_i は，各出力ポートに対して高々１回だけ《MERGE,v_i》を送信する．また，$S^{t'}(i)$ の各 PE は１回だ

け，par 以外の各出力ポートに対して《MERGE,v_i》を送信する．従って，《MERGE,v_i》は高々 $\deg(v_i)+\sum_{1\leq j\leq i-1}(\deg(v_j)-1)(<2\cdot e)$ 個通信される．故に，CST 全体で高々 $2\cdot n\cdot e$ 個の MERGE メッセージが通信される．

MERGED メッセージ：各 i，j $(1\leq j<i\leq n)$ について，$P^{t'}(i)$ の PE v_j が1回だけ，par に対して，《MERGED,v_i,v_j,m》を送信する．故に，CST 全体で $\sum_{1\leq j\leq n}|P^{t'}(j)|\leq n(n-1)/2$ 個の MERGED が通信される．

NEW メッセージ：各 i，j $(1\leq j<i\leq n)$ について，$v_j\in P^{t'}(i)$ のとき $S^{t'}(i)$ の各 PE が高々1回だけ par に対して，《NEW,v_i,v_j,m》を送信する．従って，第2成分が v_i の NEW は $O(i^2)$ 個送信される．従って，CST 全体で $O(n^3)$ 個の NEW が送信される．

ACK メッセージ：各 i，j $(1\leq j<i\leq n)$ について，$v_j\in P^{t'}(i)$ のとき $S^{t'}(i)$ の各 PE が高々1回だけ par から，《ACK,v_i,v_j》を受信する．NEW の場合と同様に，CST 全体で $O(n^3)$ 個の ACK が送信される．

REP メッセージ：各 i について，第2成分が v_i の REP は，次の場合に par に送信される．

(1) $S^{t'}(i)$ の PE が par 以外から《MERGE,v_i》を受信したとき ((1,2) または (1,3))，または，

(2) $S^{t'}(i)$ の PE が SON に属するポートから，第2成分が v_i の REP を受信したとき ((5,3))．

(1) によって送信される第2成分が v_i の REP は高々 e 個である．そして，それぞれが (2) による $O(i)$ 個の送信を生じる．従って，第2成分が v_i の REP は $O(n\cdot e)$ 個送信され，CST 全体で $O(n^2\cdot e)$ 個の REP が送信される．

HALT メッセージ：《HALT》は CST で構成された生成木の各辺を1回だけ使って送受信される．従って，CST 全体で $O(n)$ 個の《HALT》が送信される．

$\mathcal{N}_k$ の任意のネットワークにおいて，$k\cdot n/2\leq e\leq n(n-1)/2$ が成立する[15]こ

とより，アルゴリズム CST の通信量計算量は $O(n^2 \cdot e)$ である． □

【理想時間計算量】

非同期式ネットワークではメッセージの通信遅延時間がまえもって分らない（定義２．３）ので，実時間による実行時間の評価はできない．ここでは，文献（12）等で，非同期式ネットワークでの分散アルゴリズム実行時間の評価に用いられている理想時間計算量（ideal execution time）を用いて CST を評価する．理想時間計算量とは，次の２つの評価基準のもとでの時間計算量のことである．

（評価基準Ａ） PE の処理時間は無視できる．

（評価基準Ｂ） リンクの伝播遅延時間が単位時間である．

これらの評価基準は，PE u から v に送信したメッセージは，u による送信の１単位時間後に v によって受信されるとことを意味する．

［定理３．６］ アルゴリズム CST を $\mathcal{N}_k$ の任意のネットワークで実行したときの理想時間計算量は $O(n^2)$ である．但し，n はネットワークのサイズを表す．

（証明） まず，v_n が王（st=king）になってから $O(n^2)$ 時間でアルゴリズム CST が停止することを示す．

《MERGE,v_n》を受信した PE v は，その次の時刻には MERGED を par に対して送信する．これは NEW に変り，生成木の辺を用いて v_n に伝えられる．v_n はこれに対し，ACK を生成木の辺を用いて v に伝える．そして，v はこの ACK を受信して，par 以外のポートに《MERGE,v_n》を送信する．こうし

て，全ての PE が k_id=v_n かつ st=subject になるまでの時間は高々

$\sum_{1\leq j\leq n-1}(1+2\cdot d(v_n,v_j)) \leq n^2$ である．但し，$d(v_n,v_j)$ は生成木での v_n と v_j を結ぶ通路の長さを表す．k_id=v_n かつ st=subject となった PE は《MERGE,v_n》を送信し，それを受信した PE は，REP を par に対して送信するので，REP が v_n に到着して v_n において n_rep<k となるまでにさらにあと O(n) 時間かかり，《HALT》によって O(n) 時間でアルゴリズムが停止する．

次に，CST の実行を開始してから $O(n^2)$ 時間で，v_n が王になることを示す．《MERGE,v_i》を受信した PE v は，その次の時刻にはメッセージ MERGED を par に対して送信する（$v<v_i$ のとき）か，あるいは，メッセージ《MERGE,v》を各ポートに対して送信する（$v>v_i$ のとき）．従って，上と同様に，$O(n^2)$ 時間で v_n はメッセージ MERGE を受信し，王になる． □

３．３ サイズ既知アルゴリズム

ここでは，任意のネットワーク $N \in \mathcal{N}$ で生成木構成問題（サイズ分解不可能な問題）π^S を解く耐辺故障度∞（ネットワークが条件３．１，３．２を満たす限り，故障辺数に制限がないという意味）のサイズ既知のアルゴリズム SST を示す．まず，リンクが故障しているなら，必ず両方向とも故障しているという仮定（仮定３．２）の下で π^S を解くアルゴリズム SST を示す．SST を簡単に改良すれば，片方向だけが故障しているリンクが存在するネットワークで，π^S を解くアルゴリズムが得られる．(３．５節参照)

[アルゴリズム SST]

(known-Size algorithm for constructing a Spanning Tree)

CST とほぼ同じである．CST との主な相違点についてのみ述べる．

１．(終了条件) CST では，王が，変数 n_rep に自分の王国と王国外を結んでいる可能性のあるリンク数を記憶しておき，その値が CON-1 以下になると終了した．SST では，自分の王国に属する PE 数を記憶しておき，それが SIZE になることを終了条件とする．そのため，メッセージ MERGED と NEW の第４成分やメッセージ REP は用いない．

２．(k_id より大きい識別子を持つ MERGE の受け手 u のふるまい) CST では，k_id より大きい識別子を持つ MERGE を受信した PE u は MERGED を送信し，st を pending にする．そして，MERGED の内容を NEW を用いて王に伝え，王が返信した ACK を u が受信して初めてその家来になる (st を subject にする)．それに対し，SST では，u は，すぐに家来になり，親に MERGED を，親以外に MERGE を送信する．このため，SST では，pending 状態は不要である．また，メッセージ ACK が不要になり，メッセージ MERGED,

NEW の第３成分は不要となる．

アルゴリズム SST の詳細は省略する． □

定理３．３と同様にして，次の定理を示せる．

［定理３．７］ サイズ既知アルゴリズム SST は任意のネットワーク $N \in \mathcal{N}$ で耐辺故障度∞で生成木構成問題を解く． □

定理３．４，３．５，３．６と同様にして，次の定理を示せる．

［定理３．８］ アルゴリズム SST の各 PE の領域計算量は $O(\log n+d)$ ビット，１メッセージのサイズは $O(\log n)$ ビット，通信量計算量は $O(n^3)$，理想時間計算量は $O(n)$ である．但し，n はネットワークのサイズを表し，d は PE の次数を表す． □

３．４　隣接識別子既知アルゴリズム

ここでは，任意のネットワーク $N \in \mathcal{N}$ で生成木構成問題（サイズ分解不可能な問題）π^S を解く耐辺故障度∞（ネットワークが条件３．１，３．２を満たす限り，故障辺数に制限がないという意味)の隣接識別子既知アルゴリズム NST を示す．まず，リンクが故障しているなら，必ず両方向とも故障しているという仮定（仮定３．２）の下で π^S を解くアルゴリズム NST を示す．NST を簡単に改良すれば，片方向だけが故障しているリンクが存在するネットワークで，π^S を解くアルゴリズムが得られる.(３．５節参照)

[アルゴリズム NST]

(known-Neighbors algorithm for constructing a Spanning Tree)

CST とほぼ同じである．CST との主な相違点についてのみ述べる．

１．(終了条件)　NST では，王は，自分の家来の識別子の集合を変数 kingdom に，また，自分の家来の隣接 PE の識別子の集合の和集合を変数 n_id に記憶しておく．そして，kingdom=n_id が成立したときに，王国がネットワーク全体に広がったことが分るので，これを終了条件にする．そのため，メッセージ MERGED と NEW を次のように変更する．また，メッセージ REP は用いない．

《MERGED,p,q,n_list》:《MERGE,p》を受信した q がその MERGE の送り手(親）に送信する．「p の家来になる．q の隣接 PE の識別子のリストは n_list (=$NLIST_q$) である.」を意味する．

《NEW,p,q,n_list》:《MERGED,p,q,n_list》または《NEW,p,q,n_list》の受け手が親に送信する．「q が p の王国に加わるというメッセージを受信し

た．q の隣接 PE の識別子のリストは n_list である.」を意味する．

２．(k_id より大きい識別子を持つ MERGE の受け手 u のふるまい) NST では，u は，すぐに家来になり，親に MERGED を，親以外に MERGE を送信する．このため，SST では，pending 状態は不要である．また，メッセージ ACK が不要になる． □

定理３．３と同様にして，次の定理を示せる．

［定理３．９］ 隣接識別子既知アルゴリズム NST は任意のネットワーク N $\in \mathcal{N}$ で耐辺故障度∞で生成木構成問題を解く． □

定理３．４，３．５，３．６と同様にして，次の定理を示せる．

［定理３．10］ アルゴリズム SST の各 PE の領域計算量は O(n･log n) ビット，１メッセージのサイズは O(n･log n) ビット，通信量計算量は $O(n^3)$，理想時間計算量は O(n) である．但し，n はネットワークのサイズを表し，d は PE の次数を表す． □

３．５　仮定の緩和

ここでは，いくつかの仮定を緩めた場合について考察する．

(1)（仮定３．２の緩和） CST，SST，NST は，次のように変更すれば，片方向の通信だけができなくなる可能性のある故障（仮定３．２の緩和）に対しても同じ耐辺故障度(２方向とも通信できなくなるリンクの故障も，片方向だけの通信ができなくなるリンクの故障も，１つのリンクの故障と数える.)で，生成木構成問題 π^S を解くことができる．

MERGE を送信する前に，メッセージ《TEST》を送信する．《TEST》の受け手はメッセージ《ALIVE》を返信し，《ALIVE》を受信して初めて MERGE を送信する．このように変更すると，MERGE は両方向に通信可能なリンク（故障していないリンク）を通じてのみ送受信されるので，リンクが故障すると両方向の通信ができなくなると仮定した場合と同じように CST も SST も NST も π^S を解く．また，このときの PE の領域計算量，通信量計算量，理想時間計算量はそれぞれオーダ的には変わらない．

(2)（仮定２．７の緩和） CST，SST，NST はいずれも，リンクに対して，最初に MERGE を送信し，それに対する返事を受信してから，その後のメッセージの送信を行う．従って，最初の MERGE が受信されなければ，その後のメッセージの送信が行われないので，故障したリンクがアルゴリズム実行中に復旧することがある場合（仮定２．７の緩和）でも，CST も SST も NST も変更することなく生成木構成問題を解く．

(3)（仮定２．７の緩和） CST，SST，NST はいずれも，次のように変更すれば，アルゴリズムの実行中にリンクが故障することがある場合（仮定２．７の緩和）でも，同じ耐辺故障度で，N の生成木を構成することができる．但し，構成された N の生成木には故障しているリンクが含まれているかもしれ

ないという意味で，定義２．７で定義した生成木構成問題（Live(N) の生成木を求めるので，生成木が故障リンクを含んではいけない）を解くことにはならない．

MERGE を送信するときには，その送り手の識別子とポート番号を付けて送信する．そして，MERGE 以外の全てのメッセージは，送り手の識別子と受け手の識別子を（MERGED には，更に受信した MERGE に付いていたポート番号を，また，REP には，シリアル番号を）付けて，放送（全ての隣接PE にメッセージを送信し，それを受信した PE は，その行き先が自分以外ならば，そのメッセージの送り手以外の全ての隣接 PE に送信する）によって，通信を行う．

４．プロセッサ故障診断問題

この章では，PE 故障診断問題π^pについて考察する．但し，診断 PE を v とし，$target_v$=a とする．そして，PE v の a-隣接 PE を wとする．つまり，v が w の故障を診断する問題を考える．

故障診断を行う主体とも言うべき PE v が故障しているとπ^pが意味を持たなくなる．また，準健全な v-w 通路が存在しないと，v から w へメッセージを送るための通路が存在しなくなり，π^pが解けなくなる．また，ある健全 PE u に対し，健全な v-u 通路が存在しないと，u がいつ停止すればよいかの決定が難しくなる.(ＰＬ同期式でないときは決定できないことが示せる.）そして，アルゴリズムによって，w が故障しているかどうかを v が判定できても，u が停止しないために，定義上π^pを解いたことにならない（定義２．６参照）という状況が生じてしまう．問題の解を求めた後，全ての PE を停止させる方法を考えることは，それ自体が興味深い問題であるが，ここでは，w の故障を v が診断できるかどうかという点に着目したいので，本章では，次の３つの条件が成立するときにπ^pが解けるかどうかについて議論する．そして，本章の定理や補題では，特に断らないが，次の３条件が成立することを前提としている．これらの条件が成立しないネットワークで，本章のアルゴリズムを実行すると，アルゴリズムが停止しないか，または，w が故障していなくても w が故障していると診断する.(定理４．３，４．４，４．６，４．７の（注）参照.）但し，w が故障していないと診断したときは，その結果は正しい．

［条件４．１］ PE v は故障していない． □

［条件４．２］ 準健全な v-w 通路が存在する． □

［条件４．３］ 任意の健全 PE u に対して，健全な v-u 通路が存在する

(Live(N) が連結). □

表４．１に，本章の結果をまとめる．

表4．1　π^P（PE v による PE w の故障診断）を解くアルゴリズム

<table>
<tr><th>アルゴリズム
ネットワーク</th><th>基本情報
アルゴリズム</th><th>サイズ既知
アルゴリズム</th><th>辺連結度既知
アルゴリズム</th><th>隣接識別子既知
アルゴリズム</th><th>トポロジ・
アルゴリズム</th></tr>
<tr><td>P非同期式</td><td colspan="5">w 以外の PE やリンクに故障がなくても存在しない（定理4．1）</td></tr>
<tr><td>L非同期式</td><td colspan="5">w 以外の PE やリンクに故障がなくても存在しない（定理4．2）</td></tr>
<tr><td rowspan="4">PL同期式</td><td colspan="5">リンク（v,w）が故障していないとき：
PLSBp（故障 PE 数，故障リンク数にかかわらず解く）（定理4．3）</td></tr>
<tr><td colspan="5">リンク（v,w）が故障している可能性があるとき：</td></tr>
<tr><td rowspan="3">w 以外の PE や（v,w）以外のリンクに故障がなくても存在しない
（定理4．7(3)）</td><td colspan="2">w 以外の任意のあるPEが故障している可能性があるとき存在しない
（定理4．5）</td><td colspan="2" rowspan="3">PLSNp
（故障 PE 数，故障リンク数にかかわらず解く）
（定理4．4）</td></tr>
<tr><td colspan="2">w 以外の PE に故障がないとき：</td></tr>
<tr><td>PLSSp
（故障リンク数にかかわらず解く）
（定理4．6）</td><td>故障リンク数が⌈CON/2⌉-1 以下のとき：
PLSCp
（定理4．7(1)）
故障リンク数が⌈CON/2⌉ 以上のとき存在しない
（定理4．7(2)）</td></tr>
</table>

４．１　Ｐ非同期式ネットワーク

ネットワークがＰ非同期式の場合，次の定理より，w 以外の PE やリンクに故障がなく，各 PE がネットワークの形状に関していかなる情報を持っていてもπ^P は解けない．

［定理４．１］ ネットワークがＰ非同期式のとき，Ｌ同期式，Ｌ非同期式にかかわらず，w 以外の PE やリンクに故障がなくても，π^P を解くトポロジ・アルゴリズムは存在しない．

（証明） Ｐ非同期式ネットワークで，π^P を解くトポロジ・アルゴリズムＡが存在すると仮定する．

Ａはネットワーク N で w だけが故障していて他に故障がないときπ^P を解き，w が故障していると診断して停止する．このとき，w は故障しているので通信命令を実行していない．

一方，N でどの PE もリンクも故障していないときにＡを実行するとする．N がＰ非同期式なので w の動作が非常に遅い場合がある．このとき，w の最初の通信命令の実行前に，w 以外の各 PE は，w だけが故障しているときと同じ動作（定理３．１の証明参照）をし，w が故障していると診断して停止する場合がある．このことは，Ａがπ^P を解くことに矛盾する．　□

４．２　Ｌ非同期式ネットワーク

ネットワークがＬ非同期式の場合，次の定理より，w 以外の PE やリンクに故障がなく，各 PE がネットワークの形状に関していかなる情報を持っていても π^P が解けない．

［定理４．２］ ネットワークがＬ非同期式のとき，Ｐ同期式，Ｐ非同期式にかかわらず，w 以外の PE やリンクに故障がなくても，π^P を解くトポロジ・アルゴリズムは存在しない．

（証明） Ｌ非同期式ネットワークで，π^P を解くトポロジ・アルゴリズムＡが存在すると仮定する．

Ａはネットワーク N で w だけが故障していて他に故障がないとき π^P を解き，w が故障していると診断して停止する．このとき，w は故障しているので通信命令を実行していない．従って，w からのメッセージを受信する PE は存在しない．

一方，N でどの PE もリンクも故障していないときにＡを実行するとする．N がＬ非同期式なので，w に接合するリンク全ての伝播遅延が非常に大きい場合がある．このとき，w からのメッセージを受信する前に，w 以外の各 PE は，w だけが故障しているときと同じ動作（定理３．１の証明参照）をし，w が故障していると診断して停止する場合がある．このことは，Ａが π^P を解くことに矛盾する．　□

４．３　ＰＬ同期式ネットワーク

定理４．１，４．２より，ネットワークがＰ非同期式かＬ非同期式なら，w 以外の PE やリンクに故障がなくても，π^P を解くトポロジ・アルゴリズムが存在しない．この節では，ＰＬ同期式の場合について考える．

4.3.1　リンク (v,w) が故障していないとき

リンク (v,w) が故障していない (<v,w> も <w,v> も故障していない) ときに π^P を解く基本情報アルゴリズム PLSBp を示す．仮定２．４で，全ての PE のプログラムは同一であると仮定したが，分りやすさのために，診断 PE v と v 以外の PE とが実際に実行するプログラムの部分だけを示す．実際のプログラムでは，入力変数 tester の値によってどちらか一方に分岐するようになっている．また，B がポート集合を表すとき，B の各ポートに対して任意の順でメッセージ mes を送信することを send(B,mes) と略記する．また，$PORT_u$ は u の全ポートの集合 $\{1,2,...,DEG_u\}$ を表す．

[アルゴリズム PLSBp]

(PL-Synchronized known-Basic-information algorithm for π^P)

【診断 PE v の動作】

(v1)

send($target_v$,《TEST》);

repeat(2×(PSYNC+LSYNC)-1) {2×(PSYNC+LSYNC)-1 回繰り返し実行する}

  receive(MES);

    {receive 命令を実行し，受信したメッセージの集合を MES とする}

  if 《SAFE》∈MES then begin $result_v$:=false; (v2) へ end

endrepeat;

$result_v$:=true; (v2) へ.

(v2) send($PORT_v$,《HALT》), halt 命令を実行し，停止する.

【v 以外の各 PE u の動作】

repeat(∞)

  receive(MES);

  if 《TEST》∈MES then send(b,《SAFE》) (b は《TEST》を受信したポート)

  else if 《HALT》∈MES then begin

    send($PORT_u$-{c},《HALT》) (c は《HALT》を受信したポート);

    halt {停止する}

  end

endrepeat. □

［補題４．１］ ＰＬ同期式ネットワークで PLSBp を実行したときに，PE v がメッセージ《SAFE》を受信するための必要十分条件は PE v，w およびリンク (v,w) が故障していないことである．

（証明） 必要条件は明らか．以下では，十分条件であることを証明する．

Ｐ同期定数，Ｌ同期定数をそれぞれΓ，Δとする．v，w，(v,w) が故障していないネットワークで PLSBp を実行したときの任意の通信順序を $S=(s_1, s_2,\ldots)$ とし，$2\times(\Gamma+\Delta)-1=r$ と表す．v は１回の send 命令，r 回の receive 命令 ((v1)) と deg(v) 回の send 命令 ((v2)) を実行するので，$v\in s_i$ なる i ($1\leq i$) は正確に $r+1+deg(v)$ 個ある．それらを $s_{f(j)}$ ($0\leq j\leq r+deg(v)$) (但し，$f(j)<f(j+1)$ ($0\leq j<r+deg(v)$)) とする．$s_{f(0)}$ は send 命令，$s_{f(1)},\ldots,s_{f(r)}$ は receive 命令，$s_{f(r+1)},\ldots,s_{f(r+deg(v))}$ は send 命令の実行に対応する．

$\#(v,S[f(\Delta),f(\Gamma+\Delta-1)])=\Gamma$ 及びＰ同期式であることより，$w\in s_h$ なる h (但し，$f(\Delta)\leq h\leq f(\Gamma+\Delta-1)$) が存在する．一方，$f(\Delta)\leq h$ より，$\#(v,S[f(1),h])\geq\Delta$ なので，Ｌ同期式であることより，s_h 以前に，w は《TEST》を受信する．つまり，v の $\Gamma+\Delta-1$ 回目の receive 命令の実行以前に w は《TEST》を受信する．同様に，v の $2\times\Gamma+\Delta-1$ 回目の receive 命令の実行以前に w が《SAFE》を送信することおよび，v の r 回目の receive 命令の実行以前に v が《SAFE》を受信することが示せる． □

［定理４．３］ ネットワークがＰＬ同期式のとき，リンク (v,w) が故障していなければ，基本情報アルゴリズム PLSBp は，故障 PE の数，故障リンク数にかかわらず π^p を解く．

（証明） まず，PLSBp は必ず停止することを示す．v が停止するのは明らか．(v2) で v は全隣接 PE に《HALT》を送信し，《HALT》を受信した PE はさらにその隣接 PE に《HALT》を送信する．条件４．３より，任意の健全 PE

u に対し健全な v-u 通路が存在するので，v 以外の健全 PE は必ず《HALT》を受信し，停止する．つまり，PLSBp は停止する．

補題４．１より，w が故障していれば v は《SAFE》を受信しないので，w が故障している（$result_v$=true）と診断し，w が故障していなければ v は《SAFE》を受信するので，w が故障していない（$result_v$=false）と診断する．（v は条件４．１より，(v,w) は定理の仮定より故障していない．） □

（注１） リンク (v,w) が故障しているとき，v は《SAFE》を受信しないので，PLSBp は，PE w が故障していなくても w は故障していると診断する．

（注２） 条件４．３が成立しなくても，v は w が故障しているかどうか判定できる（v が停止したとき，w が故障していれば $result_v$=true が成り立ち，そうでなければ $result_v$=false が成り立つ）が，このとき，v との間に健全な通路のない健全 PE は停止しない．v 以外の PE の動作を，<u>repeat</u>(∞) の代りに，<u>repeat</u>(PSYNC+LSYNC) とし，repeat 文から抜けたときに halt 命令を実行し，停止するように変更すれば，条件４．３が成立しなくても全 PE が有限個の命令の実行後停止するので，同様のアルゴリズムで π^P が解ける．

4.3.2 リンク (v,w) が故障しているとき

以下では，リンク (v,w) に故障の可能性がある場合について考察する．まず，Live(N) と Live(N) の v を根とする根付き生成木を求める基本情報アルゴリズム PLSBLT を示す．但し，PLSBLT では，各 PEは，次の変数を持つ．

LIVEP：Live(N) のリンクに対応するポートの集合を記憶する．

par：生成木の親に通じるポートを記憶する．

SON：生成木の子に通じるポートの集合を記憶する．

【アルゴリズム PLSBLT の概略】

PLSBLT では，Live(N) の v を出発点とする深さ優先探索 (depth-first traverse)[1] で PE を訪れる順で，各健全 PE x に対し，$PORT_x$ のポートのうち，Live(N) のリンク（辺）に接続しているポート（つまり，両方向とも故障していないリンクに接続し，しかもそのリンクによって健全 PE に隣接しているポート）の集合を $LIVEP_x$ に求める（これを，ポート試験という）．ポート試験は，PLSBp と同様の方法で，メッセージ《TEST》と《SAFE》をやりとりすることによって行う．ここで，深さ優先探索の順でポート試験を行うのは，次の理由による．

２つ以上の PE が同時にポート試験を行うと，ある PE が１回の receive 命令で２個以上の《TEST》を受信する可能性がある．この PE は，受信した各《TEST》に対し《SAFE》を返信するが，同時に２個以上のメッセージを送信できない（定義２．３参照）ので，これらを順番に送信する．この場合，《TEST》を送信した PE は，隣接 PE のポート数などを知らないと，いつまで《SAFE》を待てばよいか判断できない．ここでは深さ優先探索の順でポート試験を行うことにより，任意の時刻において，ポート試験を行っている PE は高々１つであるようにしている．

深さ優先探索の順でポート試験を行うために，メッセージ《TOKEN》が Live(N) の v を出発点とする深さ優先探索によって PE を訪れさせられる．この目的のために，各 PE x には，《TEST》を受信したことがあるポートの集合を記憶するための変数 $TESTP_x$ が用意されている．$TESTP_x$ のポートを通じて隣接している PE は，既に《TOKEN》が訪れた PE である．ポート試験を終えた PE xは，$LIVEP_x - TESTP_x \neq \phi$ なら，$LIVEP_x - TESTP_x$ の任意の１つのポートから《TOKEN》を送信する．この《TOKEN》を受信した PE y は，x を生成木における自分の親と定める．そして，深さ優先生成木の y を根とする部分木を求めた後，y は x に《TOKEN》を送信する．このようにして，Live(N) の v を根とする深さ優先生成木[1] を構成する．

[アルゴリズム PLSBLT]

(PL-Synchronized known-Basic-information algorithm

for constructing Live(N) and a spanning Tree)

【診断 PE v の動作】

(v1) par_v:=0; SON_v:={}; $TESTP_v$:={}; (v2) へ.

{v は根なので v の親は存在しない．そこで，ダミー値として par_v に 0 を記憶しておく.}

(v2) {v がポート試験を行う.} send($PORT_v$,《TEST》) を実行後，receive 命令を 2×(PSYNC+LSYNC)-1 回実行する．この間に《SAFE》を受信したポートの集合を $LIVEP_v$ に代入し，(v3) へ.

(v3) まだ《TOKEN》を送信していないポートが $LIVEP_v$-$TESTP_v$ にあれば，その任意の１つを b とし，(v4) へ．なければ，halt 命令を実行し，停止する.

(v4)

SON_v:=SON_v ∪ {b}; send(b,《TOKEN》);

repeat(∞)

receive(MES);

if 《TEST》∈MES then begin

$TESTP_v$:=$TESTP_v$ ∪ {c}; send(c,《SAFE》) (c は《TEST》を受信したポート)

end

else if 《TOKEN》∈MES then (v3) へ {《TOKEN》は b から受信する}

{深さ優先の順でポート試験を行うので，１回の receive 命令で，２個以上の《TEST》，または，《TEST》と《TOKEN》の両方を受信することはない}

endrepeat.

