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

2008－DD－67（5）
2008/7/25

Adobe CID字形集合のTrueType代替可能範囲および処理負荷の評価

鈴木 俊哉†

† 〒739-8511 東広島市鏡山 1-4-2 広島大学大学院総合科学研究科
E-mail: †mpsuzuki@hiroshima-u.ac.jp

あらまし 従来、PostScript や PDF などの Adobe CID 字形集合によるグリフ資源と、TrueType フォントなど符号化文字集合に基づくグリフ資源は、厳格に区別されてきた。具体的には、Adobe CID 字形集合を利用するドキュメントは、PostScript や PDF など、符号化文字列を用いずにグリフを指定できる基盤の上で処理されてきた。しかし、2008 年に Adobe-Japan1 字形集合に由来する漢字を指定するための異体字シーケンスを Unicode Consortium が登録したことにより、Adobe CID 字形番号を指示するものの実際のグリフ資源を欠いたままのドキュメントが広く流通する可能性がある。そこで、本研究は TrueType フォントによる Adobe CID 字形集合の代替処理について、グリフ資源の対応づけアルゴリズムを検討する。また、提案するアルゴリズムを、PostScript ユーザ空間で実装した場合の処理負荷について報告する。

キーワード 漢字、PDL, PostScript, CID, 異体字, IVS

The substition of Adobe CID glyph collection by CJK TrueType font

the coverage of substitutible glyphs and its loading

suzuki toshiya†

† Faculty of Integrated Arts and Science, Hiroshima Univ., Kagamiyama 1-4-2, Higashi-Hiroshima-shi, 739–8511 Japan
E-mail: †mpsuzuki@hiroshima-u.ac.jp

Abstract The glyph collections in high quality printing systems had been discriminated from the coded character sets for information interchange. Because the former has been designed for specific printing systems, and the latter has been designed for the information interchange among the various systems. But for CJK typography, recently Unicode consortium decided to define a standard of Ideographic Variant Sequences which assign an unique numbers of the glyph shapes. As a result, it is expected that future electronic document can include additional informations to specify the glyph shape in detail, even for the low-end printing or display systems. In this paper, I evaluate the coverages that common CJK TrueType font can display for Adobe CID glyph collections, and the loading of the substituion by PostScript implementation.

Key words Kanji, PDL, PostScript, CID, Ideographic Variant, IVS

1. 背 景

情報交換用に設計された符号化文字集合における、いわゆる「例示字形」と、印刷・印字処理系に設計された字形集合は、厳密に区別されなければならない。前者においては、文字符号が第一義的なものであり、例示字形は符号化文字集合を為す閉集合の中の要素を特定するために示される二義的なものだからである。これに対し、後者は、特定の処理系を用いてある字形を出力することが念頭に置かれており、必ずしも様々な処理系の間で情報を交換するために設計されてはいない。このため、前者はプレインテキストのような、文字符号のみによる利用が広く行なわれているが、後者は特定の印刷用データ体系の中での字形指定データとして利用され、字形指定データだけがプレインテキストのように交換されることは稀であった。

さて、ISO/IEC 10646 は符号化文字集合であり、厳密な字形の規定を含まない。しかし、字形の指定をするための付加的な情報を符号化文字列の中に埋め込むための符号 (Variation Sequence、以下では VS と略す) がいくつか定義されている。ISO/IEC 10646 の中で字形が規定されているのは数学記号やモンゴル文字の数文字など非常に限定された字形だけであり、それ以外の文字に対して VS を使用した場合、表示系は単にこれを無視すると定められている。これに対し、ISO/IEC 10646 を利用した符号化方式の規格である Unicode では、上記の他に日中韓統合漢字に対する VS(Ideographic Variant Sequences、以下では IVS と略す) を登録制によって管理し、さらに未定義の VS の使用を禁止している。この IVS にどのようなものが登録されるか注目されていたが、2007 年に最初の申請として Adobe による Adobe-Japan1 字形集合が登録された。Adobe-Japan1 字形集合 [1] は本来は PostScript, PDF の中で用いられる字形集合であったが、その規格書に整理されている字形をそのまま参照字形とし、それぞれの字形番号 (CID) に対する IVS が Unicode に登録されたことにより、理論上はプレインテキストの中でも

Adobe-Japan1 字形集合が利用できることになった。

これまで、Adobe CID 字形番号を直接に指定して文字符号よりもさらに詳細な字形を指定できるものは、商品位な印刷用データを生成するアプリケーションに限られていた。たとえば、事務文書を作成するほとんどのソフトウェアでは JIS90 字形と JIS2004 字形を切り替える UI を持たない。しかし、今後は安価なアプリケーションからでも間接的に Adobe CID を指定することができ、また、Adobe CID 字形番号を指定できないアプリケーションからの出力にも、他のアプリケーションによって挿入された IVS が透過的に含まれることで、表示の際に Adobe CID による字形管理が必要となる可能性がある。

