アルゴリズム 13－2
（1990．1．31）

トポロジー変化時の重み最小生成木を再構成する分散アルゴリズムについて

朴　政鎬† 増澤　利光†† 萩原　兼一† 都倉　信樹†

†大阪大学　基礎工学部
††大阪大学　情報処理教育センター

本稿では，重み最小生成木 (OMST)が構成されているネットワークで，リンク追加と削除が生じたとき，トポロジー変化後のネットワークの MST (NMST)を再構成する分散アルゴリズムを提案する．本稿のアルゴリズムの通信計算量と理想時間計算量はそれぞれ O(nlog (f+t) +m)，O(nlog (f+t) +n)である．但し，nとeはそれぞれトポロジー変化後のネットワークのプロセッサ数とリンク数で，tは追加リンク数，fは削除されたリンクの中で OMSTに属するリンク数を表す．ここで，|f| =0のとき，m=n+tで，|f|>0のとき，m=eである．また，本稿では，プロセッサの追加と削除が生じたとき，NMSTを再構成する分散アルゴリズムも示す．

Distributed Algorithms To Reconstruct MST After Topology Change

Jungho Park† Toshimitsu Masuzawa††
Ken'ichi Hagihara† and Nobuki Tokura†

†Faculty of Engineering Science, Osaka University
††Education Center for Information Processing, Osaka University
Toyonaka, Osaka 560, JAPAN

This paper proposes distributed algorithms for reconstructing Minimum Spanning Tree (MST) when topology changes. In our algorithm, the message complexity is O(nlog (t+f) + m) and the ideal time complexity is O(nlog (t+f) +n) when t links are added and f links of old MST are deleted. Here, n(resp. e) is the number of processors (resp. links in the network after topology change). And, m=e if |f|>0, otherwise m=n+t. We also present an algorithm for reconstructing MST when processors are deleted and added.

１．　まえがき

ネットワーク環境では，メッセージの放送や情報収集などを効率よく行うため，ネットワークの重み最小生成木（Minimum Spanning Tree，以降，単にＭＳＴという）がよく利用される．ＭＳＴの例として，図１（a)のネットワークのＭＳＴを図１(b)に示す．これまでは，図１(a)のように，ＭＳＴが構成されていないネットワーク Nに対し，（b)の太線で示すＭＳＴを構成する分散アルゴリズムに関する研究が行われてきた．文献（3)では，通信計算量Ｏ（nlog n＋e'）のＭＳＴを構成する分散アルゴリズム（以下，そのアルゴリズムをＧＨＳという）を提案し，その通信計算量がオーダー的に最適であることを示した．但し，　n，e'はそれぞれネットワークのプロセッサ数とリンク数である．

実際のネットワークではプロセッサやリンクの削除や追加が起こる．従って，実際に活動しているネットワークのトポロジーは動的に変化するものと考えた方がよい．図１(a)のNに対し，(b)のＭＳＴ（ＯＭＳＴ（Old MST）という）が構成されているとする．この後に，（c)のネットワークN'に変化した状況を考える．ＯＭＳＴはN'のＭＳＴ（ＮＭＳＴ（New MST）という）ではない．従って，(d)の太線で示すＮＭＳＴを再構成しなければならない．このように，トポロジー変化後のネットワークのＭＳＴを再構成する問題をＭＳＴ更新問題という（以下，ＵＭＰ（Updating MST Problem）という）．本稿では，リンク削除と追加が生じた場合のＵＭＰと，リンクとプロセッサ削除と追加が混在する場合のＵＭＰについて考える．

アルゴリズムＧＨＳを　N'に適用すれば，ＵＭＰは通信計算量Ｏ（nlog n+e）で解ける．但し，nと eは，N'のプロセッサ数とリンク数を表す．しかし，　NのＭＳＴを求めるときに，以降に起こるトポロジー変化を考慮して，ＵＭＰを効率よく解くために利用できる情報（以下，更新補助情報という）を前もって求めておくことも考えられる．どのような更新補助情報を用いれば，ＵＭＰを効率よく解けるかを明確にすることは，理論的にも実際的にも興味深い．

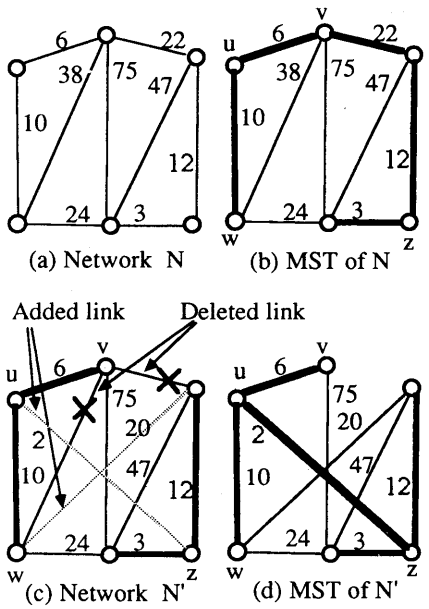

図1 ネットワークとMST

本稿では各プロセッサが持つＯＭＳＴに関する情報（どの隣接リンクがＯＭＳＴに属するか．これを前解（Old Solution）という）を更新補助情報として用いてＵＭＰを解くことを考える．前解は，メッセージ交換なしで，ＯＭＳＴから得られるので，自然な更新補助情報の一つである．