【v 以外の各 PE u の動作】

(u1)

SON_u:={}, $TESTP_u$:={};

<u>repeat</u>(∞)

receive(MES);

<u>if</u> 《TEST》∈MES <u>then</u> <u>begin</u>

$TESTP_u$:=$TESTP_u$∪{d}; send(d,《SAFE》) (d は《TEST》を受信したポート)

<u>end</u>

<u>else</u> <u>if</u> 《TOKEN》∈MES <u>then</u> (u2) へ

<u>endrepeat</u>.

(u2) par_u:=e (e は《TOKEN》を受信したポート) を実行する．その後は，(v2)-(v4) と同様．但し，halt 命令を実行する直前に，send(par_u,《TOKEN》) を実行する． □

［補題４．２］ PLSBLT は必ず停止し，停止時の各 PE の変数 LIVEP, par, SON に対し，以下のことが成立する．

(1) x を任意の健全 PE とする．このとき，

$LIVEP_x = \{b \in PORT_x | x$ の b-隣接 PE も $\mu(x,b)$ も故障していない$\}$

が成立する．

(2) $P' = \{x \in P | x$ は健全 PE$\}$, $L' = \{\mu(x,b) \in L | b \in LIVEP_x,\ x \in P'\}$

とする．このとき，

$Live(N) = (P',L')$

となる．

(3) x を v 以外の任意の健全 PE, y を x の c-隣接 PE ($1 \leq c \leq deg(x)$) とする．また，x は y の d-隣接 PE ($1 \leq d \leq deg(y)$) であるとする．このとき，

$par_x = c$ のとき，かつ，そのときに限り $d \in SON_y$

となる．

(4) $P' = \{x \in P | x$ は健全 PE$\}$, $L'' = \{\mu(x,par_x) \in L | x \in P'-\{v\}\}$

とする．このとき，$G'' = (P',L'')$ は Live(N) の生成木であり，G" を v を根とする根付き木とみなしたとき，v 以外の任意の PE u に対し，par_uは G" で u の親に通じるポートである． □

（注）条件４．３が満たされないとき，v との間に健全な通路のない健全 PE は停止しない．このとき，v との間に健全な通路のある健全 PE に関しては，補題４．２(1), (3)が成り立つ．また，

$P' = \{x \in P | x$ は v との間に健全な通路のある健全 PE$\}$

とすると，補題４．２(2), (4)の G', G" はそれぞれ，Live(N) の v を含む連結成分，その連結成分の生成木となる．PE が形状定数 SIZE を利用できるなら，v との間に健全な通路があれば，receive 命令を高々何回実行すれば《TOKEN》を受信するか分るので，v との間に健全な通路のない PE も停止するようにできる．

PLSBLT を用いて構成した生成木を利用すると，リンク (v,w) が故障している可能性があっても，故障 PE の数，故障リンクの数にかかわらず，π^p を解く隣接識別子既知アルゴリズムを次のようにして作れる．

[アルゴリズム PLSNp]

(PL-Synchronized known-Neighbors algorithm for π^p)

PLSNp は以下に示す２つのフェーズからなる．

フェーズ１：PLSBLT を用いて，Live(N) の生成木 G"=(P',L") を構成する．但し，halt 命令は実行しない．

フェーズ２：v は G" を用いて，全ての健全 PE に w の識別子 id(w) (=$NLIST_v[target_v]$) を知らせ，w∈P' かどうかを調べる．w∈P' なら w は健全 PE であると診断し，w∉P' なら w は故障していると診断する． □

補題４．２より，次の定理が成立する．

[定理４．４] ネットワークがＰＬ同期式のとき，リンク (v,w) が故障している可能性があっても，隣接識別子既知アルゴリズム PLSNp は，故障 PE の数，故障リンク数にかかわらず，π^p を解く． □

(注) 条件４．３が満たされないとき，PLSNp で v は w の故障を診断できる (v が停止したとき，w が故障していると $result_v$=true，そうでなければ $result_v$=false が成り立つ) が，停止しない PE が存在する．しかし，PE が形状定数 SIZE を利用できるなら，全ての健全 PE が停止するようにできるので (補題４．２の注参照)，同様の方法で π^p が解ける．

次に，リンク (v,w) が故障している可能性がある場合に π^P を解くサイズ既知アルゴリズム，辺連結度既知アルゴリズム，基本情報アルゴリズムについて考察する．まず，w 以外の PE が故障している可能性があるとき，次の定理が成立する．

［定理４．５］ ネットワークがＰＬ同期式で，(v,w) が故障している可能性と，w 以外の PE が故障している可能性があるとき，w 以外の故障 PE が高々１個で，(v,w) 以外のリンクに故障がなくても，次のことが成立する．

(1) π^Pを解くサイズ既知アルゴリズムは存在しない．

(2) π^Pを解く辺連結度既知アルゴリズムは存在しない．

（証明）(1)の証明：w 以外の故障 PE が高々１個で，(v,w) 以外のリンクに故障がないときに π^P を解くサイズ既知アルゴリズムＡが存在すると仮定する．Ａは N=(P,L) でPE w'(≠w)，リンク ⟨v,w⟩，⟨w,v⟩ だけが故障しているとき π^P を解き，w は故障していないと診断する．但し，N において，v は w の b-隣接 PE とする．

N に対し，N'=(P',L') を次のように定義する.(図４．１)

$P' = P,$

$L' = (L-\{(v,w)\}) \cup \{(v,w'),(w,w')\}$

L' の定義より，

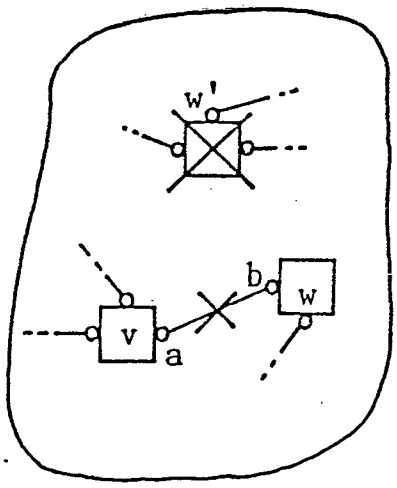
$\deg_{N'}(w') = \deg_N(w')+2,$ † 及び，

任意の PE u (≠w') に対し，

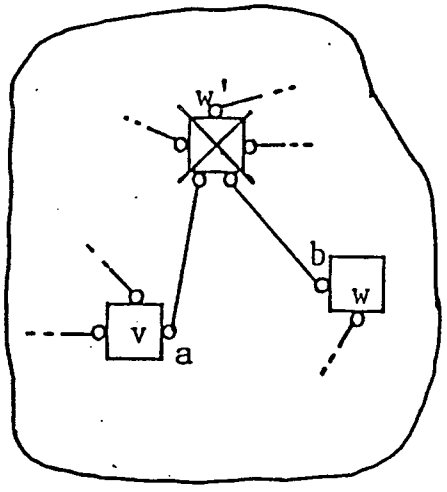
$\deg_{N'}(u) = \deg_N(u)$

が，成立する．従って，|P'|=|P| 及びＡがサイズ既知アルゴリズムであることより，N' において，w' 以外の各 PE が利用できる形状定数の値は，N の

† $\deg_N(w')$は Nにおける w' の次数を表す．



(a) Network N with (v,w) and w' failed.



(b) Network N' with w' failed.

図4．1　ネットワーク N と N'.

Fig. 4. 1　Networks N and N'.

場合と同じである．また，N' のポート対応を次のように定める．

$$\mu_{N'}(v,c) = \begin{cases} (v,w') & (c=a(=target_v)\ のとき) \\ \mu_N(v,c) & (c\neq a\ のとき,\ 1\leq c\leq deg(v)) \end{cases}$$

$$\mu_{N'}(w,c) = \begin{cases} (w,w') & (c=b\ のとき) \\ \mu_N(w,c) & (c\neq b\ のとき,\ 1\leq c\leq deg(w)) \end{cases}$$

$$\mu_{N'}(w',c) = \begin{cases} \mu_N(w',c) & (1\leq c\leq deg_N(w')=deg_{N'}(w')-2\ のとき) \\ (w',v) & (c=deg_{N'}(w')-1\ のとき) \\ (w',w) & (c=deg_{N'}(w')\ のとき) \end{cases}$$

v, w, w' 以外の各 PE u に対し，

$$\mu_{N'}(u,c) = \mu_N(u,c) \qquad (1\leq c\leq deg(u))$$

w' だけが故障している N' で，v が診断 PE で $target_v$=a のときにAを用いて π^p (v が w' の故障を診断する問題) を解くと，w' 以外の各 PEは，<v,w>，<w,v>，w' だけが故障している N で π^p を解くときと同じ動作（定理３．１の略証参照）をし，w' は故障していないと診断する場合がある．このことは，Aが π^p を解くことに矛盾する．

(2)の証明：N が k-辺連結なら (1) で作った N' も k-辺連結であることが示せる．従って，辺連結度既知アルゴリズムが N' において，w' 以外の各 PE が利用できる形状定数の値は，N の場合と同じである．このことから，(1) と同様にして証明できる． □

次に，w 以外の PE に故障がない場合について考える．

[アルゴリズム PLSSp]

(PL-Synchronized known-Size algorithm for π^p)

PLSSp は以下に示す２つのフェーズからなる．

フェーズ１：PLSBLT を用いて，Live(N) の生成木 G" を構成する．但し，

halt 命令は実行しない.

フェーズ２：v は G" の PE 数を求め（求めた PE 数を n' とする），n' <SIZE なら w は故障していると診断し，n'=SIZE なら w は故障していないと診断する. □

補題４．２より，次の定理が成立する.

［定理４．６］ ネットワークがＰＬ同期式のとき，リンク (v,w) が故障している可能性があっても，w 以外の PE に故障がなければ，サイズ既知アルゴリズム PLSSp は，故障リンク数にかかわらず，π^p を解く. □

（注１） w 以外の PE が故障しているとき，G" に含まれる PE の数は SIZE より少ないので，PLSSp は w が故障していなくても w は故障していると診断する.

（注２） 条件４．３が満たされないとき，Live(N) の v を含む連結成分の PE 数が求まり，これは SIZE より小さいので，PLSSp は w が故障していなくても w は故障していると診断する．但し，各 PE は SIZE を利用できるので，全ての健全 PE が停止するようにできる.(補題４．２の注参照)

次に，辺連結度既知アルゴリズムについて考える.

［アルゴリズム PLSCp］

(PL-Synchronized known-Connectivity algorithm for π^p)

PLSCp は以下に示す２つのフェーズからなる.

フェーズ１：PLSBLT を用いて，Live(N) とその生成木 G"=(P',L") を構成する．但し，halt 命令は実行しない．

フェーズ２：各 PE u は，変数 $count_u$ に $|PORT_u|-|LIVEP_u|$ を求める．そして，PE v は，G" の全ての PE の変数 count の値を集めることによって，変数 $sumc_v$ に $\sum_{u \in P'} count_u$ を求める．そして，$sumc_v < CON_v$ なら w は健全 PE であると診断し，$sumc_v \geq CON_v$ なら w は故障 PE であると診断する．□

［定理４．７］ ネットワークがＰＬ同期式で，リンク (v,w) が故障している可能性があるとき， w 以外の PE に故障がなければ，次のことが成立する．

(1) 故障リンク数が ⌈CON/2⌉-1 以下のとき，辺連結度既知アルゴリズム PLSCp は π^p を解く．

(2) 故障リンクの数が ⌈CON/2⌉ 以上のとき，π^p を解く辺連結度既知アルゴリズムは存在しない．

(3) (v,w) 以外のリンクに故障がなくても，π^p を解く基本情報アルゴリズムは存在しない．

（証明）(1)の証明：w が故障していない（故障している PE がない）とき，故障リンク数が h ならば，PLSCp のフェーズ２で $sumc_v$ に設定される値は 2×h（故障している全２重リンクの両端のポートの総数）である．従って，故障リンク数が ⌈CON/2⌉-1 以下のとき，

$$sumc_v \leq 2\times(\lceil CON/2 \rceil - 1) \leq CON-1$$

となり，PLSCp は w が故障していないと診断する．

一方，ネットワークが CON-辺連結であることより，deg(w)≧CON が成立する[15]．従って，w が故障しているとき，w に接続するポートからはそのリンクが故障していなくても《SAFE》を受信しないので，

$sumc_v \geqq deg(w) \geqq CON$

が成立し，PLSCp は w が故障していると診断する．

(2)の証明：故障リンクの数が⌈CON/2⌉でも，π^Pを解く辺連結度既知アルゴリズムAが存在すると仮定する．

ネットワーク N=(P,L) を次のように定義する．

$P = \{v_0(=v), v_1(=w), \ldots, v_{n-1}\}$

$L = \{(v_i, v_j) \mid i \neq j\}$

つまり，N は n 個の PE からなり，任意の相異なる２つの PE 間にリンクがあるネットワークである．このとき，N は (n-1)-辺連結で ある[15]．

$Lf = \{(v_{2i}, v_{2i+1}) \mid 0 \leqq i \leqq \lceil (n-1)/2 \rceil - 1\}$

とする．Aは，CON=n-1 のとき，Lf のリンクだけが全て両方向とも故障していて，他に故障がない N でπ^P を解き，w が故障していないと診断する．

N に対し，N'=(P',L') を次のように定義する．(図４．２)

$P' = P \cup \{w'\}$

$L' = (L - Lf) \cup \{(v_i, w') \mid 0 \leqq i \leqq 2 \times \lceil (n-1)/2 \rceil - 1\}$

また，ポート対応を次のように定める．

各 i，b $(0 \leqq i \leqq n-1,\ 1 \leqq b \leqq deg(v_i))$ に対して，

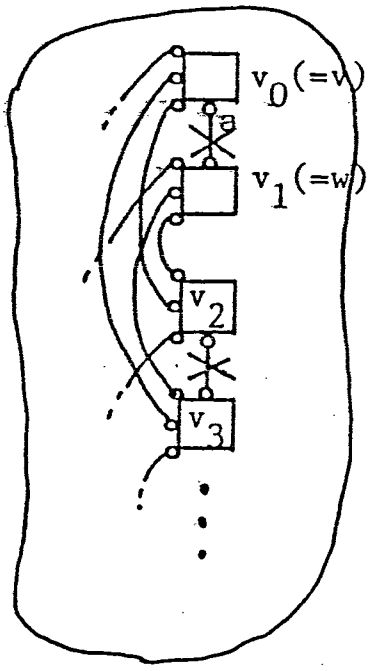
$$\mu_{N'}(v_i, b) = \begin{cases} (v_i, w') & (\mu_N(v_i, b) \in Lf \text{ のとき}) \\ \mu_N(v_i, b) & (\mu_N(v_i, b) \notin Lf \text{ のとき}) \end{cases}$$

各 b $(1 \leqq b \leqq deg(w'))$ に対し，

$\mu_{N'}(w', b) = (w', v_{b-1})$

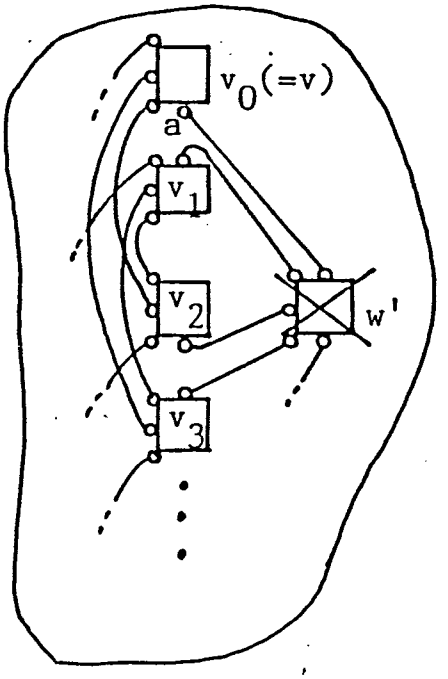
このとき，N' は (n-1)-辺連結となる．w' だけが故障している N' において，v が診断 PE で，$target_v$=a かつ CON=n-1 のときに，Aを用いてπ^P (v が w' を故障診断する問題) を解くと，定理４．５と同様にして，w' は故障していないと診断する場合があることが示せ，Aがπ^P を解くことに矛盾する．

(3)の証明：ネットワーク N=(P,L) を (2) の証明で用いた N とする．N'=(P',L') を次のように定義する．



(a) Network N with all links in

$Lf=\{(v_{2i},v_{2i+1})|0\leq i\leq \lceil (n-1)/2\rceil -1\}$ failed.



(b) Network N' with w' failed.

図4．2　ネットワーク N と N'.

Fig. 4. 2 Networks N and N'.

$P' = P \cup \{w'\}$

$L' = (L-\{(v,w)\}) \cup \{(v,w'),(w,w')\}$

また，ポート対応を次のように定める．

各 i，b $(0 \leqq i \leqq n-1,\ 1 \leqq b \leqq \deg(v_i))$ に対して，

$$\mu_{N'}(v_i,b) = \begin{cases} (v_i,w') & (\mu_N(v_i,b)=(v,w) \text{ のとき}) \\ \mu_N(v_i,b) & (\mu_N(v_i,b) \neq (v,w) \text{ のとき}) \end{cases}$$

各 b $(1 \leqq b \leqq 2)$ に対して，

$$\mu_{N'}(w',b) = (w',v_{b-1}) \quad (\text{但し，} v_0=v,\ v_1=w)$$

このとき，(2) と同様にして，(3) が成立することが示せる． □

（注１） w 以外の PE が故障しているとき，または故障リンク数が ⌈CON/2⌉ 以上のとき，PLSCp は，w が故障していなくても，w は故障していると診断する．

（注２） w が健全 PE で（故障 PE がなく），故障リンク数が ⌈CON/2⌉-1 以下のとき，ネットワークが CON-辺連結であることより，条件４．３は成り立つ．w の故障のために条件４．３が成り立たないとき，PLSCp によって，v は w の故障を診断できるが，停止しない PEが存在する．しかし，形状定数 SIZE を利用できるなら，全ての健全 PE を停止させることができ，同様の方法で π^p が解ける．

５．リンク故障診断問題

本章では，リンク故障診断問題π^1について考察する．但し，診断 PE を v とし，$target_v$=a とする．そして，μ(v,a)=(v,w) とする．つまり，v がリンクl=(v,w) の故障を診断する問題を考える．

故障診断を行う主体とも言うべき PE v が故障しているとπ^1が意味を持たなくなる．また，PE w が故障していると，lが故障していなくてもlを用いる通信ができないので，π^1が解けなくなる．また，ある健全 PE u に対し，健全な v-u 通路が存在しないと，u がいつ停止すればよいかの決定が難しくなる.(ＰＬ同期式でないときは決定できないことが示せる.) そして，アルゴリズムによって，lが故障しているかどうかを v が判定できても，u が停止しないために，定義上π^1を解いたことにならない（定義２．６参照）という状況が生じてしまう．ここでは，lの故障を v が診断できるかどうかという点に着目したいので，以下では，次の２つの条件が成立するときにπ^1が解けるかどうかについて議論する．そして，本章の定理や補題では，特に断らないが，次の２条件が成立することを前提としている．　本章のアルゴリズムを，次の条件が成り立たないネットワークで実行すると，アルゴリズムが停止しないか，あるいは，lが故障していなくても故障と診断する.(定理５．２，５．４，５．６，５．７の（注）参照）但し，故障がないと診断したときには，その結果は正しい．

［条件５．１］ PE v，w はともに故障していない． □

［条件５．２］ 任意の健全 PE u に対して，健全な v-u 通路が存在する (Live(N) が連結)． □

表５．１に，本章の結果をまとめる．

表５．１　π^1 (PE v によるリンク l=(v,w) の故障診断) を解くアルゴリズム

<table>
<tr><th>アルゴリズム
ネットワーク</th><th>基本情報
アルゴリズム</th><th>サイズ既知
アルゴリズム</th><th>辺連結度既知
アルゴリズム</th><th>隣接識別子既知
アルゴリズム</th><th>トポロジ・
アルゴリズム</th></tr>
<tr><td>L非同期式</td><td colspan="5">l 以外のリンクや PE に故障がなくても存在しない（定理５．１）</td></tr>
<tr><td>ＰＬ同期式</td><td colspan="5">PLSB1（故障 PE 数，故障リンク数にかかわらず解く）（定理５．２）</td></tr>
<tr><td rowspan="3">L同期式かつ
Ｐ非同期式</td><td rowspan="3">PE や l 以外のリンクに故障がなくても存在しない
（定理５．７(3)）</td><td colspan="2">v，w 以外の任意のある PE が故障している可能性があるとき，
l 以外のリンクに故障がなくても存在しない（定理５．５）</td><td colspan="2" rowspan="3">LSN1
（故障 PE 数，故障リンク数にかかわらず解く）（定理５．４）</td></tr>
<tr><td colspan="2">PE に故障がないとき：</td></tr>
<tr><td>LSS1
（故障リンク数にかかわらず解く）
（定理５．６）</td><td>故障リンク数が⌈CON/2⌉-1 以下のとき：
LSC1
（定理５．７(1)）
故障リンク数が⌈CON/2⌉ 以上のとき存在しない
（定理５．７(2)）</td></tr>
</table>

５．１　Ｌ非同期式ネットワーク

ネットワークがＬ非同期式の場合，次の定理より，l 以外のリンクや PE に故障がなく，各 PE がネットワークの形状に関していかなる情報を持っていても π^1 が解けない．

［定理５．１］ ネットワークがＬ非同期式のとき，Ｐ同期式，Ｐ非同期式にかかわらず，l 以外のリンクや PE に故障がなくても，π^1 を解くトポロジ・アルゴリズムは存在しない．

（証明）Ｌ非同期式ネットワークで，π^1 を解くトポロジ・アルゴリズムＡが存在すると仮定する．

Ａはネットワーク N で l だけが故障していて他に故障がないとき π^1 を解き，l は故障していると診断して停止する．但し，議論を簡単にするため，l は両方向とも（$\langle v,w\rangle$ も $\langle w,v\rangle$ も）故障しているとする．このとき，l は故障しているので，v も w も l からはメッセージを受信していない．

一方，N でどの PE もリンクも故障していないときにＡを実行するとする．N がＬ非同期式なので l の通信遅延が非常に大きい場合がある．このとき，v，w が l からメッセージを受信する前に，各 PEは，l だけが両方向とも故障しているときと同じ動作（定理３．１の証明参照）をし，l が故障していると診断して停止する場合がある．このことは，Ａが π^1 を解くことに矛盾する．　□

５．２　ＰＬ同期式ネットワーク

次に，ネットワークがＰＬ同期式の場合について考える．ここでは，π^1 を解く基本情報アルゴリズム PLSB1 を示す．

まず，PLSB1 の基本方針とそこで使われるメッセージについて，簡単に説明する．このアルゴリズムは，次の３段階からなる．

(1)　〈v,w〉の故障診断を行う．このためにメッセージ《TEST1》が v から w へ〈v,w〉を通じて送信される．このメッセージを w が受信したとき，〈v,w〉が故障していないことが分る．

(2)　(1) で〈v,w〉が故障していないと分ったとき，〈w,v〉の故障診断を行う．このためにメッセージ《TEST2》が w から v へ〈w,v〉を通じて送信される．このメッセージを v が受信したとき，〈w,v〉が故障していないことがわかる．

(3)　l が故障しているかどうかを v が分ると，v は全ての PE を停止させる．このために用いられるメッセージが，《HALT》である．

(注)　この基本方針は本章の他のアルゴリズムにおいても，その基本方針となる．他のアルゴリズムにおいても，メッセージ《TEST1》，《TEST2》，《HALT》を，同じ目的に用いる．

アルゴリズム PLSB1 を以下に示す．但し，B がポート集合を表すとき，B の各ポートに対して任意の順でメッセージ mes を送信することを send(B, mes) と略記する．また，$PORT_u$ は u の全ポートの集合 $\{1,2,...,DEG_u\}$ を表す．

[アルゴリズム PLSB1]

(PL-Synchronized known-Basic-information algorithm for π^1)

【診断 PE v の動作】

(v1)

send(target$_v$,《TEST1》); {<v,w> の故障診断を開始}

repeat(2×(PSYNC+LSYNC)-1)

  receive(MES);

  if 《TEST2》∈MES then begin result$_v$:=false; (v2) へ end

endrepeat;

result$_v$:=true; (v2) へ.

(v2) send(PORT$_v$,《HALT》), halt 命令を実行し，停止する.