ISO/IEC 10646 でも Unicode でも、処理系は VS を単に無視することが許されるが、無視した場合には何のスペースも配置してはならない (いわゆるゲタなどで代替表示してはならない) 点が、他の文字符号とは異なる。このため、印刷用データにおいては、これまでの各処理系の字形集合を用いた字形指定から、VS による字形指定へ移行する利点は、情報交換用文字符号における VS に比べると小さいと予想される。

現在一般に流通している TrueType フォントは ISO/IEC 10646 や JIS X 0208, JIS X 0213 といった情報交換用符号化文字集合には準拠しているものの、Adobe CID 字形番号を念頭に置いて設計されているわけではないので、Adobe CID による字形指定を反映するには、いわゆる CID-keyed フォントや、CFF OpenType といった特殊なフォントを用いなければならない。しかし、これらのフォントは TrueType フォントとは異なる字形記述方式をとるため、広く普及するに至っていない。プリンタに組み込むフォント資源が限られているため、印刷用 PDL の IVS サポートもあまり活発ではなく、しばらくの間はホスト内での IVS サポートが先行し、PS や PDF などの IVS に依らずに Adobe CID 字形空間を直接指定するような従来の PDL での印字処理が続くと予想される。

そこで、本研究では一般に流通している CJK TrueType フォントによって Adobe CID 字形集合を表示することを考え、代替表示が可能な字形集合の範囲と、印字処理における代替処理のオーバヘッドを評価する。

2. Adobe-Japan1 IVS の概略と規格化の動向

本節では、Adobe-Japan1 IVS の基礎となっている Adobe-Japan1 字形集合と、それに基づく VS 群である Adobe-Japan1 IVS について簡単に整理する。

2.1 Adobe CID 字形集合

Adobe CID 字形集合は、PostScript および PDF で用いるために策定された字形集合である [2]。各字形には 16 ビットの CID 番号がわりあてられている。伝統的な利用方法としては、各処理系の文字符号位置と CID 番号の対応関係を定義する符号表 (CMap) と組み合わせて用いていたが、文字符号と独立に CID 番号を直接指定して字形を特定することも可能である。この方式は、日本の漢字処理系のように、情報交換用文字集合における字形規定が緩やかで、様々な実装字形がありえるが、実際の利用においては細かな字形の差異が問題視される状況を考慮し、情報交換用文字集合では包摂して区別しないような差異をも区別するために導入された。当初、Adobe CID はほとんどの漢字が重複しているが符号位置が異なる文字符号をできるだけ小さいフォント資源で提供するための方式として導入された。ISO-2022-JP, Shift-JIS, EUC-JP など大半の符号位置を数学的演算で関係づけられる場合には問題ないが、ここに Unicode などを含めようとした場合、資源の流用が難しい。そこで、Adobe CID 字形集合を利用するために、文字符号に対し Adobe CID 字形番号をわりあてる対応表として CMap テーブルが定められ、これによって特定の文字符号で記述されたテキストを適切な字形で表示することが可能となる。その後、PDF に Adobe CID 字形集合が採用されるにあたり、文書の検索やプレインテキスト抽出などを可能とするため、CID で指定される字形に文字符号の属性をわりあてることが期待され、CMap とは別に ToUnicode テーブルとして配布されている。これらの関係を図 1 に示した。

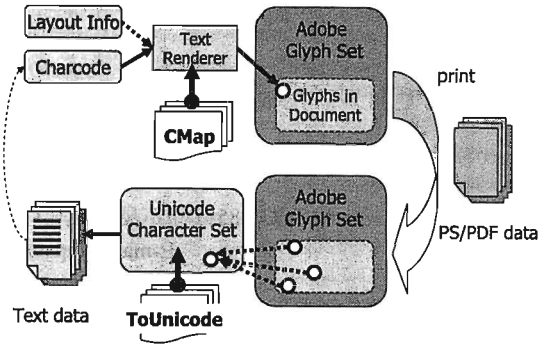


図 1　CMap と ToUnicode テーブルの役割

CMap テーブルは特定のビットパターンに対して別のビットパターンをわりあてる対応表で、一般には文字符号ビットパターンに対して 16 ビットの CID 番号を対応づけている。基本的には最短一致のエントリが利用されるので、プラフミ系文字のような符号化順序と印字順序を交換するような処理は実現できず、Adobe CID が定義されているのは符号化方式が比較的簡単であるが多数の字形が必要な漢字類に限られている。

Adobe-Japan1 に準拠する字形をデザインする際に、規格書印字に用いられた小塚明朝字形を参照するべきなのか、原規格となった JIS X 0208-1990 や JIS X 0213:2004 などを参照するべきなのか、Adobe-Japan1 規格書が原規格との対応づけを含まないためはっきりしなかったが、Adobe-Japan1 IVS 登録以後に更新された最新の Adobe-Japan1 規格書には原規格との対応表が含まれ、原規格を参照するべきことが明示された。

