Vol.2011-AL-134 No.13
2011/3/7

情報処理学会研究報告
IPSJ SIG Technical Report

資源増加を許したOVSF符号割当問題に対する2競合アルゴリズム

朝廣雄一[†1]　上米良謙太[†2]　宮野英次[†2]

直交可変拡散率（OVSF）符号の割当てをオンライン問題として定式化した問題，特に，Erlebach らにより STACS'04 で導入された資源増加を許すモデル[8] について考える．本稿では，OVSF 符号木の高さ h に対して $2\lg^* h$ 個の符号木を利用する 2-競合アルゴリズムを提案する．このアルゴリズムでは，Chan らによって COCOON'09 で提案された，$3h/8+2$ 個の符号木を利用する 2-競合アルゴリズム[2] と比較して，符号木の数を大きく減らすことができた．

2-Competitive Algorithm for Online OVSF Code Assignment with Resource Augmentation

Yuichi Asahiro ,[†1] Kenta Kanmera [†2]
and Eiji Miyano[†2]

This paper studies the online OVSF (Orthogonal Variable Spreading Factor) code assignment problem with resource augmentation introduced by Erlebach *et al.* in STACS'04[8]. We propose a 2-competitive algorithm with help of $2\lg^* h$ trees for the height h of the OVSF code tree, which substantially improves the previous resource of $3h/8+2$ trees shown by Chan *et al.* in COCOON'09[2].

1. はじめに

直交可変拡散率（Orthogonal Variable Spreading Factor，略して OVSF）符号は W-CDMA （Wideband Code-Division Multiple-Access） 方式で用いられている．OVSF 符号は，マルチメディアサービスにおいて，通信内容に応じて異なる品質でデータを送受信することを可能とする．OVSF-CDMA システムにおいては，それぞれの通信に割り当てられる符号が**直交**している必要があり，各々の符号はその長さや拡散率が異なる．ここで，高いデータ通信率は，低い拡散率での通信により実現される．OVSF 符号は再帰的に生成され，それは高さ h の完全 2 分木（OVSF 符号木と呼ぶ）により表現される．OVSF 符号木の各頂点は，正確に 1 つの符号に対応するので，以下では頂点と符号を区別せずに説明を行なう．OVSF 符号木のレベル h にある根頂点は，符号 1 に対応する．ある頂点が符号 c を持つとき，その左右の子はそれぞれ符号 cc と $c\bar{c}$ を持つというように，再帰的に符号が生成される．ここで $\bar{c}$ は c を反転したものである．これにより，OVSF 符号木のレベル 0 にある，2^h 個の葉は，それぞれ長さ が 2^h ビットの符号を持つことになる．

OVSF-CDMA システムにおいては，新しい通信要求が届いた時，その要求を OVSF 符号木の一つの頂点に割り当てるが，その際，新しく割り当てられた符号と，それ以前に割り当てられ利用されている符号が直交している必要がある．ここで直交とは，OVSF 符号木における根から葉に至るどの道においても，要求が割り当てられている頂点は高々1 個しかないことを言う．Erlebach らは，この要求を割り当てる問題をオンライン問題として定式化し，オンライン OVSF 符号割当問題と呼んだ[8]．この問題においては，**割当**と**解放**の要求（**リクエスト**）の列がオンライン的に入力される．割当リクエストは，レベルを表す整数 ℓ $(0 \le \ell \le h)$ を持ち，OVSF 符号木のレベル ℓ にある頂点に割り当てられる必要がある．この割当処理に対して，コストは 1 かかると考える．一方，解放リクエストに対しては，そのリクエストに対応する割当済の頂点を解放することを行ない，この処理にはコストはかからない（コスト 0）と考える．頂点にリクエストを割り当てる際には，直交条件を満たす必要があり，既にどこかの頂点に割り当てられているリクエストを別の頂点に再割当（移動）する必要が出るかもしれない．この再割当処理のコストも 1 であると考える．以上により，アルゴリズムのコストは，割当と再割当の合計回数と定義される．オンライン OVSF 符号割当問題の目的は，コストを最小化できるような割当と再割当の処理列を探す（そのようなアルゴリズムを設計する）ことである．

オンラインアルゴリズムのコストは競合比解析により評価される．オンラインアルゴリズム ALG の**競合比**は，ALG が必要とするコストと最適オフラインアルゴリズムのコストの比の最悪値と定義される．アルゴリズム ALG の競合比が σ の時，ALG は σ-競合であるという．高さ h の OVSF 符号木は帯域幅 2^h を持っており，レベル ℓ にある割当済の頂点は，

†1 九州産業大学情報科学部情報科学科
Department of Information Science, Kyushu Sangyo University
†2 九州工業大学情報工学研究院システム創成情報工学系
Department of Systems Design and Informatics, Kyushu Institute of Technology

