マルチメディアプロシーディングパッケージ再編集のための MPP ビットストリーム変換方式に関する一検討

笠井 裕之　　高屋 和幸　　児玉 明　　富永 英義

早稲田大学 理工学部 電子・情報通信学科

〒169 東京都新宿区大久保 3–4–1
kasai@tom.comm.waseda.ac.jp

筆者らは，利用者の利用面，機能面に沿って構造化されたマルチメディアパッケージ情報によりサービス提供する，" マルチメディアプロシーディング (MP)" を提案している．MP への要求条件としてパッケージ情報 (MPP) の柔軟な編集・加工機能が挙げられる．本稿では，コンテント情報が内容的階層構造をもつ MPP を対象とした，コンテント情報単位での編集処理手法について検討した．特に MPP 再編集・加工を実現するためのデータとして論理構造データを導入し，論理構造データを編集することにより，一括して MPP ビットストリーム変換を実現する手法について述べた．最後に処理量の観点から変換処理について考察し，変換処理時間が MPP ビットストリーム内におけるデータ検索のためのスキャニング処理に大きく依存することを明らかにした．

A Study on MPP Bitstreams Transformation Method for the Re-Edition of Multimedia Proceedings Package

Hiroyuki KASAI　Kazuyuki TAKAYA　Mei KODAMA　Hideyoshi TOMINAGA

Dept. of Electronics, Information and Communication Engineering, WASEDA University

3–4–1 Ohkubo Shinjuku-ku, Tokyo, 169 JAPAN
kasai@tom.comm.waseda.ac.jp

In this paper, we study the Multimedia Proceedings(MP) bitstream transfomation method for re-edition of Mulitmedia Proceedings Package(MPP). At first, we summarize the requirement for re-editting function in MP, and show the logical structure data for re-edition. Next the re-editting processing for the logical structure data is showed, and MPP bitstream transformation method using this data is explained. In particular, we indicate the function of each transformation processores, which are MPP Parser, Decoder, Encoder and Controller, and transformation processing order. At last, we realize the transformaaation, and estimated thie transformation from the viewpoint of processing times.

1. はじめに

マルチメディアサービスにおいては，利用者は有意な情報を容易に取得・再生でき，さらに利用者の意志に基づいて情報を変換・編集することにより情報を再利用できることが要求される．そのためには，マルチメディア情報を予め利用面に沿って構造化しておくことが重要である．

筆者らは，利用者の利用面，機能面に沿って構造化されたマルチメディアパッケージ情報によりサービス提供する，”マルチメディアプロシーディング”を提案している(1)．マルチメディアプロシーディングへの要求条件として，パッケージ構成情報の編集，加工が可能であり，パッケージ情報の再利用が可能であることが挙げられる．本稿では，マルチメディアプロシーディングの編集・加工に着目し，編集・加工への要求条件を整理し，編集・加工のためのデータ構造を明らかにする．そして編集・加工を実現するビットストリーム変換方式について述べ，処理量の観点から変換処理について考察する．

2. MPP 編集・加工

最初に，MPにおけるMPP編集・加工機能の意義を述べ，MPP編集・加工機能への要求条件をまとめる．最後に，本稿で対象とするMPP変換方式について概説する．

2.1 MP における編集・加工機能の意義

マルチメディアプロシーディングでは，パッケージ情報の自由かつ柔軟な編集・加工機能性，再利用性が要求される．情報の編集・加工機能により，より有効的な情報の再利用が可能となり，より円滑な情報流通が促進される．ユーザ自身の意志に基づいて，パッケージ情報の編集・加工・合成が可能となり，新たな情報を作成することが容易となる．例えばMPP構造を持つ電子本を考えた場合，新たな内容付加によるMPP電子本の更新や，一部分の情報の抽出による新たなMPP電子本の作成，複数の電子本の内容を凝縮した新たなMPP電子本の作成が可能となる．またユーザによるMPP再生時の動作履歴を，再生履歴情報としてパッケージ内に保存，つまり再生履歴情報の内容付加によるMPP編集により，前回終了ポイントからの再生や履歴情報に基づく自動再生が可能となる．またMPPへのネットワークを経由したメディア情報，操作情報の付加・更新により，リモートアクセス，リモートコントロールが可能となり，MPPの遠隔操作サービスへの拡張も考えられる．さらにネットワークを経由したMPPの分配・伝送を考えた場合，ネットワーク帯域に基づいて，パッケージ内のメディア情報を厳選し，新たなパッケージを作成することで，帯域に応じたMPP伝送が可能となる．また以上のような編集・加工処理において，編集履歴情報をパッケージ内に保存することにより，編集されたMPPから編集前のMPPへ遡ることが必要となる(2)．