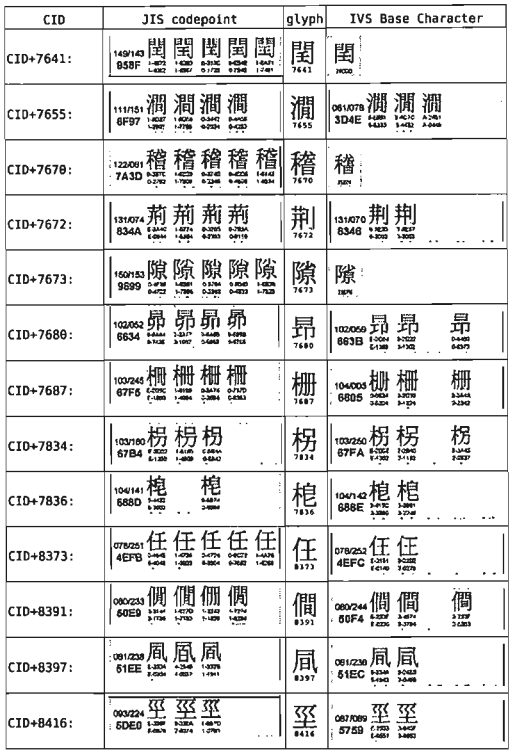
2.2 Adobe-Japan1 IVD および IVS

IVS は、ISO/IEC 10646 が定義する日中韓統合漢字に付加する VS を Unicode の枠内では登録制で管理するものである。従って、Adobe-Japan1 字形集合のうち、IVS によって指定できるのは漢字のみである。Unicode 規格に IVS を申請する際、字形を示す必要がありこれは IVD(Ideographic Variant Database) に別途登録管理されている(注*)。ISO/IEC 10646 における VS は、符号化文字列の処理中の削除は許されないが、表示においては VS を削除した場合の表示と同じであっても構わない。

JIS 漢字に対するベンダ外字の一部は、日中韓統合漢字ではなく日中韓互換漢字として ISO/IEC 10646 で採録されているが、IVS は日中韓統合漢字に対するもので、日中韓互換漢字には割り当てることができない。これは、統合漢字は様々な字形差を統合した抽象的な字体に対して符号を割り当てているが、互換漢字はそのような統合を行なっていない位置付けのためと説明されている(注†)。このため、Adobe-Japan1 字形集合の漢字のうち、ベンダ外字のために導入した字形には IVS を割り当てないという選択肢もありうるが、最終的には Adobe-Japan1 の全ての漢字に対して IVS が申請された。従って、現在では Adobe

(注*) 1 つの IVD インスタンスを指定する IVS が複数あっても構わない。

(注†) 互換漢字は本来 KS C 5601 の同一漢字が発音の差異のために複数の符号位置にあるものや CNS 11643 に規格策定のミスにより複数の符号位置に同じ漢字が符号化されているなど、字形の差異がない重複に対する対応として導入された。



CID	JIS codepoint	glyph	IVS Base Character
CID+7641:	149/143 858F	7641	
CID+7655:	111/151 6F97	7655	061/078 3D4E
CID+7670:	122/091 7A3D	7670	
CID+7672:	131/074 834A	7672	131/070 8346
CID+7673:	150/153 9699	7673	
CID+7680:	102/052 6634	7680	102/050 663B
CID+7687:	103/245 67F5	7687	104/003 6805
CID+7834:	103/180 67B4	7834	103/250 67FA
CID+7836:	104/141 688D	7836	104/142 688E
CID+8373:	078/251 4EFB	8373	078/252 4EFC
CID+8391:	080/233 50E9	8391	080/244 50F4
CID+8397:	081/238 51EE	8397	081/236 51EC
CID+8416:	093/224 5DE0	8416	087/089 5759

表 1　JIS X 0208-1990 の異体字として登録されているが、IVS の基底符号位置がどの JIS 漢字とも対応しないもの (一部)。

CID で指定される字形の指定方法として、従来のように互換漢字の符号位置を用いる方法と統合漢字の符号位置に IVS を加えて指定する方法の二つが存在する [3]。