© 2011 Information Processing Society of Japan

そのうちの $1/2^{\ell}$，すなわち帯域幅 $2^{h-\ell}$ を消費する．本稿では，割当済の頂点が消費する帯域幅の合計は，常に 2^h 以下であると仮定する．また，一般性を失わずに，レベル h の割当リクエストは存在しないことも仮定できる．よってオンラインアルゴリズムは，必要ならば既に割り当てられているリクエストを再割当することも含め，全ての割当リクエストをどこかの頂点に割り当てねばならない．割当リクエストのためのコストは必ずかかるので，再割当処理の回数をできるだけ少なくすることが，アルゴリズム設計の方針となる．

オンライン OVSF 符号割当問題には，例えば，無線通信やメモリ管理など，様々な応用があり，活発に研究されている（関連研究のまとめとして 3) がある）．オンライン OVSF 符号割当問題は，利用できる資源の増加を許すか否かによって，主に 2 種類の問題設定に区分される．最初に，資源増加を許さない問題設定において，Erlebach らは $O(h)$-競合アルゴリズムを与えるとともに，どのような決定性オンラインアルゴリズムの競合比も 1.5 以上になることを示した[8]．その後の研究の進展[4),5),9),11)] により，現在のところ競合比の上下限として知られている最良のものは 6[5] と 2[11] である．

本稿では**資源増加**を許したオンライン OVSF 符号割当問題について考える．資源増加の考え方は，Kalyanasundaram らによってオンラインスケジューリング問題に対する解析手法の一つとして導入された[10]．近年では，資源増加はよく知られた手法の一つとなっており，スケジューリング，ページング，ビンパッキングなど様々な問題に適用されている[1),7),12)]．資源増加を許す問題設定においては，オンラインアルゴリズムの性能を，ある意味，不公平にオフラインアルゴリズムの性能と比較する．オンライン OVSF 符号割当問題においては，オフラインアルゴリズムは高さ h の符号木（主符号木と呼ぶ）を 1 個だけ利用できるのに対して，オンラインアルゴリズムは高さ h の符号木を $k(\geq 1)$ 個，すなわち，主符号木に加え，高さ h の符号木をさらに $k-1$ 個余分に利用してよい．しかしながら，アルゴリズムの目標は相変わらずコスト，すなわち割当と再割当の合計回数の最小化である．容易に分かるように，多くの符号木を利用して良いなら，より良い競合比を得ることが可能である．資源増加を許したオンライン OVSF 符号割当問題も，資源増加を許さない問題設定と同様に Erlebach らによって研究が始められ，2 個の符号木を利用すると 4-競合アルゴリズムを設計できることが示された[8]．その後，Chin らによって，9/8 個の符号木（1 個の主符号木と，高さ $h-3$ の符号木）を用いた 5-競合アルゴリズムが提案された[6]．また，最近になって，Chan らにより，再割当をしないどのようなオンラインアルゴリズムも $(h+1)/2$ 個以上の符号木を必要とすることが示されるとともに，$(h+1)/2$ 個の符号木を利用する 1-競合アルゴリズム，すなわち符号木の数が最適なアルゴリズムも提案された[2]．さらに同じ論文で，$3h/8+2$ 個の符号木を利用する 2-競合アルゴリズムと，$0<\delta\leq 4/3$ を満たす定数 δ に対して，$(11/4+4/(3\delta))$ 個の符号木を利用する $(4/3+\delta)$-競合アルゴリズムも提案された．ここで注意したいのは，前者の 2-競合アルゴリズムは，各割当リクエストに対してコスト 1 で処理し，かつ各解放リクエストに対してもコスト高々1 で処理するが，後者の $(4/3+\delta)$-競合アルゴリズムは各リクエストに対して必要とするコストが定数に限定されておらず，ならし解析により評価されていることである．

本稿の目的は，資源増加を許したオンライン OVSF 符号割当問題に対して，各リクエストに対するコストが定数であり，かつ利用する符号木の数が $3h/8+2$ よりも小さい 2-競合アルゴリズムを提案することである．より正確に述べると，提案するアルゴリズムは，Chan らのアルゴリズムと同様に，各割当リクエストに対してコスト 1 で処理し，かつ各解放リクエストに対してもコスト高々1 で処理し，利用する符号木の数は高々 $2\lg^* h$ 個である．

2. 諸 定 義

簡単のために，符号木やその部分木を単に（部分）木と書くが，それらは完全 2 分木とする．例えば，高さ $h-1$ の木を 3 個利用する場合，それらが持つ帯域幅は高さ h の木の帯域幅の半分であるので，高さ h の木を $1/2\times 3=3/2$ 個利用すると考える．葉頂点 v のレベル $l(v)$ は 0 とし，葉以外の頂点 v のレベル $l(v)$ は，v の子 u のレベル $l(u)$ を用いて $l(u)+1$ と定義する．頂点 v の **利用可能帯域幅** $abw(v)$ は $2^{l(v)}$ である．もし割当リクエスト R が頂点 v に割り当てられている（R が解放されていない）ならば，v は**割当済**という．頂点 v が**自由**とは，根頂点から葉頂点に至る道のうち v を通るものすべてが，割当済の頂点を含まないことをいう．なお，オンラインアルゴリズムは常に **直交条件**，すなわち，根頂点から葉頂点に至るどの道においても割当済の頂点は高々1 個であるという条件を満たさねばならない．以下では（部分）木の状態を色（白，黒，灰の 3 色）を用いて表現する: 木 T のすべての頂点が自由なとき T は白色であり，自由な頂点がない場合に黒色であり，そのどちらでもない（すなわち自由な頂点と割当済頂点の両方がある）場合に灰色とする．