【v 以外の各 PE u の動作】

repeat(∞)

  receive(MES);

  if 《TEST1》∈MES then send(b,《TEST2》) (b は《TEST1》を受信したポート)

    {u=w であり，u は <v,w> に故障のないことがわかる．そして，<w,v> の故障診断を開始する.}

  else if 《HALT》∈MES then begin

    send(PORT$_u$-C,《HALT》) (C は《HALT》を受信したポートの集合);

    halt {停止する}

  end

endrepeat. □

アルゴリズム PLSB1 は，4.3.1 節のアルゴリズム PLSBp と使われるメッセージが異なる（PLSB1 の《TEST1》，《TEST2》はそれぞれ，PLSBp の《TEST》，《SAFE》に対応する）だけで，ほとんど同一のアルゴリズムである．

補題４．１と同様に，次の補題が成り立つ．

［補題５．１］ ＰＬ同期式ネットワークで PLSB1 を実行したときに，PE v がメッセージ《TEST2》を受信するための必要十分条件は PE v，w およびリンク l=(v,w) が故障していないことである． □

［定理５．２］ ネットワークがＰＬ同期式のとき，基本情報アルゴリズム PLSB1 は，故障 PE の数，故障リンク数にかかわらず π^1 を解く．

（証明） PLSB1 が停止することは，定理４．３と同様にして示せる．

次に，補題５．１より，l が故障していれば v は《TEST2》を受信しないので，l は故障している（$result_v$=true）と診断し，l が故障していなければ v は《TEST2》を受信するので，l が故障していない（$result_v$=false）と診断する． □

（注１） PE w が故障しているとき，v は《TEST2》を受信しないので，PLSB1 は，l が故障していなくても l は故障していると診断する．

（注２） 条件５．２が成立しなくても，v は l が故障しているかどうか判定できる（v が停止したとき，l が故障していれば $result_v$=true が成り立ち，そうでなければ，$result_v$=false が成り立つ）が，このとき，v との間に健全な通路のない健全 PE は停止しない．v 以外の PE の動作を，<u>repeat</u>(∞) の代りに，<u>repeat</u>(PSYNC+LSYNC) とし，repeat 文から抜けたときに halt 命令を実行し，停止するように変更すれば，条件５．２が成立しなくても全 PE が有限個の命令の実行後停止するので，同様のアルゴリズムで π^1 が解ける．

５．３　Ｌ同期式かつＰ非同期式ネットワーク

条件５．１より，w は健全 PE である．このことと条件５．２より，健全な v-w 通路が存在する．条件５．２は，全ての健全 PE を停止させるために設けた条件であった．しかし，次の定理より，Ｌ同期式かつＰ非同期式ネットワークにおいては，健全な v-w 通路が存在することは，w を停止させるためだけに必要な条件ではなく，v が l が故障しているかどうかを判定するために必要な条件であることが分る．

［定理５．３］ ネットワークがＰ非同期式のとき，Ｌ同期式，Ｌ非同期式にかかわらず，健全な v-w 通路が存在しなければ，故障 PE が存在しなくても，v は l=(v,w) が故障しているかどうかを判定できない．つまり，v が停止したとき（v 以外の PE は停止しなくてもよい），l が故障しているとき，かつ，そのときに限り，$result_v$=true となるトポロジ・アルゴリズムは存在しない．

（証明） Ｐ非同期式ネットワークで，健全な v-w 通路がなくても，v が l の故障を判定できるトポロジ・アルゴリズムＡが存在すると仮定する．

PE w が PE v だけに隣接しているようなネットワーク N で l=(v,w) だけが故障していて他に故障がないときＡを実行すると，Ａの仮定より，$result_v$=true で v は停止する．但し，議論を簡単にするため，l は両方向とも故障しているとする．このとき，v は l からメッセージを受信しない．

一方，どの PE もリンクも故障していない N でＡを実行するとする．N がＰ非同期式なので w の動作が非常に遅い場合がある．このとき，v が l からメッセージを受信する前に，w 以外の各 PE は，l だけが両方向とも故障しているときと同じ動作（定理３．１の証明参照）をし，l が故障していないにもかかわらず，v が $result_v$=true で停止する場合がある．このことは，v が l の故障を判定できることに矛盾する．　□

定理５．３より，Ｐ非同期式ネットワークにおいては，健全な v-w 通路がなければ，v は l が故障しているかどうかを判定できない．一方，w 以外のある健全 PE u に対し，健全な v-u 通路がなくても，v は l が故障しているかどうかを判定できることがある．(但し，u を停止させることができないことがある．) そこで，条件５．２を次のように条件５．２ａと条件５．２ｂに分けることがある．以下でも，これまで通り，条件５．２が成り立つ場合について議論するが，アルゴリズムを示す場合，条件５．２ｂが成り立たないときにどうなるかを，定理や補題の（注）で述べる．

［条件５．２ａ］ 健全な v-w通路が存在する． □

［条件５．２ｂ］ w 以外の任意の健全 PE u に対して，健全な v-u 通路が存在する． □

４．２で，Ｌ非同期式ネットワークでは，PE 故障診断問題を解くトポロジ・アルゴリズムが存在しないことを示した．これは，Ｌ非同期式ネットワークでは，PE の故障と，その PE に接合する全てのリンクの伝播遅延が大きいこととが区別できないためである．しかし，リンク故障診断の場合は，PE が利用できる形状定数によっては，Ｐ非同期式であっても，リンクの故障と，そのリンクに接続する PE の動作が遅いこととが区別できることがある．まず，故障 PE 数，故障リンク数にかかわらず π^1 を解く隣接識別子既知アルゴリズム LSN1 を示す．このアルゴリズムでは，《TEST1》，《TEST2》，《HALT》以外に，以下のメッセージを用いる．

《START1,id(w)》：v が《TEST1》の送信後に送信する．〈v,w〉以外の健全な v-w 通路が存在するなら，その通路を通って w に伝えられる．w が《START1,id(w)》を受信すると，w は自分に接合するいずれかのリンクに《TEST1》が送信されたことが分る．

《START2》: w が《TEST2》の送信後に送信する．〈w,v〉以外の健全な w-v 通路が存在するなら，その通路を通って v に伝えられる．v が《START2》を受信すると，v は〈w,v〉に《TEST2》が送信されたことが分る．

《FAIL》: w が〈v,w〉が故障していると分ったときに，そのことを v に知らせるために使われる．健全な w-v 通路を通って v に伝えられる．

(注) 本章の他のアルゴリズムにおいても，《START1,id(w)》，《START2》，《FAIL》を同じ目的で用いる．

また，各 PE は，論理型の変数 s1，s2 を持つ．s1，s2 の値はプログラム実行開始時にそれぞれ true に初期設定される．次の目的でこれらの変数を利用する．

s1:《START1,id(w)》を受信したときに，このメッセージを中継するために送信するかどうかの判定に用いる．true のとき中継し，false のとき中継しない．《START1,id(w)》を送信するとき，または，〈v,w〉の故障診断が終了したことが分ったとき，false に設定する．

s2:《START2》または《FAIL》を受信したときに，これらのメッセージを中継するために送信するかどうかの判定に用いる．true のとき中継し，false のとき中継しない．《START2》または《FAIL》を送信するとき，false に設定する．

[アルゴリズム LSN1]

(L-Synchronized known-Neighbors algorithm for $\pi_{\underline{1}}$)

【診断 PE v の動作】

(v1)

send(target$_v$,《TEST1》); send(PORT$_v$-{target$_v$},《START1,id(w)》);

repeat(∞)

receive(MES);

if 《TEST2》∈MES then begin result$_v$:=false; (v3) へ end

else if 《FAIL》∈MES then begin result$_v$:=true; (v3) へ end

else if 《START2》∈MES then (v2) へ

endrepeat.

(v2)

repeat(LSYNC-1)

receive(MES);

if 《TEST2》∈MES then begin result$_v$:=false; (v3) へ end

endrepeat;

result$_v$:=true; (v3) へ.

(v3) send(PORT$_v$,《HALT》), halt 命令を実行し，停止する.

【v 以外の各 PE u の動作】

```
(u1)
repeat(∞)
  receive(MES);
  if 《HALT》∈MES then (u3) へ
  else if 《TEST1》∈MES then (u2) へ
  else if 《FAIL》∈MES ∧ s2 then begin
    send(PORT_u-B,《FAIL》) (B は《FAIL》を受信したポートの集合);
    s1:=false; s2:=false
  end
  else if 《START2》∈MES ∧ s2 then begin
    send(PORT_u-C,《START2》) (C は《START2》を受信したポートの集合);
    s1:=false; s2:=false
  end
  else if 《START1,id(w)》∈MES ∧ s1 then
    if id(u)=id(w) then begin
      repeat(LSYNC-1)
        receive(MES);
        if 《TEST1》∈MES then (u2) へ
      endrepeat;
      send(PORT_u,《FAIL》); s1:=false; s2:=false
    end
    else begin {id(u)≠id(w)}
      send(PORT_u-D,《START1,id(w)》)
                    (D は,《START1,id(w)》を受信したポートの集合);
      s1:=false
    end
endrepeat.
```

(u2) {u=w が成り立つ．〈v,w〉が故障していないことがわかる．〈w,v〉の故障診断を開始する．}

send(e,《TEST2》); send($PORT_u$-{e},《START2》)

(e は《TEST1》を受信したポート);

s1:=false; s2:=false; (u1) へ．

(u3) 《HALT》を受信したポートの集合を F とする．send($PORT_u$-F,《HALT》), halt 命令を実行し，停止する． □

アルゴリズムより，以下の性質が成り立つ．

［性質５．１］ LSN1 に関して，以下のことがが成り立つ．

(1) v，w 以外の PE は受信したメッセージを中継する（受信したメッセージと同じメッセージを送信する）だけである．

(2) w が《TEST1》または《START1,id(w)》を受信するまで，他の種類のメッセージは送信されない．

(3) w は《TEST2》か《FAIL》の一方だけを送信する． □

［補題５．２］ L同期式ネットワークで LSN1 を実行したとき，任意の時刻において，次の(1)，(2)が成り立つ．

(1) 任意の PE u において，$s1_u$=false ならば，このとき $s2_u$=false が成り立つか，または，このとき以前に u は《START1,id(w)》を送信したことがある．

(2) ある PE u において $s2_u$=false が成り立つならば，このとき以前に w は《TEST1》または《START1,id(w)》を受信し，《FAIL》または《START2》を送信したことがある．

（証明）(1) の証明：$s1_u$:=false を実行するとき，《START1,id(w)》の送信，$s2_u$:=false のいずれかを実行する．

(2) の証明：$s2_u$:=false は u(=w) が《START2》または《FAIL》を送信するとき，あるいは，u が《START2》または《FAIL》を受信したときに実行される．性質５．１(1)より，u が受信したこれらのメッセージは，w が送信したものが中継されてきたものである．従って，u が $s2_u$:=false を実行する以前に w は《FAIL》または《START2》を送信する．また，w が《FAIL》または《START2》を送信するのは，《TEST1》または《START1,id(w)》を受信したときである． □

［定理５．４］ ネットワークがＬ同期式のとき，隣接識別子既知アルゴリズム LSN1 は，故障 PE 数，故障リンク数にかかわらず π^1 を解く．

（証明）(1)　l に故障がないとき，LSN1 は必ず停止し，停止時に $result_v$= false が成立することを示す．

(v1) で，v は l を通じて w に《TEST1》を送信し，その後，v 以外の全ての隣接 PE に《START1,id(w)》を送信する．性質５．１(2) より，w が《TEST1》または《START1,id(w)》を受信するまで，他のメッセージは送信されないので，w は《TEST1》または《START1,id(w)》を受信するまで，receive 命令を繰り返し実行する．

w が《START1,id(w)》を受信する以前に《TEST1》を受信すれば，w は《TEST2》をリンク l を通じて v に送信する．一方，w が《TEST1》を受信するより前に《START1,id(w)》を受信したとする．性質５．１(1) より，w が受信した《START1,id(w)》は v が送信したものが中継されてきたものである．v が《TEST1》の送信後に《START1,id(w)》を送信したこと，及び，ネットワークがＬ同期式であることより，w が《START1,id(w)》を受信後に receive 命令を LSYNC-1 回実行すれば，この間に《TEST1》を受信する．そして，w は (u2) で《TEST2》をリンク l を通じて v に送信する．

w が《TEST2》を送信すること及び性質５．１(1)，(3) より，v が《FAIL》を受信することはない．従って，v は《TEST2》か《START2》を受信するまで，(v1) で receive 命令を繰り返し実行する．このことから，w が《TEST1》を受信するのと同様に，v が《TEST2》を受信する．

v が《TEST2》を受信すると，v は $result_v$:=false, send($PORT_v$,《HALT》) を実行後，停止する．定理５．２と同様にして，LSN1 が停止することを示せる．

(2)　l が故障しているとき，LSN1 は必ず停止し，停止時に $result_v$=true が成立することを示す．

v は停止する前に，$result_v$:=false または $result_v$:=true を実行する．

そして，$result_v$:=false を実行するのは，《TEST2》を受信したときである．《TEST2》は，l を通じて w から v に送信されるだけなので，〈w,v〉が故障していると $result_v$:=false を実行することはない．また，w が《TEST2》を送信するのは，w が《TEST1》を受信したときである．《TEST1》は，l を通じて v から w に送信されるだけなので，〈v,w〉が故障していると w は《TEST2》を送信しないので，v が $result_v$:=false を実行することはない．従って，l が故障している場合は，LSN1 が停止すれば，停止時に $result_v$= true が成立する．

以下では，〈v,w〉が故障している場合と，〈v,w〉に故障がなく，〈w,v〉が故障している場合に場合分けして，LSN1 が停止することを示す．

(a) 〈v,w〉が故障している場合．

まず，w が必ず《START1,id(w)》を受信することを示す．そのために，w が《START1,id(w)》を受信しないと仮定すると，矛盾が生じることを示す．

v は l を通じて w に《TEST1》を送信し，その後，v 以外の全ての隣接 PE に《START1,id(w)》を送信する．(1) の場合と同様，w は《START1,id(w)》を受信するまで，receive 命令を繰り返し実行する．(〈v,w〉の故障のため，w が《TEST1》を受信することはない．)

条件５．２ａより，健全な v-w 通路が存在する．任意の健全な v-w 通路を P とする．v が w 以外の全ての隣接 PE に《START1,id(w)》を送信すること，v，w 以外の PE は《START1,id(w)》を送信するなら，これを受信したポート以外の全てのポートから送信すること，及び，w が receive 命令を繰り返し実行するにもかかわらず《START1,id(w)》を受信しないことより，この v-w 通路には《START1,id(w)》を受信するが，これを送信しない PE u' が存在する．従って，$s1_{u'}$=false が成り立つ．u' が《START1,id(w)》を送信しないので，補題５．２(1) より，$s2_{u'}$=false が成り立つ．一方，w が《TEST1》も《START1,id(w)》も受信しないことと補題５．２ (2)より，$s2_{u'}$= true が成り立つ．よって，矛盾を生じる．

w は《START1,id(w)》を受信すると，LSYNC-1 回 receive 命令を実行する

が，〈v,w〉の故障のために《TEST1》を受信しないので，send($PORT_w$,《FAIL》)を実行する．条件５．２ａより健全な v-w 通路が存在するので，w が《START1,id(w)》を受信するのと同様に，v は《FAIL》を受信する．v が《FAIL》を受信すると LSN1 が停止することを，(1) と同様にして示せる．

(b) 〈v,w〉に故障がなく，〈w,v〉が故障している場合．

〈v,w〉に故障がないことより，(1) と同様に，w は《TEST1》を受信する．そして，w は send(b,《TEST2》)，send($PORT_w$-{b},《START2》) (b は《TEST1》を受信したポート) を実行する．条件５．２ａより，健全な v-w 通路が存在するので，(a)で w が《START1,id(w)》を受信するのと同様に，v は《START2》を受信する．v が《START2》を受信すると LSN1 が停止することを，(1) と同様にして示せる． □

(注１) w が故障 PE のとき，v は《TEST2》も《FAIL》も受信しないので，アルゴリズム LSN1 は停止しない．

(注２) 条件５．２ｂが満たされなくても，LSN1 で v は l が故障しているかどうか判定できる (定理５．２の (注２) 参照) が，停止しない PE が存在する．

以下では，π^1 を解くサイズ既知アルゴリズム，辺連結度既知アルゴリズム，基本情報アルゴリズムについて考察する．

[定理５．５] ネットワークがＬ同期式かつＰ非同期式で，v，w 以外の PE が故障している可能性があるとき，v，w 以外の故障 PE が高々１個で，l 以外のリンクに故障がなくても，次のことが成り立つ．

(1) π^1 を解くサイズ既知アルゴリズムは存在しない．

(2) π^1 を解く辺連結度既知アルゴリズムは存在しない．

（証明）(1) の証明：l 以外のリンクに故障がなく，v，w 以外の故障 PE が高々１個のときに π^1 を解くサイズ既知アルゴリズムＡが存在すると仮定する．Ａは，ネットワーク N=(P,L) の l，w'（v，w 以外の任意の PE）だけが故障しているとき，π^1 を解き，l が故障していると診断して停止する．但し，l は両方向とも故障しているとする．また，N において，v は w の b-隣接 PE とする．

Nに対し，N'=(P',L') と N' でのポート対応を定理４．５(1) の証明と同様に定義する.(図４．１参照) このとき，w' 以外の各 PE が利用できる形状定数の値は，N の場合と同じである．

どの PE もリンクも故障していない N'で，v が診断 PE，$target_v$=a のときにＡを用いて π^1（v が（v,w'）の故障を診断する問題）を解くことを考える．ネットワークがＰ非同期式なので，w' の動作が非常に遅い場合がある．このとき，w' 以外の PE は，l，w' だけが故障している N で π^1 を解いたときと同じ動作（定理３．１の証明参照）をし，l が故障していると診断して停止する場合があり，Ａが π^1 を解くことに矛盾する．

(2) の証明：N が k-辺連結なら (1) で作った N' も k-辺連結であることが示せる．従って，辺連結度既知アルゴリズムが N' において，w' 以外の各 PE で利用できる形状定数は，N の場合と同じである．このことから，(1) と同様にして証明できる． □

定理５．５より，故障 PE が存在するときは，l 以外のリンクに故障がなくても，π^1 を解くサイズ既知アルゴリズムも辺連結度既知アルゴリズムも基本情報アルゴリズムも存在しない．以下では，故障 PE がないとき，π^1 を解くアルゴリズムについて考える．第３章で示したアルゴリズム SST（故障 PE がない非同期式ネットワークで，故障リンク数にかかわらず Live(N) の生成木を構成するサイズ既知アルゴリズム），CST（故障 PE がない非同期式ネットワークで，故障リンク数が ⌈CON/2⌉-1 以下のとき Live(N) の生成木を構成する辺連結度既知アルゴリズム）を利用して π^1 を解くアルゴリズ

ムを示す．SST，CST は，時間定数 LSYNC を利用しないとみなせば，L同期式かつP非同期式ネットワークにおいても Live(N) の生成木を構成する．

[アルゴリズム LSS1]

(L-Synchronized known-Size algorithm for π^{1})

フェーズ1：SST を用いて Live(N) の根付き生成木 T1 を構成する．このとき，各 PE u は，変数 par_u に親に通じるポートを求め，SON_u に子に通じるポートの集合を求めるものとする．

フェーズ2：各 PE は，次の変数を持つ．

$FAILP_u$：《FAIL》を受信したことのあるポートの集合を記憶する．プログラム実行開始時に，空集合に初期値設定される．

s1：論理型の変数で《START1》を受信したときに，このメッセージを中継するために送信するかどうかの判定に用いる．true のとき中継し，false のとき中継しない．〈v,w〉の故障診断が終了したことが分ったとき，false に設定する．s1 の値はプログラム実行開始時に true に初期設定される．

【診断 PE v の動作】

```
(v1)
if target_v∈SON_v then begin result_v:=false; (v3) へ end
else begin {target_v∉SON_v}
  send(target_v,《TEST1》);
  repeat(LSYNC-1)
    receive(MES);
    if 《TEST2》∈MES then begin result_v:=false; (v3) へ end
    else if 《START2》∈MES then (v2) へ
  endrepeat;
  send(SON_v,《START1》);
  repeat(∞)
    receive(MES);
    if 《TEST2》∈MES then begin result_v:=false; (v3) へ end
    else if 《START2》∈MES then (v2) へ
    else if 《FAIL》∈RMES then begin
      FAILP_v:=FAILP_v∪B (B は《FAIL》を受信したポートの集合);
      if FAILP_v=SON_v then begin result_v:=true; (v3) へ end
    end
  endrepeat.

(v2)
repeat(LSYNC-1)
  receive(MES);
  if 《TEST2》∈MES then begin result_v:=false; (v3) へ end
endrepeat;
result_v:=true; (v3) へ.
```

(v3) send(SON_v,《HALT》), halt 命令を実行する.

【v 以外の各 PE u の動作】

```
repeat(∞)
  receive(MES);
  if 《HALT》∈MES then begin
    send(SON_u,《HALT》); halt {停止する}
  end
  else if 《TEST1》∈MES then begin
    send(b,《TEST2》) (b は《TEST1》を受信したポート);
    send(par_u,《START2》); s1:=false
      {u=w であり, <v,w> が故障していないことがわかる. <w,v> の故障診
      断を開始する.}
  end
  else if 《FAIL》∈MES then begin
    FAILP_u:=FAILP_u ∪ C (C は《FAIL》を受信したポートの集合);
    if FAILP_u=SON_u then send(par_u,《FAIL》)
  end
  else if 《START2》∈MES then begin send(par_u,《START2》); s1:=false end
  else if 《START1》∈MES ∧ s1 then
    if SON_u={} then send(par_u,《FAIL》)
    else send(SON_u,《START1》)
endrepeat.                                                          □
```

［定理５．６］ ネットワークがＬ同期式で，故障 PE がないとき，故障リンク数にかかわらず，サイズ既知アルゴリズム LSS1 は π^1 を解く．

（証明）(1) l が故障していないとき，LSS1 は $result_v$=false で停止することを示す．SST で構成される Live(N) の生成木を T1=(P,L') （故障 PE がないので，頂点集合は P となる）とする．

(a) l=(v,w)∈L' のとき：$target_v$∈SON_v であり，v は l が故障していないと診断し，$result_v$:=false，send(SON_v,《HALT》) を実行後，halt 命令を実行する．この《HALT》は，T1 のリンクを用いて全ての PE に伝えられ，《HALT》を受信した PE は halt 命令を実行する．従って，LSS1 は停止し，停止時に，$result_v$=false が成立する．

(b) l=(v,w)∉L' のとき：send($target_v$,《TEST1》) を実行後，receive 命令を LSYNC-1 回実行する．この間に《TEST2》も《START2》も受信しなければ，v は send(SON_v,《START1》) を実行する．この《START1》は T1 のリンクを用いて w に伝えられる．ネットワークがＬ同期式なので，w は遅くとも《START1》を受信する以前に《TEST1》を受信する．従って，w は send(c,《TEST2》) (c は《TEST1》を受信したポート)，send(par_u,《START2》) を実行する．この《START2》は T1 のリンクを用いて v に伝えられる．v 以外の PE は T1 における全ての子から《FAIL》を受信したときに親に《FAIL》を送信するが，w は《FAIL》を送信しないので，v が全ての子から《FAIL》を受信することはない．従って，v は《TEST2》か《START2》を受信するまで，receive 命令を繰り返し実行する．《START2》を受信すると，receive 命令を LSYNC-1 回実行する．w は《TEST2》送信後に《START2》を送信するので，ネットワークがＬ同期式であることより，この間に，v は《TEST2》を受信する．従って，v は $result_v$:=false を実行する．そして，send(SON_v,《HALT》)，halt 命令を実行する．このとき，LSS1 が停止することは，(a) と同様にして示せる．

(2)　l が故障しているとき，LSS1 は停止し，停止時に，$result_v$=true が成立することを示す．

定理５．４の証明と同様にして，l が故障しているとき，LSS1 が停止すれば，停止時に，$result_v$=true が成立することが示せる．

(a)　〈v,w〉 が故障している場合．

《TEST1》は v から w へ l を用いて送信されるだけなので，〈v,w〉 が故障していることより，《TEST1》を受信する PE は存在しない．また，T1 のリンクを用いて《START1》は全ての PE に伝えられる．そして，《TEST1》を受信する PE は存在しないので，葉から順に v まで，《FAIL》が伝えられる．従って，v は SON_v の全ての PE から《FAIL》を受信する．v が SON_v の全ての PE から《FAIL》を受信すると LSS1 が停止することを，(1) と同様にして示せる．

(b)　〈v,w〉 に故障がなく，〈w,v〉 が故障している場合．

〈v,w〉 に故障がないことより，(1) と同様にして，w が必ず《TEST1》を受信することを示せる．w は《TEST1》を受信すると，send(b,《TEST2》) (b は《TEST1》を受信したポート)，send(par_w,《START2》) を実行する．この《START2》は T1 のリンクを用いて v まで伝えられる．v が《START2》を受信すると LSS1 が停止することを，(1) と同様にして示せる．　□

(注)　故障 PE が存在するとき，または，条件５．２ｂが満たされないとき，アルゴリズム SST が停止しないので，アルゴリズム LSS1 は停止しない．

次に，辺連結度既知アルゴリズムについて考える．

[アルゴリズム LSC1]

(L-Synchronized known-Connectivity algorithm for π^1)

まず，CST を用いて Live(N) の生成木 T1 を構成する．その後の各 PE の動作は LSS1 と同じ． □

[定理５．７] リンク故障診断問題π^1について，ネットワークがＬ同期式かつＰ非同期式で，故障 PE がないとき，次のことが成り立つ．

(1) 故障リンクの数が ⌈CON/2⌉-1 以下のとき，辺連結度既知アルゴリズム LSC1 はπ^1を解く．

(2) 故障リンクの数が ⌈CON/2⌉ 以上のとき，π^1を解く辺連結度既知アルゴリズムは存在しない．

(3) l 以外のリンクに故障がなくても，π^1を解く基本情報アルゴリズムは存在しない．

(証明) (1) の証明：CST は，故障リンクの数が ⌈CON/2⌉-1 以下のとき，Live(N) の生成木を構成する辺連結度既知アルゴリズムである．定理５．６と同様にして，証明できる．

(2) の証明：故障リンクの数が ⌈CON/2⌉ でも，π^1を解く辺連結度既知アルゴリズムＡが存在すると仮定する．

ネットワーク N=(P,L)，N の辺集合 Lf を定理４．７(2) と同様に定義する．Ａは，CON=n-1 のとき，Lf のリンクだけが全て両方向とも故障していて，他に故障がない N でπ^1を解き，lが故障していると診断する．

N に対し，N'=(P',L') と N' のポート対応を定理４．７(2) と同様に定義する．(図４．２参照) N' において，Ａを用いて v が (v,w') を故障診断する問題π^1を解くことを考える．ネットワークがＰ非同期式なので，w' の動作が非常に遅い場合がある．このとき，w' 以外の PE は，Lf のリンクだけが全て両方向とも故障している N でπ^1を解いたときと同じ動作（定理３．

１の証明参照）をし，l が故障していると診断する場合がある．このことは，Ａが π^1 を解くことに矛盾する．

(3) の証明：定理４．７(3) と同様にして，(3) が成立することが示せる.□

（注１） 故障 PE が存在するとき，または故障リンク数が ⌈CON/2⌉ 以上のとき，CST が停止しないので，アルゴリズム LSC1 は停止しない．

（注２） 故障 PE が存在しなければ，故障リンク数が ⌈CON/2⌉-1 以下のとき，ネットワークが CON-辺連結であることより，条件５．２ｂは成り立つ．

６．ｋ－頂点連結拡大構成問題

６．１　定義および記法

ここでは，有向グラフに関する第６章で新たに用いる用語と記法を導入する．また，この章では，有向グラフのみを扱うので，有向グラフのことを，単に，グラフといい，有向辺のことを，単に，辺という．

［記法６．１］

$\langle a_1,a_2,\ldots,a_p\rangle$：$a_1,a_2,\ldots,a_p$ のこの順の系列を表す．そして，p をこの系列の長さという．

有限集合 A に対し，

$\langle\langle A\rangle\rangle$：A の全ての要素からなる長さ $|A|$ の任意の系列を表す．

$\langle\langle A\rangle\rangle_i$：系列 $\langle\langle A\rangle\rangle$ の i 番目の要素（$1\leq i\leq|A|$）を表す．

$m\leq n$ なる任意の整数 m，n に対し，

$[m..n]=\{m,\ldots,n\}$：整数の集合 $\{i\mid m\leq i\leq n\}$ を表す．　□

以下では，特に断らない限り，m，n，i，j，k は正整数を表し，グラフを $G=(V,E)$ と表す．

［記法６．２］

$G=(V,E)$ に対し，

$V(G)$：G の頂点集合を表す．つまり，$V(G)=V$．

$E(G)$：G の辺集合を表す．つまり，$E(G)=E$．

辺 $e=\langle u,v\rangle\in E(G)$ に対し，

$t(e)$：e の始点を表す．つまり，$t(e)=u$．

$h(e)$：e の終点を表す．つまり，$h(e)=v$．

辺集合 $E'(\subseteq E(G))$ に対し，

$t(E')$：E' に属する辺の始点の集合 $\{t(e)|e\in E'\}$ を表す．

$h(E')$：E' に属する辺の終点の集合 $\{h(e)|e\in E'\}$ を表す．

頂点 $v\in V(G)$ に対し，

$IE_G(v)$：v を終点とする辺の集合 $\{e\in E(G)|h(e)=v\}$ を表す．

$OE_G(v)$：v を始点とする辺の集合 $\{e\in E(G)|t(e)=v\}$ を表す．

$Fin_G(v)$：v を終点とする辺の始点の集合 $\{u|(u,v)\in E(G)\}(=t(IE_G(v)))$ を表す．

$Fout_G(v)$：v を始点とする辺の終点の集合 $\{w|(v,w)\in E(G)\}(=h(OE_G(v)))$ を表す．

$ideg_G(v) = |Fin_G(v)|$：v の入次数という．

$odeg_G(v) = |Fout_G(v)|$：v の出次数という． □

G が文脈から明らかなときは，添字 G を省略することがある．

２つの辺 e_1，e_2 が，$t(e_1)\neq t(e_2)$ かつ $h(e_1)\neq h(e_2)$ を満たすとき，e_1，e_2 は独立であるという．また，辺集合 $E'(\subseteq E(G))$ において，E' の任意の相異なる辺が独立のとき，辺集合 E' が独立であるという．