これまでに，リンク追加だけが生じた場合とリンク削除だけが生じた場合のＵＭＰを解くアルゴリズムがそれぞれ文献(6)，(2)に示されている．しかし，リンク追加と削除が混在する場合のＵＭＰについては，これまでに研究されていない．本稿ではリンク追加と削除が混在する場合のＵＭＰに対して，通信計算量Ｏ（nlog（f＋t)＋m），理想時間計算量Ｏ（nlog（f＋t)＋n）の分散アルゴリズム（以下，ＭＡという）を示す．但し，　tは追加リンク数，　fは削除されたリンクの中で，ＯＭＳＴに属するリンク数で，　|f|＝0の場合，m＝n＋tで，|f|＞0の場合，m＝eである．以下，リンク追加だけが生じた場合とリンク削除だけが生じた場合に，アルゴリズムＭＡを適用したときの計算量と，文献(6)，(2)の結果との比較を行う．

リンク追加だけが生じた場合，文献（6)は通信計算量と理想時間計算量が共にＯ（nt＋t^2）の分散アルゴリズムを示している．この場合，アルゴリズムＭＡの通信計算量はＯ（nlog t＋（n+t)），理想時間計算量はＯ（nlog t＋ n）になるので，通信計算量と理想時間計算量に関して，アルゴリズムＭＡが文献（6)のアルゴリズムより優れている．

リンク削除だけの場合，文献（2)は通信計算量と理想時間計算量が共にＯ（j^2p）の分散アルゴリズム（以下，ＣＫという）を示している．但し，　jは削除されたリンク数，pは N'での最長閉路の長さを表す．この場合，ＭＡの通信計算量はＯ（nlog f＋ e），理想時間計算量はＯ（nlog f ＋ n）になる．ＭＡがＣＫより優れている点を列挙する．

・１メッセージの長さに関して優れている．ＭＡでの１メッセージ長は Ｏ（log n）であるが，ＣＫでの１メッセージ長はＯ（elog n）である．

・ビット計算量に関して優れている．ＣＫのビット計算量は Ｏ（j^2elog n）であるが，ＭＡのビット計算量はＯ（nlog k·log n＋elog n）である．

・ j＝Ω（$\sqrt{\text{nlog f＋e／p}}$）のとき，通信計算量に関して優れている．また，　j＝Ω（$\sqrt{\text{nlog f＋n／p}}$）のとき，理想時間計算量に関して優れている．

・領域計算量に関して優れている．ＣＫでは更新補助情報として前解以外に，　Replacement Set（ＯＭＳＴの各リンクについて，そのリンクが削除されたとき，そのリンクの代わりにＮＭＳＴのリンクになりうるリンクの集合）を用いるので，多くの領域を必要とする．

２．　諸定義

２．１．　グラフに関する定義

重み付き無向グラフ Gは３項組G＝（V,E,W）で定義され

る．Vは空でない頂点集合で，Eは無向辺（相異なる頂点の順序のない対）の集合である．Wは関数 W：E→N（Nは自然数の集合）で，各 e（∈E）に対して，W（e）をeの重みという．e＝(u,v)のとき，W(e)をW(u,v)と表すこともある．以下，簡単のため，辺の重みを特に気にしないときは，重み付き無向グラフ G＝（V,E,W）を単に G＝（V,E）と表すことがある．本稿では，グラフとして重み付き無向グラフのみを扱うので，重み付き無向グラフを単にグラフと呼ぶ．定義より，グラフには多重辺や自己閉路は存在しない． Gで，(u,v)∈Eのとき，uと vは隣接するといい，uと隣接する頂点の集合 {v∈V| (u,v)∈E} を $NEG_G(u)$と表す． uの次数 $deg_G(u)$を $deg_G(u)=|NEG_G(u)|$と定義する．

Gで，相異なる頂点系列$\langle v_1,\cdots,v_m\rangle$が各 $i(1\leqq i\leqq m-1)$に対して，$(v_i,v_{i+1})\in E$を満たすとき，この系列を v_1-v_m道といい，m-1をこの道の長さという．特に，１点vだけからなる系列<v>も長さ0のv-v道と考える．Gの任意の２頂点 u，vに関し，u-v道が存在するとき，Gは連結という．Gの２頂点 u，v間の距離 $d_G(u,v)$を長さ最小のu-v道の長さと定義する．但し，u-v道が存在しないとき，$d_G(u,v)=\infty$とする．

長さ２以上の v_1-v_m道$\langle v_1,\cdots,v_m\rangle$と辺 (v_1,v_m)が存在するとき，頂点系列 $\langle v_1,\cdots,v_m,v_1\rangle$を閉路という．閉路がない連結な無向グラフを木という．特に，根と呼ばれる頂点（それをiとする）が指定されている木を，iを根とする根付き木という． iを根とする根付き木 Tに関して，次の用語と記法を定義する．

・親，子：u，vを隣接する任意の２頂点とする．$d_T(i,v)=d_T(i,u)-1$が成り立つとき， vを uの親といい，uを vの子という．

・子孫：任意の２頂点を u，vとする．Tで， i-v道は一意に決まるが，この i-v道に頂点 uが現れるとき， vを uの子孫という（ u自身も uの子孫である）．

・葉：子を持たない頂点 uを葉という．

G＝(V,E,W)と G'＝(V',E',W')に対し，V'⊆Vかつ，E'⊆Eが成立ち，更に，W'がWの定義域をE'に制限した（つまり，各e（∈E'）に対し，W'(e)＝W(e)が成り立つ）とき，G'を Gの部分グラフといい， G'⊆Gと表す．部分グラフに関する用語と記法を次のように定義する．

・生成木：G'＝(V',E')を G＝(V,E)の部分グラフとする．V'＝Vかつ，G'が木のとき，G'をGの生成木という．以下，本稿では根付き生成木だけを扱い，簡単のため，根付き生成木を単に生成木という．