（部分）木 T に対して，T 内の割当済頂点の集合を $A(T)$ で表す．また（部分）木の集合 S に対して，$A(S)$ は S に含まれる木にあるすべての割当済頂点の集合を表す．高さ h' を持つ（部分）木 T の**利用可能帯域幅** $abw(T)$ は，その根頂点が持つ帯域幅と考えればよく，$2^{h'}$ である．（部分）木の集合 S の利用可能帯域幅 $abw(S)$ は $\sum_{T\in S} abw(T)$ で定義される．（部分）木 T 内のレベル $\ell(v)$ にある割当済頂点 $v\in A(T)$ の**消費帯域幅** $cbw(v)$ は $2^{\ell(v)}$ である．よっ

© 2011 Information Processing Society of Japan

て，(部分) 木 T の消費帯域幅は $\sum_{v\in A(T)} cbw(v)$ と定義され，$cbw(A(T))$ または $cbw(T)$ で表現する．また（部分）木の集合 S に対して，消費帯域幅 $cbw(S)$ は $\sum_{T\in S} cbw(T)$ である．

記号 $\lg n$ を，底が 2 の対数，すなわち $\lg n = \log_2 n$ とする．関数 $f(n)$ について，値 n に対して $f(n)$ を繰返し i 回適用するものを $f^{(i)}(n)$ で表す．例えば，$\lg^{(3)} n = \lg\lg\lg n$ である．記号 $\lg^* n$ を $\lg^* n = \min\{i \geq 0 \mid \lg^{(i)} n \leq 1\}$ と定義する．主符号木の高さ h に対して，$\ell_0 = h$ とし，さらに $1 \leq i \leq \lg^* h$ に対して ℓ_i で $\lceil \lg^{(i)} h \rceil$ を表すことにする．つまり，$\ell_1 = \lceil \lg^{(1)} h \rceil$，$\ell_2 = \lceil \lg^{(2)} h \rceil$，...，$\ell_{\lg^* h - 1} = \lceil \lg^{(\lg^* h - 1)} h \rceil (= 2)$，$\ell_{\lg^* h} = \lceil \lg^{(\lg^* h)} h \rceil (= 1)$ となる．ここで，高さ $h - \ell_i - 1$ と利用可能帯域幅 $2^{h-\ell_i-1}$ を持つ部分木を i-**部分木**と呼ぶことにする．

本稿では，既存の研究[2),6),7)] にならい，入力について以下の 2 個の制限を設ける．最初の仮定では，最適オフラインアルゴリズムは，主符号木の帯域幅 2^h しか消費しないことを保証する．そして次の仮定は，レベル h の割当リクエストについては考えないというものである．そのような割当リクエストがあると，それまで割当てられているリクエストすべてを解放しないといけないので，結局そのようなリクエストの箇所で入力列を区切り，それぞれの部分列ごとに対処すればよいと考えられるので，この仮定が設けられる．

仮定 1　全体での消費済帯域幅は常に 2^h 以下である．

仮定 2　レベル h の割当リクエストは存在しない．

3. $2\lg^* h$ 個の木を用いる 2-競合アルゴリズム

本節では，高さ h である木を $2\lg^* h$ 個だけ用いる 2-競合アルゴリズム ALG を設計する．主なアイデアは，以下の通りにまとめられる:

(i)　まず 0 から $h-1$ までのレベルを $\lg^* h$ 個のグループに分割する．

(ii)　次に，これらのグループそれぞれに対して，ある高さの部分木をある個数ずつ用意する．ここで用意する部分木は，そのグループに属するレベルを持つ割当リクエスト間で共有され，割り当てが行なわれるが，一旦あるレベルのリクエストが割り当てられると，他のレベルのリクエストはその部分木には割り当てられない．ただし割り当てられていた全てのリクエストが解放された場合，その部分木は，その後，他のレベルのリクエストを割り当てても良い．これら部分木の高さや個数については後で詳しく説明するが，各グループで利用される部分木の帯域幅の合計は高々2×2^h であり，高さ h である 2 個の木を利用すると言える．