また、ISO/IEC 10646 における統合漢字の字形については統合規準が規定されているが、VS は本来は漢字に対して規定されたものではないため、VS による字形の分散はどの程度許されるのか、ISO/IEC 10646 は全く規定していない。特に、日中韓統合漢字および拡張 A, B においては、既存の漢字符号との互換性を重視し原規格分離のための統合例外が存在するため、統合例外がある漢字は統合範囲がはっきりせず、字形の分散の議論が難しい(注*)。中国や台湾の漢字符号では区別されているものが JIS X 0208 では同一視される場合、JIS 漢字の符号位置は JIS X 0208-1990 の規格票字形に近いものに対応づけられる。ここで対応づけられなかった字形について、JIS 漢字の観点では同一の符号位置の異字形と見るが、日中韓統合漢字の原規格を特に尊重せずに最も近い字形の符号位置に対応づけ、JIS 漢字で表現できるかどうかを重視しないという立場もありうる。そのような対応づけの例を表 1 に示す。

さて、VS は通常の文字符号と異なり、指示される字形を揃えていなくても代替表示が可能となるように設計されている。Adobe-Japan1 IVS の表示系のレベルには以下のようなものが考えられる。

（1） IVS に対応しており、Adobe-Japan1 IVD と日中韓統合漢字の全ての符号位置を表示できる。

（2） IVS に対応しており、Adobe-Japan1 IVD の全ての字形を表示できる。日中韓統合漢字のうち、Adobe-Japan1 IVS で用いる部分集合のみを表示できる。

（3） IVS に対応していない、または、字形集合を完全には揃えていないが、日中韓統合漢字 (拡張領域を含む) の全ての符号位置を表示できる。

（4） IVS に対応していない、または、字形集合を完全には揃えていないが、JIS 漢字の全ての符号位置を表示することができる。

Adobe-Japan1 IVS が割り当てられる符号位置は、これらのうち、どのレベルでの代替表示を重視するかによる。正式に規定された Adobe-Japan1 IVS では、このうち 2 番目のものが重視されている。ただし、安岡らが指摘するように [4]、Adobe-Japan1 IVD にもっとも近い ISO/IEC 10646 例示字形の符号位置が選択されているわけではない。

当初の申請案では、字形差を無視した意味的な異体字を同一の符号位置にわりあてたものも散見されたが、最終的に決定した Adobe-Japan1 IVS では、以下のような原則に従っている。

- ISO/IEC 10646 における日中韓統合漢字の統合規準を越えるような字形群を同一の符号位置に割り当てることはしない。ある符号位置にわりあてた字形群は統合規準の下では区別できないものとする。
- 原規格分離による字形統合範囲の変化は考慮しない。したがって、統合規準の範囲内であれば、複数の符号位置が考えられる場合、類似性が低い方にわりあてられる可能性もある。
- ISO/IEC 10646 で符号化済みのどの字形とも統合できないと思われる字形については、新たに ISO/IEC 10646 へ申請する。

2.3 OpenType における VS 対応

TrueType フォントについては、先頃公開された OpenType 規格 1.5 で cmap テーブル (TrueType フォントにおける符号位置-グリフ番号の対応表) が拡張され、VS に正式に対応した。その概略を図 2 に示す。規格書では ISO/IEC 10646 の VS ではなく、Unicode の VS が用いられることが定められている。OpenType における字形切り換えは cmap テーブルによって得たグリフ番号に対する置換として実装されているが、UVS への対応は cmap テーブルのみで実装されているので、いわゆる OpenType レイアウト機能への指定を与えなくとも利用できることが特徴である。

この拡張では、ある符号位置に対して

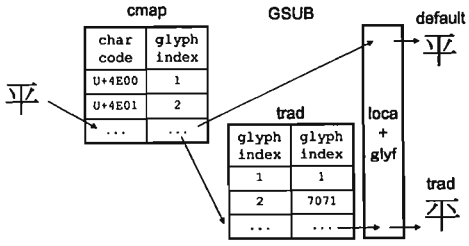
- 既定字形 (VS が付加されていない場合、または、フォントが対応していない VS が付加された場合に代替として表示するための字形)
- 非既定字形群 (特定の VS に対して用いる字形の集合)

の 2 種類の対応表が格納されている。既定字形は必ずしも非既定字形の 1 つである必要はない。また、現状の IVS 規格は完全に同一の字形に対して複数の VS を登録することを許しており、この拡張でも非既定字形群の差異の大小を評価しない。そのため、ある非既定字形がフォントに含まれない場合、仮にその字形に近い非既定字形があっても既定字形で代替表示される。

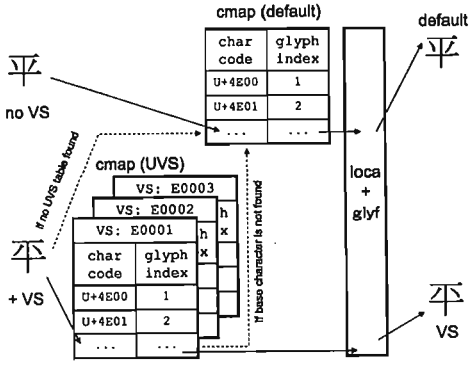
3. Adobe CID 字形空間の CJK TrueType フォントによる代替表示

CFF OpenType 形式で Adobe-Japan1 字形集合を既に揃えている場合には IVS と CID の対応表を作成して cmap テーブルに追加するだけで良いが、CJK TrueType フォントは OpenType 拡張を含めても字形集合が少なく、また、符号位置も JIS 漢字といわゆる Microsoft CodePage 932 しか考慮されていない場合が多い。これらのフォントでは cmap を追加しただけでは Adobe-Japan1 IVS の基底符号位置を全て表示できない可能性がある。

(注*)統合規準は、日中韓統合漢字の策定後に統合規準が事後的に文書化されたため、実際には適用された例のない統合規則や、統合規則では分離する筈のものが統合されている例などが見つかっている。



(a) Conventional glyph substitution in OpenType.