2.2 MP における編集・加工機能への要求条件

本節では，**2.1.** で述べた編集・加工機能を実現する際の，変換処理への要求条件について以下にまとめる．

(1) 自由かつ柔軟な編集が可能であること
MPPに対する編集・加工処理の分類化に関しては文献[2]において述べられているが(3)，MPP編集・加工処理において，特にコンテント情報単位での編集・加工機能への要求が高まると考えられる．コンテント情報の削除，追加・挿入，変換・入れ換え処理が自由かつ柔軟に行なわれなければならない．そして複数のMPPに属する複数コンテント情報を結合し，一つのMPPを作成することも必要である．さらにコンテント情報が内容的階層構造を持つMPPにおいては，単純なコンテント情報削除，挿入，変換を行なうことは困難であり，内容的階層構造を考慮した変換処理が必須である．

(2) 変換処理が簡易であること
特にデータ構造について考えた場合，データそのものを編集対象とするデータ部と，データは編集対象とせずデータ単位で扱うことのできるデータ部とが互いに独立であることが望まれる．これにより，特定データのみの編集処理により変換が実現されると考えられる．

(3) 再生を考慮したデータ構造に変換すること
パッケージ情報の再生を考えた場合，目的コンテントデータへのアクセス時間が問題となる．そこで変換処理においては，再生時のシナリオに基づいて，MPPビットストリーム内の各コンテント情報の配置を考慮する必要がある．

(4) 変換されたMPPから元のMPPへ遡ることが可能であること

以上の要求条件から，本稿では，内容的階層構造を有するインタラクティブ再生モードを持つMPPを編集対象として，コンテント情報単位での編集処理手法について検討する．特に編集・加工処理に適した論理構造データ記述法とその編集処理について述べ，編集データを利用したMPPビットストリーム変換方式について述べる．同時に，編集処理における編集処理履歴情報のビットストリーム内への付加手法についても述べる．

3. 論理構造モデルとデータ記述

本稿では，編集・加工機能性の向上を目的として，コンテント情報間の動作記述・階層記述データとして，"コンテント論理構造データ"を導入する．論理構造データは編集・加工機能だけでなく，MPP再生時の全体内容把握，他項へのランダムアクセス性の向上を実現する．本稿における論理構造データとは，複数コンテントの内容に基づく順序関係，詳細度の関係に関する構造をツリー構造で表現したものである．階層構造を示す構造記述データや時間関係データ，各種属性データとデータ実体を分けて管理する方式は様々な分野，アプリケーションサービスにおいて用いられており[4][5]，文書処理分野ではODA[6]やSGML[7]に見られる．

以下では，今回想定するコンテントツリー構造(論理構造)モデルを示し，想定モデルを記述する際の論理構造データ記述について述べる．ただし想定論理構造モデルの拡張により，他のモデルの実現は可能であり，汎用性は損なわれないと考える．

3.1 論理構造データモデル

図1に示すようなツリー構造・論理構造モデルを想定する．図1において，横方向は順序的・時間的方向を示し，一方，縦方向は詳細度を示し，下方は詳細方向である．また図1中，四角形で表現されるコンテントを"基幹コンテント"，角丸四角形を"付随コンテント"と定義する．さらに，内容あるいは時間に基づく順序関係を示すコンテント間リンクを"順序リンク"，詳細度関係を示すコンテント間リンクを"詳細度リンク"，付随関係を示すリンクを"付随リンク"と定義する．