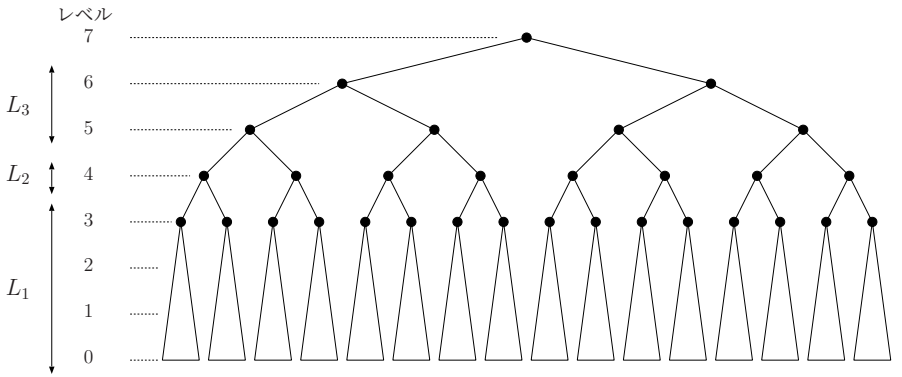

図 1　レベルとグループ分け
Fig. 1　Levels and groups of levels

(iii)　再割当の処理は，1 回の解放リクエストについて，高々1 回しか行なわない．

まず，3.1 節で，レベルのグループ分けと，用意する部分木の高さや個数を説明する．アルゴリズムの動作としてリクエストを処理する際に，部分木に対してその状態である色を表現するラベルを付けたり変更したりするので，その処理について，3.2 節で説明する．そして，3.3 節において，アルゴリズム ALG と，その競合比について述べる．

3.1 部分木の準備

まず，$1 \leq j \leq \lg^* h - 1$ に対して，$L_j = \{i \mid h - \ell_{j-1} \leq i \leq h - \ell_j - 1\}$ とし，$L_{\lg^* h} = \{h-2,\ h-1\}$ とすることにより，レベルを $\lg^* h$ 個のグループに分ける．例えば，$h = 7$ とすると，$\ell_0 = 7$，$\ell_1 = \lceil \lg 7 \rceil = 3$，$\ell_2 = \lceil \lg\lg 7 \rceil = 2$ であるので，$L_1 = \{0,1,2,3\}$，$L_2 = \{4\}$，$L_3 = \{5,6\}$ となる（図 1 参照）．

$1 \leq j \leq \lg^* h - 1$ に対して，L_j に属するレベルのリクエストを処理するために，$(\ell_{j-1} - \ell_j) + 2^{\ell_j+1}$ 個の j-部分木を用意し，これらの集合を $\mathcal{T}_j$ と名付ける．この j-部分木の数は，$(\ell_{j-1} - \ell_j)$ が L_j に属するレベルの個数であり，この数は高々 2^{ℓ_j+1} である（この事実については，補題 1 の証明で確認する）とともに，2^{ℓ_j+1} 個の j-部分木の総帯域幅が 2^h （すなわち高さ h の木の帯域幅）であることに由来する．最後のグループ $L_{\lg^* h}$ に対しては，高さ h の木を 2 個用意し，それぞれ T_{h-1} と T_{h-2} と名付ける．ここで，高さ h の主符号木は，この T_{h-1} と T_{h-2} のいずれかとして利用されるものとする．以下の補題は，以上のように用意した（部分）木の総量に対する上界を与える．

© 2011 Information Processing Society of Japan

補題 1 ALG のために用意した高さ h の木は，高々$2\lg^* h$ 個である．

証明. まず，$1 \le j \le \lg^* h - 1$ に対して，$\ell_{j-1} \le 2^{\ell_j+1}$ が成り立つことが，以下の式から分かる．

$$\lg \ell_{j-1} = \lg\lceil \lg^{(j-1)} h \rceil \le \lg(2\lg^{(j-1)} h) = \lg^{(j)} h + 1 \le \lceil \lg^{(j)} h \rceil + 1 = \ell_j + 1$$

これを踏まえて，グループ L_j のために用意した $\mathcal{T}_j$ の利用可能帯域幅 $abw(\mathcal{T}_j)$ を求める．$\mathcal{T}_j$ には，$(\ell_{j-1} - \ell_j) + 2^{\ell_j+1}$ 個の j-部分木が含まれており，それらの高さは $h - \ell_j - 1$ であるので，$abw(\mathcal{T}_j)$ は以下の通り見積もれる．

$$\begin{aligned} abw(\mathcal{T}_j) &= ((\ell_{j-1} - \ell_j) + 2^{\ell_j+1}) \cdot 2^{h-\ell_j-1} \\ &< \ell_{j-1} \cdot 2^{h-\ell_j-1} + 2^h \\ &\le 2^{\ell_j+1} \cdot 2^{h-\ell_j-1} + 2^h \\ &\le 2^{h+1} \end{aligned}$$

ここで，2 個目の不等号は，上で求めた事実 $\ell_{j-1} \le 2^{\ell_j+1}$ による．以上により，L_1 から $L_{\log^* h-1}$ までの $\log^* h - 1$ 個のグループに対して，高さ h の木が 高々 2 個ずつ用意されていると考えられる．グループ $L_{\lg^* h}$ については，高さ h の木を 2 個用意したので，アルゴリズム ALG のために用意した高さ h である木の個数は，高々$(\lg^* h - 1) \cdot 2 + 2 = 2\lg^* h$ である． □