G の相異なる頂点の系列 $\langle v_1,v_2,\ldots,v_m\rangle$ が，各 i $(1\leq i<m)$ に対し $\langle v_i, v_{i+1}\rangle\in E(G)$ となるとき，この系列を v_1-v_m 通路（あるいは，単に，通路）という．そして，頂点 v_1，v_m を，それぞれ，v_1-v_m 通路の始点，終点とよぶ．また，このとき，$m-1$ をこの通路の長さという．但し，１頂点だけからなる系列も長さ０の通路とみなす．

［定義６．１］ 要素数が同じの頂点集合 $V_1(\subseteq V(G))$，$V_2(\subseteq V(G))$ $(m=|V_1|=|V_2|$ とする）に対し，次の (1)，(2) を満たす m 個の通路の集合を (V_1, V_2)-リンクという．

(1) (V_1,V_2)-リンクの任意の相異なる２つの通路は，始点，終点を含めて，頂点を共有しない．

(2) (V_1,V_2)-リンクの任意の通路は，ある頂点 $v_1 \in V_1$，$v_2 \in V_2$ に対する v_1-v_2 通路である．但し，$v_1=v_2$ の場合には，この通路は長さ０の通路になる．

頂点集合 $V_1(\subseteq V(G))$ と頂点 v $(\in V(G))$ に対し，次の (1)，(2) を満たす m 個の通路の集合を (v,V_1)-ファンアウトという．

(1) (v,V_1)-ファンアウトの任意の相異なる２つの通路は，v 以外には，頂点を共有しない．

(2) (v,V_1)-ファンアウトの任意の通路は，ある頂点 $v_1 \in V_1$ に対する v-v_1 通路である．但し，$v=v_1$ の場合には，この通路は長さ０の通路になる．

頂点集合 $V_1(\subseteq V(G))$ と頂点 v $(\in V(G))$ に対し，次の (1)，(2) を満たす m 個の通路の集合を (V_1,v)-ファンインという．

(1) (V_1,v)-ファンインの任意の相異なる２つの通路は，v 以外には，頂点を共有しない．

(2) (V_1,v)-ファンインの任意の通路は，ある頂点 $v_1 \in V_1$ に対する v_1-v 通路である．但し，$v=v_1$ の場合には，この通路は長さ０の通路になる． □

［例６．１］ グラフ G における (V_1,V_2)-リンク，(v,V_1)-ファンアウト，(V_1,v)-ファンインの例を図６．１に示す． □

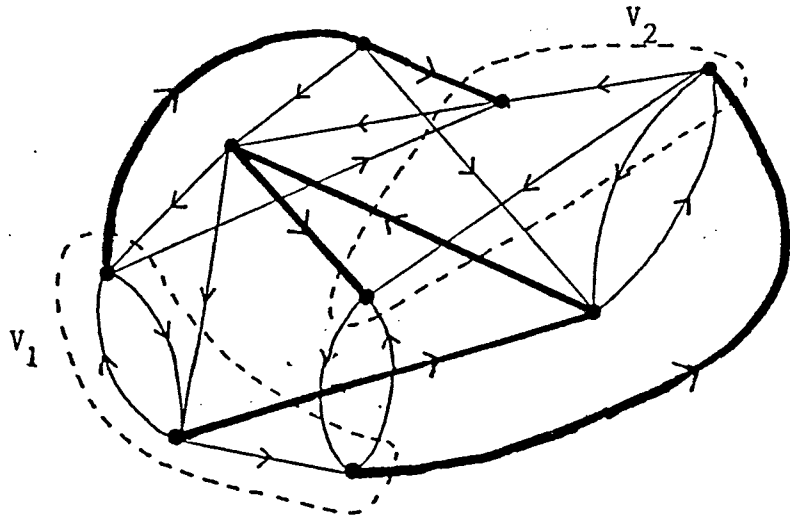
グラフ $G=(V,E)$ と $G'=(V',E')$ に関して，$V \supseteq V'$ かつ $E \supseteq E'$ が成り立つとき，G' を G の部分グラフという．特に，$V'=V$ のとき，G' を G の生成部分グラフという．

２つのグラフ G_1，G_2 が，$V(G_1) \cap V(G_2)=\phi$ を満たすとき，G_1，G_2 が独立であるという．

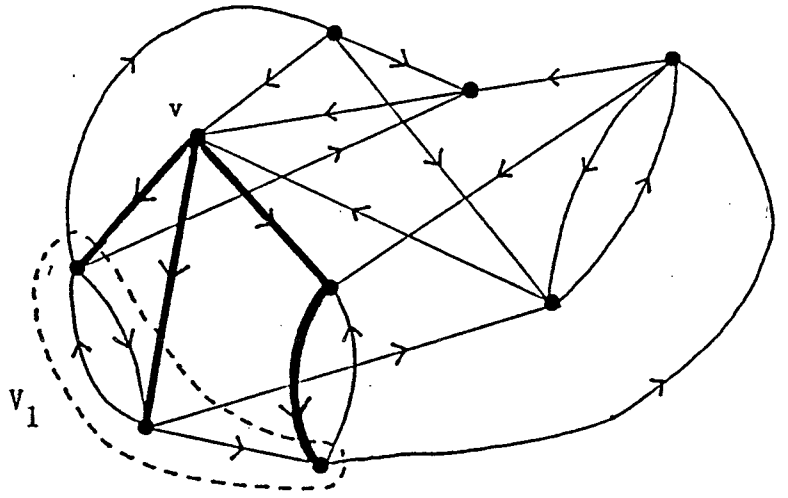
［記法６．３］

グラフ $G=(V,E)$，頂点集合 $V'(\subseteq V)$ と 辺集合 $E'(\subseteq E)$ に対し，

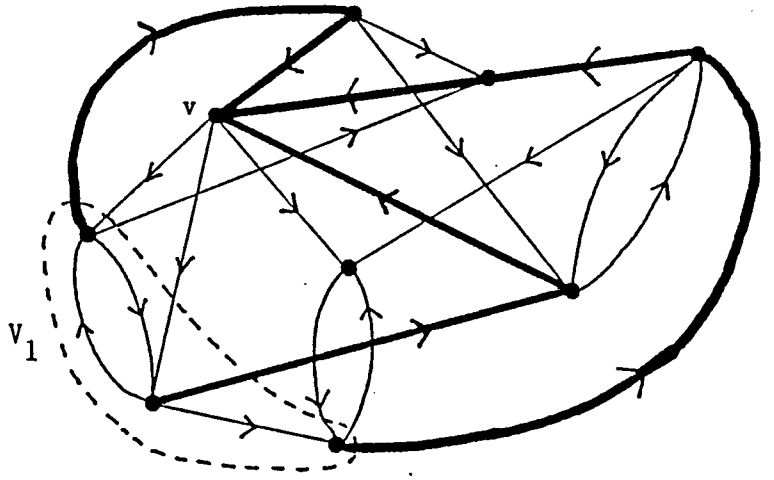
$G[V'] = (V',\{e \in E \mid t(e) \in V'$ かつ $h(e) \in V'\})$：このグラフを頂点集合



(a) A (V_1,V_2)-link.



(b) A (v,V_1)-fanout.



(c) A (V_1,v)-fanin.

図6．1 (V_1,V_2)-リンク，(v,V_1)-ファンアウト，(V_1,v)-ファンイン．

Fig. 6. 1 A (V_1,V_2)-link, a (v,V_1)-fanout and a (V_1,v)-fanin.

を V' とする G の誘導部分グラフという．

G-V'：グラフ G[V-V'] を表す．

G-E'：グラフ (V,E-E') を表す．

u-v[V'] 通路：G[V'] の u-v 通路を表す．つまり，各 i ($1 \leq i \leq m$) に対して $u_i \in V'$ となる u-v 通路 $\langle u(=u_1), u_2, \ldots, v(=u_m) \rangle$ を表す．

(V_1, V_2)[V']-リンク：G[V'] の (V_1, V_2)-リンクを表す．

(v, V_1)[V']-ファンアウト：G[V'] の (v, V_1)-ファンアウトを表す．

(V_1, v)[V']-ファンイン：G[V'] の (V_1, v)-ファンインを表す． □

［定義６．２］ グラフ G=(V,E) において，任意の相異なる２頂点 u，v に対し $\langle u,v \rangle$ も $\langle v,u \rangle$ も E に属するとき，G を完全グラフといい，$K_{|V|}$ と表す．

グラフ G=(V,E) が，次の３つの条件を満たすとき，G を根付き木（あるいは，単に，木）という．

(1) $\mathrm{ideg}(v_0)=0$ となる頂点 v_0 が存在する．このとき，v_0 を根という．

(2) v_0 以外の任意の頂点 v に対し，ideg(v)=1 となる．

(3) 任意の頂点 $v \in V(G)$ に対し，v_0-v 通路が存在する．

木 G において，各頂点 $v \in V(G)$ に対し $\mathrm{odeg}(v) \leq m$ となるとき，G を m-進木という．

$\langle u,v \rangle$ が木の辺のとき，u を v の親といい，v を u の子という．木の頂点 v に対し odeg(v)=0 となるとき，v を葉という．そして，葉以外の木の頂点を内部頂点という．

v_0 を根とする木の頂点 v の深さを，v_0-v 通路の長さと定義する．

v を木 G の頂点とする．{u|G において v-u 通路が存在する} を頂点集合とする G の誘導部分グラフを G の v を根とする部分木という． □

［記法６．４］

$G = (V,E,v_0)$：v_0 を根とする木 G=(V,E) を表す．

$depth_G(v)$：木 G における頂点 v の深さを表す．

$G(v) = (V(v),E(v))$：G の $v(\in V(G))$ を根とする部分木を表す．

$n_G(v) = |V(v)|$：G の v を根とする部分木の頂点数を表す．つまり，$n_G(v)=|V(G(v))|$ である． □

［定義６．３］ 木 $G=(V,E)$ の頂点 u が次の２つの条件を満たすとき，u を G の m-弱重心という．

(1) $n(u)\geq|V|-m$

(2) u の各子 v に対し，$n(v)\leq m$

つまり，木 G からその m-弱重心 u を取り除いて得られるグラフ $G-\{u\}$ は，各連結成分に高々 m 個の頂点を含む．

また，木 $G=(V,E)$ の辺 $\langle u,v\rangle$ が $m\leq n(v)\leq|V|-m$ を満たすとき，$\langle u,v\rangle$ を G の m-橋という．

つまり，木 G からその m-橋 e を除去して得られるグラフ $G-\{e\}$ は，各連結成分に m 個以上の頂点を含む． □

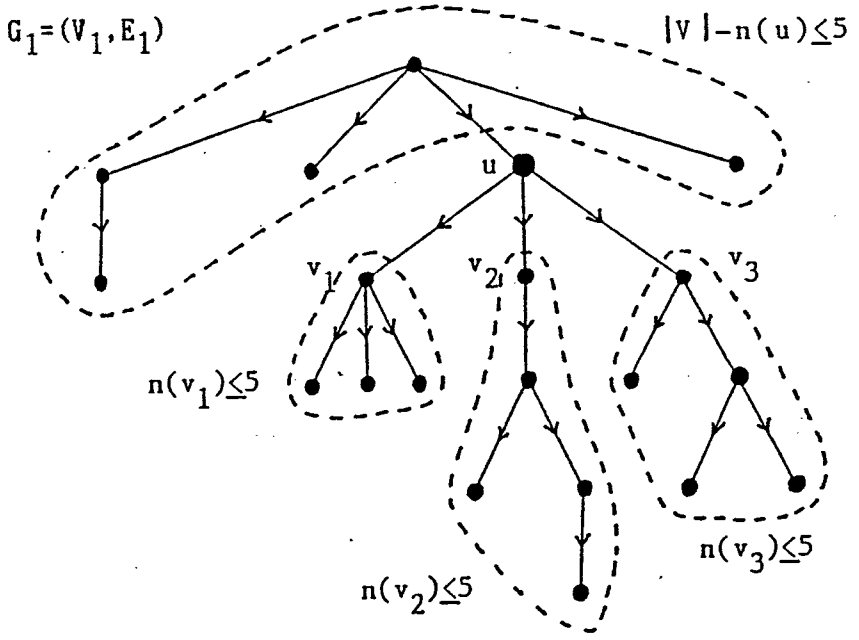
［例６．２］ 図６．２に，5-弱重心と 5-橋の例を示す． □

［定義６．４］ 各内部頂点 v に対し，v の子の間に順序が定義されている木を順序木という．順序木の頂点 v の j 番目の子 $(1\leq j\leq od(v))$ を第 j 子といい，v^j と表す．

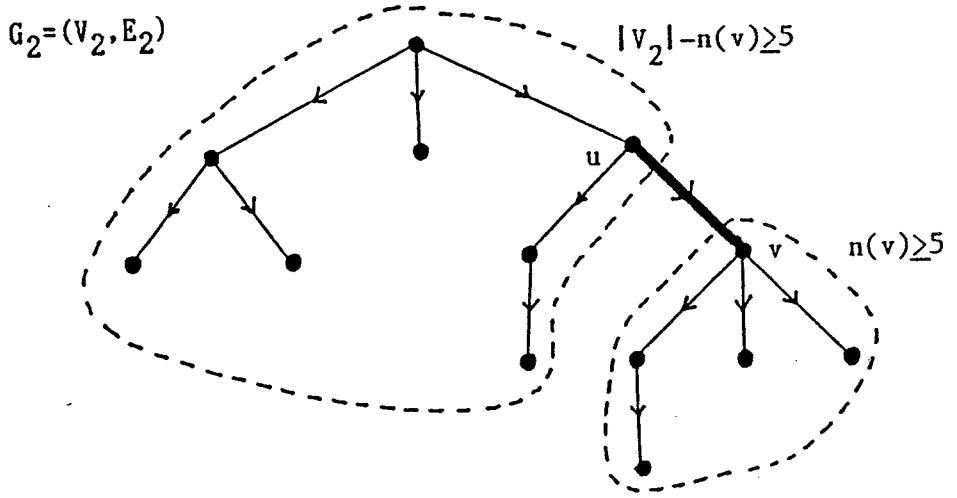
順序木 G における通路 $\langle u_1,u_2,\ldots,u_m\rangle$ が，各 i $(1\leq i\leq m-1)$ に対し $u_{i+1}=u_i^{odeg(u_i)}$ となるとき，この通路のことを G の最右通路という．

順序木 G が，各頂点 v と各 j $(1\leq j\leq odeg(v)-1)$ に対し $n(v^j)\leq n(v^{j+1})$ を満たすとき，G は整列表現されているという． □

［例６．３］ 図６．３に整列表現された木とその最右通路を示す． □



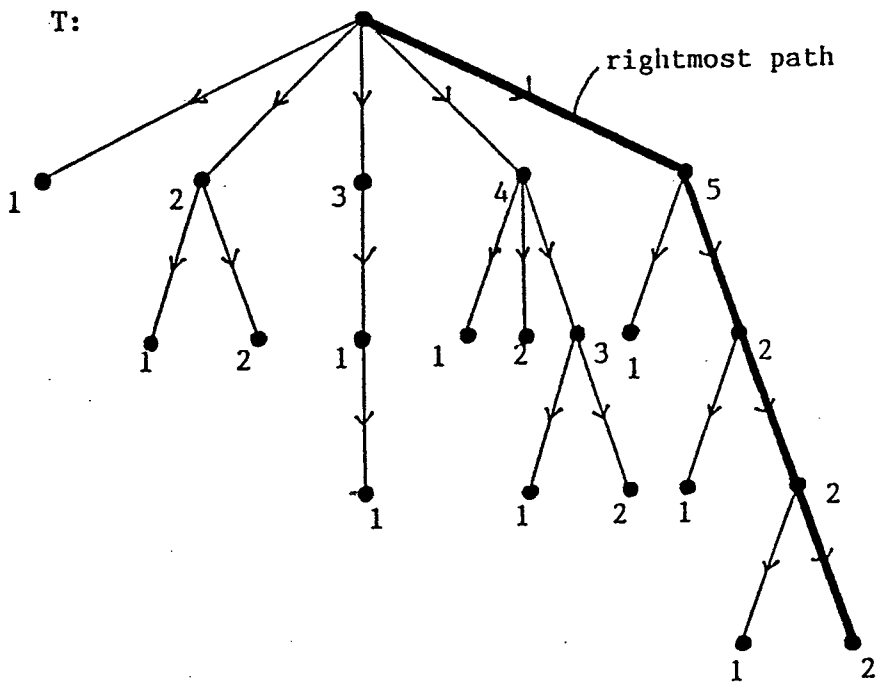
(a) u is a 5-weak-centroid-vertex in G_1.



(b) (u,v) is a 5-bridge in G_2.

図６．２　5-弱重心，5-橋.

Fig. 6. 2　A 5-weak-centroid-vertex and a 5-bridge.



The number represents the order of sons.

図６．３　整列表現された木 T とその最右通路.

Fig. 6. 3 A tree T in the normal-ordering and its rightmost path.

［記法６．５］ グラフ G に対し，$\bar{m}$ で m mod |V(G)| を表す． □

［定義６．５］ グラフ G=(V,E) に対し，全単射 f：V→[0..|V|-1] を考える．f が各辺 $\langle u,v\rangle \in E$ に対し $0<\overline{f(v)-f(u)}\leq m$ を満たすなら，f を m-幅順序という．特に，f が $0<f(v)-f(u)\leq m$ を満たすなら，f を強 m-幅順序という．

順序木 $G=(V,E,v_0)$ に対し，全単射 f：V→[0..|V|-1] が次の２つの条件を満たすとき，f を G の深さ優先順序という．

(1) $f(v_0) = 0$

(2) $f(v) = \sum_{j=1}^{i-1} n(v_0{}^j) + f'(v)+1 \quad (v\in V(v_0{}^i))$

ここで，f' は $G(v_0{}^i)$ の深さ優先順序である．

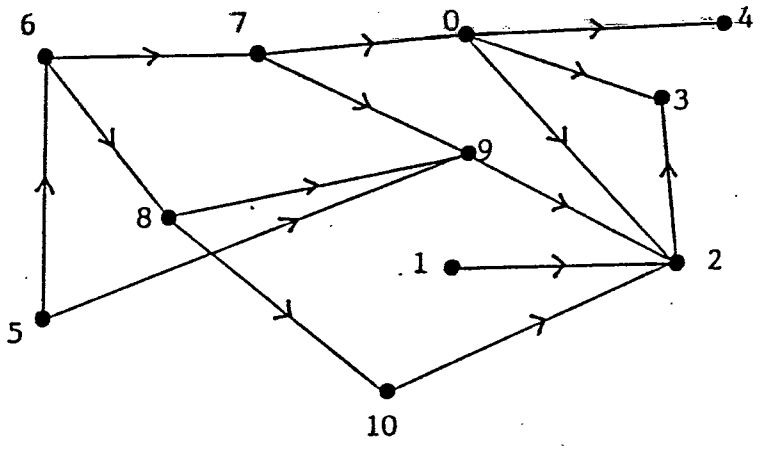
つまり，深さ優先探索(1)で頂点を訪れる順に従って，頂点に 0 から |V|-1 までの整数を割当てるのが深さ優先順序である．

順序木 $G=(V,E,v_0)$ に対し，次の全単射 f：V→[0..|V|-1] を G の幅優先順序という．

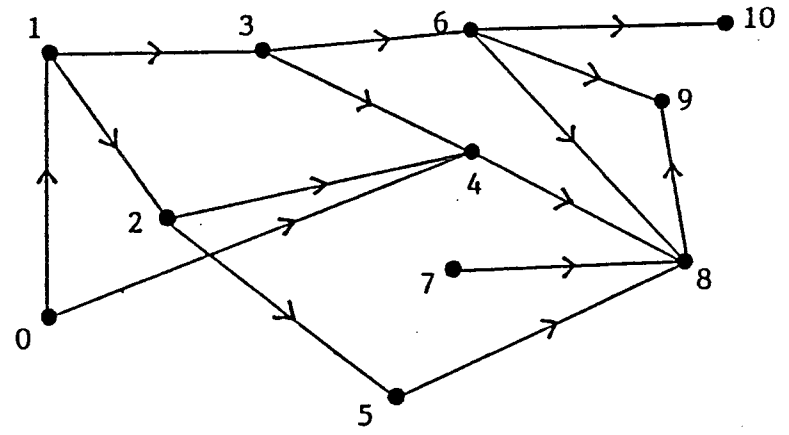
$$
\begin{aligned}
f(v) = {} & |\{w \mid depth(w)<depth(v)\}| \\
& + |\{w \mid depth(w)=depth(v) \text{ かつ} \\
& \qquad w \text{ の親 } w' \text{ と } v \text{ の親 } v' \text{ に対し，} f(w')<f(v')\}| \\
& + |\{w \mid depth(w)=depth(v) \text{ かつ } w \text{ と } v \text{ は同一の頂点 } u \text{ を親とし，} \\
& \qquad i<j \text{ なるある } i,\ j \text{ に対し，} w=u^i,\ v=u^j \text{ となる}\}|
\end{aligned}
$$

つまり，幅優先探索(1)で頂点を訪れる順に従って，頂点に 0 から |V|-1 までの整数を割当てるのが幅優先順序である． □

［例６．４］ 図６．４に，G の 4-幅順序と強 4-幅順序の例を示す．また，図６．５に，順序木 G の深さ優先順序と幅優先順序の例を示す． □



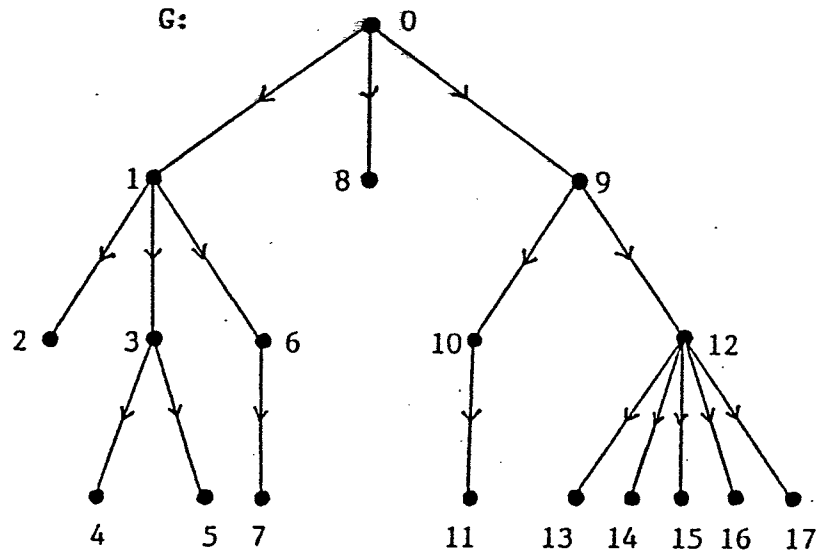
(a) A 4-bandwidth-order.



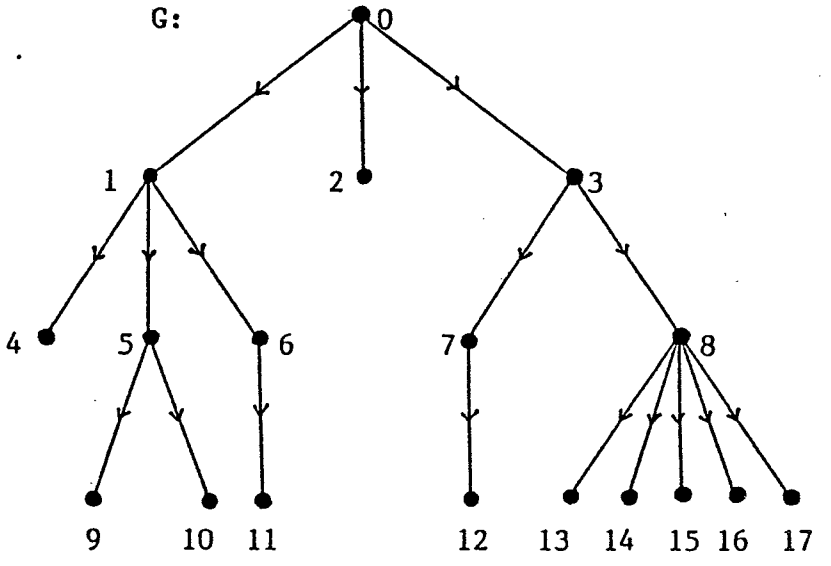
(b) A 4-strong-bandwidth-order.

図６．４　4-幅順序と強 4-幅順序.

Fig. 6. 4　A 4-bandwidth-order and a 4-strong-bandwidth-order.



(a) The depth-first-order of G.



(b) The breadth-first-order of G.

図６．５　深さ優先順序と幅優先順序．

Fig. 6. 5 The depth-first-order and the breadth-first-order.

［定義６．６］ ２つの u-v 通路が，u，v 以外の頂点を共有しないとき，これらの通路は内素であるという．G の相異なる２頂点 u，v に対し，G における互いに内素な u-v 通路の最大数を $P_G(u,v)$ と表す．

グラフ G の任意の相異なる２頂点 u，v に対し，少なくとも m 個の互いに内素な u-v 通路があるとき，G が m-頂点連結であるという．

また，グラフ G の各頂点 $v(\in V(G))$ に対し，$odeg(v)=ideg(v)=m$ が成り立つなら，G は m-正則であるという． □

P(u,v) に関して，次の命題が成り立つ．

［命題６．１］[8] G=(V,E) が完全グラフでなければ，

$$\min_{\substack{\langle u,v\rangle\in V\times V-E\\ u\neq v}} P(u,v)=\min_{\substack{u,v\in V\\ u\neq v}} P(u,v)$$

が成立する． □

つまり，命題６．１は P(u,v) が最小となるような $\langle u,v\rangle\notin E$ なる相異なる u，v が存在することを示している．

［定義６．７］ G=(V,E) と $\langle u,v\rangle\notin E$ なる相異なる u，v に対し，$C_G(u,v)$ は，u から v を分離する（$G-C_G(u,v)$ において u-v 通路が存在しない）要素数最小の頂点集合の要素数を表す． □

［命題６．２］[8] G=(V,E) において，$\langle u,v\rangle\notin E$ なら，

$$|C_G(u,v)| = P_G(u,v)$$

が成立する． □

命題６．１，６．２より，完全グラフでないグラフが m-頂点連結であるこ

とを示すためには，$\langle u,v\rangle \notin E$ なる任意の相異なる２頂点 u，v に対し $P(u,v) \geq m$ または $|C(u,v)| \geq m$ が成り立つことを示せば十分である．

文献 (3) の無向グラフに対する議論と同様にすれば，次の命題が証明できる．

［命題６．３］ グラフ $G=(V,E)$ が m-頂点連結であるための必要十分条件は，$|V| \geq m+1$ が成立し，かつ，任意の頂点 v と $|V'|=m$ なる任意の頂点集合 V' ($\subseteq V$) に対し (v,V')-ファンアウトが存在することである． □

［命題６．４］ グラフ $G=(V,E)$ が m-頂点連結であるための必要十分条件は，$|V| \geq m+1$ が成立し，かつ，任意の頂点 v と $|V'|=m$ なる任意の頂点集合 V' ($\subseteq V$) に対し (V',v)-ファンインが存在することである． □

［定義６．７］ グラフの k-頂点連結拡大構成問題 (k-vertex-connectivity augmentation problem) とは，グラフ $G=(V,E)$ と，重み関数 $f: V \times V \rightarrow R$ (R は非負実数の集合) および正整数 k が与えられたとき，G が k-頂点連結になるように辺を付加する (辺拡大するという) とき，付加辺の重みの和が最小となる辺集合の１つを求める問題である．また，特に，全ての辺の重みが等しいとき，重みなし k-頂点連結拡大構成問題 (k-vertex-connectivity unweighted augmentation problem; 以下，k-VCUAP と略記する) という． □

定義より，本論文のグラフには，多重辺も自己ループも存在しないが，このために，一般性が失われることはない．なぜなら，多重辺を１つの辺に置換し，さらに，自己ループを除去して得られる縮退グラフの k-VCUAP の解が，多重辺や自己ループのあるグラフの k-VCUAP の解となるからである．

６．２　辺拡大に必要な付加辺数

［定理６．１］ グラフ $G=(V,E)$ から辺拡大されたグラフが k-頂点連結ならば，付加された辺の集合 E' について，次式が成り立つ．

$$|E'| \geq \max\left(\sum_{v\in V} \max(k-\mathrm{ideg}_G(v),\ 0),\ \sum_{v\in V} \max(k-\mathrm{odeg}_G(v),\ 0)\right)$$

（証明） G から辺拡大された k-頂点連結グラフを G' とする．任意の頂点 v に対し，次の２つの式が成り立つ．

$$\mathrm{ideg}_{G'}(v) \geq \max(\mathrm{ideg}_G(v),\ k)$$

$$\mathrm{odeg}_{G'}(v) \geq \max(\mathrm{odeg}_G(v),\ k)$$

これらから，定理６．１は証明される． □

定理６．１から，次の系が得られる．

［系６．１］ $G'=(V,E\cup E')$ $(E\cap E'=\phi)$ を $G=(V,E)$ から辺拡大されたグラフとする．このとき，G' が k-正則かつ k-頂点連結なら，辺集合 E' は，G の k-VCUAP の解の１つである． □

系６．１より，与えられたグラフ G を k-正則かつ k-頂点連結グラフに辺拡大することが，G の k-VCUAP を解くための１つの方法である．次の２つの節では，以下の議論に重要な役割を果たす k-正則かつ k-頂点連結グラフを導入する．

６．３　超デイジーチェイン

ここでは，超デイジーチェインとよぶ k-正則かつ k-頂点連結のグラフを定義する．そして，そのいくつかの性質を示す．

［定義６．８］ n，kを $n>k\geq 1$ なる正整数とする．超デイジーチェイン $D_{n,k}=(V,E)$ は次のように定義される．

$V = \{0,1,\ldots,n-1\}$,

$$E = \bigcup_{i=1}^{k} E_i(D_{n,k})$$

但し，$E_i(D_{n,k})=\{\langle j,\overline{j+i}\rangle \mid 0\leq j\leq n-1\}$.