以下に想定モデルの設定諸条件をまとめる．

- 内容的・時間的順序方向(横方向)と詳細度方向(縦方向)の2次元である．
- n階層にコンテントが存在する場合，必ず$(n-1)$階層にコンテントは存在する．

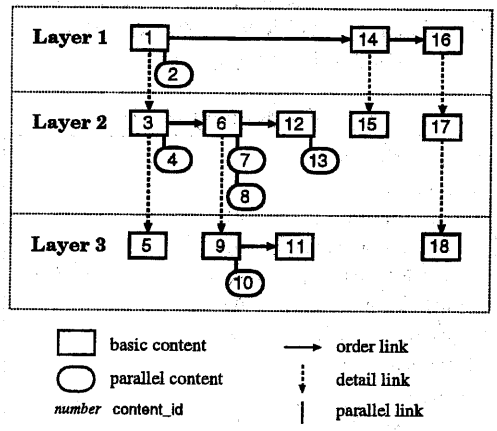

図 1: 想定論理構造モデル

- 基幹コンテントから張られるリンク数は，順所リンク及び詳細度リンクは0本もしくは1本，付随リンクは0本以上である．
- 付随コンテントから，いかなるリンクも張られることはない．

3.2 論理構造データ記述

論理構造データに含まれるデータを以下に示す．

- content_id
  コンテントの識別番号．コンテント情報内の各コンテントとの相互参照番号．
- content_layer_level
  階層番号，詳細度データ
- parallel_content_id
  付随コンテントの識別番号
- detail_content_id
  詳細度方向に存在するコンテントの識別番号
- next_content_id
  内容に基づく順序関係において次に来るコンテントの識別番号

構造記述は，各コンテントの階層構造を基本とし，各コンテントのリンク情報を同時に記述する．図1に示した構造の内，コンテント1～6の記述例を表1に示す．表記法はSGML表記法による．表1に示されるように，一つのコンテント情報に対する記述は<content>開始タグから</content>終了タグで囲まれ，各データは<データ名>によって表現される．ただし理解しやすいように，コンテント間に空行を入れ，インデント・空白を入れ，

さらに終了タグは一部割愛した．また図中の(a)～(f)記号は4.1の論理構造データ編集で説明するための表記であり実際には記述されない．

表 1: 論理構造データ記述例

```
<logical_structure>
<content>
  <content_id>              01
  <content_layer_level>     01
  <parallel_content_id>     02          ..... (a)
  <detail_content_id>       03
  <next_content_id>         14
</content>
<content>
  <content_id>              02
  <content_layer_level>     01          ..... (b)
</content>
<content>
  <content_id>              03
  <content_layer_level>     02
  <parallel_content_id>     04          ..... (c)
  <detail_content_id>       05
  <next_content_id>         06
</content>
<content>
  <content_id>              04
  <content_layer_level>     02          ..... (d)
</content>
<content>
  <content_id>              05
  <content_layer_level>     03          ..... (e)
</content>
<content>
</logical_structure>
```

4. MPP 再編集処理

MPP再編集処理は，以下の2つの過程から構成される．

(1) 論理構造データの編集処理

(2) MPPビットストリーム変換処理

(1)については以下4.1において，(2)に関しては4.2において述べる．

4.1 論理構造データ編集処理

本節では，次節4.2で述べるMPPビットストリーム変換で利用するための，論理構造データの編集処理手法について述べる．以下4.1.1～4.1.4では，一つのMPPを対象としたコンテント情報の削除，追加・挿入，置換，抽出などのMPP編集処理を実現するための論理構造データ編集手法について述べる．さらに4.1.5では4.1.1～4.1.4を踏まえながら，複数のMPPからのMPP作成を実現するための，論理構造データ編集について述べる．

4.1.1 削除

基幹コンテントの削除処理においては，削除対象コンテントの< content> ～ </content> 部を削除する．その際，対象コンテントへの順序リンク，詳細リンクが存在する場合には，それぞれリンク元の <next_content_id>， <detail_content_id> から対象コンテント識別子を削除し，新たなリンク先の識別子へ修正する．さらに対象リンク中に詳細リンクが存在する場合には，対象コンテント階層以下のコンテントは削除するものとする．一方，付随コンテントの削除処理においては，削除対象コンテントの <content> ～ </content> を削除し，付随リンク元の <parallel_content_id> を削除する．