・重み最小生成木： Gの生成木の中で，辺の重みの総和が最小の生成木を重み最小生成木（以下，ＭＳＴという）といい，MT(G)と表す．Gの辺の重みが全て異なるときは，MT(G)は一意に定まる[3]．

・誘導部分グラフ：G'＝（V',E'）を G＝（V,E）の部分グラフとする．E'＝｛(u,v)∈E| u，v∈V'｝のとき，G'を GのV'による誘導部分グラフといい，G[V']と表す．

・部分木：根付き木 Tの任意の頂点を uとする．$T[DES_T(u)]$を，Tの uを根とする部分木という．

・連結成分：グラフ Gの連結な部分グラフで極大なものをグラフ Gの連結成分という．

２．２．　ネットワークと分散アルゴリズムに関する定義

ネットワークと分散アルゴリズムに関する定義と仮定を示す．詳細は，文献（4)参照のこと．

ネットワーク Nは，３項組N＝(P,L,W)で定義される．Pはプロセッサの集合，Lはリンク（相異なるプロセッサの順序のない対）の集合を表す．リンク(u,v)は，uと vが両方向に独立にメッセージを送れる全２重リンクである．Wは関数 W：L→N（Nは自然数の集合）で，各 ℓ（∈L）に対して，W(ℓ)を ℓの重みという．定義より，ネットワーク Nを無向グラフとみなせるので，無向グラフに対する用語や記法を Nに対しても使う．

ネットワークと分散アルゴリズムに関して以下の仮定を置く．

①ネットワークに共有メモリは存在せず，プロセッサ間の通信はメッセージの交換でのみ行われる．

②ネットワークは非同期式である．即ち，メッセージの伝送遅延は有限であるが，予測不能で，しかも，可変である．

③プロセッサ uが uの隣りの vに送信したメッセージは損失なしに vに届く．

④各プロセッサは初期状態において，基本情報（自分の識別子と隣接リンクの重み）と，どの隣接リンクが削除され，どの隣接リンクが追加されたかを知っている．さらに，更新補助情報を利用できる場合がある．

⑤プロセッサは Ｏ（log n）ビットの互いに異なる識別子を持つ．

⑥リンクの重みはすべて異なり，各リンクの重みは Ｏ（log n）ビットで表せる†．

⑦任意のプロセッサ（複数個かもしれない）が始動プロセッサ（自発的にプログラムの実行を開始するもの）となる．始動プロセッサ以外のものは，他からのメッセージを受信するとプログラムの実行を開始する．

⑧分散アルゴリズムの実行の間，プロセッサとリンクの増設や削除および，それらの故障は起こらない．

本稿では，まず，リンク追加とリンク削除が生じたときのＭＳＴ更新問題について考える．リンクとプロセッサの追加と削除が混在する場合のＭＳＴ更新問題に対しては，５節で述べる．

［定義１］ＭＳＴ更新問題（Updating MST Problem．以下，ＵＭＰという）：

Nを任意のネットワークとする．複数個のリンクが追加または，削除されることによって，Nがネットワーク N'に変化したとき，N'のＭＳＴを構成する（各プロセッサはど

---

† これは，説明を簡単にするために置いた仮定であり，本質的でない．同じ重みのリンクが存在するときは，リンク(u,v)の重みを３項組 <W(u,v),uの識別子,vの識別子>（但し， uの識別子＜vの識別子とする）とすれば，全てのリンクの重みが異なることになる．このために隣りのプロセッサ間で識別子の交換などが必要であるが，その計算量が本稿の結果に影響しないことを簡単に示せる．

の隣接リンクがN'のＭＳＴに属するかを求める）問題がＵＭＰである．但し，ＵＭＰにおいては，ＵＭＰを効率よく解くため，何らかの情報（更新補助情報という）を前もって各プロセッサに持たせて，その情報を用いることができる．各プロセッサが更新補助情報を持ち，その情報を利用する場合には，ＵＭＰを解いたとき，更新補助情報もN'に対応するように更新しておかなければならない．□

一般に，分散アルゴリズムは通信計算量とビット計算量および，理想時間計算量を用いて評価する．しかし，更新問題を通信とビット計算量及び理想時間計算量に関して効率よく解くため，更新補助情報を前もって求めておくことがある．従って，本稿では更新問題を解く分散アルゴリズムの評価尺度として，上記の尺度以外に，局所領域計算量と全領域計算量も考える．

通信計算量は全てのプロセッサ間で交換される最悪時のメッセージの総数で，ビット計算量は全てのプロセッサ間で交換される最悪時のメッセージのビットの総和である．理想時間計算量はプロセッサ内での処理時間を無視し，メッセージがリンクを伝わる伝送時間を１単位時間としたとき，アルゴリズムが終るまでの単位時間数である．また，局所領域計算量は各プロセッサが用いるローカルメモリ量で，全領域計算量は全てのプロセッサが用いるローカルメモリの総和である．

３．重み最小生成木更新問題を解く分散アルゴリズム

３．１　リンク追加と削除が混在する場合

ここでは，トポロジー変化として，リンクの追加と削除が混在する場合を扱う．以下，任意のネットワークを N=（P,L），Nのトポロジー変化後のネットワークを　N'=（P,L'）とする．但し，本稿では，連結なネットワークのみを扱う．つまり，Nと N'のいずれもが連結であるとする．また，NのＭＳＴを MT(N)=（P,LT(N)），N'のＭＳＴを MT(N')=（P，LT(N')）と表す．ここでは，リンクの追加と削除によって，ネットワークが NからN'に変化したとき，N'でＵＭＰを解く分散アルゴリズムＭＡ（Minimum Spanning TreeUpdating Algorithm）を示す．