3.2 部分木へのラベル付け

アルゴリズム ALG は $\mathcal{T}_j$ に含まれる各 j-部分木に対して，その状態である，白色または灰色，黒色を示すためにラベルを付けたり消したりする．また，それらのラベルは，色と共に，その j-部分木に対してどのレベルのリクエストが割り当てられているかも示す．ここで，先にも述べたように，灰色や黒色の各 j-部分木には，それぞれある単独のレベルのリクエストのみしか割り当てされていない状態になる．最初は，$\mathcal{T}_j$ 中の各 j-部分木は白色であり，ラベルがついていない．j-部分木につけるラベルとして次の 2 種類を用いる．

ラベル $B_{i,k}$ ：その j-部分木はレベル i のリクエストが割り当てられており，黒色である．また黒い j-部分木の中で k 番目のものである[⋆1]．

ラベル G_i ：その j-部分木はレベル i のリクエストが割り当てられており，灰色である．

ラベル $B_{i,k}$ が付いている j-部分木，あるいはラベル G_i が付いている j-部分木と書くかわりに，j-部分木 $B_{i,k}$ （または単に $B_{i,k}$），j-部分木 G_i（または単に G_i）とそれぞれ書くことにする．レベル i のリクエストのために使われている 黒 j-部分木と灰 j-部分木の個数を，それぞれ b_i と g_i で表す．さらに，$\mathcal{T}_j$ 内の白 j-部分木の個数を w_j で表す．ここで，$\mathcal{T}_j$ 内に含まれる j-部分木の個数より，$w_j + \sum_{i \in L_j}(b_i + g_i) = |\mathcal{T}_j| = (\ell_{j-1} - \ell_j) + 2^{\ell_j+1}$ を満たすことが言える．後で示す観察 1 により，アルゴリズムの実行中には，任意の i に対して， $g_i \in \{0, 1\}$ である．これにより，灰色の j-部分木についてのラベルは，各レベル i について，G_i の 1 個だけしか使わないで済むことが分かる．

各 j-部分木の個数を表す b_i，g_i，w_j の値は，アルゴリズム ALG の実行中に変化（増加または減少）する．例えば，白 j-部分木にリクエストが割り当てられることにより灰色に変わったり，黒 j-部分木の頂点からリクエストが解放されることにより灰色に変わったりする．よって，アルゴリズムはこれらの変化を管理せねばならない．各リクエストを処理した直後に，アルゴリズムは各 j-部分木のラベルならびに，各変数 b_i，g_i，w_j の値を更新する．j-部分木 T の色変更に関わる処理として，下記の 7 種類がある:

- 白 j-部分木 T が，レベル i のリクエストを割り当てられたことにより黒色に変わる:
  T にラベル B_{i,b_i+1} を付け，$w_j := w_j - 1$，$b_i := b_i + 1$ とする．
- 白 j-部分木 T が，レベル i のリクエストを割り当てられたことにより灰色に変わる:
  T にラベル G_i を付け，$w_j := w_j - 1$，$g_i := 1$ とする．
- ラベル $B_{i,k}$ を持つ黒 j-部分木 T が白色に変わる:
  ラベル $B_{i,k}$ を T から消し，もし $b_i \ge 2$ かつ $k \ne b_i$ ならば，黒 j-部分木 B_{i,b_i} のラベルを $B_{i,k}$ に変える．そして，$b_i := b_i - 1$，$w_j := w_j + 1$ とする．
- ラベル $B_{i,k}$ を持つ黒 j-部分木 T が灰色に変わる:
  ラベル $B_{i,k}$ を G_i に変え，もし $b_i \ge 2$ かつ $k \ne b_i$ ならば，黒 j-部分木 B_{i,b_i} のラベルを $B_{i,k}$ に変える．そして $g_i := 1$，$b_i := b_i - 1$ とする．
- ラベル G_i を持つ灰 j-部分木 T が白色に変わる:
  ラベル G_i を T から消し，$g_i := 0$，$w_j := w_j + 1$ とする．
- ラベル G_i を持つ灰 j-部分木 T が黒色に変わる:
  ラベル G_i を B_{i,b_i+1} に変え，$g_i := 0$，$b_i := b_i + 1$ とする．
- T の状態（色）に変更なし:
  何もしない．

以上のそれぞれの更新は定数時間で実行できる．この更新手続きを Update(T) として，ALG の記述の中で用いる．処理対象の j-部分木 T が黒色の場合には，Update(T) は，直接の処理

⋆1 黒い j-部分木の順序については任意で構わない．添字の k を導入する理由は，黒い j-部分木を単に区別するためである．

© 2011 Information Processing Society of Japan

対象である T 以外にも j-部分木 B_{i,b_i} のラベルを変更することに注意されたい．

3.3 アルゴリズム ALG

リクエストのレベルに応じて，アルゴリズム ALG は，以下に記述する手続きのうち適切なものを選択して実行する．具体的には，各 L_j に対して，それぞれ 1 個ずつ手続きを用意する．まず，リクエストのレベルが $h-1$ または $h-2$ の場合には，アルゴリズムは非常に単純である．