(b) UVS cmap since OpenType version 1.5.

図 2 従来の OpenType による異体字指定方法 (a) と、UVS cmap による異体字指定方法 (b)。

PostScript および PDF は従来の PostScript フォントや CFF OpenType 形式に相当する CIDFontType0 形式の他に、従来の TrueType フォントに相当する CIDFontType2 形式が利用可能である [5]。CIDFontType2 形式のフォントは、一般に Adobe CID 空間を無視して設計されている TrueType フォントを描画形式の再変換をせずにプリンタに送信するか、あるいは文書中に埋め込むためのもので、その CID 字形番号は Adobe CID とは互換性がない (一般には TrueType フォントのグリフ番号がそのまま使用される)。しかし、CIDFontType2 オブジェクトはオブジェクト外に公開する CID 番号と内部の TrueType フォントのグリフ番号の対応づけを定義する CIDMap テーブルを持つことができるので、この対応表を適切に調整することで、Adobe CID と互換性のある CID を持った CIDFontType2 オブジェクトを生成することができる。

著者らは、Adobe-Japan1 IVS の登録以前に、Adobe が公開している字形・文字符号関連のデータベースにより Adobe 外の規格解釈を含めずに日中台韓の Adobe CID 字形空間を埋めるアルゴリズムについて検討を行ない、また、オープンソースソフトウェアで用いられている CJK TrueType フォントを用いてアルゴリズムの評価を行なった [6]。Adobe の提供する CMap テーブルは、文字符号位置から Adobe CID 字形番号への対応表であり、IVS の枠組では既定字形群に相当する。同様に、ToUnicode テーブルは、IVS の枠組では非既定字形群に相当する (ただし、ToUnicode テーブルは IVS 規格化以前に設計されたものであり、統合規準を越えた関係づけも含まれる)。CMap のみを使用して対応づけを行なう手法 (以下、Reversal CMap と呼ぶ) では、表 2 に示すように多数の字形が表示できないことがわかった。

これを解消するために非既定字形群の対応表に相当する ToUnicode テーブルを利用すると、表 3 のように表示できる Adobe CID 空間を拡大することができる。

オープンソースの PostScript, PDF 処理系である Ghostscript は、当初 CMap テーブルと ToUnicode テーブルを併用していた (ghostscript-6.53 および 7.07) が、ghostscript-8.00 以降は Reversal CMap アルゴリズムに単純化されたため、大量のグリフ欠けが発生することになった。OpenPrinting Asia Forum ではこの問題に対応するため、上記の提案アルゴリズムを汎用的な PostScript 処理系で実行できるようユーザ空間の PostScript 言語で実装した。この実装は Ghostscript の派生ソフトウェアである ESP Ghostscript-8.15.4 に採用された。

ToUnicode table	CMap table	$N_{\mathrm{CID}}^{\mathrm{CID}}$	$N_{\mathrm{CID}}^{\mathrm{CID}} - N_{\mathrm{UCS}}^{\mathrm{CID}}$
Adobe-CNS1-UCS2	UniCNS-UCS2-H	18698	425
Adobe-CNS1-UCS2	UniCNS-UTF16-H	18698	271
Adobe-GB1-UCS2	UniGB-UCS2-H	29143	518
Adobe-GB1-UCS2	UniGB-UTF16-H	29143	485
Adobe-Japan1-UCS2	UniJIS-UCS2-H	23057	13567
Adobe-Japan1-UCS2	UniJIS-UTF16-H	23057	7676
Adobe-Korea1-UCS2	UniKS-UCS2-H	18075	1019
Adobe-Korea1-UCS2	UniKS-UTF16-H	18075	1016

表 2 ToUnicode テーブルにより、PostScript および PDF 中での利用が予想されている CID 数 ($N_{\mathrm{CID}}^{\mathrm{CID}}$) と、CMap テーブルによって文字符号位置と対応づけられる CID 数 ($N_{\mathrm{UCS}}^{\mathrm{CID}}$) の対比。2 つの CID 数の差が、PostScript および PDF 中での利用の可能性があるが、Reversal CMap アルゴリズムではグリフを対応づけられない CID 数である。