頂点系列 $\langle u,\overline{u+1},\overline{u+2},\ldots,\overline{u+(m-1)}\rangle$ を m-連続頂点系列とよぶ．そして，m-連続頂点系列 $\langle u,\overline{u+1},\overline{u+2},\ldots,v\rangle$ (但し，$v=\overline{u+(m-1)}$) を $\mathrm{Int}(u,v)$ と表す．　□

［例６．５］ 図６．６に $D_{8,3}$ を示す．$D_{8,3}$ において，$\langle 6,7,0,1,2\rangle$ は 5-連続頂点系列であり，$\mathrm{Int}(6,2)=\{6,7,0,1,2\}$ である．　□

［補題６．１］ $D_{n,k}$ $(n>k\geq 1)$ において，V_1, V_2 を $|V_1|=|V_2|\leq k$ なる任意の頂点集合とする．このとき，$D_{n,k}$ において，(V_1,V_2)-リンクが存在する．

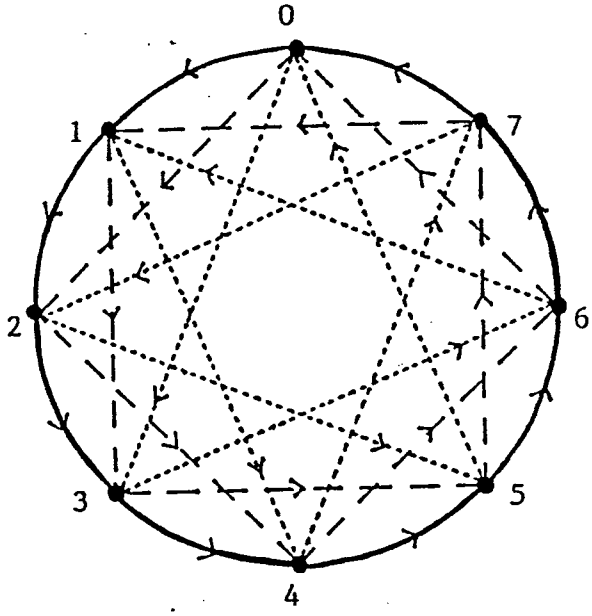
（証明） $|V_1|=|V_2|=p$ $(1\leq p\leq k)$ とする．また，j を $j\in V_1$ なる頂点とする．$0\leq i<n$ なる各 i に対し，

$s_i = |V_1 \cap \mathrm{Int}(j,\overline{j+i})|$,

$t_i = |V_2 \cap \mathrm{Int}(j,\overline{j+i})|$,

$d_i = s_i-t_i$

とする．各 i $(0\leq i<n)$ に対し，$d_i=s_i-t_i\geq 0$ が成立するように j が選べる．一般性を失うことなく，$j=0$ と仮定できる．



⟶: $E_1(D_{8,3})$　　- - -→ : $E_2(D_{8,3})$　　·····→: $E_3(D_{8,3})$

図６．６　超デイジーチェイン $D_{8,3}$.

Fig. 6. 6　A super daisy chain $D_{8,3}$.

$t_i=p$ となる i の中で最小のものを q $(p-1\leqq q\leqq n-1)$ と表す．以下では，p に関する数学的帰納法によって，次のより強い形で補題を証明する．

$D_{n,k}$ において，$(V_1,V_2)[\mathrm{Int}(0,q)]$-リンクが存在する．

$p=1$ のとき：このとき，$V_1=\{0\}$，$V_2=\{q\}$ であり，$(V_1,V_2)[\mathrm{Int}(0,q)]$-リンク $\langle 0,1,\ldots,q\rangle$ が存在する．

帰納仮定：$p=m$ $(1\leqq m<k)$ のとき，$(V_1,V_2)[\mathrm{Int}(0,q)]$-リンクが存在すると仮定する．

$p=m+1$ のとき：q に関する数学的帰納法で証明する．

($q=p-1$ のとき)　このとき，$V_1=V_2=\{0,1,\ldots,p-1\}$ であり，

$(V_1,V_2)[\mathrm{Int}(0,q)]$-リンク $\{\langle 0\rangle,\langle 1\rangle,\ldots,\langle p-1\rangle\}$ が存在する．

(帰納仮定) $p-1\leqq q\leqq h$ $(p-1\leqq h<n-1)$ のとき，$(V_1,V_2)[\mathrm{Int}(0,q)]$-リンクが存在すると仮定する．

($q=h+1$ のとき)

(1)　$q\in V_1$ のとき：$V_1'=V_1-\{q\}$，$V_2'=V_2-\{q\}$ とする．

$(V_1',V_2')[\mathrm{Int}(0,q-1)]$-リンクが存在することを示せばよい．このとき，

$|V_1'|=|V_2'|=p-1=m$ が成立する．

各 i $(0\leqq i\leqq n-1)$ に対し，

$$s_i' = |V_1' \cap \mathrm{Int}(0,i)|,$$

$$t_i' = |V_2' \cap \mathrm{Int}(0,i)|,$$

$$d_i' = s_i'-t_i'$$

とする．明らかに，$t_h'=t_h=p-1$ が成り立つ．

$t_i'=p-1$ となる i の中で最小のものを q' と表す．このとき，$q'\leqq h$ となる．

$$d_i'=s_i'-t_i'=s_i-t_i\geqq 0 \quad (0\leqq i\leqq h),$$

$$d_i'=s_i'-t_i'=(s_i-1)-(t_i-1)=d_i\geqq 0 \quad (h+1\leqq i\leqq n-1)$$

より，各 i $(0\leqq i\leqq n-1)$ に対し，$d_i'\geqq 0$ となる．$p=m$ のときの帰納法の仮定より，$(V_1',V_2')[\mathrm{Int}(0,q-1)]$-リンクが存在する．従って，

$(V_1,V_2)[\mathrm{Int}(0,q)]$-リンクが存在する．

(2) $q \notin V_1$ のとき：$i \in V-V_2$ かつ $0 \leq i \leq h$ となる i の中で最大のものを r と表す．$|V_2|=p$ より，$h+1-p \leq r$ となる．各 i $(r<i \leq h+1)$ に対し，$i \in V_2$ より，$d_i \geq 1$ かつ $d_r \geq 1$ となる．$V_2' = (V_2-\{h+1\}) \cup \{r\}$ とする．このとき，$|V_2'|=p$ となる．

各 i $(0 \leq i \leq n-1)$ に対し，

$t_i' = |V_2' \cap Int(0,i)|$,

$d_i' = s_i-t_i'$

とする．このとき，$t_h'=t_h+1=p$ が成り立つ．

$t_i'=p$ となる i の中で最小のものを q' と表す．このとき，$q' \leq h$ となる．

$d_i'=s_i-t_i'=s_i-t_i=d_i \geq 0$ $(0 \leq i \leq r-1)$,

$d_i'=s_i-t_i'=s_i-(t_i+1)=d_i-1 \geq 0$ $(r \leq i \leq h)$,

$d_i'=s_i-t_i'=s_i-t_i=d_i \geq 0$ $(h+1 \leq i \leq n-1)$

より，各 i $(0 \leq i \leq n-1)$ に対し，$d_i' \geq 0$ が成立する．$p-1 \leq q \leq h$ のときの帰納仮定より，$(V_1,V_2')[Int(0,q-1)]$-リンクが存在する．$1 \leq h+1-r \leq p \leq k$ より，$\langle r, h+1\rangle \in E_{h+1-r}(D_{n,k})$ であり，$(V_1,V_2)[Int(0,q)]$-リンクが存在する． □

［定理６．２］ 超デイジーチェイン $D_{n,k}$ $(n>k)$ は，k-正則かつ k-頂点連結である．

（証明） 定義６．８より，$D_{n,k}$ は k-正則である．

(1) $n=k+1$ のとき：このとき，$D_{n,k}=K_n$ である．故に，$D_{n,k}$ は，明らかに k-頂点連結である．

(2) $n>k+1$ のとき：u, v を $\langle u,v\rangle \notin E(D_{n,k})$ を満たす $D_{n,k}$ の任意の異なる２頂点とする．$U=Fout(u)$, $V=Fin(v)$ とすると，$|U|=|V|=k$ となる．補題６．１より，(U,V)-リンクが存在する．このことから，$D_{n,k}$ において，k 個の互いに内素な u-v 通路が存在する．故に，$D_{n,k}$ は k-頂点連結である． □

系６．１と定理６．２より，G を $D_{|V(G)|,k}$ に辺拡大できると，G に対す

る k-VCUAP は解けたことになる．しかし，$D_{|V|,k}$ に辺拡大できないような k-進木が存在する．例えば，図６．７の 3-進木は， $D_{10,3}$ に辺拡大できない．さらに，次の定理が成立する．

［定理６．３］ 正整数 k とグラフ G=(V,E) が与えられたとき，G を $D_{|V|,k}$ に辺拡大できるかどうかを判定する判定問題は，NP-完全である．G を 2-進木に制限しても，NP-完全である．

定理６．３を証明する前に，２つの補題を示す．

［補題６．２］ G=(V,E) (|V|>k) が $D_{|V|,k}$ に辺拡大できるための必要十分条件は，k-幅順序 $f:V\to[0..|V|-1]$ が存在することである． □

補題６．２は明らかに成り立つ．

［補題６．３］ 木 $T=(V,E,v_0)$ に対し，強 k-幅順序 $f:V\to[0..|V|-1]$ が存在するための必要十分条件は k-幅順序 $g:V\to[0..|V|-1]$ が存在することである．

（証明）必要条件：強 k-幅順序が存在すれば，k-幅順序が存在するのは明らか．

十分条件：$g':V\to[0..|V|-1]$ を $g'(v)=\overline{g(v)-g(v_0)}$ と定義する．g は全単射なので，g' も全単射である．任意の相異なる２頂点 u，v に対し

$$\overline{g'(v)-g'(u)} = \overline{g(v)-g(u)}$$

が成り立つので，g' は k-幅順序である．辺集合 E_1，E_2 を次のように定義する．

$$E_1 = \{\langle u,v\rangle\in E \mid g'(v)>g'(u)\}$$

$$E_2 = E-E_1 = \{\langle u,v\rangle\in E \mid g'(v)\leq g'(u)\}$$

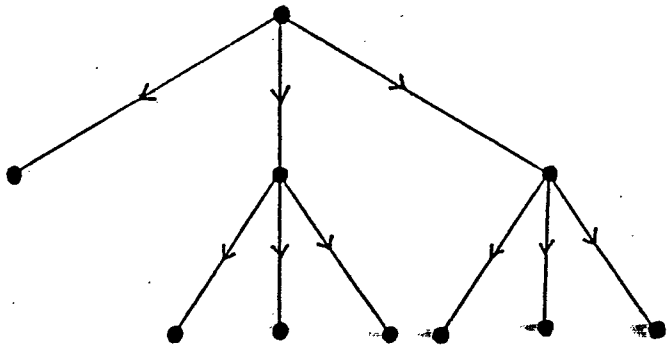


図６．７　$D_{10,3}$ に辺拡大できない 3-進木.

Fig. 6. 7　A ternary tree from which $D_{10,3}$ cannot be constructed by adding arcs.

各 i $(0 \leqq i \leqq |V|-1)$ に対し，$g'(v)=i$ となる頂点 v を v_i と表す．T は木なので，各 i に対し，ただ1つの v_0-v_i 通路が存在する．この v_0-v_i 通路に現れる E_2 の辺数を m_i と表す．

$f:V \to [0..|V|-1]$ を定義するために，V 上の線形順序 $\prec$ を次のように定義する．

$v_i \prec v_j$ となるのは，$m_i < m_j$，または，$m_i = m_j$ かつ $g'(v_i) < g'(v_j)$ のとき，かつ，このときに限る．

つまり，$\prec$ は m_i を第1キー，$g'(v_i)$ を第2キーとする辞書式順序による順序である．V の全ての頂点を $\prec$ によって単調増加の順に並べた頂点系列を考え，f を次のように定義する．

$f(v_i)=i'$ (v_i がこの頂点系列において，$(i'+1)$-番目の頂点のとき)

明らかに，f は全単射である．また，f を次式のように表せる．

$$
\begin{aligned}
f(v_i) &= |\{j \mid 0 \leqq j \leqq |V|-1 \text{ かつ } m_j < m_i\}| \\
&\qquad + |\{j \mid 0 \leqq j \leqq |V|-1 \text{ かつ } m_j = m_i \text{ かつ } g'(v_j) < g'(v_i)\}| \\
&= |\{j \mid 0 \leqq j \leqq |V|-1 \text{ かつ } m_j < m_i\}| + |\{j \mid 0 \leqq j \leqq i-1 \text{ かつ } m_j = m_i\}| \\
&= |\{j \mid 0 \leqq j \leqq i-1 \text{ かつ } m_j \leqq m_i\}| + |\{j \mid i+1 \leqq j \leqq |V|-1 \text{ かつ } m_j < m_i\}| \\
&= i - |\{j \mid 0 \leqq j \leqq i-1 \text{ かつ } m_j > m_i\}| + \\
&\qquad |\{j \mid i+1 \leqq j \leqq |V|-1 \text{ かつ } m_j < m_i\}|
\end{aligned}
$$

以下では，任意の辺 $\langle v_p, v_q \rangle \in E$ に対し $0 < f(v_q)-f(v_p) \leqq k$ が成立することを示すことによって，f が強 k-幅順序であることを示す．

(1) $\langle v_p, v_q \rangle \in E_1$ の場合：$0 < q-p \leqq k$ かつ $m_p = m_q$ が成り立つ．このとき，

$$f(v_q)-f(v_p) = q - p - |\{j \mid p+1 \leqq j \leqq q-1 \text{ かつ } m_j \neq m_p\}|$$

となる．

$$0 \leqq |\{j \mid p+1 \leqq j \leqq q-1 \text{ かつ } m_j \neq m_p\}| \leqq q-p-1$$

より，

$$0 < f(v_q)-f(v_p) \leqq k$$

が成立する．

(2) $\langle v_p, v_q \rangle \in E_2$ の場合：このとき，$p > q$，$0 < \overline{q-p} \leqq k$，$m_q = m_p + 1$ が成り立つ．

このとき，

$$
\begin{aligned}
f(v_q)-f(v_p) = q - p &+ |\{j \mid 0 \leqq j \leqq q \text{ かつ } m_j=m_q\}| \\
&+ |\{j \mid q+1 \leqq j \leqq p-1 \text{ かつ } m_j>m_p\}| \\
&+ |\{j \mid q+1 \leqq j \leqq p-1 \text{ かつ } m_j<m_q\}| \\
&+ |\{j \mid p \leqq j \leqq |V|-1 \text{ かつ } m_j=m_p\}|
\end{aligned}
$$

となる．

$1 \leqq |\{j \mid 0 \leqq j \leqq q \text{ かつ } m_j=m_q\}| \leqq q+1,$

$1 \leqq |\{j \mid p \leqq j \leqq |V|-1 \text{ かつ } m_j=m_p\}| \leqq |V|-p,$

$|\{j \mid q+1 \leqq j \leqq p-1 \text{ かつ } m_j>m_p\}| + |\{j \mid q+1 \leqq j \leqq p-1 \text{ かつ } m_j<m_q\}| = p-q-1$

より，

$0 < f(v_q)-f(v_p) \leqq k$

が成立する． □

（定理６．３の証明） 正整数 k とグラフ G=(V,E) が与えられたときに，G に対する強 k-幅順序が存在するかどうかを決定する決定問題は，有向バンド幅問題 (directed bandwidth problem) とよばれる．文献 (13) に，G を 2-進木に制限しても，有向バンド幅問題が NP-完全であることが示されている．このことと，補題６．２，補題６．３より，定理６．３が証明される． □

次に，$D_{n,k}$ からいくつかの頂点や辺を除去して得られるグラフに関して，そのいくつかの性質を示す．

［定義６．９］ V', E' をそれぞれ，$V' \subseteq V(D_{n,k})$，$E' \subseteq E(D_{n,k})$ なる任意の頂点集合，辺集合とする．また，$D' = (D_{n,k}-V')-E'$ とする．V(D') の頂点からなる系列 $\langle v_1, v_2, \ldots, v_m \rangle$ が，次の性質を満たすとき，この頂点系列を m-擬連続頂点系列とよぶ．

各 i $(1 \leqq i \leqq m-1)$ に対して，$\mathrm{Int}(v_i, v_{i+1})-\{v_i, v_{i+1}\} \subseteq V'$ となる． □

［例６．６］ 図６．８に示す $D_{12,3}-\{3,5,9,10\}$ において，頂点系列 $\langle 2,4,6,7\rangle$ は 4-擬連続頂点系列である． □

［補題６．４］ $D_{n,k}=(V,E)$ $(n>k\geq 2)$ に対し，V' を $|V'|=k-2$ なる V の任意の部分集合，E' を E の任意の独立な部分集合とする．また，$D'=(D_{n,k}-V')-E'$ とする．v_0 を D' の任意の頂点とし，$\langle v_1,v_2\rangle$ を D' における任意の 2-擬連続頂点系列とする．このとき，以下のことが成り立つ．

(a) D' に，v_0-$v_1[\mathrm{Int}(v_0,v_1)]$ 通路，あるいは，v_0-$v_2[\mathrm{Int}(v_0,v_2)]$ 通路が存在する．つまり，$\mathrm{Int}(v_0,v_1)$ の頂点だけを通る v_0-v_1 通路，あるいは，$\mathrm{Int}(v_0,v_2)$ の頂点だけを通る v_0-v_2 通路が存在する．

(b) D' に，v_1-$v_0[\mathrm{Int}(v_1,v_0)]$ 通路，あるいは，v_2-$v_0[\mathrm{Int}(v_2,v_0)]$ 通路が存在する．

（証明）ここでは，(a) だけを証明する．(b) については，同様に証明できる．

一般性を失うことなく，$\overline{v_1-v_0}<\overline{v_2-v_0}$ と仮定できる．$q=|\mathrm{Int}(v_0,v_2)-V'|$ とし，q に関する帰納法で証明する．

$q=2$ のとき：このとき，$v_0=v_1$ であり，(a) は明らかに成り立つ．

帰納仮定：$q=m$ $(m\geq 2)$ のとき，(a) が成立すると仮定する．

$q=m+1$ のとき：$\langle v_1',v_1\rangle$ を D' における 2-擬連続頂点系列とする．このとき，$|\mathrm{Int}(v_0,v_1)-V'|=m$ となる．帰納仮定より，v_0-$v_1[\mathrm{Int}(v_0,v_1)]$ 通路，あるいは，v_0-$v_1'[\mathrm{Int}(v_0,v_1')]$ 通路が存在する．

(1) v_0-$v_1[\mathrm{Int}(v_0,v_1)]$ 通路が存在するとき：(a) は明らかに成り立つ．

(2) v_0-$v_1'[\mathrm{Int}(v_0,v_1')]$ 通路 p が存在するとき：$|V'|=k-2$ 及び $\{v_1',v_1,v_2\}\subseteq V-V'$ より，$e_1=\langle v_1',v_1\rangle$ も $e_2=\langle v_1',v_2\rangle$ も $D_{n,k}-V'$ の辺である．また，E' は独立辺集合なので，e_1，e_2 のいずれかは，E' に属さない．

$e_1\notin E'$ の場合，通路 p と辺 e_1 から v_0-$v_1[\mathrm{Int}(v_0,v_1)]$ 通路が構成できる．

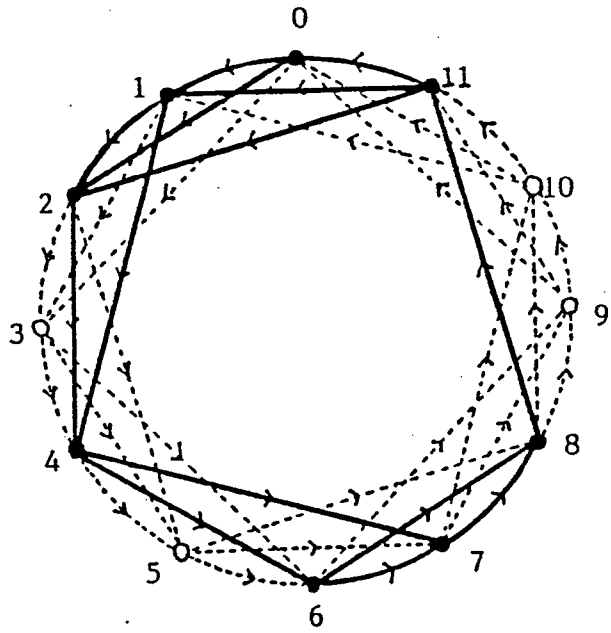
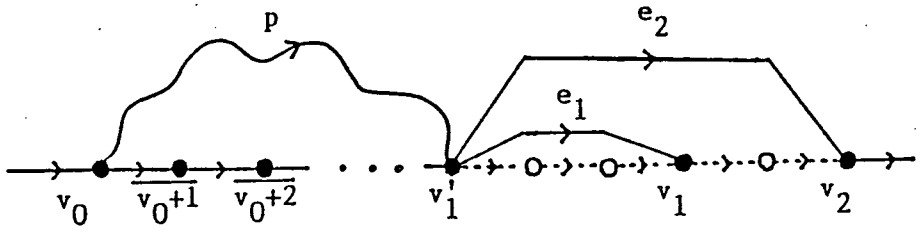


図６．８　$D_{12,3}-\{3,5,9,10\}$ における 4-擬連続頂点系列 $\langle 2,4,6,7\rangle$.

Fig. 6. 8　A pseudo-consecutive vertex sequence $\langle 2,4,6,7\rangle$ in $D_{12,3}-\{3,5,9,10\}$.



o: a vertex in V'

図６．９　v_0-v_1' 通路 p と辺 e_1, e_2.

Fig. 6. 9　A v_0-v_1' path p and arcs e_1 and e_2.

$e_2 \notin E'$ の場合，通路 p と辺 e_2 から v_0-v_2[Int(v_0,v_2)] 通路が構成できる．(図６．９) □

［補題６．５］ $D_{n,k}=(V,E)$ $(n-3 \geq k \geq 3)$ に対し，E' を E の任意の独立な部分集合とする．このとき，$D_{n,k}$-E' は (k-1)-頂点連結である．

（証明） $D=D_{n,k}-E'$ とする．$\langle v_1,v_2\rangle \notin E(D)$ なるある相異なる２頂点 v_1, v_2 に対し，$|C_D(v_1,v_2)| \leq k-2$ であると仮定する．以下では，この仮定の下で，$D'=D-C_D(v_1,v_2)$ において，v_1-v_2 通路が存在することを示し，$|C_D(v_1,v_2)| \leq k-2$ と仮定したことが矛盾を導くことを示す．

(1) $|Fin_{D_{n,k}}(v_2) \cap C_D(v_1,v_2)| \leq k-3$ (6.1)

のとき：$\langle w_1,w_2,w_3,v_2\rangle$ を D' における 4-擬連続頂点系列とする．式 (6.1) より，各 i $(1 \leq i \leq 3)$ に対し，$\overline{v_2-w_i} \leq k$ が成り立つ．従って，各 i $(1 \leq i \leq 3)$ に対し，$e_i=\langle w_i,v_2\rangle \in E$ となる．E' は独立なので，これらの３つの辺のうち，高々１つの辺だけが E' に属する．(図６．10(a))

(1.1) $e_3 \in E(D')$ つまり，$e_3 \notin E'$ のとき：補題６．４より，D' において，v_1-w_3 通路，または，v_1-v_2 通路が存在する．従って，いずれの場合も v_1-v_2 通路が存在することになる．

(1.2) $e_3 \notin E(D')$ のとき：このとき，e_1 も e_2 も D' の辺である．補題６．４より，D' において，v_1-w_1通路，または，v_1-w_2通路が存在する．従って，いずれの場合も v_1-v_2 通路が存在する．

(2) $|Fin_{D_{n,k}}(v_2) \cap C_D(v_1,v_2)| = k-2$ (6.2)

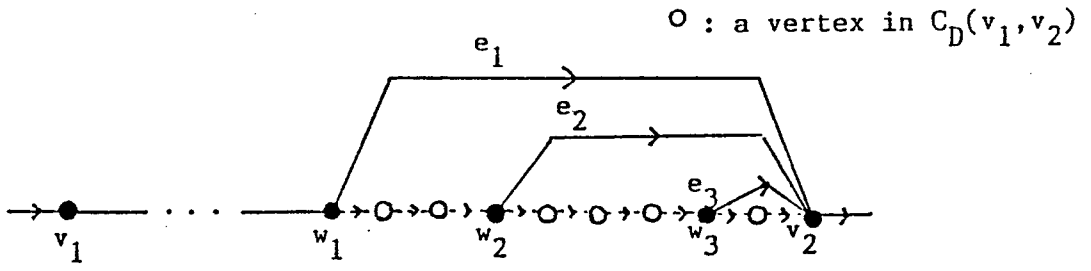
のとき：$\langle w_1,w_2,v_2\rangle$ を D' における 3-擬連続頂点系列とする．式 (6.2) より，

$$e_1 = \langle w_1,v_2\rangle \in E$$

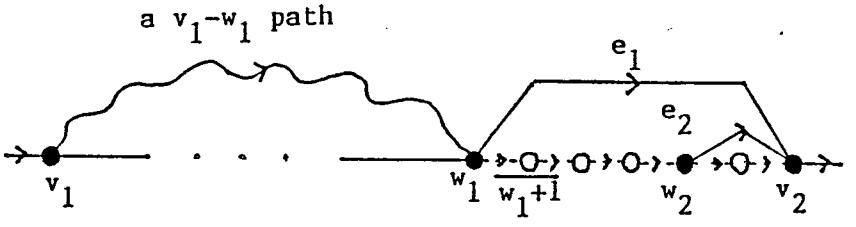
$$e_2 = \langle w_2,v_2\rangle \in E$$

$$C_D(v_1,v_2) = Fin_{D_{n,k}}(v_2)-\{w_1,w_2\} \quad (6.3)$$

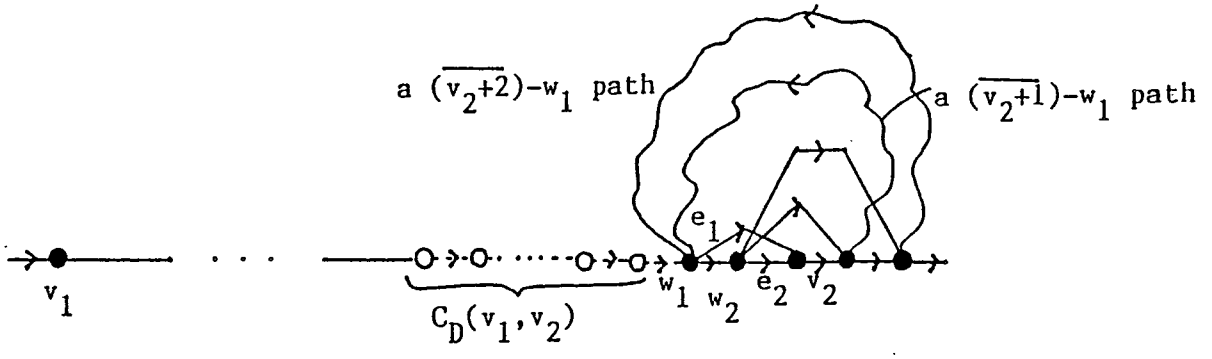
が，成立する．



(a) Case of $|Fin_{D_{n,k}}(v_2) \cap C_D(v_1,v_2)| \leq k-3$.