複数個のリンクが削除されたとき（以降，削除されたリンクの中で MT(N)に属するリンクを削除枝といい，削除枝の集合を E_dと表す），MT(N)は複数個の連結成分に分割される．MT(N')の任意の連結な部分グラフを”（MT(N')の）破片”と呼ぶ．著者らは，MT(N)-E_d=（P,LT(N)-E_d）の各連結成分がMT(N')の破片であることに着目して，MT(N')を効率よく構成する分散アルゴリズムを文献（5)に示している．文献(5)のアルゴリズムでは，r個の破片が存在する状況から始めて，破片の結合を繰り返すことで，通信計算量Ｏ（nlog r+e）で MT(N')を構成する．一方，１つのリンク（u,v)が追加されると，MT(N)上のu-v道とリンク（u,v)による閉路が生じる．このとき，閉路の中から最大の重みのリンクを削除することで，MT(N')が構成できる[1]．複数個のリンクが追加されたとき（以降，追加されたリンクを追加リンクといい，追加リンクの集合をE_aと表す），MT(N)+E_a（=（P,LT(N)∪E_a））に複数個の閉路ができる．文献（1)は任意の閉路を発見し，その閉路から重み最大のリンクを削除することを MT(N)+E_aに閉路が存在しなくなるまで繰り返すことによってMT(N')を構成するアルゴリズムを提案している．しかし，閉路を効率よく求めるのは容易ではないので，文献（1)のアルゴリズムの効率はよくない．次に，アルゴリズムＭＡのアイデアを述べる．

リンク追加と削除が生じたとき（追加リンクの集合をE_a，削除枝の集合を E_dとする），MT(N)は複数個の連結成分に分割される．しかし，追加リンクが存在するので，MT(N)-E_dの各連結成分がMT(N')の破片であるとは限らない．アルゴリズムＭＡでは，まず，削除枝以外に，さらに MT(N)からいくつかのリンクを取り去り（ MT(N)から取り去られるリンク（削除枝も含む）を分割枝（定義２参照）と呼び，分割枝の集合をE_pと表す），次の性質①と②が成り立つようにしている．

①MT(N)-E_pはMT(N')の部分グラフである(補題１参照)．

②MT(N)-E_pは，Ｏ（ｆ＋ｔ）個の連結成分からなる（補題２参照)．但し，　ｆ=|E_d|，　ｔ=|E_a|である．

アルゴリズムＭＡでは，分割枝を除去した後，上記の①と②が成り立つことを利用して，Ｏ（ｆ＋ｔ）個の破片が存在する状況で，文献（5)のアルゴリズムを適用し，効率よくＮＭＳＴを構成する．

まず，分割枝を次のように定義する．

［定義２］追加リンクまたは，削除枝と接続するプロセッサを隣接プロセッサ（Adjacent Processor）という（図２参照)．また，異なる２つ以上の子の子孫の中に隣接プロセッサを持つプロセッサを分岐プロセッサ（Branch Processor）という．　u，vを任意の隣接または分岐プロセッサとする．　u-v道上に他の隣接または分岐プロセッサが存在しないとき，この u-v道を u，　vを端とする主要道（Primary path）という．各主要道上の重み最大のリンクを分割枝という（図２)．□

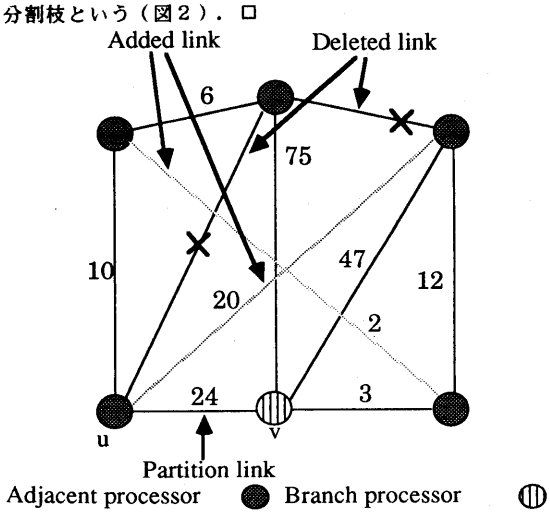

図２ 定義２の説明図

分割枝の定義より，削除枝は分割枝である．

分割枝の集合を E_pとする． MT(N)-E_pに関して，次の補題が成り立つ．

［補題１］ MT(N)-E_pは MT(N')の部分グラフである．

（証明）MT(N)-E_pの連結成分の中にMT(N')の破片でないものが存在すると仮定する．このとき，その連結成分はMT(N')に属さないリンクを含む．そのリンクを (u,v)とし，vを uの子とする．

MT(N)から (u,v)を削除すると，MT(N)は vを根とするMT(N)の部分木（T_vと表す）とそれ以外の部分グラフ（G'と表す）に分割され，次のいずれかが成り立つ．