Adobe CID	TrueType font	Reversal CMap	Proposed Map
Adobe-CNS1	bkai00mp.ttf	13683	13888
Adobe-GB1	dwfszk.ttf	28611	28912
Adobe-Japan1	ipam.ttf	7653	11510
	sazanami-mincho.ttf	9238	13855
Adobe-Korea1	batang.ttf	16991	17303
	hline.ttf	16991	17185

表 3 各 TrueType フォントに対し、Reversal CMap アルゴリズムと提案アルゴリズムを適用して Adobe CID 空間を表示した場合の CID 数。

3.1 代替表示の処理負荷

ghostscript-8.00 以降に CIDFontType2 オブジェクトの生成アルゴリズムが単純化された理由の一つに、メモリ消費量および処理負荷の軽減がある。そこで、本実装がどの程度の影響を与えるかを Reversal CMap アルゴリズムに基く実装である ESP Ghostscript-8.15.1 と、提案アルゴリズムのユーザ空間実装である ESP Ghostscript-8.15.4 での処理負荷を比較する。また、提案アルゴリズムを PostScript システム空間で実装した ghostscript-7.07 とも比較を行なった。評価は汎用的な GNU/Linux 環境として CPU Pentium-M 1.2GHz、主記憶 512MB 上の Turbo Linux 11D を用いた。

3.1.1 CIDFontType2 オブジェクトの生成における負荷

CJK TrueType フォントを読み込み Adobe CID 字形番号によってアクセスできるフォントオブジェクトを生成する際、Reversal CMap アルゴリズムは CMap テーブルのみをもとに Unicode 符号位置を Adobe CID 字形番号に対応づける。これに対し、提案手法では CMap テーブルと ToUnicode テーブルを読み込み、CMap テーブルでは対応づけられなかった Adobe CID 字形番号について、ToUnicode テーブルによって Adobe CID 字形番号に対応づける。従って、ToUnicode テーブルを読み込む分のメモリ消費量が増加し、さらに Reversal CMap アルゴリズムによって対応づけられなかった Adobe CID 字形番号に対して表引きの処理が加わる。これによって処理時間の増大も予想される。評価に用いたフォントと、各アルゴリズムでの VM 消費量を表 4 に示した。

Adobe-GB1 や Adobe-Korea1 のフォントを見ると、CIDFontType2 オブジェクトを生成した際に、グリフ数は同じであるにも関わらず VM 消費量は大きく異なっている。このことから、10MB 以上の VM 消費は符号位置-Adobe CID 字形番号対応表の

font	N_{glyph}	espgs-8.15.1	espgs-8.15.4	gs-7.07
bkai00mp.ttf	14148	12693059	14054908	14082522
bsmi00lp.ttf	14148	15386337	16706694	16798244
dwfszk.ttf	30396	19345533	21314235	20697014
dwhtzk.ttf	30396	13311409	14761071	14698014
dwktzk.ttf	30396	21207045	22500507	22560794
dwstzk.ttf	30396	14183395	15599227	15538968
ipagp.ttf	9217	2521381	4259474	3592926
ipagui.ttf	9217	2521416	4259513	3593029
ipag.ttf	9217	2521122	4259211	3593391
ipamp.ttf	8628	3395753	5128948	4451300
ipam.ttf	8628	3395748	5128899	4454671
sazanami-gothic.ttf	13584	4474236	6201177	5520083
sazanami-mincho.ttf	14679	7892392	9579423	8924411
batang.ttf	17570	16949389	18323763	17472272
dotum.ttf	17570	3800224	5406892	4850569
gulim.ttf	17570	12269248	13756963	13318945
hline.ttf	17573	1202570	2372662	2241419

表 4　実験に用いた CJK TrueType フォントと CIDFontType2 オブジェクトを生成する際に消費するメモリ量

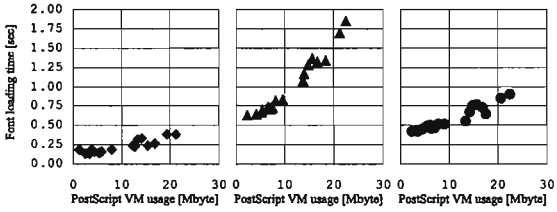


図 3　PostScript 処理系のメモリ消費量と CIDFontType2 オブジェクト生成時間の関係。左から ESP Ghostscript-8.15.1(Reversal CMap)、ESP Ghostscript-8.15.4(提案手法)、Ghostscript-7.07(提案手法の PostScript システム空間実装)。