以下にコンテント3を削除する際の編集例を以下に示す．

(1) コンテント6の <content> ～ </content> 部(表1中(c)部)を削除する．

(2) コンテント4，5の <content> ～ </content> 部(表1中(d)，(e)部)を削除する．

(3) コンテント1の <detail_content_id> を06に変更する．

4.1.2 挿入・追加

基幹コンテントを挿入・追加する場合は，削除処理と同様に，挿入・追加位置へのリンクを持つコンテントの <next_content_id>， <detail_content_id> を修正する．ただしその際，挿入するコンテントの実データファイル名を <insert_content_name> として記述し，さらに挿入コンテントの属性データファイル名を <substitution_file_name> として記述する．属性データファイル表記例を表2に示す．

以下にコンテント3，6の間にコンテント19を挿入する際の編集表記例を表3に示す．

付随コンテントを挿入・追加する場合には，付随リンク元の <parallel_content_id> を追加する．

表 2: 属性データ記述例

```
<substitution>
<content>
  <content_id>                19
  <content_title>             Japanese History
  <content_author>            Hiroyuki KASAI
  <content_date>              5/Dec./1997
  <content_version>           1.0
  <content_affiliation>       Waseda University
  <content_publisher>         -
  <content_user_extention>    -
</content>
</substitution>
```

表 3: 論理データ編集例 (挿入処理)

```
<content>
  <content_id>                03
  <content_layer_level>       02
  <parallel_content_id>       04          ..... (c')
  <detail_content_id>         05
  <next_content_id>           19
</content>
<content>
  <content_id>                19
  <content_layer_level>       02
  <insert_content_name>       filename
  <substitution_file_name>    filename
  <next_content_id>           06
</content>
```

4.1.3 置換

置換処理は，置換対象コンテントを異なるコンテントに入れ換える処理である．基本的に挿入・追加処理と同様であるが，異なる点は，<content_id>を受け継ぐことができること，変換対象コンテントへのリンク元のデータを修正する必要がないことである．ただし変換によりコンテント属性データを変換する必要がある場合には，挿入・追加処理と同様に属性データファイル名を記述する

4.1.4 内容・階層構造に基づく抽出

内容・階層構造に基づく抽出処理は，任意の階層に属するコンテントのみを抽出，あるいは任意コンテントの詳細リンクに属するコンテントのみを抽出する処理である．図2に抽出処理例を示すが，領域Aの抽出においては，上位階層コンテントのみの抽出によりパッケージ情報の要約集を作成することが可能となり，また領域Bの抽出においては，ある内容・項目に着目したパッケージを作成することが可能となる．抽出処理においては，抽出対象コンテント以外のコンテント削除と共に，コンテントの階層レベルを示す<content_layer_level>の修正を行なう．また抽出対象コンテント以外へのリンク情報が存在する場合には削除する．

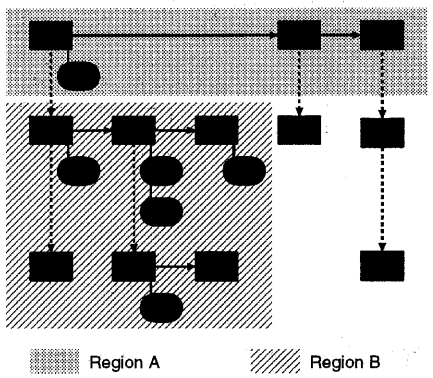

図 2: 抽出例

4.1.5 複数MPPからのMPP作成

複数のMPPから新たなMPPを作成する際は，論理構造データ中に元のMPPを示すMPPファイルネーム，あるいは識別子を記述する必要がある．さらにMPPから抽出するコンテント情報の元のcontent_idを記述する必要がある．これらはそれぞれ<mpp_file_name>，<org_content_id>として記述する．

4.2 MPPビットストリーム変換処理

本節では，編集された論理構造データを用いたMPPビットストリーム変換について述べる．まずMPPビットストリーム変換処理器の全体構成について述べ，各処理器の処理方法について述べる．

4.2.1 MPPビットストリーム変換処理器

MPPビットストリーム変換処理器の全体構成を図3に示す．変換処理器は大きく4つのブロックから構成され，それぞれパーサ，MPP復号器，MPP符号化器，MPPコントローラである．以下では，各処理器について述べる．