レベル $i \in \{h-1, h-2\}$ に対する ALG

レベル i の割当リクエスト R ：R を T_i 内のレベル i にある自由な頂点に割り当てる．
レベル i の解放リクエスト R ：T_i 内にある，R に対応する割当済頂点を解放する

グループ L_j $(1 \le j \le \lg^* h - 1)$ に含まれるレベルのリクエストに対しては，アルゴリズムは以下のように動作する．

レベル $i \in L_j$ に対する ALG

レベル i の割当リクエスト R ：g_i の値に応じて，次のどちらかを実行する．

場合 A1 （$g_i = 1$）：
R を G_i 内のレベル i にある自由な頂点に割り当て，Update(G_i) を行う．

場合 A2 （$g_i = 0$）：
$\mathcal{T}_j$ 内にある白 j-部分木を 1 個選び，それを T と呼ぶことにする．R を T 内のレベル i にある自由な頂点に割り当て，Update(T) を行う．

レベル i の解放リクエスト R ：まず R を，それが割り当てられている頂点 u （j-部分木 T 内にあるとする）から解放する（これにより u が自由になる）．そして，g_i の値に応じて，次のどちらかを実行する．

場合 R1 （$g_i = 1$）：さらに 2 つの場合に分けて処理する．
（i）もし T に ラベル $B_{i,k}$ が付いているなら，G_i 中の任意の割当済頂点 v を 1 個選び，そこに割り当てられているリクエストを v から u へ再割当する．そして，Update(G_i) を実行する．
（ii）もし T にラベル G_i が付いているなら，Update(G_i) を実行する（再割当は行わない）．

場合 R2 （$g_i = 0$）：Update(T) を実行する（再割当は行わない）．

図 2 は ALG の動作例を示している．例えば，OVSF 符号木の高さ h が 7 とすると，グループ L_1 のための $\mathcal{T}_1$ として 20 $(= (\ell_0 - \ell_1) + 2^{\ell_1+1} = (7-3) + 2^4)$ 個の 1-部分木を用意することになる．アルゴリズム ALG は次のように動作する．(1) 割当リクエストが 19 個届いたとする．その内訳は，レベル 3 が 1 個，レベル 2 が 5 個，レベル 1 が 6 個，レベル 0 が 7 個とする．そしてその後，4 個の解放リクエスト，すなわち，レベル 1 の頂点 v_{12}，レベル 0 の頂点 v_{17}，v_{19}，v_{20} が届いたとする．そうすると，割当済頂点は図中の黒丸で示したものである．レベル 3，2，1，0 にはそれぞれ，1 個の黒 1-部分木 $B_{3,1}$，2 個の黒 1-部分木 $B_{2,1}$，$B_{2,2}$ と 1 個の灰 1-部分木 G_2，1 個の黒 1-部分木 $B_{1,1}$ と 1 個の灰 1-部分木 G_1，1 個の灰 1-部分木 G_0 がある．$\mathcal{T}_1$ 中には，13 個の白 1-部分木が残っている（すなわち $w_1 = 13$）が，それらは図中には記載していない．(2) 次に，レベル 2 の頂点 v_4 に対する解放リクエストが届いたと仮定する．$B_{2,2}$ から v_4 の解放が行われ，ALG は **場合 R1**-（i）を実行する．すなわち，灰 1-部分木 G_2 内の v_6 に割当てられているリクエストを v_4 に再割当てする．灰色だった 1-部分木 G_2 は白色に変わったので，ラベル G_2 が消される．そして，g_2 と w_1 はそれぞれ 0 と 14 となる．(3) そして，例えば，レベル 3 の割当リクエストが届いたとする．アルゴリズム ALG は**場合 A2** を実行し，白 1-部分木が一つ選択され，ラベル $B_{3,2}$ が付けられる．そして，届いたリクエストは，$B_{3,2}$ の根に割当てられ，w_1 は 13 になり，b_3 は 2 となる．

アルゴリズム ALG は，割当リクエストが届いた時に必ず自由な頂点を発見できることを以下で示す．いくつかの証明は，仮定 1 に基づく背理法により行われる．レベル $h-1$ と $h-2$ については，以下の補題が成立する．

補題 2 任意の $i \in \{h-1, h-2\}$ に対して，レベル i の割当リクエストが届いた時，T_i 中のレベル i に自由な頂点が必ず存在する．

証明. T_i 中のレベル i に自由頂点が存在しないと仮定する．T_i はレベル i の割当リクエストにしか利用されないので，この仮定は，$cbw(T_i) = 2^h$ を意味する．すなわち，既に消費されている帯域幅と，届いたリクエストの分の帯域幅を合計すると $2^h + 2^i > 2^h$ となり，これは仮定 1 に反する．すなわち，T_i 中のレベル i に自由な頂点が必ず存在する． □

$h-2$ よりも低いレベルに関して，以下で議論を行う．ALG は 全ての i について，$g_i \in \{0, 1\}$ であることを仮定して記述されているので，g_i に関する観察からまず始める．