作成ではなく、glyf テーブルに格納されている描画データの大きさに由来すると考えられる。CMap テーブルの他に ToUnicode テーブルを利用したことによる VM 消費量の増加は 1~3MB と見つもられる。これは Adobe-CNS1, Adobe-GB1, Adobe-Korea1 のような CJK TrueType フォント自体の収録字数が多く、マッピングアルゴリズムによらずメモリを大量に消費している場合には影響が少ないが、Adobe-Japan1 を日本の TrueType フォントから生成する場合などでは TrueType フォントの収録字数が少ない場合には相対的に影響が大きくなると予想される。

次に、この VM 消費の増大が CIDFontType2 のオブジェクト生成所用時間に対する影響を見る。マッピングアルゴリズムを適用する前に TrueType フォントのバイナリデータを PostScript 空間に読み込む処理に時間がかかっているとすれば、VM 消費量の大きいフォントではマッピングアルゴリズムの差異による影響は小さくなるため、小さいフォントで差異が顕著になると考えられる。この場合は、VM 消費量が比較的小さい TrueType フォントを用いる Adobe-Japan1 が不利となる筈である。しかし、巨大なフォントでは収録字数が多いために、Adobe CID 字形番号-文字符号位置対応の表を作成するための表引きの時間が大きく影響するとすれば、収録字数の多い TrueType フォントを用いる Adobe-CNS1, Adobe-GB1, Adobe-Korea1 が不利となる。表 4 に示したフォントについて、VM 消費量と CIDFontType2 オブジェクト生成に要した時間の関係を図示したものを図 3 に示す。所用時間は連続して 100 回の処理を行なった平均である。

この図から、まず、VM 消費量が 10MB を越える領域では、VM 消費量にほぼ比例して CIDFontType2 オブジェクト生成時間も増加することがわかる。次に、VM 消費量が 10MB 以上のフォントにのみ注目すると、たとえば 12~13MB を消費する bkai00mp, gulim, dwhtzk は、収録字数が 14148, 17570, 30396 字と大きく異なっているにもかかわらず、生成時間に大きな差がない。このことから、マッピングアルゴリズムの違いによって表引きの回数が変わっても CIDFontType2 生成時間への影響は比較的小さいと言える。

No.	Data size (bytes)	Pages	N_{glyph}	備考
J1	18024	1	826	文字のみのワープロ文書
J2	45252	2	669	数字のみの表計算文書
J3	36210	2	554	数字のみの表計算文書
J4	19268	1	826	図を含むワープロ文書
J5	154128	1	907	図を含むワープロ文書
J6	46564	1	1310	図、文字修飾を含む文書
J7	20296	1	71	簡単なプレゼンスライド
J9	843814	5	3570	図、文字修飾を含む文書
J10	212160	1	6754	複雑な表計算文書
J11	878025	12	975	複雑なプレゼンスライド
J12	3026015	20	15049	複雑なワープロ文書

表 5　評価に用いた JEITA のプリンタ評価用データ。J8 は文字を含まない画像データであるため除外した。

しかし、Reversal CMap による実装である espgs-8.15.1 と、著者らが提案したアルゴリズムによる実装である espgs-8.15.4 では CIDFontType2 生成時間は 3 倍以上異なる。考えられる理由として、espgs-8.15.4 は全ての処理を PostScript のユーザ空間で行なうのに対し、espgs-8.15.1 は TrueType フォントの読み込みの一部を C 言語による実装で行なっていたことの影響が考えられる。この確認のため同一アルゴリズムを PostScript システム空間で行なっていた gs-7.07 で計測した VM 消費量と CIDFontType2 生成時間の関係を図??に示す。処理時間が espgs-8.15.4 の約半分に軽減されており、数十 MB のデータを PostScript 空間に読み込む処理の実装方法によることが裏付けられる。

3.1.2　文書レンダリングにおける遅延

提案手法を全て PostScript ユーザ空間に実装した場合、CIDFontType2 オブジェクトの生成時間が 3 倍以上、実時間で最長 1.5 秒程度遅延することがわかった。日本の TrueType フォントから Adobe-Japan1 CID を生成した場合の遅延は 0.22~0.33 秒である。しかし、この遅延はモニタ表示に際する遅延とは位置付けが異なる。まず、この遅延は処理系が最初にフォントを読み込む際の遅延であり、文字ごとに必要なラスタライズの時間ではない。次に、処理を行なっている ghostscript は PostScript や PDF の処理系であり、単純なテキストを表示するだけの処理系に比べると初期化やラスタライズが重い。そこで、この遅延が実際の PostScript や PDF の文書レンダリング時間に対してどの程度であるかを調べる。本稿では、JEITA が提供するベクトルプリンタの試験用のデータ 11 種を利用した。所用時間は 100 回の処理を行なった平均である。出力は BMP 形式のラスタデータであるため、高解像度では純粋な処理時間の他にディスクアクセスの所用時間が含まれる。