(1) パーサ (Parser)

編集済みの論理構造データはパーサに入力され

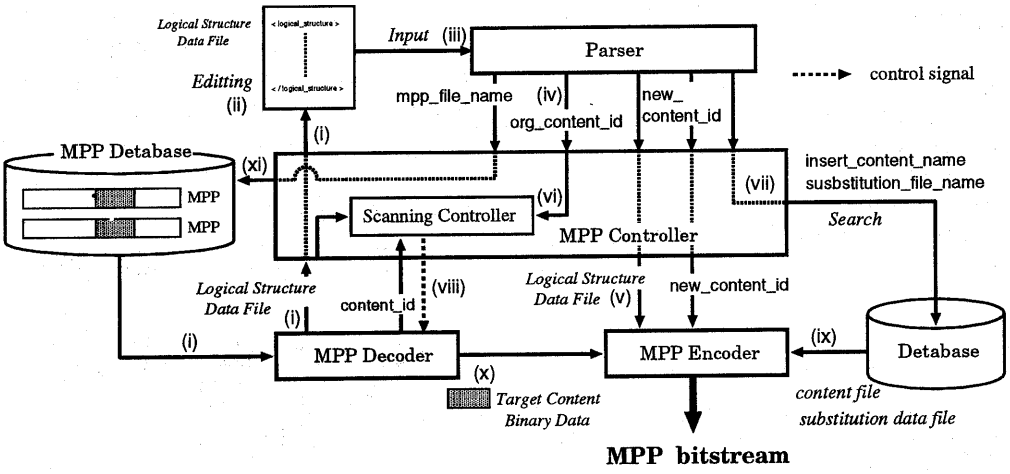

図 3: MPP ビットストリーム変換処理器の全体構成と処理フロー

解析される．全ての MPP ビットストリームは論理構造データに基づいて作成されるため，パーサからの解析データは全て MPP コントローラに入力され，全変換処理過程が制御される．

(2) MPP コントローラ (MPP Controller)
MPP コントローラは，パーサから入力された各制御データに基づいて MPP データベースからの検索処理，MPP 復号器の復号処理，MPP 符号化器の符号化処理を制御する．またビットストリーム内におけるスキャン制御を司る”スキャニングコントローラ”が存在する．

(3) MPP 復号器 (MPP Decoder)
MPP 復号器は，コントローラからの制御信号を基に，論理構造データ，コンテント識別番号の復号，コンテント実データの検索，スキャニング処理を行なう．各復号データをコントローラ，符号化器に伝達する．

(4) MPP 符号化器 (MPP Encoder)
MPP 符号化器は，MPP 復号器からのバイナリデータを新たなビットストリームに付加する．また挿入・追加処理の場合は，属性データを MPP ビットストリームシンタックスに基づいてヘッダ情報として符号化し，さらに実データも符号化し付加する．

4.2.2 MPP ビットストリーム変換処理手順

MPP ビットストリーム変換処理手順を図 3と対応付けながら説明する．

1. 変換対象 MPP(MPP_{Target}) を MPP データベースから検索し，MPP デコーダにおいて論理構造データを復号する (i).
2. 復号論理構造データを編集する (ii).
3. 編集された論理構造データは MPP パーサに入力され (iii)MPP 符号化器に伝達される (v)．ただしこの際，編集されたデータを 3.2で示したデータ記述法に準拠したものに変換する．次に論理構造データを解析し，次に符号化対象コンテント ($Content_{Target}$) の org_content_id(3.2における content_id と同義) を抽出する (vi).
4. MPP 符号化器に入力された論理構造データが符号化される (v).

5.(a) $Content_{Target}$ が MPP_{Target} に存在する場合には，org_content_id がスキャニングコントローラに入力さる (vi).

5.(b) $Content_{Target}$ が新たなコンテントである場合，insert_content_name，substitution_file_name がデータベースに伝達される (vii).

6.(a) スキャニングコントローラは，org_content_id と MPP デコーダから復号される content_id を照らし合わせ，$Content_{Target}$ のバイナリデータを検索する (viii).

6.(b) 符号化対象ファイルデータとその属性ファイルデータがMPP符号化器に伝達され符号化される (ix).