① T_vと G'のプロセッサ間を結ぶ追加リンクが存在する場合（図３参照）．

T_vと G'のプロセッサ間を結ぶ追加リンクが存在することより， uの先祖の中に隣接プロセッサまたは分岐プロセッサは存在する．その中で， uに最も近いプロセッサを tとする．同様に， vの子孫の中にも隣接プロセッサまたは，分岐プロセッサは存在し，その中で， vに最も近いものをsとする．分割枝の定義より，t-s道上の重み最大のリンクが分割枝（(g,g')とし，gは g'の子とする）である．一般性を失うことなく，(g,g')は u-t道上に存在すると仮定できる．MT(N)から(u,v)と (g,g')を削除すると，MT(N)は u,gを含む部分グラフ（G1とする）， vと g'をそれぞれ含む部分グラフに分割される． tの選び方より， u-g道上に隣接プロセッサと分岐プロセッサは存在しないので，G1とG1以外のプロセッサ間を結ぶ任意のリンク(z,z')に対して，W(z,z')＞W(u,v)または，W(z,z')＞W(g, g')である[3]．また，(g,g')が t-s道上の重み最大のリンクであることより，W(g,g')＞W(u, v)なので， W(z,z')＞W(u,v)が成り立つ．MT(N')に(u,v)を加えると，MT(N')上の u-v道と(u,v)によって閉路が生じる．この閉路で，(u,v)以外に G1とG1以外のプロセッサ間を結ぶリンクは存在し，その任意の１つを (p,q)とする．（P, (LT(N')－{(p,q)}) ∪ {(u,v)}）は生成木で，W(p,q)＞W(u,v)より，この生成木上のリンクの総和は，MT(N')の総和より小さい．よって，MT(N')がN'のＭＳＴであることに矛盾が生じる．

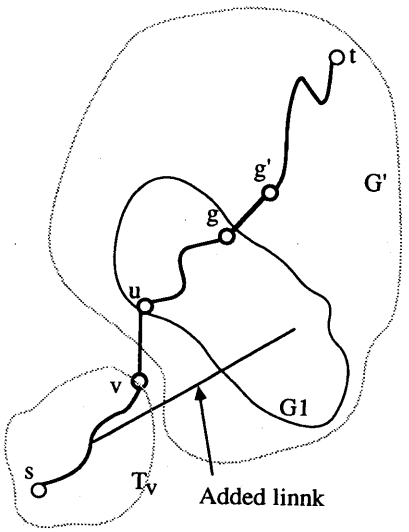

図３補題１の証明のグラフ

② T_vと G'のプロセッサ間に追加リンクが存在しない場合．(u,v)が MT(N)に属し， MT(N)はＭＳＴなので， T_vとG'のプロセッサ間を結ぶ任意のリンク (p,q)に対して，W(p,q)＞W(u,v)が成り立つ[3]．MT(N')に (u,v)を加えると，MT(N')上のu-v道と(u,v)によって，閉路が生じる．この閉路には (u,v)以外に T_vと G'のプロセッサ間を結ぶリンクは存在し，その任意の１つを(z,z')とする．以下，①と同様にして矛盾を示せる．■

MT(N)-E_pの連結成分の数に対して，次が成り立つ．

［補題２］MT(N)-E_pの連結成分の数は，０（t+f）である．但し，t=|E_s|，f=|E_d|である．

（証明） MT(N)-E_pの連結成分を頂点とするグラフ G'=（V', E'）を次のように定義する．MT(N)-E_pの連結成分を C_i（i=1,2,･･･,z）とする．頂点集合 V'= {C_i|i=1,･･･, z}，辺集合E'= {(C_i,C_j)| C_i, C_j（i≠j）に属する頂点間に分割枝が存在する} とする．G'は明らかに木である．各連結成分には１つの隣接プロセッサまたは分岐プロセッサが存在する．隣接プロセッサ数は高々2(t+f)個なので，隣接プロセッサを含む連結成分に対応する頂点数は高々2(t+f)個である．分岐プロセッサを含む連結成分に対応する頂点（以下，分岐点という）は，分岐プロセッサの定義より，G'で，２つ以上の子を持つ．従って，分岐点の数は高々2(t+f)-1個で，連結成分の数は０（t+f）個である．■

また，リンクの追加のみによってネットワーク NがN'に変化した場合（|f|=0，|t|>0）について考える．このとき，次の性質より，MT(N)に含まない Nのリンクは MT(N')に含まれない．この性質を利用して，ＭＡでは，メッセージを節約するために， MT(N)に含まれないリンクにはメッセージを送らない．

［性質１］LT(N')⊆LT(N)∪E_aが成り立つ．

（証明）MT(N)に属さないNのリンクの中で，MT(N')に属するリンクが存在するとする．その中の１つを (u,v)とする．

MT(N')から (u,v)を削除すると，MT(N')は２つの部分グラフ（G1，G2とする）に分割される．(u,v)は MT(N')のリンクなので， G1と G2のプロセッサを結ぶ任意のリンク（s,t)（∈L'-{(u,v)}）に対し，W(u,v)＜W(s,t)が成り立つ．(u,v)は MT(N)に属さないので，MT(N)に(u,v)を加えると，閉路ができ，この閉路には，(u,v)以外に G1と G2のプロセッサを結ぶリンクが存在する．その任意の１つを (x,y)とすると，W(u,v)＜W(x,y)である．（P,LT(N)－{(x,y)}∪{(u,v)}）は生成木で， W(u,v)＜W(x,y)より，この生成木のリンクの重みの総和は， MT(N)の総和より小さい．よって， MT(N)が NのＭＳＴであることに矛盾が生じる．■

次に，アルゴリズムＭＡの概略を示す．

３．２ ＭＡの概略

ＭＡは２つのフェーズからなる．第１フェーズでは，

MT(N)から分割枝を除去し，第２フェーズでは，文献(5)のアルゴリズムを適用することで，MT(N')を構成する．ここでは，フェーズ１の前処理とフェーズ１に関してのみ簡単に述べる．フェーズ２に関しては文献 (5)参照のこと．

（フェーズ１の前処理の概略）

まず，始動プロセッサは，前処理として MT(N)のリンクを用いて，削除枝があるかどうかを調べる．

①削除枝がない場合：削除枝の存在を調べるためのメッセージによって，全てのプロセッサは起動される．この場合，１つの連結成分だけが存在するので，１つの連結成分の中から任意の１つのプロセッサ（ iとする）をその連結成分のリーダーとして選ぶ．