観察 1 任意の $i \in L_j$ $(1 \le j \le \lg^* h - 1)$ に対して，常に $g_i \in \{0, 1\}$ である．

割当と解放リクエストの列について詳しく考えてみる．最初は，明らかに $g_i = 0$ である．(1) レベル i の割当リクエスト R が届いた時に $g_i = 0$ だったとする．ALG は **場合 A2** を実行し，$\mathcal{T}_j$ 中の一つの白 j-部分木にラベル G_i （または B_{i,b_i+1}）を付ける．これにより

© 2011 Information Processing Society of Japan

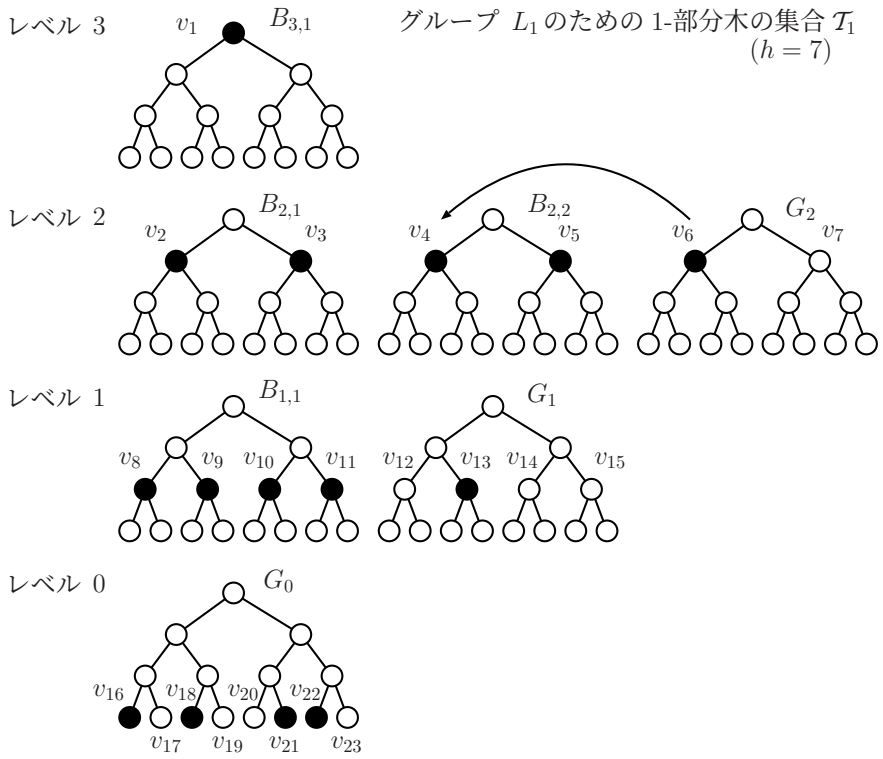

図 2 アルゴリズム ALG の動作例
Fig. 2 Algorithm ALG's behavior

$g_i = 1$（または 0 のまま）となる．(2) レベル i の解放リクエスト R が届いた時に $g_i = 0$ だったとする．ALG は **場合 R2** を実行し，一つの黒 j-部分木が灰色（または白色）になり，g_i は 1（または 0 のまま）となる．(3) レベル i の割当リクエスト R が届いた時に $g_i = 1$ だったとする．ALG は **場合 A1** を実行し，R は灰 j-部分木 G_i に割り当てられる．もし，この j-部分木が灰色のままであれば $g_i = 1$ であるし，自由な頂点がなくなってしまった（黒色になった）のであれば，$g_i = 0$ となる．(4) 最後に，レベル i の解放リクエスト R が届いた時に $g_i = 1$ だったとする．R は黒 j-部分木 $B_{i,k}$ から解放される（**場合 R1**-（i)），または灰 j-部分木 G_i から解放される（**場合 R1**-（ii））のいずれかである．どちらの場合でも G_i 内にある割当済の頂点数は減少する．なぜならば，**場合 R1**-（i）においても，1 つのリクエストが G_i から $B_{i,k}$ へ再割当されるからである（これにより $B_{i,k}$ は黒のままである）．もし j-部分木 G_i が灰色のままであるなら，$g_i = 1$ であるし，もし白色になったのなら $g_i = 0$ となる．以上により，常に $g_i \in \{0, 1\}$ であることが分かる．

アルゴリズム ALG の正しさ，すなわち割当リクエストが届いた時に必ず自由な頂点を発見できることは，以下の補題により保証される．

補題 3 レベル $i \in L_j$ の割当リクエスト R が届いた時，以下の （i）または（ii）のどちらかが必ず成立する．すなわち ALG は R のための自由な頂点を必ず発見できる:

（i） 灰 j-部分木 G_i が $\mathcal{T}_j$ 内に存在する．

（ii） 白 j-部分木が $\mathcal{T}_j$ 内に残っている．

証明. 背理法により証明するため，次の仮定を行う:

（仮定 3） レベル i のための灰 j-部分木，白 j-部分木のいずれも $\mathcal{T}_j$ には存在しない．

$\mathcal{T}_j$ 中における灰 j-部分木の総数について考えてみると，高々$\ell_{j-1} - \ell_j - 1$ のはずである．その理由は，任意の $k \in L_j$ （$k \neq i$）について g_k は高々1 であることが観察 1 により分かることと，$|L_j| = \ell_{j-1} - \ell_j$ だからである．(仮定 3）により，$\mathcal{T}_j$ 中の他の $2^{\ell_j+1} + 1$ 個の j-部分木はすべて黒である．j-部分木の帯域幅は $2^{h-\ell_j-1}$ なので，これらの黒 j-部分木が消費する帯域幅の合計は $(2^{\ell_j+1} + 1) \times 2^{h-\ell_j-1} > 2^h$ となり，仮定 1 に反する．以上により（仮定 3）が誤り，すなわちレベル i のための灰 j-部分木，白 j-部分木のいずれかが $\mathcal{T}_j$ に存在することが言える． □

これまでの議論から，次の定理を得る．

定理 1 ALG は 2-競合であり，高さ h の木を高々 $2\log^* h$ 個だけ利用する．

証明. （木の個数について）高さ h の木の個数は補題 1 で与えられ，$2\lg^* h$ である．

（正しさについて）補題 3 により，割当リクエストが届いた時 ALG は必ず自由頂点を発見できる．また，解放リクエストについての処理が行えることは明らかである．

（競合比について）各割当リクエストに対して ALG はそのリクエストをどこかの自由頂点に割当し，再割当処理は行わないので，コスト 1 だけかかる．一方，解放リクエストに対しては，再割当処理は 1 個の解放リクエストに対して高々1 回（**場合 R1**-（i）のみ）しか行わない．すなわち，入力中の割当リクエストと解放リクエストの数をそれぞれ n と m とすると，ALG の総コストは $n + m$ である．最適オフラインアルゴリズムも n 以上のコストは必要であり，解放リクエストの数は割当リクエストの数よりも小さい，すなわち $m \leq n$

© 2011 Information Processing Society of Japan

であるので，ALG の競合比は高々$(n+m)/n \leq 2$ である．□

4. おわりに

本稿では，資源増加を許したオンライン OVSF 符号割当問題に対して，主符号木の高さ h に対して，$2\lg^* h$ 個の符号木を利用する 2-競合アルゴリズムを提案した．今後の研究課題としては，より少ない個数の符号木を用いて同等の競合比を実現するアルゴリズムや，より小さい競合比を持つアルゴリズムの開発が挙げられる．

謝辞 本研究の一部は，文部科学省の科学研究費補助金 (20500017 と 22700019) の助成を受けた．

参考文献

1) Albers, S., Arora, S., and Khanna, S.: Page Replacement for General Caching Problems. In *Proc. SODA*, pp.31–40 (1999).
2) Chan, J.W.T., Chin, F.Y.L., Ting, H.F., and Zhang, Y.: Online Tree Node Assignment with Resource Augmentation. In *Proc. COCOON 2009*, pp.358–367 (2009).
3) Chan, J.W.T., Chin, F.Y.L., Ting, H.F., and Zhang, Y.: Online Problems for Frequency Assignment and OVSF Code Assignment in Wireless Communication Networks. *ACM SIGACT News*, Vol. 40 (3), pp.86–98 (2009)
4) Chin, F.Y.L., Ting, H.F., and Zhang, Y.: A Constant-competitive Algorithm for Online OVSF Code Assignment. In *Proc. ISAAC 2007*, pp.452–463 (2007).
5) Chin, F.Y.L., Ting, H.F., and Zhang, Y.: Constant-competitive Tree Node Assignment. *manuscript*.
6) Chin, F.Y.L., Zhang, Y., and Zhu, H.: Online OVSF Code Assignment with Resource Augmentation. In *Proc. AAIM 2007*, pp.191–200 (2007).
7) Epstein, L. and Stee, R.v.: Online Bin Packing with Resource Augmentation. *Discrete Optimization*, 4, pp.322-333 (2007).
8) Erlebach, T., Jacob, R., Mihalák, M., Nunkesser, M., Szabó, G., and Widmayer, P.: An Algorithmic View on OVSF Code Assignment. In *Proc. STACS 2004*, 270–281 (2004) (conference version of *Algorithmica*, 47(3), 269–298 (2007)).
9) Forišek, M., Katreniak, B., Katreniaková, J., Královič, R., Královič, R., Koutný, V., Pardubská, D., Plachetka, T., and Rovan, B.: Online Bandwidth Allocation. In *Proc. ESA 2007*, pp.546–557 (2007).
10) Kalyanasundaram, B., and Pruhs, K.: Speed is as Powerful as Clairvoyance. *J. ACM*, 32, pp.617–643 (2000)
11) Miyazaki, S., and Okamoto, K.: Improving the Competitive Ratio of the Online OVSF Code Assignment Problem. In *Proc. ISAAC 2008*, pp.64–76 (2008).
12) Phillips, C.A., Stein, C., Torng, E., and Wein, J.: Optimal Time-critical Scheduling via Resource Augmentation. *Algorithmica*, 32(3), pp.163–200 (2002).

© 2011 Information Processing Society of Japan