7.(a) 検索された $Content_{Target}$ のバイナリデータはMPP符号化器に伝達され符号化される (x).

上記 5. ～ 7. の処理過程をコンテントの個数分繰り返す．また複数のMPPからの変換に関しては，過程 3. において符号化対象 $Content_{Target}$ が属する mpp_file_name も同時に抽出する (xi).

4.2.3 コンテント情報符号化順序と編集履歴情報保存

コンテント情報内の各コンテントの符号化順序 (ビットストリーム位置) については，3.2で述べた論理構造データに基づいて決定する．特に縦方向から横方向の順で位置する．これは通常の再生時においてビットストリーム中のスキャンが逆方向になるのを極力避けるためである．

編集履歴情報に関しては，各コンテント情報符号化の際，コンテント情報のヘッダ情報として符号化する．この際，編集者，日時，元のMPPのファイル名，編集処理に関する情報を編集履歴情報として保存する．

4.2.4 スキャニング処理手法

MPPビットストリーム変換処理において，特に処理量が増大すると考えられるのは，前節処理過程 6.(a) である．ここでは，各MPPから復号された論理構造データに基づく適応的スキャニング手法を用いる．

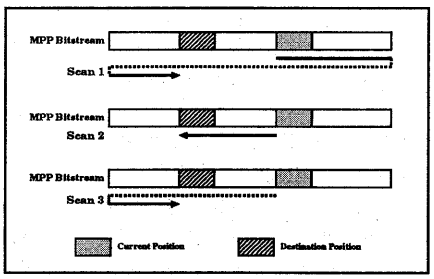

図 4: スキャニング処理手法

通常のスキャニング手法では現在位置から順方向スキャンを行ない，ビットストリーム終端に到達した場合，ストリーム先頭へ移動し，再度順方向スキャニングを行なう (図 4: Scan 1)．そこで本稿では，次の符号化対象コンテントの位置が，現在位置に対して，順方向であるか，あるいは逆方向であるかの判定を，復号論理構造データを用いることにより行なう．これにより予めスキャニング方向を決定し，処理量の増加を低減できるものと考えられる．さらに次の符号化対象コンテントが逆方向に存在すると判定された場合のスキャニング手法として，(i) 逆方向スキャニング (図 4: Scan 2) と (ii) 先頭位置へのリセットからの順方向スキャニング (図 4: Scan 3) の2手法を用いる．

5. シミュレーション実験

本稿では，MPP編集処理として 4.1 で述べたように，コンテント情報の削除，追加・挿入，置換，内容・構造に基づいた抽出処理を実装した．さらに複数のMPPからの新たなMPPの作成も実装した．以下では，特に，変換対象とするコンテント数，アクセスMPP数の変動に伴うMPPビットストリーム変換処理の処理量についてシミュレーション実験を行なう．本シミュレーション実験では，コンテント情報としてJPEG符号化情報を用いた．一つのコンテント情報は約 100[kbyte] である．

5.1 コンテント数とMPP変換処理量の関係

アクセスMPP数を1と固定し，コンテント数を変動させた時の処理時間を算出した．コンテント数は 10 ～ 200[個] である．この実験において，MPP_{Target} 内における $Content_{Target}$ の位置，つまり org_content_id を最大コンテント数内でランダムに発生させた．シミュレーション実験結果を図5に示す．同時にコンテント数の変動に伴う符号化処理時間を図6に示す．ただし，JPEG符号化時間は処理時間に含まない．