②削除枝がある場合：始動プロセッサは全てのリンクを用いて全てのプロセッサを起動させる．そして， f個の削除枝によって分割された ０（ｆ）個の各連結成分の中からその連結成分のリーダーを選ぶ．

（フェーズ１の概略）

各連結成分のリーダー iは iの属する連結成分のリンクを介し，<START>を連結成分内の全てのプロセッサに送り，フェーズ１の開始を知らせる．<START>を受けた葉 uは，uが隣接プロセッサでないなら，uの子孫の中に隣接プロセッサは存在しないことを知らせる<NO>を親に送信する．また，uが隣接プロセッサなら，uの子孫の中に隣接プロセッサが存在することを知らせる<EXT(W(u,u'))>を親に送信する． W(u,u')は uを端とする主要道の分割枝の候補を表す．

葉以外の各 uは２つ以上の子から<EXT(val)>を受けたとき， uが分岐プロセッサであることが分かり，それ以外の場合，分岐プロセッサでないことが分かる．

分岐プロセッサ uは， uを端とする主要道の分割枝を MT(N)から削除するために， uに<EXT(val)>を送ってきたそれぞれの子に <CUT(val)>（<CUT>メッセージの valの値は<EXT>メッセージの valの値）を送信する．そして，親 u'に<EXT(W(u,u'))>を送信する．W(s,t)＝valなる<CUT(val)>を受けたプロセッサ s，tは(s,t)が分割枝であることが分かる．また，子から<NO>を受けた各プロセッサは，直ちにその子に<CUT(0)>を送る．

分岐プロセッサでない各 uは，次のいずれかを行う．①uが<EXT(val)>を受けてない場合：uが隣接プロセッサなら親の u'に<EXT(W(u,u'))>を送り，そうでないなら，u'に<NO>を送る．但し，uが根なら何もしない．

②uが１つの<EXT(val)>を受けた場合：

(a)uが根の場合．隣接プロセッサでないなら，何もせず，隣接プロセッサなら，<EXT(val)>を送ってきた子に<CUT(val)>を送る．

(b)uが根でない場合．隣接プロセッサでないなら，親u'に<EXT(val)>を送り，隣接プロセッサなら，<EXT(val)>を送ってきた子に<CUT(val)>を，親 u'に<EXT(W(u,u'))>を送る

根以外の各プロセッサは親から <CUT>メッセージを受けたとき，フェーズ１を終了し，根は全ての子からメッセージを受け，それに対する処理を終えたとき，フェーズ１を終了する．

アルゴリズムＭＡのフェーズ１（の前処理以外）での各プロセッサの厳密な動作だけを表１に示す．

３．３　メッセージ

ここでは，ＭＡのフェーズ１で用いるメッセージと変数に関して簡単に述べる．

(1)<START>メッセージ：各連結成分の根が発信するメッセージで，フェーズ１の実行開始を知らせる．

(2)<NO>メッセージ： 自分を根とする部分木上には隣接プロセッサが存在しないことを親に知らせる．

(3)<EXT(val)>メッセージ： 自分を根とする部分木上に隣接プロセッサが存在することを親に知らせる．

valは，主要道上の分割枝の候補を表す．

(4)<CUT(val)>メッセージ： 分割枝は重み valを持つリンクであることを知らせる．

３．４　局所変数

各プロセッサ uは次の局所変数を持つ．

(1) FA_u：uの親の識別子を記憶する．フェーズ１の開始時，フェーズ１の前処理で求めた根付き生成木上の uの親の識別子を記憶している．フェーズ１の開始時に，FA_u＝vとする．(u,v)が分割枝のときは，フェーズ１の終了時，FA_u＝0が成り立つ．

(2) CH_u：uの子の識別子の集合を記憶する．フェーズ１の開始時，フェーズ１の前処理で求めた根付き生成木上の uの全ての子の識別子を記憶している．フェーズ１で，値が変わることはない．

(3) MN_u：子から受けた <EXT(val)>の数を記憶する．

(4) ACK_u：<EXT(val)>，<NO>のいずれも送ってきてない子の識別子を記憶する．

(5)DIV_u：子から受けた最初の<EXT(val)>のvalを記憶．

(6)FST_u：最初の<EXT>を送ってきた子の識別子を記憶．

(7) ADJ_u：隣接プロセッサかどうかを表す論理型変数．フェーズ１開始時，隣接プロセッサのとき， 1に，隣接プロセッサでないとき， 0に初期設定されている．

(8) ST_u：uの状態を表す．状態は次のいずれかである．

・判定状態：フェーズ１の初期状態で，分岐プロセッサかどうかを調べる状態．

・通過状態：隣接でも分岐プロセッサでないことが判明し，親からの<CUT(val)>メッセージを待つ状態．

・切断状態：隣接または，分岐プロセッサであることが判明し，親からの<CUT(val)>メッセージを待つ状態．

・終了状態：フェーズ１実行終了後の状態．

４．アルゴリズムＭＡの正当性と評価

フェーズ１が有限時間内に終了し，フェーズ１でＯＭＳＴから除去されるリンクが分割枝であることは明らか．補題１より，分割枝を除去することによって得られた各連結成分は破片なので，文献 (5)の方法を用いると，フェーズ２において，有限時間内にMT(N')を正しく構成できる．従って，次の定理が成り立つ．