(b) Case of $C_D(v_1,v_2)=Fin_{D_{n,k}}(v_2)-\{w_1,w_2\}$ and $e_2 \in E'$ and $\overline{w_1+1} \neq w_2$.



(c) Case of $C_D(v_1,v_2)=Fin_{D_{n,k}}-\{w_1,w_2\}$ and $e_2 \in E'$ and $\overline{w_1+1}=w_2$ and $\overline{w_2+1}=v_2$.

図６．10　補題６．５の証明のための図．

Fig. 6. 10 Figures for the proof of Lemma 6. 5.

E' は独立なので，e_1，e_2 の高々一方だけが E' に属する．

(2.1) $e_2 \in E(D')$ のとき：補題６．４より，D' において，v_1-w_2 通路，または，v_1-v_2 通路が存在する．従って，いずれの場合も v_1-v_2通路が存在する．

(2.2) $e_2 \notin E(D')$，つまり，$e_2 \in E'$：このとき，$e_1 \in E(D')$ が成り立つ．

（ｉ） $\langle w_1, w_2, v_2 \rangle$ が $D_{n,k}$ において，3-連続頂点系列でないとき：

$\overline{w_1+1} \neq w_2$ の場合，式 (6.3) より，$\overline{w_1+1} \in C_D(v_1, v_2)$ が成り立つ．$n \geq k+3$ より，$\overline{w_1+1} \notin Fin_{D_{n,k}}(w_1)$ となる．従って，$|Fin_{D_{n,k}}(w_1) \cap C_D(v_1, v_2)| \leq k-3$ が成立し，(1) より，D' において v_1-w_1 通路が存在する．$e_1 = \langle w_1, v_2 \rangle \in E(D')$ なので，v_1-v_2 通路が存在する．(図６．10(b))

同様にして，$\overline{w_1+1} = w_2$ かつ $\overline{w_2+1} \neq v_2$ の場合にも，v_1-v_2 通路が存在することを示せる．

（ii） $\langle w_1, w_2, v_2 \rangle$ が $D_{n,k}$ において，3-連続頂点系列のとき：このとき，$w_2 = \overline{w_1+1}$，$v_2 = \overline{w_2+1}$，$C_D(v_1, v_2) = Int(\overline{v_2-k}, \overline{v_2-3})$ が成り立つ．$e_2 \notin E(D')$，$n-3 \geq k \geq 3$ 及び E' が独立であることより，$\langle w_2, \overline{v_2+1} \rangle \in E(D')$，$\langle w_2, \overline{v_2+2} \rangle \in E(D')$ が成立する．補題６．４より，D' において，$(\overline{v_2+1})$-w_1 通路，または，$(\overline{v_2+2})$-w_1 通路が存在する．従って，D' において，w_2-w_1 通路が存在する．$e_1 \in E(D')$ より，w_2-v_2 通路が存在する．補題６．４より，v_1-w_2 通路，または，v_1-v_2 通路が存在するので，D' において，v_1-v_2 通路が存在する．(図６．10(c)) □

６．４　グラフの合成と頂点連結度

ここでは，２つのグラフ合成法を示す．その１つは，２つの k-正則かつ k-頂点連結のグラフから１つの k-正則かつ k-頂点連結のグラフを構成するものであり，他の１つは，$D_{n,k}$ と１頂点から１つの k-正則かつ k-頂点連結のグラフを構成するものでありる．

また，上述の構成法によって $D_{n,k}$ から構成される k-B-グラフとよぶグラフの族を定義する．

［定義６．10］ グラフ合成法 $\mathrm{dmerge1}(G_1,IE_1,OE_1,G_2,OE_2,IE_2)$ は，次のように定義されるグラフを構成する．ここで，$G_1=(V_1,E_1)$，$G_2=(V_2,E_2)$ は独立なグラフであり，$IE_1=\langle\langle IE_{G_1}(v_1)\rangle\rangle$，$OE_1=\langle\langle OE_{G_1}(v_1)\rangle\rangle$，$OE_2=\langle\langle OE_{G_2}(v_2)\rangle\rangle$，$IE_2=\langle\langle IE_{G_2}(v_2)\rangle\rangle$ である．但し，$v_1\in V_1$，$v_2\in V_2$ は，$\mathrm{ideg}_{G_1}(v_1)=\mathrm{odeg}_{G_2}(v_2)=j$，$\mathrm{odeg}_{G_1}(v_1)=\mathrm{ideg}_{G_2}(v_2)=m$ を満たす頂点である．(図６．11)

$$V = V_1 \cup V_2 - \{v_1,v_2\}$$

$$\begin{aligned} E = ((E_1 \cup E_2) &- (IE_{G_1}(v_1) \cup OE_{G_1}(v_1) \cup IE_{G_2}(v_2) \cup OE_{G_2}(v_2)) \\ &\cup \{\langle t((IE_1)_i),h((OE_2)_i)\rangle \mid 1\le i\le j\} \\ &\cup \{\langle t((IE_2)_i),h((OE_1)_i)\rangle \mid 1\le i\le m\} \end{aligned}$$ □

［定義６．11］ グラフ $G_1=(V_1,E_1)$，頂点 $v\notin V_1$，独立な辺集合 $E_1'(\subseteq E_1)$ に対し，グラフ合成法 $\mathrm{dmerge2}(G_1,E_1',v)$ は，次のようなグラフ $G=(V,E)$ を構成する．(図６．12)

$$V = V_1 \cup \{v\}$$

$$E = (E_1-E_1') \cup \{\langle t(e),v\rangle \mid e\in E_1'\} \cup \{\langle v,h(e)\rangle \mid e\in E_1'\}$$ □

以下では，$\mathrm{dmerge1}(G_1,IE_1,OE_1,G_2,OE_2,IE_2)$ で構成されたグラフを $\mathrm{dmerge1}(G_1,IE_1,OE_1,G_2,OE_2,IE_2)$ と表すことがある．dmerge2 に関しても同

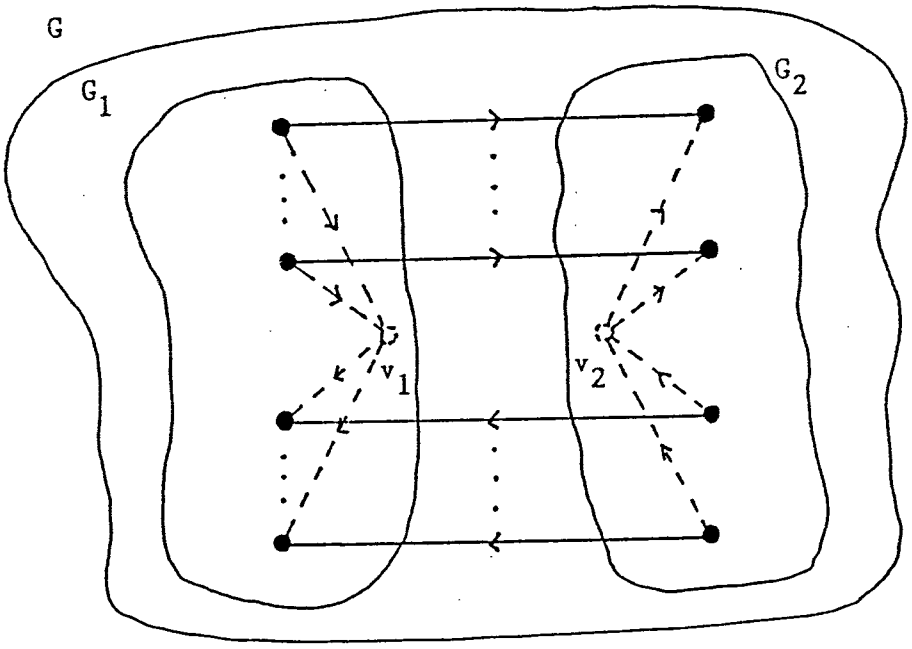
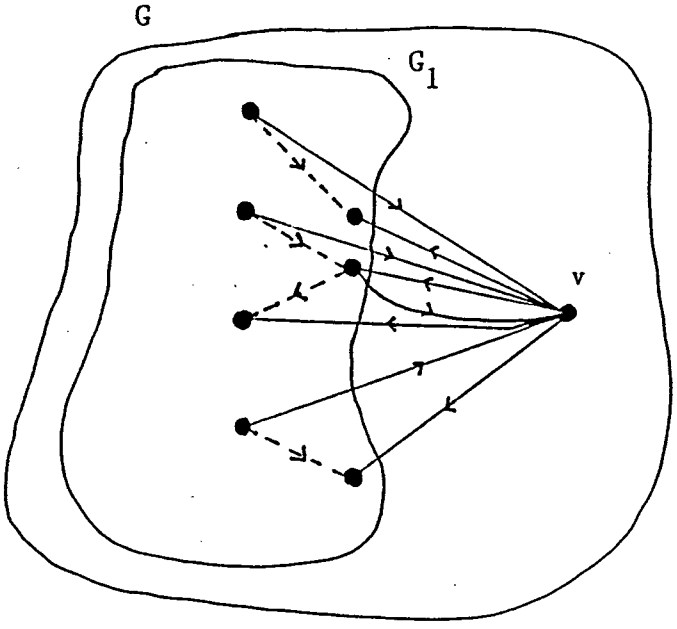


図６．11　グラフの合成法 dmerge1(G_1,IE_1,OE_1,G_2,OE_2,IE_2).

Fig. 6. 11　A digraph composition operation dmerge1(G_1,IE_1,OE_1,G_2,OE_2,IE_2).



---→: an arc in E_1'

図６．12　グラフの合成法 dmerge2(G_1,E_1',v).

Fig. 6. 12　A digraph composition operation dmerge2(G_1,E_1',v).

様である．また，辺系列に現れる辺の順序を気にしないときは，dmerge1(G_1, IE_1,OE_1,G_2,OE_2,IE_2) を，単に，dmerge1(G_1,v_1,G_2,v_2) と略記することがある．

［定理６．４］ 独立なグラフ $G_1=(V_1,E_1)$，$G_2=(V_2,E_2)$ が，ともに k-頂点連結 ($k\geq 2$) であるとする．そして，v_1，v_2 をそれぞれ，

$$odeg_{G_1}(v_1) = ideg_{G_1}(v_1) = odeg_{G_2}(v_2) = ideg_{G_2}(v_2) = k$$

を満たす V_1，V_2 の頂点とする．このとき，グラフ G=dmerge1(G_1,v_1,G_2,v_2) は k-頂点連結である．

（証明）$G=(V,E)$ とする．V の任意の相異なる頂点 u，v に対し，$P_G(u,v)\geq k$ であることを示す．

(1) $u,v\in V_1-\{v_1\}$ のとき：G_1 は k-頂点連結なので，G_1 において，$|P|\geq k$ なる内素な u-v 通路の集合 P が存在する．G_1 において，P の通路のうち，高々１つの通路が v_1 を通る．

P に v_1 を通る通路がない場合，$P_G(u,v)\geq k$ は明らかに成り立つ．

P に v_1 を通る通路が１つだけある場合，この通路を p とすると，p は $\langle u,\ldots,w_1,v_1,w_1',\ldots,v\rangle$ と表せる．ここで，$w_1\in Fin(v_1)$，$w_1'\in Fout(v_1)$ である．

dmerge1 の定義より，$\langle w_1,w_2\rangle\in E$ かつ $\langle w_2',w_1'\rangle\in E$ となる w_2，w_2' が $V_2-\{v_2\}$ に存在する．$P_{G_2}(w_2,w_2')\geq k$ より，少なくとも１つの w_2-$w_2'[V_2-\{v_2\}]$ 通路が存在する．それらの１つを p' とする．u-v 通路 p の $\langle w_1,v_1,w_1'\rangle$ の部分を３つの通路 $\langle w_1,w_2\rangle$，p'，$\langle w_2',w_1'\rangle$ で置換することにより，G において，u-v 通路が構成でき，その通路は P-{p} の任意の通路と独立である．従って，$P_G(u,v)\geq k$ が成り立つ．

(2) $u,v\in V_2-\{v_2\}$ のとき：(1) と同様にして，$P_G(u,v)\geq k$ が成り立つことが示せる．

(3) $u \in V_1-\{v_1\}$ かつ $v \in V_2-\{v_2\}$ のとき：まず最初に，G_1 において，$(u,Fin(v_1))[V_1-\{v_1\}]$-ファンアウトが存在することを示す．

$u \in V_1-\{v_1\}-Fin(v_1)$ の場合，命題６．３及び $ideg(v_1)=k$ より，$(u,Fin(v_1))[V_1-\{v_1\}]$-ファンアウトが存在する．

$u \in Fin(v_1)$ の場合，命題６．３より，$(u,Fin(v_1) \cup \{v_1\}-\{u\})$-ファンアウトが存在する．$ideg(v_1)=k$ より，このファンアウトに含まれる u-v_1 通路は $\langle u,v_1 \rangle$ である．従って，このファンアウトの通路で，その終点が $Fin(v_1)-\{u\}$ に属する通路は，v_1 を通らない．つまり，$(u,Fin(v_1))[V_1-\{v_1\}]$-ファンアウトが存在する．

同様にして，G_2 において，$(Fout(v_2),v)[V_2-\{v_2\}]$-ファンインが存在することが示せる．

また，新たに付加した辺だけからなる $(Fin(v_1),Fout(v_2))[Fin(v_1) \cup Fout(v_2)]$-リンクが存在する．

従って，G において，$(u,Fin(v_1))$-ファンアウト，$(Fin(v_1),Fout(v_2))$-リンク，$(Fout(v_2),v)$-ファンインから，k 個の内素な u-v 通路が構成できる．故に，$P_G(u,v) \geq k$ が成り立つ．

(4) $u \in V_2-\{v_2\}$ かつ $v \in V_1-\{v_1\}$ のとき：同様にして，$P_G(u,v) \geq k$ が示せる． □

［定理６．５］ E_1' をグラフ $D_{n,k}$ の $|E_1'|=k$ なる任意の独立辺集合とする．また，v を $v \notin V(D_{n,k})$ $(n-3 \geq k \geq 3)$ なる任意の頂点とする．このとき，グラフ $G = dmerge2(D_{n,k},E_1',v)$ は，k-正則かつ k-頂点連結である．

（証明） $|E_1'|=k$ 及び dmerge2 の定義より，G は明らかに k-正則である．以下では，G が k-頂点連結であることを示すために，$\langle v_1,v_2 \rangle \notin E$ なる任意の相異なる頂点 v_1，v_2 に対し，$P_G(v_1,v_2) \geq k$ あるいは $|C_G(v_1,v_2)| \geq k$ を示す．

(1) $v_1=v$ のとき：$\langle v_1,v_2 \rangle \notin E$ より，$v_2 \notin h(E_1')$ である．$(v_1,h(E_1'))$-

ファンアウト $\{\langle v,h(e')\rangle | e'\in E_1'\}$ を FO とする．$D_{n,k}$ は k-頂点連結なので，$D_{n,k}$ において，$(h(E_1'),v_2)$-ファンインが存在し，これを FI と表す．$v_2\notin h(E_1')$ より，FI は E_1' の辺を用いていない．従って，FO，FI から，G において，$k(=|h(E_1')|)$ 個の内素な v_1-v_2 通路が構成できる．つまり，$P_G(v_1,v_2)\geq k$ となる．

(2) $v_2=v$ のとき：同様に，$P_G(v_1,v_2)\geq k$ となる．

(3) $v_1\neq v$ かつ $v_2\neq v$ のとき：$|C_G(v_1,v_2)|\leq k-1$ と仮定する．補題６．５より，$D_{n,k}-E_1'$ は (k-1)-頂点連結である．従って，$C_G(v_1,v_2)\subseteq V(G)-\{v\}$，つまり，$v\notin C_G(v_1,v_2)$ となる．一方，$D_{n,k}$ は k-頂点連結なので，$D_{n,k}-C_G(v_1,v_2)$ において，v_1-v_2 通路が存在する．その v_1-v_2 通路を，$p=\langle w_1(=v_1),w_2,\ldots,w_q(=v_2)\rangle$ と表す．$I=\{i|1\leq i\leq q-1$ かつ $\langle w_i,w_{i+1}\rangle\in E_1'\}$ とする．

$I=\phi$ の場合，$G-C_G(v_1,v_2)$ において，明らかに，v_1-v_2 通路 p が存在し，矛盾を生じる．

$I\neq\phi$ の場合，$i_1=\min I$，$i_2=\max I$ とする．このとき，$v\notin C_G(v_1,v_2)$ 及び dmerge2 の定義より，$G-C_G(v_1,v_2)$ において，$\langle w_{i_1},v\rangle$，$\langle v,w_{i_2+1}\rangle$ が存在する．　従って，$G-C_G(v_1,v_2)$ において，v_1-v_2 通路 $\langle w_1(=v_1),w_2,\ldots,w_{i_1},v,w_{i_2+1},\ldots,w_q(=v_2)\rangle$ が存在し，矛盾を生じる．　□

次に，k-B-グラフと呼ばれる k-正則かつ k-頂点連結グラフを導入する．

［定義６．12］ 任意の正整数 k ($k\geq 2$) に対して，$\{D_{n,k}|n>k\}$ を含み，$k=2$ の場合，次の操作 M1 について，また，$k\geq 3$ の場合，操作 M1，M2 について閉じている最小のグラフの族を BD_k と定義する．

M1：BD_k の任意のグラフを $B_1=(V_1,E_1)$，$B_2=(V_2,E_2)$ とし，任意の頂点を $v_1\in V_1$，$v_2\in V_2$ とする．このとき，グラフ $\mathrm{dmerge1}(B_1,v_1,B_2,v_2)$ は，BD_k に属する．

M2：$n-3\geq k\geq 3$ なる 任意の n に対し，$D_{n,k}=(V,E)$ とする．$|E'|=k$ なる任

意の独立辺集合 $E'(\subseteq E)$，任意の頂点 $v \notin V$ に対し，グラフ $dmerge2(D_{n,k}, E', v)$ は，BD_k に属する．

また，BD_k に属するグラフを k-B-グラフと呼ぶ． □

［定理６．６］ 任意の k-B-グラフ（$k \geq 2$）は，k-正則，かつ，k-頂点連結である．

（証明） k-B-グラフを得るために用いた（定義６．12の）操作 M1 の回数に関する数学的帰納法を用いて，定理６．６を証明できる． □

6．5　k-進木の k-頂点連結拡大構成問題

この節では，任意の k－進木 (k≧2) を k-B-グラフ に辺拡大するアルゴリズムを示し，その計算時間がオーダ的に最適であることを示す．

6.5.1 アルゴリズム KDTB

[アルゴリズム KDTB]

(Algorithm for augmenting a k-ary directed tree to a k-B-digraph)

【入力】 k-進木 Tin=(Vin,Ein,v_0) の隣接リスト L，及び，|Vin|>k≧2 なる正整数 k．但し，Vin の頂点は正整数 [1..|Vin|] で表されているものとする．

【出力】 Tin から辺拡大された k-B-グラフ Bout=(Vout,Eout)．但し，グラフ Bout は，次のデータ構造を持つ．

```
type
  vertex=1..maxvertices;
  barc=record
         head:vertex;
         flag:boolean
       end;
  BGRAPH=array [vertex,1..maxk] of barc;
var
  Bout:BGRAPH;
```

アルゴリズムの実行時に，頂点数は |Vin| から 高々3|Vin| まで増加するが (6.5.3 節参照)，定数 maxvertices は，それら全ての頂点を表すのに充分大きいものとする．

頂点 v∈Vout(=Vin)，正整数 j (1≦j≦k) に対し，Bout[v,j]=b とする．このとき，アルゴリズム終了時の b.flag は，

(v,b.head)∈Ein⊆Eout のとき，かつこのときに限り b.flag=true

となり，

(v,b.head)∈Eout-Ein のとき，かつこのときに限り b.flag=false

となる．

Bout は，Tin から辺拡大された k-B-グラフなので，辺集合

{<v,u>|Bout[v,j].head=u かつ Bout[v,j].flag=false, 1≦j≦k, 1≦v≦|Vin|}

は，Tin の k-VCUAP の解の１つである．

【方法】 KDTB アルゴリズムは２つのフェーズからなる．

（フェーズ１） 前処理：L で表されている Tin を整列表現で表し，それを T_0 とする．フェーズ１の手続はここでは省略する．

フェーズ１の出力は，次のデータ構造を持つ．

```
type
  arc=record
        t,h:vertex
      end;
  TREE=record
        odeg,size:array [vertex] of integer;
        s:array [vertex,1..maxk] of vertex
      end;
```

var

T_0:TREE;

型 arc の変数 e は，

e.t=u かつ e.h=v のとき，辺 $\langle u,v \rangle$ を表す．

また，T_0 を型 TREE の変数とする．このとき，

T_0.odeg[v] は，odeg(v) を表し，

T_0.size[v] は，n(v)，つまり，v を根とする部分木の頂点数を表す．

T_0.s[v,j] ($1 \leq j \leq$ odeg(v)) は，v^j，つまり，v の第 j-子を表す．

（フェーズ２） T_0 を k-B-グラフ Bout に辺拡大する：以下に示す手続き bmake(T_0,v_0,k,Bout) によって，T_0 を Bout$\in BD_k$ に辺拡大する．

手続き bmake は，まず，与えられた k-進木をいくつかの k-進木に分割する．この分割は，以下に示す手続き bmake のステップ (3)-(14)，およびステップ (31) で行なわれる．次に，分割して得られた各 k-進木を超デイジーチェインに辺拡大する．この超デイジーチェインへの辺拡大は，ステップ (23)-(24)，(26)-(27)，(29)-(30)，及び (32)-(36) で行なわれる．そして，最後に，ステップ (19)，及び (37)-(38) で，これらの，超デイジーチェインから，グラフ合成法 dmerge1 と dmerge2 を用いて，１つの k-B-グラフを構成する．このとき，与えられた k-進木をいくつかの k-進木に分割するときに除去された辺が，加えられるように k-B-グラフを構成する．

以下では，Pascal 風の言語を用いて手続き bmake を表すが，読みやすさのために，グラフ理論の記法も用いる．特に，手続き update_Bs(B) は，グラフ理論の記法で表されたグラフ B をパラメタとして持つ．実際には，手続き update_Bs(B) は，B の構造を記憶するために，型 BGRAPH の変数 Bs の一部分だけを更新する．但し，手続き bmake 実行時に，変数 Bs が複数個のグラフの構造を記憶していることがあるが，update_Bs(B) は，グラフ B と頂点を共有しないグラフには影響を与えないものとする．

```
procedure bmake(Ts:TREE; v:vertex; k1:integer; var Bs:BGRAPH);
  {最初，Ts は v を根とする木を表す．アルゴリズム実行中には，Ts は最
   初の木から得られるいくつかの順序木を表す.}

  var last:integer;
      {頂点を表す最大の整数を表す.}

  procedure submake(v1,v2:vertex);
    {v1 を根とする木 T から k1-B-グラフを構成する．ここで，v2 は T の
     k1-橋全てを含む部分木の根を表す.}

    var brd:arc;
        w1,w2,vc:vertex;
        sn:integer;

    procedure make_daisy(W0:vertex set; g:order W0→[0..|W0|-1]);
      {W0 を頂点集合とする超デイジーチェインを構成する.}
      begin
        let B=(W0, {<u,v>|0<g(v)-g(u)≦k1});
        update_Bs(B)
      end; {make_daisy}
```

```
procedure dmerge_1(u1,u2:vertex; e:arc);
  {規則 M1 により，k1-B-グラフを構成する.}
  begin
    u1, u2 を含む k1-B-グラフをそれぞれ，G1, G2 とし，B=
    dmerge1(G1,u1,G2,u2) (但し，e を付加するために，
    <<IE(u1)>>1=(e.t,u1)，<<OE(u2)>>1=(u2,e.h)) とする；
      {dmerge1 は，６．４節で導入したグラフ合成法である.}
    update_Bs(B)
  end; {dmerge_1}

procedure dmerge_2(W1:vertex set; Ed1:arc set; u:vertex);
  {規則 M2 により，k1-B-グラフを構成する.}
  begin
    頂点集合を W1 とするグラフを G1 とし，B=dmerge2(G1,Ed1,u) と
    する；
      {dmerge2 は，６．４節で導入したグラフ合成法である.}
    update_Bs(B)
  end; {dmerge_2}
```

```
begin {submake}
(1)   Ts の v1 を根とする木を T=(V,E,v1) とする;
(2)   if T(v2) の最右通路に k1-橋がある then begin
        {規則 M1 により, k1-B-グラフを構成する.}
(3)     brd:=(T(v2) の最右通路上で, v2 に最も近い k1-橋);
(4)     sn:=Ts.odeg[brd.t]; {Ts.s[brd.t,sn]=brd.h}
(5)     w1:=last+1; w2:=last+2; last:=last+2;
          {w1, w2 は新たに導入した頂点を表す.}
(6)     with Ts do begin
(7)       s[brd.t,sn]:=w1;
(8)       odeg[w1]:=0;
(9)       size[w1]:=1;
(10)      s[w2,1]:=brd.h;
(11)      odeg[w2]:=1;
            {ステップ (7)-(11) で, T を２つの木に分割する.}
(12)      T1=(V1,E1), T2=(V2,E2) を次のように定まる木とする:
          {図６．13}
            V1 = (V-V(brd.h)) U {w1}
            E1 = E(T[V-V(brd.h)]) U {<brd.t,w1>}
            V2 = V(brd.h) U {w2}
            E2 = E(T[V(brd.h)]) U {<w2,brd.h>};
(13)      size[v1]:=size[v1]-size[brd.h]+1;
(14)      size[w2]:=size[brd.h]+1
            {size[v1], size[w2] をそれぞれ |V1|, |V2| に設定する. こ
             れらの値は, (2) で k1-橋を求めるとき, (21) で (k1-1)-弱
             重心を求めるときに利用する.}
        end;
```

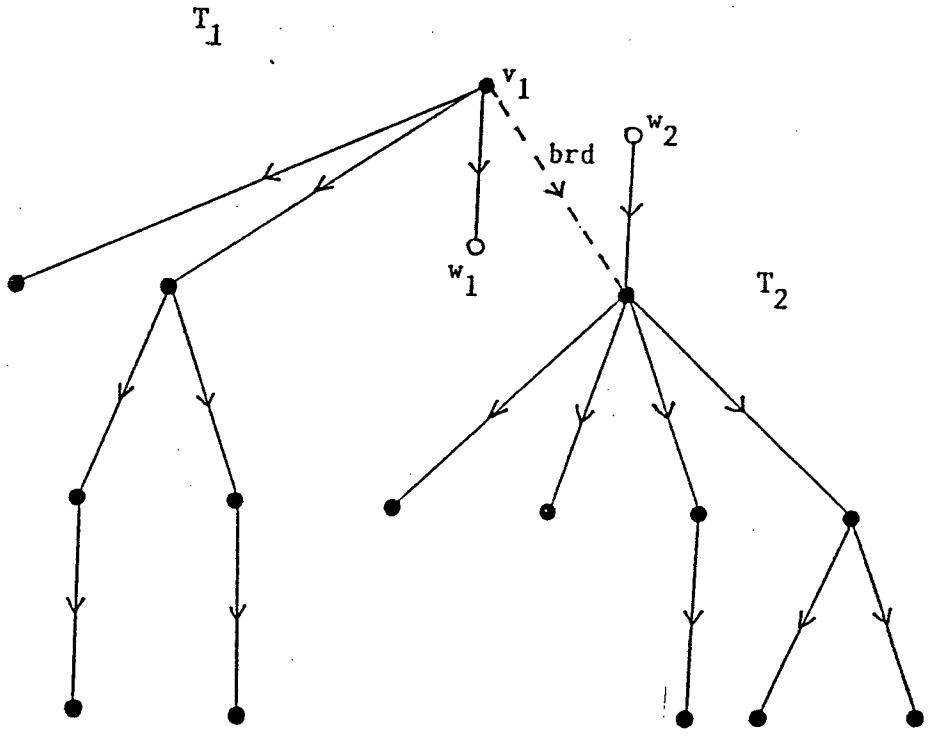


図６．13　ステップ (7)-(11) で作られる k1-進木 T_1, T_2.(k1=4)

Fig. 6. 13 The k1-ary trees T_1 and T_2 produced in the step (7)-(11).