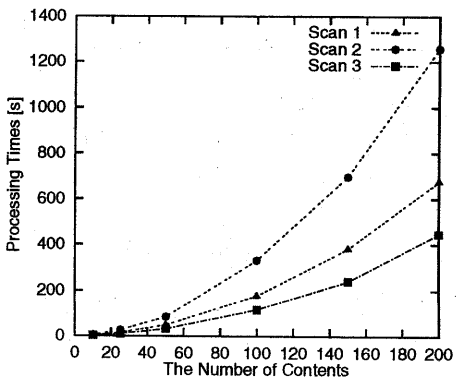

図 5: コンテント数とMPP変換処理時間の関係

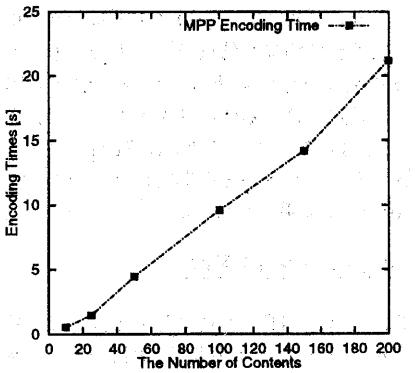

図 6: コンテント数と MPP 符号化処理時間の関係

図 6から，符号化処理時間はコンテント数にほぼ比例して増加していることが分かる．一方，図 5から，コンテント数の増加に伴い変換処理時間の増加率が大きくなっている．図 5から，変換処理時間とコンテント数の関係は定数 a を用いて，以下の式で近似されることが分かった．

$$(\text{変換処理時間}) = a \times (\text{コンテント数})^2$$

さらに各スキャニング手法の特性として，**Scan 3** が最も処理量が軽減されており，**Scan 2** は **Scan 1** よりも処理量が増大している．これは逆方向スキャニングでは，逆方向移動と復号による順方向移動の繰り返しであるためであると考えられる．

5.2 アクセス MPP 数と MPP 変換処理時間の関係

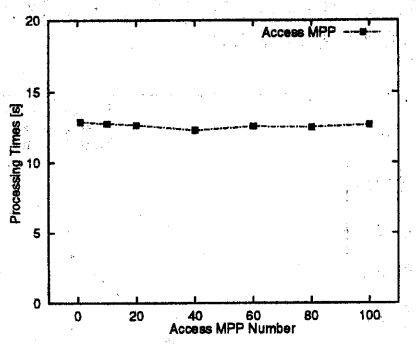

図 7: アクセス MPP 数と MPP 変換処理時間の関係

コンテント数を 100[個] と固定し，アクセス MPP 数を変動させた時の処理時間を算出した．ただし本シミュレーション実験において，コンテント情報の検索のための MPP 内でのスキャニングを避けるため，$Content_{Target}$ を MPP_{Target} 内において順番に配置した MPP を用いた．実験結果を図 7に示す．

図 7から，変換処理時間はアクセス MPP 数の増加にほぼ依存しないことが分かる．

以上の結果から，変換処理量は MPP 内におけるコンテント情報検索のためのスキャニング処理に費やされることが理解でき，より高速なスキャニング処理法が要求されることが分かる．

6. まとめ

本稿では，MPP 再編集を実現するためのデータ構造とその編集処理手法について述べた．特に論理構造データを導入し，論理構造データを編集することにより，一括して MPP ビットストリーム変換を実現する手法について述べた．最後に処理量の観点から変換処理について考察し，処理時間が MPP ビットストリーム内におけるデータ検索のためのスキャニング処理に大きく依存することを明らかにした．今後は，より適応的なスキャニング処理手法の検討とともに，ネットワークを介した MPP 制御法についても検討していく．

参考文献

(1) 児玉 明, 笠井 裕之, 村井正人, 富永 英義: “マルチメディアプロシーディングとその情報構成に関する検討”, 信学技報, IN96-122,OFS96-60, pp. 99–106 (1997).
(2) 田中 譲: “Meme Media and a Meme Pool”, PCSJ 97, pp. 1–7 (1997).
(3) 笠井 裕之, 児玉 明, 村井正人, 富永 英義: “マルチメディアプロシーディングパッケージの再編集処理に関する検討”, 信学技報, IN96-123,OFS96-61, pp. 31–37 (1997).
(4) 滝川 啓, 権田 亜紀子, 福留 治隆: “ビデオシーケンスの構造化”, 情報処理学会論文誌, Vol. 37 No.8, pp. 1592–1599 (1996).
(5) L. Hardman, G. van Rossum and D. Bulterman: “Structured Multimedia Authoring”, ACM Multimedia 93, ACM, pp. 283–289 (1993).
(6) ISO-IEC 8613-1: “Information Technology - Open Document Architecture(ODA) and Interchange Format: Introduction and General Principles (2nd edition)” (1994).
(7) ISO-IEC 8879: “Information Processing - Text and Office Systems - Standard Generalized Markup Language(SGML)” (1986).