［定理１］アルゴリズムＭＡはＵＭＰを正しく解く．□

次に，分散アルゴリズムＭＡの通信計算量と理想時間計算量を評価する．以下，t＝|E_e|，f＝|E_d|で，　n＝|P|，e＝|L'|を表す．

［定理２］分散アルゴリズムＭＡの通信計算量は，Ｏ（nlog（t+f)＋m）である．但し，　|f|＝0のとき，m＝n+tで，|f|＞0のとき，m＝eである．□

（証明）|f|＝0のときと，|f|＞0のときに分けて示す．

①|f|＝0のとき，Ｏ（nlog（t+f)＋（n+t））：フェーズ１の前処理の通信計算量は Ｏ（n）である．フェーズ１で，<START>は MT(N)の各リンクに高々１回送信され，１つの<START>に対し，<NO>または，<EXT(val)>が１回送り返される．また，<CUT(val)>は MT(N)の各リンクに高々１回送信される．従って，フェーズ１の通信計算量は Ｏ（n）である．次に，フェーズ２の通信計算量を示す．LT(N')⊆LT(N)∪E_eより，MT(N)＋E_e（（t＋n）個のリンクからなる）のＭＳＴを構成すればよい．また，LT(N)-E_pの連結成分数はＯ（t＋f）個（補題２）で，それらはMT(N')の破片である（補題１）．従って，フェーズ２の通信計算量は Ｏ（nlog（t+f)＋（n+t））である[5]．

②|f|＞0のとき，Ｏ（nlog（t+f)＋ e）：フェーズ１の通信計算量は Ｏ（n）ある．次に，フェーズ２の通信計算量を示す．LT(N')⊆L'より，すべてのリンク（つまり，　e個のリンク）が　MT(N')のリンクになる可能性がある．①と同様に，フェーズ２の通信計算量はＯ（nlog（t+f)＋ e）であることを示せる．■

定理２より，次の定理が成り立つ．

［定理３］分散アルゴリズムＭＡのビット計算量は，Ｏ（nlog（t+f)･log n＋mlog n）である．□

［定理４］分散アルゴリズムＭＡの理想時間計算量は，Ｏ（nlog（t+f)＋n）である．□

（証明）フェーズ１とフェーズ１の前処理の理想時間計算量は共に Ｏ（n）である．フェーズ２の開始時，Ｏ（t＋f）個の破片が存在するので，フェーズ２の理想時間計算量は Ｏ（nlog（t+f)）である[5]．従って，アルゴリズムＭＡの理想時間計算量はＯ（nlog（t+f)＋ n）である．■

各プロセッサは自分が属する破片のリーダーの識別子と，どのリンクがＯＭＳＴまたは，ＮＭＳＴに属するかを記憶すればよいので，次の定理が成り立つ．

表１　アルゴリズムＭＡのフェーズ１

(a) 根となるプロセッサ iのプログラム

ST	vからのメッセージ	動　　作
開始時		DIV:=∞, MN:=0, SON:=0, ST:=判定, ACK:=CH, S(ACK,<START>).
判	<NO>	Del(v,ACK), S(v,<CUT(0)>), |ACK|＝0→ST:=終了.
定	<EXT(val)>	Del(v,ACK), MN:=MN＋1, |MN|＞2→S(v,<CUT(val)>), |ACK|＝0→ST:=終了. |MN|＝2→S(v,<CUT(val)>), S(FST,<CUT(DIV)>), |ACK|＝0→ST:=終了. |MN|＝1→ADJ＝1→S(v,<CUT(val)>), |ACK|＝0→ST:=終了. ADJ＝0→FST:=v, DIV:=val, |ACK|＝0→S(v,<CUT(0)>), ST:=終了.

(b) 根以外のプロセッサ uのプログラム

ST	vからのメッセージ	動　　作
判	<START>	DIV:=∞, MN:=0, FST:=0, |CH|≠0→ACK:=CH, S(ACK,<START>). |CH|＝0→ADJ＝1→S(FA,<EXT(W(u,FA))>), ST:=通過. ADJ＝0→S(FA,<NO>), ST:=通過.
定	<NO>	Del(v,ACK), S(v,<CUT(0)>), |ACK|＝0→ADJ＝1→ST:=切断, S(FA,<EXT(W(u,FA))>). ADJ＝0→MN≧2→S(FA,<EXT(W(u,FA))>), ST:=切断. MN＝1→S(FA,<EXT(max(W(u,FA),DIV))>), ST:=通過. MN＝0→S(FA,<NO>), ST:=通過.
	<EXT(val)>	Del(v,ACK), MN:=MN＋1, |MN|＞2→S(v,<CUT(val)>), |ACK|＝0→S(FA,<EXT(W(u,FA))>), ST:=切断. |MN|＝2→S(v,<CUT(val)>),S(FST,<CUT(DIV)>),|ACK|＝0→S(FA,<EXT(W(u,FA))>), ST:=切断. |MN|＝1→ADJ＝1→S(v,<CUT(val)>), |ACK|＝0→S(FA,<EXT(W(u,FA))>), ST:=切断. ADJ＝0→FST:=v, DIV:=val,|ACK|＝0→S(FA,<EXT(max(W(u,FA),DIV))>), ST:=通過.
通過	<CUT(val)>	val≠0→val＝W(u,FA)→FA:=0, S(FST,<CUT(0)>), ST:=終了. val≠W(u,FA)→S(FST,<CUT(val)>), ST:=終了. val＝0→ST:=終了.
切断	<CUT(val)>	val＝W(u,FA)→FA:=0, ST:=終了. val≠W(u,FA)→ST:=終了.