(k1=4)

(15) T_1(brd.t) を整列表現に直す;

{T_2 は既に整列表現されている.}

(16) submake(v_1,brd.t);

{T_1 を k1-B-グラフに辺拡大する.}

(17) submake(w_2,w_2);

{T_2 を k1-B-グラフに辺拡大する.}

(18) B_1, B_2 をそれぞれ, T_1, T_2 から構成された k1-B-グラフとする;

(19) dmerge_1(w_1,w_2,brd);

{グラフ合成法 dmerge1 によって, B_1, B_2 から k1-B-グラフを構成する. このとき, (7) で除去された辺 brd が付加される.}

<u>end</u>

<u>else</u> {k1-橋がない} <u>begin</u>

(20) T を整列表現に直す;

(21) vc:=(T の最右通路上の (k1-1)-弱重心);

(22) <u>if</u> Ts.odeg[vc]≦2 <u>then</u> <u>begin</u>

{T を $D_{|V|,k1}$ に辺拡大する.}

(23) T の深さ優先順序 f:V→[0..|V|-1] を求める;

(24) make_daisy(V,f)

<u>end</u>

(25) <u>else</u> <u>if</u> (T の葉の数)≦k1 <u>then</u> <u>begin</u>

{T を $D_{|V|,k1}$ に辺拡大する.}

(26) T の幅優先順序 f:V→[0..|V|-1] を求める;

(27) make_daisy(V,f)

<u>end</u>

(28) <u>else</u> <u>if</u> $|V|=k1+3$ <u>then</u> <u>begin</u>

{T を $D_{|V|,k1}$ に辺拡大する.}

(29) T の深さ優先順序 $f:V\to[0..|V|-1]$ を求める;

(30) make_daisy(V,f)

<u>end</u>

<u>else</u> <u>begin</u>

{規則 M2 によって, k1-B-グラフを構成する.}

(31) $T_3=(V_3,E_3)$ をグラフ $T[V-\{vc\}]$ とする;

{T_3 を $B_3=D_{|V_3|,k1}$ に辺拡大する.}

(32) $T[V-V(vc)]=(V_{30},E_{30})$ の深さ優先順序 $f_0:V_{30}\to[0..|V_{30}|-1]$ を求める;

(33) $T(vc^j)=(V_{3j},E_{3j})$ $(1\leq j\leq odeg_T(vc))$ の深さ優先順序 $f_j:V_{3j}\to[0..|V_{3j}|-1]$ を求める;

(34) $f:V_3\to[0..|V_3|-1]$ を次のように定める:

$$f(u):=f_i(u)+\sum_{j=0}^{i-1}|V_{3j}| \quad (u\in V_{3i} \text{ のとき});$$

(35) make_daisy(V_3,f);

(36) $B_3=(V_3,E_3')$ をステップ (35) で構成された $D_{|V_3|,k1}$ とする;

(37) 以下の３条件を満たす独立辺集合 $Ed(\subseteq E_3'-E_3)$ を求める:

$|Ed| = k1$,

$Ed \supseteq \{\langle vc',vc^1\rangle\}\cup\{e\in E_1(B_3)\,|\,h(e)=vc^i\ (2\leq i\leq odeg(vc))\}$,

$Ed-\{\langle vc',vc^1\rangle\} \subseteq E_1(B_3)$

(但し, $vc\neq v_1$ なら vc' は vc の親,

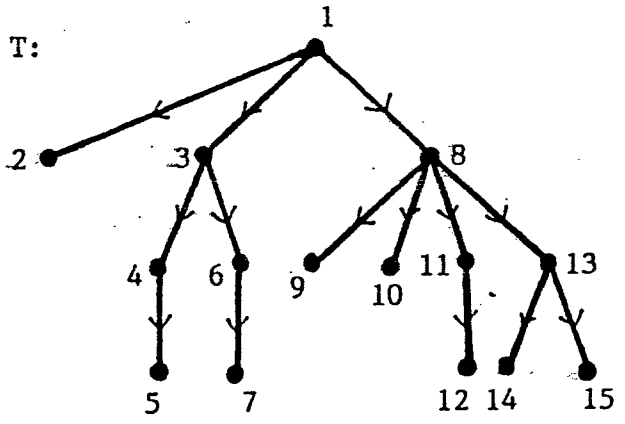
$vc=v_1$ なら $\langle vc',vc^1\rangle\in E_1(B_3)$);

{$E_1(B_3)$ は, 定義６．８で定義した超デイジーチェイン B_3 の辺集合}

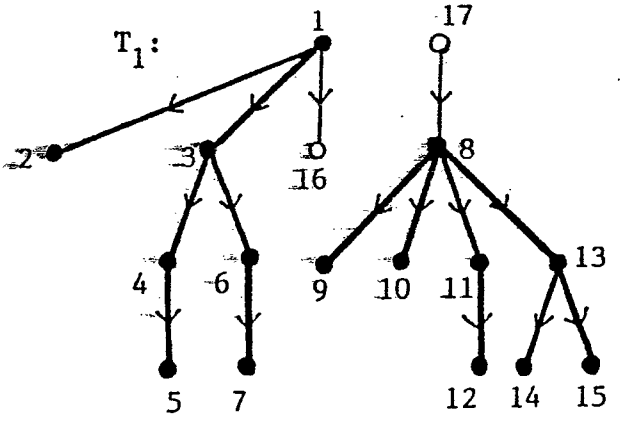
```
(38)        dmerge_2(V3,Ed,vc)
          end
        end; {submake}

    begin {bmake}
(39)    last:=(最初の木の頂点数);
(40)    submake(v,v)
    end; {bmake}
```

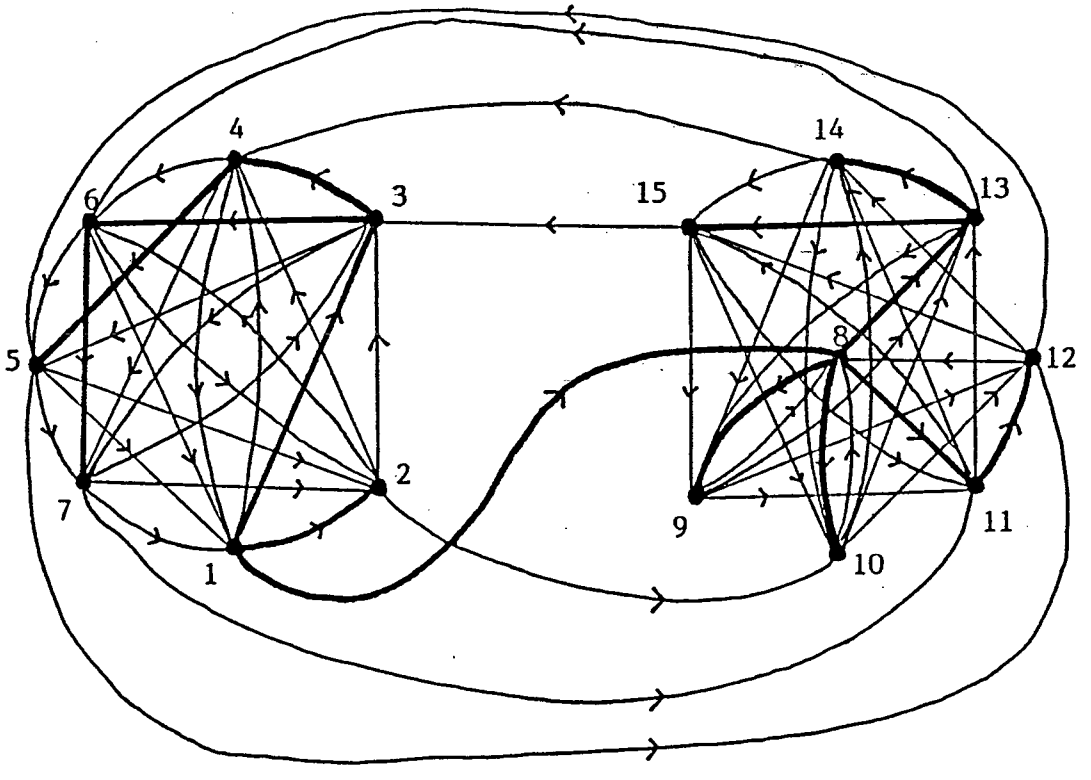
［例６．７］ アルゴリズム KDTB によって， 4-進木を 4-B-グラフに辺拡大する様子を図６．14 に示す．



(a) A given 4-ary tree T.



(b) Division of T.



(c) A 4-B-digraph constructed from T.

図6．14 KDTB アルゴリズムによる 4-進木の 4-B-グラフへの辺拡大．

Fig. 6. 14 The process of augmentation of a 4-ary tree to a 4-B-digraph by KDTB algorithm.

6.5.2 アルゴリズム KDTB の正当性

手続き submake によって，木 $T=(V,E,v_1)$ から辺拡大されたグラフを $B=(V_B,E_B)\in BD_{k1}$ とする．

アルゴリズム KDTB の正しさを証明するために，ステップ (16) で手続き submake を呼ぶ回数に関する数学的帰納法を用いて，B が次の条件Cを満たすことを示す．

C：$B\in BD_{k1}$ かつ $V_B=V$ かつ $E_B\supseteq E$

B が条件Cを満たすことを示す前に，手続き submake が無限に繰り返すような木は存在しないことを示す．

［定理６．７］ 手続き submake が無限に繰り返すような木は存在しない．

（証明） k1-橋の定義より，k1-橋 brd ($k1(=k)\geq 2$) に対し，

$|V-V(brd.h)| \geq k1$ かつ $|V(brd.h)| \geq k1$

が成り立つ．$T_1=(V_1,E_1)$，$T_2=(V_2,E_2)$ をステップ (7)-(11) で得られた木とする．このとき，$V_1=(V-V(brd.h))\cup\{w_1\}$ より，

$|V_1| \leq |V|-k1+1 \leq |V|-1$

が成り立つ．同様に，

$|V_2| \leq |V|-1$

が成り立つ．従って，手続き submake の再帰呼出しを繰り返すと，submake が適用される木の頂点数は，単調減少する．従って，いつかは k1-橋が存在しなくなるので，submake が無限に繰り返されることはない． □

数学的帰納法の基礎として，手続き submake のステップ (24)，(27)，

(30) 及び (38) で構成された B が条件Cを満たすことを示す.

まず最初に，手続き submake のステップ (20) における任意の木 T には，k1-橋が存在しないことを示す.

［補題６．６］ T を k1-橋 が存在する木とする．そして，T の全ての k1-橋が $T(v_2)$ に存在するような頂点を v_2 とし，$T(v_2)$ は整列表現で表されているとする．このとき，T の全ての k1-橋が $T(t(e))$ に存在するような k1-橋 e が $T(v_2)$ の最右通路に存在する.

（証明） $T(v_2)$ の最右通路を $\langle u_0(=v_2),u_1,\ldots,u_m\rangle$ とする．そして，$T(v_2)$ の最右通路以外に存在する任意の k1-橋を e' とする．v_2-$h(e')$ 通路を $\langle u_0(=v_2),u_1,\ldots,u_p,u'_{p+1},\ldots,u'_q(=h(e'))\rangle$ (但し，$u'_{p+1}\neq u_{p+1}$) と表す.(図６．15)

$$n(u_{p+1}) \geqq n(u'_{p+1}) \geqq n(h(e')) \geqq k1,$$

$$|V(T)|-n(u_{p+1}) \geqq n(u'_{p+1}) \geqq k1$$

より，$\langle u_p,u_{p+1}\rangle$ は k1-橋である．つまり，e' を $T(u_p)$ に含むような k1-橋 $\langle u_p,u_{p+1}\rangle$ が最右通路に存在する．従って，各 j $(0\leqq j<i)$ に対し $\langle u_j,u_{j+1}\rangle$ が k1-橋でなく，$\langle u_i,u_{i+1}\rangle$ が k1-橋であるような $\langle u_i,u_{i+1}\rangle$ を e とすれば，全ての k1-橋は $T(t(e))$ に存在する. □

［補題６．７］ 手続き submake のステップ (20) における任意の木 T に対し，次のことが成り立つ.

$T(v_2)$ の最右通路に k1-橋がなければ，T には k1-橋がない.

（証明） T の全ての k1-橋が $T(v_2)$ に存在するなら，補題６．６より，証明できる.

submake がステップ (40) で呼ばれたなら，v_2 は T の根に設定されているので，T の全ての k1-橋が $T(v_2)$ に存在する．同様のことが，ステップ

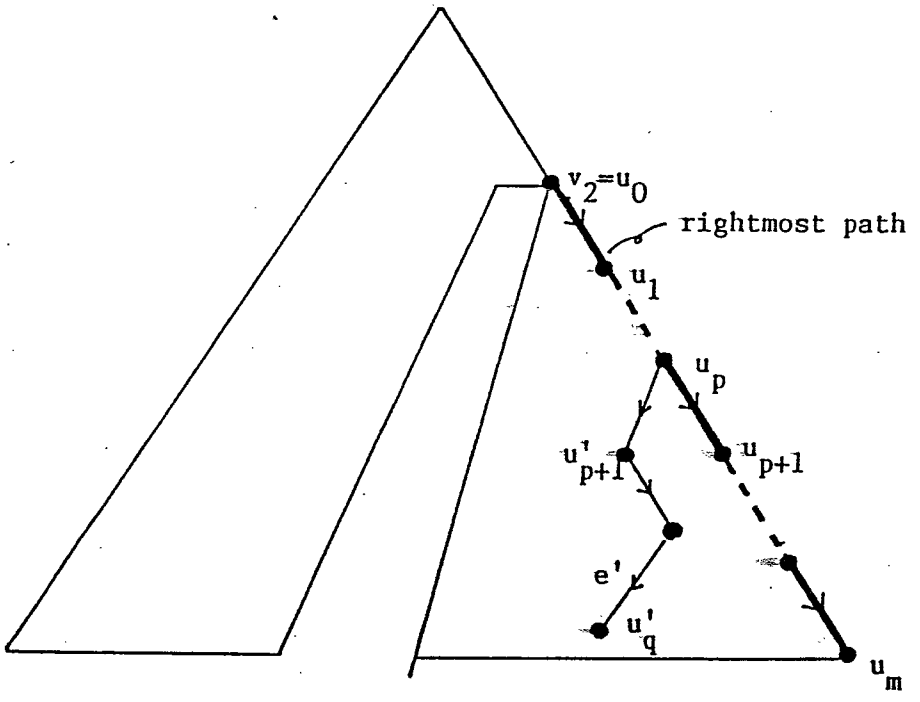


図6．15　e' が k1-橋の場合の最右通路の k1-橋 (u_p, u_{p+1}).

Fig. 6. 15　A k1-bridge (u_p, u_{p+1}) in the rightmost path in case e' is a k1-bridge.

(17) で呼ばれた submake についても成り立つ.

補題6．6及び，ステップ (3) で brd が $T(v_2)$ の最右通路上で v_2 に最も近い k1-橋に設定されることから，T の全ての k1-橋は T(brd.t) に存在する．ステップ (7)-(11) で，T は T_1，T_2 に分割されるが，T_1 の k1-橋は，T においても k1-橋である．従って，T_1 の全ての k1-橋は T_1(brd.t) に存在するので，ステップ (16) で呼ばれた submake に関しても，補題が成立する． □

次に，ステップ (21) の任意の木 T において，(k1-1)-弱重心が最右通路に存在することを示す．

［補題6．8］ k1-橋が存在せず，かつ，$|V(T)|>k1$ となる任意の木を T とする．T を，整列表現で表したとき，(k1-1)-弱重心がその最右通路に存在する．

（証明） T を整列表現で表したとき，その最右通路に，

$$n(vc) \geqq k1 \text{ かつ } n(vc^{odeg(vc)}) \leqq k1-1$$

となる頂点 vc が存在する．このとき，各 i ($1 \leqq i \leqq odeg(v)$) に対し，

$$n(vc^{i}) \leqq n(vc^{odeg(vc)}) \leqq k1-1$$

が成り立つ．

vc が T の根 v_1 の場合，明らかに，vc は (k1-1)-弱重心である．

$vc \neq v_1$ の場合，vc の親 v' が存在する．$n(vc) \geqq k1$，及び，T には k1-橋が存在しないことから，

$$|V|-n(vc) \leqq k1-1$$

が成り立つ．従って，vc は (k1-1)-弱重心である． □

次に，ステップ (24)，(27)，(30)；及び，(38) で構成された B が条件Cを満たすことを示す．

［補題６．９］ ステップ (24) で k1-進木 T=(V,E) から構成された B は条件Ｃを満たす．

（証明） ステップ (23) で求められる深さ優先順序 $f:V\to[0..|V|-1]$ が強 k1-幅順序であることを示せば十分である．

f は深さ優先順序なので，任意の辺 $\langle u_1,u_2\rangle\in E$ に対し，$f(u_2)-f(u_1)>0$ は成り立つ．

ステップ (24) における k1-進木 T=(V,E) には k1-橋が存在せず，(k1-1)-弱重心 vc に対し，$odeg(vc)\leq 2$ が成り立つ．

$|V-V(vc)|\leq k1-1$, $n(vc^1)\leq k1-1$, 及び,

$n(vc^2)\leq k1-1$ (vc^2 が存在するなら)

より，任意の辺 $\langle u_1,u_2\rangle\in E$ に対し，$f(u_2)-f(u_1)\leq k1$ が成り立つ．(図６．16)

従って，f は強 k1-幅順序である． □

［補題６．10］ k1-進木 T=(V,E) からステップ (27) で辺拡大された B は，条件Ｃを満たす．

（証明） ステップ (26) で求められる幅優先順序 $f:V\to[0..|V|-1]$ が強 k1-幅順序であることを示せば十分である．

f は幅優先順序なので，任意の辺 $\langle u_1,u_2\rangle\in E$ に対し，$f(u_2)-f(u_1)>0$ は成り立つ．

以下では，任意の辺 $\langle u_1,u_2\rangle\in E$ に対し，$f(u_2)-f(u_1)\leq k1$ が成り立つことを示す．

ある辺 $\langle x_1,x_2\rangle\in E$ に対し，$f(x_2)-f(x_1)>k1$ が成り立つとする．$p=depth(x_1)$ とすると，$depth(x_2)=p+1$ となる．各 i ($1\leq i\leq f(x_2)-f(x_1)$) に対し，$y_i$ を $f(y_i)=f(x_1)+i$ となる頂点とする．このとき，x_2 は $y_{f(x_2)-f(x_1)}$ と表される． q を $depth(y_q)=p$ かつ $depth(y_{q+1})=p+1$ とな

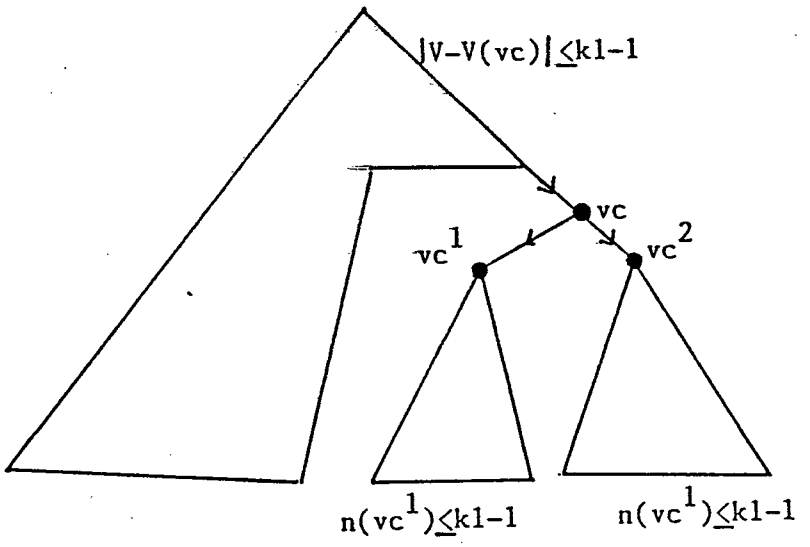


図6．16 ステップ (24) で B に辺拡大される木 T.

Fig. 6. 16 A tree T from which B is constructed in the step (24).

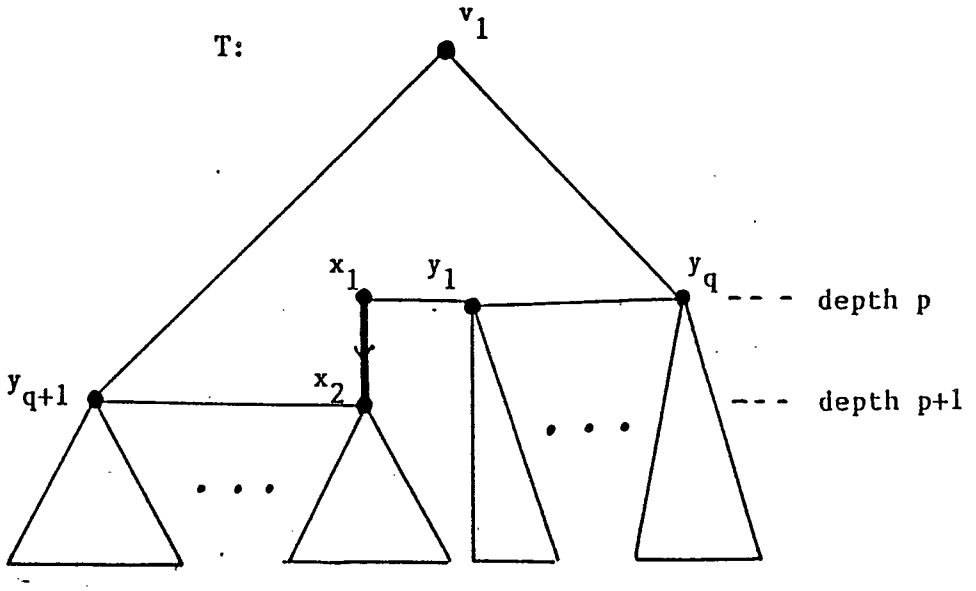


図6．17 x_1, x_2, y_i $(i=1,\ldots,f(x_2)-f(x_1))$ を根とする部分木.

Fig. 6. 17 Subtrees with the roots x_1, x_2 and y_i $(i=1,\ldots,f(x_2)-f(x_1))$.

る正整数とする.(図６．17) 幅優先順序の定義，および，$\langle x_1,x_2\rangle\in E$ より，$1\leq i\leq q<j\leq f(x_2)-f(x_1)$ なる任意の i，j に対し，$(y_i,y_j)\notin E$ となる．従って，$1\leq i<j\leq f(x_2)-f(x_1)$ なる任意の i，j に対し，y_i を根とする部分木の葉は，y_j を根とする部分木の葉と異なる．従って，T には少なくとも $f(x_2)-f(x_1)(>k1)$ 個の葉が存在することになるが，アルゴリズムより，ステップ (26) の T には 高々 k1 個の葉があるだけなので，矛盾を生じる．故に，任意の辺 $\langle u_1,u_2\rangle\in E$ に対し，$f(u_2)-f(u_1)\leq k1$ が成り立つ． □

［補題６．11］ k1-進木 T=(V,E) からステップ (30) で辺拡大された B は，条件Ｃを満たす．

（証明） ステップ (29) で求められる深さ優先順序 $f:V\rightarrow[0..|V|-1]$ が強 k1-幅順序であることを示せば十分である．

f は深さ優先順序なので，任意の辺 $\langle u_1,u_2\rangle\in E$ に対し，$f(u_2)-f(u_1)>0$ は成り立つ．

アルゴリズムの条件式より，T には k1+1 個以上の葉が存在し，かつ，$|V|=k1+3$ が成り立つ．T は k1-進木なので，T は ２個の内部頂点と k1+1 個の葉からなる．$odeg(v_1)\leq k1$，$odeg(v_1^{odeg(v_1)})\leq k1$ より，任意の辺 $\langle u_1,u_2\rangle\in E$ に対し，$f(u_2)-f(u_1)\leq k1$ が成り立つ． □

［補題６．12］ $T_3=(V_3,E_3)=T[V-\{vc\}]$ からステップ (35) で構成された $B_3=(V_3,E_3')$ は $D_{|V_3|,k1}$ であり，

$$V_3 = V-\{vc\} \text{ かつ } E_3' \supseteq E_3 = E-(IE(vc)\cup OE(vc))$$

が成り立つ．

（証明） $V_3 = V-\{vc\}$ が成り立つのは明らか．従って，ステップ (34) で求められる $f:V_3\rightarrow[0..|V_3|-1]$ が T_3 の強 k1-幅順序であることを示せば十分である．

任意の辺 $\langle u_1,u_2\rangle \in E_3$ に対し，$f(u_2)-f(u_1)>0$ が成り立つのは明らかである．

vc は (k1-1)-弱重心なので，各 i ($1 \leq i \leq odeg(vc)$) に対し，

$|V-V(vc)| < k1$ かつ $n(vc^i) < k1$

が成り立つ．従って，任意の辺 $\langle u_1,u_2\rangle \in E(T_3)$ に対し，$f(u_2)-f(u_1) \leq k1$ が成り立つ． □

［補題６．13］ ステップ (38) で，B_3 と vc から dmerge_2 によって構成された B は条件Ｃを満たす．

（証明） 補題を証明する前に，ステップ (37) の辺集合 Ed を B_3 で確かに見つけることができることを示す．

$Ed' = \{e \in E_1(B_3) \mid t(e)$ は T の葉であり，$h(e) \neq vc^1\}$

とする．ここで，B_3 は超デイジーチェインであり，辺集合 $E_1(B_3)$ は定義６．８で定義した辺集合である．このとき，Ed' は独立辺集合であり，

$Ed' \subseteq E_1(B_3)-E$

となる．さらに，T には少なくとも k1+1 個の葉が存在するので

$|Ed'| \geq k1$

となる．また，次の（ｉ），（ii）より，

辺 $\langle vc',vc^1\rangle$ は，Ed' の任意の辺と独立である．

（ｉ） $vc \neq v_1$ の場合，ステップ (37) より，vc' は vc の親であるので，vc' は T の葉でない．従って，e を Ed' の任意の辺とすると，$t(e) \neq vc'$ であり，Ed' の定義より，$h(e) \neq vc^1$ となる．従って，辺 $\langle vc',vc^1\rangle$ は Ed' の任意の辺と独立である．

（ii） $vc=v_1$ の場合，ステップ (37) より，$\langle vc',vc^1\rangle \in E_1(B_3)$ が成り立つ．辺集合 $E_1(B_3)$ が独立辺集合であること，及び，$Ed' \subseteq E_1(B_3)-\{\langle vc',vc^1\rangle\}$ より，辺 $\langle vc',vc^1\rangle$ は Ed' の任意の辺と独立である．

$Ed_1 = \{\langle vc',vc^1\rangle\} \cup \{e \in E_1(B_3) \mid h(e)=vc^i \ (2 \leq i \leq odeg(vc))\}$

とし，Ed_2 を $|Ed_2|=k1-|Ed_1|$ なる $Ed'-Ed_1$ の任意の部分集合とする．次に，

$$Ed_1-\{\langle vc',vc^1\rangle\}(=\{e\in E_1(B_3)\,|\,h(e)=vc^i\ (2\leq i\leq odeg(vc))\})\subseteq Ed'$$

が成り立つことを示す．

$e\in E_1(B_3)$ かつ $h(e)=vc^i$ $(2\leq i\leq odeg(vc))$ なる任意の辺を e とする．$t(e)$ は $T(vc^{i-1})$ の頂点であり，ステップ (34) における f の定義より，

$$f_{i-1}(t(e)) = \max\{f_{i-1}(v)\,|\,v\in V(vc^{i-1})\}$$

となる．ここで，f_{i-1} は，$T(vc^{i-1})$ の深さ優先順序である．従って，$t(e)$ は $T(vc^{i-1})$ の葉である．

$$Ed = Ed_1\cup Ed_2$$

とする．以下では，辺集合 Ed がステップ (37) の以下に示す条件を満たすことを示す．

(a) Ed は独立辺集合である．

(b) $Ed\subseteq E_3'-E_3$

(c) $|Ed| = k1$

(d) $Ed\supseteq\{\langle vc',vc^1\rangle\}\cup\{e\in E_1(B_3)\,|\,h(e)=vc^i\ (2\leq i\leq odeg(vc))\}$

(e) $Ed-\{\langle vc',vc^1\rangle\}\subseteq E_1(B_3)$

辺集合 Ed の定義より，(b)，(c) 及び (d) が成り立つのは明らか．また，

$$Ed_1-\{\langle vc',vc^1\rangle\}\subseteq Ed'\subseteq E_1(B_3)-E$$

より，条件 (e) が成り立つ．また，Ed' が独立辺集合なので，$Ed-\{\langle vc',vc^1\rangle\}(\subseteq Ed')$ は独立辺集合である．このことと，辺 $\langle vc',vc^1\rangle$ が Ed' の任意の辺と独立であることより，Ed は独立辺集合である．従って，条件 (a) も成立する．

次に，補題を証明する．

補題６．12より，グラフ B_3 は $D_{|V_3|,k1}$ であり，明らかに，

$$|V_3|\geq k1+3,\ k1(\geq odeg(vc))\geq 3,\ |Ed| = k1$$

が成り立ち，Ed は独立辺集合である．従って，k1-B-グラフの定義より，ステップ (38) で構成される B は BD_{k1} に属する．

また，明らかに，$V_B = V_3 \cup \{vc\} = V$ となる．

補題６．12と Ed の定義より，

$$E_3'-Ed \supseteq E_3 = E-(IE(vc) \cup OE(vc))$$

が成り立つ．また，$Ed \supseteq \{\langle vc', vc^1\rangle\} \cup \{e \mid h(e)=vc^i \ (2 \leq i \leq odeg(vc))\}$ より，dmerge_2(V_3,Ed,vc) によって，IE(vc)∪OE(vc) の辺は全て付加される．故に，$E_B \supseteq E$ が成り立つ． □

ステップ (16) で高々 m (≧0) 回 submake を実行して得られる B が条件Ｃを満たすと仮定し，m+1 回 submake を実行した後で，ステップ (19) で dmerge_1 によって構成される B が条件Ｃを満たすことを示す．

［補題６．14］ ステップ (16)，(17) で構成された B_1，B_2 が条件Ｃを満たすならば，ステップ (19) で dmerge_1 によって構成される B は条件Ｃを満たす．

（証明） ステップ (7)-(11) で，T を分割することにより得られた k1-進木を，$T_1=(V_1,E_1)$，$T_2=(V_2,E_2)$ とする．そして，$B_1=(V_1',E_1')$，$B_2=(V_2',E_2')$ を，それぞれ，ステップ (16)，(17) で構成されたグラフとする．補題の仮定より，

$$B_1 \in BD_{k1},\ V_1' = V_1,\ E_1' \supseteq E_1$$

が成り立つ．同様に，

$$B_2 \in BD_{k1},\ V_2' = V_2,\ E_2' \supseteq E_2$$

が成り立つ．

k1-B-グラフの定義より，ステップ (19) で構成された B は，BD_{k1} に属する．また，$V_1 = (V-V(brd.h)) \cup \{w_1\}$，及び，$V_2 = V(brd.h) \cup \{w_2\}$ より，

$$V_B = (V_1 \cup V_2)-\{w_1,w_2\} = V$$

となる．また，T を T_1，T_2 に分割した方法より，

$$E_1' \cup E_2' \supseteq E_1 \cup E_2 \supseteq E-\{brd\}$$

となる．さらに，グラフ合成法 dmerge1 の定義より，

$$E_B = (E_1' \cup E_2' - (IE(w_1) \cup OE(w_1) \cup IE(w_2) \cup OE(w_2)))$$
$$\cup \{(t(\langle\langle IE(w_1)\rangle\rangle_i), h(\langle\langle OE(w_2)\rangle\rangle_i)) \mid 1 \leq i \leq k1\}$$
$$\cup \{(t(\langle\langle IE(w_2)\rangle\rangle_i), h(\langle\langle OE(w_1)\rangle\rangle_i)) \mid 1 \leq i \leq k1\}$$

が成り立つ．$w_1 \notin V$，$w_2 \notin V$ より，$IE(w_1) \cup OE(w_1) \cup IE(w_2) \cup OE(w_2)$ は E の辺を含まない．従って，$brd = (t(\langle\langle IE(w_1)\rangle\rangle_1), h(\langle\langle OE(w_2)\rangle\rangle_1))$ より，

$$E_B \supseteq E$$

となる．

故に，ステップ (19) で構成される B は条件Cを満たす． □

［定理６．８］ 任意の正整数 k (k≧2) と任意の k-進木 Tin=(Vin,Ein) を入力としたときのアルゴリズム KDTB の出力を Bout=(Vout,Eout) とする．このとき，辺集合 Eout-Ein は Tin の k-VCUAP の解の１つである．

（証明） 補題６．９，６．10，６．11，６．13 及び ６．14 より，bmake$(T_0, v_0, k, Bout)$ で構成される Bout は，

$$Bout \in BD_k,\ Vout = V(T_0) = Vin,\ Eout \supseteq E(T_0) = Ein$$

を満たす．Bout は k-正則かつ k-頂点連結なので，系６．１より，Eout-Ein は Tin の k-VCUAP の解の１つである． □

6.5.3 アルゴリズム KDTB の時間計算量

［定理６．９］ 任意の k-進木 $T=(V,E)$ $(k\geq 2)$ に対し，k-VCUAP の時間計算量は，$\theta(k|V|)$ である．

（証明） $O(k|V|)$ の証明：T と k を入力としたときの，アルゴリズム KDTB の時間計算量 time(T,k) を評価する．

フェーズ i (i=1,2) の時間計算量を $time_i(T,k)$ と表す．

（フェーズ１）：T の深さ優先探索を行えば，$O(|V|)$ 時間で，各頂点 v に対し，n(v) が求められる[(1)]．頂点 v に対し，整列表現で表現するために，その子を n(v) によってソートするには，$O(odeg(v)\cdot\log odeg(v))$ 時間かかる．$\sum_{v\in V} odeg(v) = |V|-1$，$odeg(v)\leq k$ より，T を整列表現で表すには，$O(|V|\cdot\log k)$ 時間かかる．従って，

$$time_1(T,k)=O(|V|\cdot\log k) \tag{6.4}$$

となる．

（フェーズ２）：ステップ (7)-(11) で，T を T_1 と T_2 に分割した回数を r とする．木をステップ (7)-(11) で分割する毎に，頂点数は２だけ増加し，辺数は１だけ増加する．従って，木を r 回分割すると，頂点数は $|V|+2r$ に，辺数は $|E|+r$ になる．そして，r+1 個の k-進木ができる．一方，ステップ (2) の条件より，これらの r+1 個の k-進木は，それぞれ，少なくとも k+1 個の頂点を含む．従って，$(|V|+2r)/(r+1)\geq k+1$ となり，このことから，

$$r \leq |V|/(k-1)$$

となる．つまり，頂点数，辺数は，それぞれ，高々 $|V|+2|V|/(k-1)$，$|E|+|V|/(k-1)$ となる．但し，T は木なので，$|E|=|V|-1$ が成り立つ．

ステップ (1)-(3) の計算時間は，アルゴリズム全体を通して，$O(|V|)$ である．なぜなら，各辺は，k-橋であるかどうかを高々１回調べられるが，アルゴリズムに現れる辺の数は，高々 $|V|+2|V|/(k-1)$ であるからである．

(1) submake を実行しないとき（ステップ (20)-(38)）：フェーズ1と同様の方法で，ステップ (20) は $O(|V|\cdot\log k)$ 時間で実行できる．ステップ (21) が $O(|V|)$ 時間で実行できるのは明らか．また，T の深さ優先順序も幅優先順序も $O(|V|)$ 時間で求められるので，ステップ (24)，(27)，(30) の make_daisy(V,f) は，$O(k|V|)$ 時間で実行できる．従って，ステップ (22)-(30) は，$O(k|V|)$ 時間で実行できる．また，ステップ (31)-(34) が $O(|V|)$ 時間で実行できるのは明らかである．また，ステップ (35) は $O(k|V|)$ 時間で実行でき，ステップ (36)-(37) は $O(|V|)$ 時間で実行できる．$|Ed|=k$ より，ステップ (38) は $O(k)$ 時間で実行できる．従って，

$$time_2(T,k)=O(k|V|) \tag{6.5}$$

が成り立つ．

式 (6.4)，(6.5) より，

$$time(T,k)=time_1(T,k)+time_2(T,k)=O(k|V|) \tag{6.6}$$

が成り立つ．

(2) submake を実行するとき（ステップ (4)-(19)）：ステップ (4)-(14) は定数時間で実行できる．また，ステップ (15) で，T_1(brd.t) を整列表現に表現し直すのは $O(k)$ 時間でできる．

$time'(T_1,k)$，$time'(T_2,k)$ を，それぞれ，T_1 から B_1 を構成するのにかかる（ステップ (1)-(3) 以外の）時間，T_2 から B_2 を構成するのにかかる（ステップ (1)-(3) 以外の）時間とする．手続き dmerge_1 は $O(k)$ 時間で実行できるので，

$$time(T,k)=time_1(T,k)+O(|V|)+time'(T_1,k)+time'(T_2,k)+O(k) \tag{6.7}$$

となる．

ステップ (7)-(11) を r+1 回繰り返し実行して得られた木を $Ts_i=(Vs_i,Es_i)$ $(0\leq i\leq r)$ とする．式 (6.4)，(6.6)，(6.7) より，

$$time(T,k) = O(|V|\cdot\log k)+O(|V|)+\sum_{i=1}^{r} O(k|Vs_i|)+\sum_{i=1}^{r} O(k)$$

が成り立つ．$\sum_{i=0}^{r} |Vs_i| \leq |V|+2|V|/(k-1)$，$r \leq |V|/(k-1)$ より，

$$time(T,k) = O(k|V|)$$

となる．

T から $B \in BD_k$ を構成するときに，各辺 e に対し，e.flag は，次の条件を満たすように操作される．

e.flag = true となるのは，$e \in E$ のとき，かつ，そのときに限る．

従って，B から辺集合 E(B)-E を求めるのは，O(k|V|) 時間でできる．

故に，任意の k-進木 T=(V,E) に対し，アルゴリズム KDTB を用いれば，k-VCUAP は O(k|V|) 時間で解ける．

Ω(k|V|) の証明：定理６．１より，任意の k-進木 T=(V,E) の k-VCUAP の任意の解は少なくとも $k|V|-|E| = (k-1)|V|+1$ 個の辺を含む．従って，任意の k-進木 T=(V,E) の k-VCUAP の計算時間の下界は，Ω(k|V|) となる．□

６．６　木の k-頂点連結拡大構成問題

この節では，一般の木の k-VCUAP ($k \geq 2$) を解くアルゴリズム DT を示し，その計算時間がオーダ的に最適であることを示す．アルゴリズム DT は，前節のアルゴリズム KDTB を利用している．

6.6.1　アルゴリズム DT

[アルゴリズム DT]

(Algorithm for augmenting a directed tree to a k-connected digraph)

【入力】 木 $T=(V,E,v_0)$ の隣接リスト L，及び，$|V|>k\geq 2$ なる正整数 k.

【出力】 T から辺拡大された k-頂点連結グラフ $B=(V_B,E_B)$．但し，グラフ B は，アルゴリズム KDTB における Bout と同じデータ構造を持つ．

【方法】 アルゴリズム DT は，３つのフェーズからなる．

（フェーズ１）前処理：以下のようにして，T から k-進木 $T'=(V',E')$ を構成する．

$odeg_T(v)>k$ なる T の各頂点 v に対し，$OE_T(v)$ の辺のうち，任意の $odeg_T(v)-k$ 個の辺を削除する．このとき，削除した辺の集合を Ed とする．T-Ed は，$|Ed|+1$ 個の k-進木 $T_0,T_1,\ldots T_{|Ed|}$ からなる．各 i ($0\leq i\leq |Ed|-1$) に対し，始点を T_i の任意の葉とし，終点を T_{i+1} の根とする辺を付加する．このとき，付加する辺の集合を Ea と表し，$T'=(V,(E-Ed)\cup Ea)$ とする．ここで，T' は，k-進木になっている．

（フェーズ２） アルゴリズム KDTB を用いて，T' を k-B-グラフ $B'=(V, E_B')$ に辺拡大する．

（フェーズ３） グラフ B を次のように構成する．

$B = (V, E_B' \cup Ed)$ □

6.6.2 アルゴリズム DT の正当性と時間計算量

［補題６．15］ 任意の木 $T=(V,E)$ と 任意の正整数 k ($k \geq 2$) を入力としたときのアルゴリズム DT の出力を $B=(V_B,E_B)$ とする．このとき，B は k-頂点連結であり，$V_B = V$ かつ $E_B \supseteq E$ となる．

（証明） グラフ $B' \in BD_k$ は，B の生成部分グラフである．B' は k-頂点連結であるので，B も k-頂点連結である．また，明らかに，$V_B = V$ となる．また，$E_B = E_B' \cup Ed$ 及び $E_B' \supseteq E-Ed$ より，$E_B \supseteq E$ となる． □

［定理６．10］ 任意の木 $T=(V,E)$ と 任意の正整数 k ($k \geq 2$) を入力としたときのアルゴリズム DT の出力を $B=(V_B,E_B)$ とする．このとき，辺集合 E_B-E は，T の k-VCUAP の解の１つである．

（証明） 補題６．15より，B は T から辺拡大されたグラフであり，かつ，k-頂点連結である．従って，付加した辺数の最小性を示せば十分である．B において，次の（ｉ），（ii）が成り立つのは明らか．

（ｉ） $odeg_T(v) \leq k$ なる各頂点 v に対し，$odeg_B(v) = k$

（ii） $odeg_T(v) > k$ なる各頂点 v に対し，$odeg_B(v) = odeg_T(v)$

（ｉ），（ii）より，

$$|E_B - E| = \sum_{v \in V} \max(k - \mathrm{odeg}_T(v), 0)$$

が成り立つ．従って，定理６．１より，B は最小数の辺付加により T から辺拡大された k-頂点連結グラフである． □

［定理６．11］ 任意の木 T=(V,E) に対し，k-VCUAP (k≧2) の時間計算量は，$\theta(k|V|)$ である．

（証明） アルゴリズム DT のフェーズ１，フェーズ３が $O(|V|)$ 時間で実行できるのは明らか．また，定理６．９より，フェーズ２は $O(k|V|)$ 時間で実行できる．従って，アルゴリズム DT を用いると，任意の木 T=(V,E) の k-VCUAP (k≧2) は，$O(k|V|)$ 時間で解ける．

また，定理６．９と同様にして，下界 $\Omega(k|V|)$ が証明できる． □

７．結　論

故障を考慮した分散アルゴリズムとネットワーク・トポロジに関する基本的な問題の考察を行った．

最初に，非同期式ネットワークにおいて，プロセッサやリンクが故障している可能性のあるときに，これらの故障にかかわらず生成木構成問題を解く分散アルゴリズムが存在するかどうかについて議論した．まず，プロセッサが故障しているかもしれないネットワークでは，この問題は解けないことを示した．次に，プロセッサに故障がなく，リンクだけが故障している可能性のあるネットワーク上での生成木構成問題について考察し，この問題を解く分散アルゴリズムが存在するかどうかが，各プロセッサが利用できるネットワークに関する情報（隣接プロセッサの識別子，ネットワークのプロセッサ数，ネットワークの辺連結度）に真に依存することを示した.(表３．１参照)

次に，プロセッサ故障診断問題（任意の１つのプロセッサが自分に隣接する任意の１つのプロセッサが故障しているかどうかを診断する問題）とリンク故障診断問題（任意の１つのプロセッサが自分に接合する任意の１つのリンクが故障しているかどうかを診断する問題）について考察した．まず，これらの問題が非同期式ネットワークでは解けないことを示し，プロセッサや通信に関する同期性と各プロセッサで利用できるネットワークに関する情報が，これらの問題を解く分散アルゴリズムの存在性に真に影響を及ぼすことを示した.(表４．１，表５．１参照)

最後に，ネットワークの耐故障性を効率よく向上させる問題をモデル化した問題であるグラフの k-頂点連結拡大構成問題について考察した．そして，有向木の k-頂点連結性に関する重みなしの拡大構成問題をオーダ的に最適な時間で解くアルゴリズムを示した．

今後に残された課題の１つとして，本論文で示した分散アルゴリズムの効率を改良することがある．本論文では，問題を解く分散アルゴリズムが存在するかどうかについて議論することに主な目標をおいたため，本論文の分散

アルゴリズムの効率には改善の余地がある．

また，本論文では，隣接プロセッサの識別子，ネットワークのプロセッサ数，ネットワークの辺連結度に関して，これらの情報を各プロセッサが利用できることが，ネットワークの故障を考慮する際に役立つことを示したが，分散アルゴリズムの耐故障性を向上させるのに役立つ情報として，これら以外にどのような情報があるかを研究することも，今後に残された重要な課題である．

また，拡大構成問題に関しては，まだまだ未解決な問題が多く，さまざまなクラスのグラフに対して，その時間計算複雑度を求めることも，今後の研究課題である．

謝　　辞

本研究の全過程を通じて，直接理解ある御指導を賜わり，つねに励ましていただいた都倉信樹教授に心から深謝いたします．

大学院前期および後期課程において御教示，御指導いただいた情報工学科の嵩忠雄教授，藤沢俊男教授，鳥居宏次教授，谷口健一教授，大阪大学産業科学研究所の豊田順一教授，大型計算機センターの宮原秀夫教授，並びに，故高島堅助先生に心から感謝いたします．

大学院を通じて御指導いただいた田村進一助教授，柏原敏伸助教授，藤井護助教授に心から感謝いたします．

本研究を通じて有益な御助言，御指導いただいた荒木俊郎助教授に心から感謝いたします．

本研究の全過程を通じて，有益な御討論，御指導いただいた萩原兼一講師，拡大構成問題に関して，有益な御討論，御指導いただいた名古屋工業大学の和田幸一講師に心から感謝いたします．

種々の面で御助言，御援助いただいた辻野嘉宏助手に心から感謝いたします．

さらに，著者の在学中，御討論いただいた都倉研究室の方々に心から感謝いたします．

文　　献

(1) A.V. エイホ, J.E. ホップクロフト, J.D. ウルマン:"アルゴリズムの設計と解析Ⅰ", 野崎, 野下訳, サイエンス社 (1977).

(2) 朴, 増澤, 萩原, 都倉:"幅優先生成木構成の分散アルゴリズムについて", 電子通信学会技術研究報告, COMP86-44 (1986-11).

(3) B. Bollobás:"Extremal Graph Theory", Academic Press, New York (1978).

(4) D. Dolev:"The Byzantine generals strike again", Journal of Algorithms, 3, 1, pp.14-30 (March 1982).

(5) D. Dolev and M. Klawe and M. Rodeh:"An O(nlogn) unidirectional distributed algorithm for extrema finding in a circle", Journal of Algorithms, 3, 3, pp.245-260 (September 1982).

(6) D. Dolev, C. Dwork and L. Stockmeyer:"On the minimal synchronism needed for distributed consensus", IEEE 24th Annual Symposium on Foundations of Computer Science, pp.393-402 (November 1983).

(7) K.P. Eswaran and R.E. Tarjan:"Augmentation problems", SIAM Journal on Computing, 5, 4, pp.653-665 (December 1976).

(8) S. Even:"Graph Algorithms", Computer Science Press, Maryland (1979).

(9) M.J. Fischer, N.A. Lynch and M.S. Paterson:"Impossibility of distributed consensus with one faulty process", Journal of the Association for Computing Machinery, 32, 2, pp.374-382 (April 1985).

(10) G.N. Frederickson, and J. Ja'ja:"Approximation algorithms for several graph augmentation problems", SIAM Journal on Computing, 10, 2, pp.270-283 (May 1981).

(11) G.N. Frederickson and N.A. Lynch:"The impact of synchronous communication of the problem of electing a leader in a ring", Proceedings 16th Annual ACM Symposium on Theory of Computing, pp.493-503 (May 1984).

(12) R.G. Gallager, P.A. Humblet and P.M. Spira:"A distributed algorithm for minimum-weight spanning trees", ACM Transactions on Programming Languages and Systems, 5, 1, pp.66-77 (January 1983).

(13) M.R. Garey and D.S. Johnson:"Computers And Intractability - A Guide To The Theory Of NP-Completeness", W.H. Freeman and company, San Francisco (1979).

(14) 萩原，和田，池田，増澤，都倉:"衛星を利用した通信網の経路多重性について"，電子通信学会論文誌(D)，J67-D，10，pp.1155-1162 (1984-10).

(15) F. Harary:"Graph Theory", Addison-Wesley, Reading, Mass (1969). (池田訳:"グラフ理論"，共立出版(1971)).

(16) 池田，増澤，萩原，都倉:"１辺付加によるｋ連結化可能なグラフについて"，電子通信学会技術研究報告，CAS84-227 (1985-03).

(17) 梶谷，上野:"有向木から拡大構成される最小k-枝連結有向グラフ"，電子通信学会技術研究報告，CAS83-3 (1983-05).

(18) E. Korach, D. Rotem and N. Santoro:"Distributed Ranking", Carleton University SCS-TR-22 (January 1984).

(19) E. Korach, D. Rotem and N. Santoro:"Distributed algorithms for finding centers and medians in networks", ACM Transactions on Programming Languages and Systems, 6, 3, pp.380-401 (July 1984).

(20) E. Korach, D. Rotem and N. Santoro:"Distributed election in a circle without a global sense of orientation", International Journal of Computer Mathematics, Vol.16, pp.115-124 (November 1984).

(21) 増澤，和田，萩原，都倉:"グラフの拡大構成問題に関する一考察"，電子通信学会技術研究報告，CAS83-72 (1983-08).

(22) 増澤，萩原，和田，都倉:"ｋ－頂点連結性に関する有向２進木の拡大構成問題"，電子通信学会論文誌(D)，J67-D，1，pp.77-84 (1984-01).

(23) 増澤，萩原，和田，都倉:"頂点連結性に関する拡大構成問題(I)"，電子通信学会技術研究報告，CAS84-2 (1984-04).

(24) 増澤，萩原，都倉:"無向グラフの点連結性に関する拡大構成問題について"，電子通信学会技術研究報告，AL84-11 (1984-06).

(25) 増澤，萩原，都倉:"有向木の連結性に関する拡大構成問題について"，電子通信学会技術研究報告，AL84-27 (1984-09).

(26) T. Masuzawa, K. Hagihara, K. Wada and N. Tokura:"The graph classes with optimal algorithms for k-connectivity augmentation problems", Proceedings of International Conference on Computers, Systems and Signal Processing, Bangalore, India, pp.195-198 (December 1984).

(27) 増澤，萩原，和田，都倉:"有向３進木のｋ－連結拡大構成問題"，電子通信学会論文誌(D)，J68-D，5，pp.1003-1010 (1985-05).

(28) T. Masuzawa, K. Hagihara and N. Tokura:"An optimal time algorithm for the k-vertex-connectivity unweighted augmentation problem for rooted directed trees", (to appear in Discrete Applied Mathematics)

(29) J. Misra and K.M. Chandy:"A distributed graph algorithm : knot detection", ACM Transactions on Programming Languages and Systems, 4, 4, pp.678-686 (October 1982).

(30) J. Pachl, E. Korach and D. Rotem:"Lower bounds for distributed Maximum-Finding Algorithms", Journal of the Association for Computing Machinery, 31, 4, pp.905-918 (October 1984).

(31) M. Pease, R. Shostak and L. Lamport:"Reaching agreements in the presence of faults", Journal of the Association for Computing Machinery, 27, 2, pp.228-234 (April 1980).

(32) N. Santoro:"On the message complexity of distributed problems", International Journal of Computer and Information Sciences, 13, 3, pp.131-147 (June 1984).

(33) N. Santoro:"Sense of direction, topological awareness and communication complexity", SIGACT NEWS, 16, 2, pp.50-56 (1984).

(34) 上野，梶谷，和田:"木から拡大構成される最小ｋ－枝連結グラフ"，電子通信学会技術研究報告，IN83-6 (1983-05).

(35) 渡辺，中村:"辺の付加による点－連結度の増加問題"，電子通信学会技術研究報告，AL81-26 (1981-07).

(36) 渡辺，中村，高橋:"グラフの点部分集合に対する拡大構成問題"，電子通信学会技術研究報告，AL83-89 (1984-03).

(37) 渡辺，中村:"辺の付加によるｋ－辺連結グラフ構成問題"，電子通信学会技術研究報告，AL83-90 (1984-03).