注１）A→Bは，条件 Aが成り立つときに，　Bを実行することを意味する．
注２）S(X,m)は，各 x∈Xに変数 mの値を送ることを意味する．
注３）コンマで区切ったステートメントは逐次実行を意味する．
注４）集合型変数 Sとその要素型変数 xに対する命令 Del(x,S)は Sから xを除去することを意味する．
注５）表では現れないメッセージは，実際にそのメッセージは送られてこないを意味する．
注６）(a)では根 iの変数を，(b)では根以外のプロセッサ uの変数を表す．

［定理５］分散アルゴリズムＭＡの局所領域計算量はＯ（d＋log n），全領域計算量はＯ（e＋ nlog n）である．但し，d＝max（$deg_N(u)$,$deg_{N'}(u)$）である．□

５．リンクとプロセッサの追加と削除が混在する場合

ここでは，リンクの追加，削除だけでなく，リンクとプロセッサの追加と削除（プロセッサが削除されるときは，そのプロセッサに接続するリンクもすべて削除されるものとする）によって，ネットワーク N＝（P，L）がN'＝（P'，L'）に変化された場合のＭＳＴ更新問題を解く通信計算量Ｏ（nlog（g+h)＋r），理想時間計算量Ｏ（nlog（g+h）＋ n）の分散アルゴリズムのアイデアを示す．但し，追加リンクの数（追加プロセッサに接続するリンクを含む）を g，削除リンクの数（削除プロセッサに接続するリンクを含む）を hとする．また，g＝0の場合，r＝n+hで，g＞0の場合，r＝e（ eはトポロジー変化後のネットワークのリンク数）である．

［定義３］Nのプロセッサ u∈Pについて，次のいずれかが成り立つとき， uをプロセッサを隣接プロセッサという．

a)削除されたリンクと， Nで接続するプロセッサ．

b)追加されたプロセッサと，N'で接続するプロセッサ．

c)追加されたリンクと，N'で接続するプロセッサ．

分岐プロセッサ，主要道，分割枝は定義２と同様に定義する．□

プロセッサが削除されるときは，接続するすべてのリンクも削除されるので，削除されたプロセッサと Nにおいて隣接するプロセッサも隣接プロセッサである．

MT(N)からすべての削除プロセッサ（P_dと表す）と分割枝を取り去り，追加されたプロセッサ（P_aと表す）も追加して得られるネットワーク MT'＝（$(P-P_d)\cup P_a$，$L-E_p$）（但し，E_pは分割枝の集合）について，次の①と②が成り立つ（補題１，２と同様にして示せる）．

①MT'は MT(N')の部分グラフ．

②MT'はＯ（g+h）個の連結成分からなる．

アルゴリズムＭＡのフェーズ１と同様にして，分割枝は求まる．分割枝を N'から取り去ってから，文献(5)のアルゴリズムを適用することにより，リンクとプロセッサの追加と削除が混在する場合のＭＳＴ更新問題を解くアルゴリズムが作れる．

定理２～５と同様にして，次の定理を示せる．

この結果は前節の定理２～５をより一般化したものであり，$P_a=P_d=\phi$の場合，前節の結果と一致する．

［定理６］リンクとプロセッサの追加と削除が混在する場合のＭＳＴ更新問題を通信計算量Ｏ（nlog（g+h)＋ r），ビット計算量Ｏ（nlog（g+h)･log n＋rlog n），理想時間計算量Ｏ（nlog（g+h)＋ n），局所領域計算量Ｏ（d＋log n），全領域計算量Ｏ（e＋nlog n）で解く分散アルゴリズムが存在する．□

６．あとがき

本稿では，リンクの削除と追加が生じた場合のＭＳＴ更新問題を解く通信計算量Ｏ（nlog（t+f)＋ m），理想時間計算量Ｏ（nlog（t+f)＋ n）の分散アルゴリズムを提案した．但し， tは追加リンク数， fは削除されたリンクの中で，ＯＭＳＴに属するリンク数で，|f|＝0の場合，m＝n＋tで，|f|＞0の場合， m＝eである．また，リンクとプロセッサの削除と追加が混在する場合のＭＳＴ更新問題を解く通信計算量Ｏ（nlog（g+h)＋r），理想時間計算量Ｏ（nlog（g+h)＋ n）の分散アルゴリズムを提案した．但し，gは追加されたリンクの総数で，hは削除されたリンクの総数で，|h|＝0の場合，r＝n＋tで，|h|＞0の場合，r＝eである．

参考文献

(1) C.Cheng,I.A.Cimet and S.P. Kumar:" A Protocol to Maintain a Minimum Spanning Tree in a Dynamic Topology", Proc. of SIGCOMM '88 Symposium, Communications Architectures & Protocols, California, pp.330-337 (1988).

(2) I.A.Cimet and S.P.Kumar:" A Resilient Distributed Algorithm for Minimum Weight Spanning Trees", Proc. of the 1987 International Conference on Parallel Processing, pp.196-203 (1987).

(3) R. Gallager, P. Humblet and P. Spira : " A Distributed Algorithm for Minimum Weight Spanning Trees", ACM TOPLAS, 5, 1, pp.66-77 (1983).

(4) 萩原兼一：" 分散アルゴリズムの複雑度について", Proc. of The Logic Programming Conference '87, pp.11-20 (1987).

(5) 朴 政鎬，増澤利光，萩原兼一，都倉信樹：" 重み最小生成木再構成の分散アルゴリズムについて－リンク削除の場合－", 信学技報，COMP 89-25 (1989).

(6) Y.H.Tsin : " An Asynchronous Distributed MST Updating Algorithm For Handling Vertex Insertions in Networks", Proc. of the Int. Conference on Parallel Processing and Applications (1987).