まず、Reversal CMap による実装である espgs-8.15.1 でのラスタ処理時間を図 4 に示す。

ワープロ文書における文字は 1 ページあたり 900 文字程度で、このような文書を 1 ページ処理する速度はモニタ表示解像度 (75dpi) では 0.38~0.60 秒、プリンタ解像度 (600dpi) では 0.46~1.08 秒であった。従って、文書 1 枚の印刷ではフォント読み込み時の遅延が処理時間と同程度か、あるいはそれ以上になる。しかし、細かい数表を含む表計算 (J10) や、10 ページ以上の文書 (J11, J12) ではモニタ表示の時間が数秒であるため、相対的に遅延の影響は少なくなると予想される。提案手法による実装である espgs-8.15.4 での計測結果を図 5 に示す。各データ処理時間の解像度依存性は espgs-8.15.1 とほぼ同じであるが、短い文書であっても処理時間は全て 1 秒以上かかっている。

また、提案手法を PostScript システム空間で実装していた gs-7.07 での計測結果を図 6 に示す。この場合も、CIDFontType2

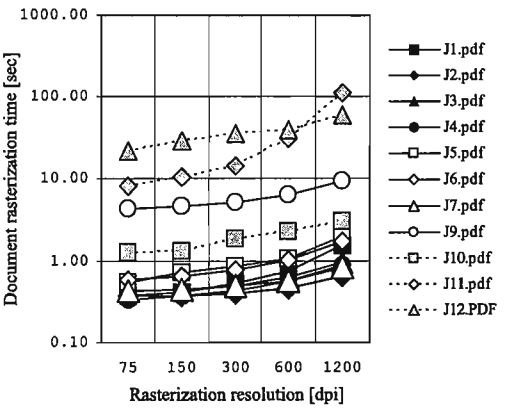


図 4　ESP Ghostscript-8.15.1 での文書ラスタライズ時間。

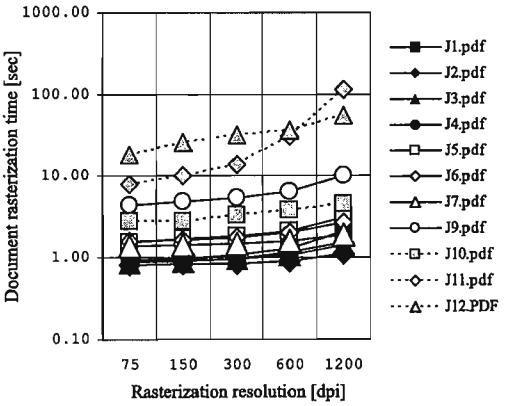


図 5　ESP Ghostscript-8.15.4 での文書ラスタライズ時間。

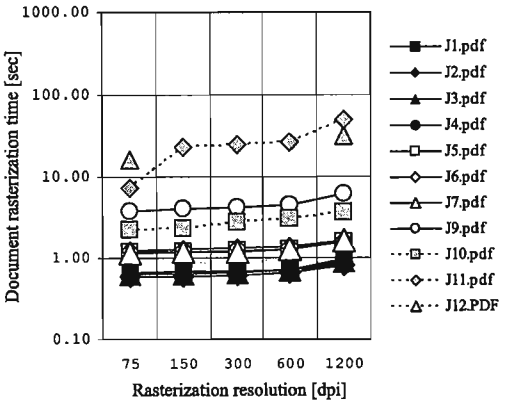


図 6　Ghostscript-7.07 での文書ラスタライズ時間。

オブジェクト生成時間と同様に espgs-8.15.1 より若干遅延しているが、PostScript システム空間で処理しているために espgs-8.15.4 よりは若干高速である。J11 の処理時間について、解像度依存性が若干異なるが、gs-7.0 系列と gs-8.0 系列は TrueType ラスタライザおよびグリフキャッシュが変更されているためと考えられる。

4. まとめ・今後の課題

本稿では、Adobe-Japan1 IVS の普及による Adobe CID 字形空間の利用の拡大を念頭におき、一般的な CJK TrueType フォントによって Adobe CID 字形空間を代替表示するためのアルゴリズムと、提案するアルゴリズムを PostScript ユーザ空間で実装した場合の処理負荷の評価を行なった。その結果、以下のことが明らかとなった。

- 提案アルゴリズムは、複数の表を用いるため CIDFontType2 オブジェクト生成の際に従来手法に比べて PostScript 空間で 1~3MB のメモリを余分に消費する。JIS X 0208 や CodePage 932 のみを収録した CJK TrueType フォントではこの増加によりメモリ消費量が倍加するが、Big5 や GB-2312 準拠のフォントのように収録字数が多いフォントや、ハングルのようにグリフあたりの描画データが増えるフォントでは、メモリ消費量が数十 MB になるため、影響は相対的に小さくなる。
- 提案アルゴリズムで CIDFontType2 オブジェクトを生成する際、符号対応を調べるための表引きの回数が数倍に増加するが、これによる処理時間への影響は小さい。
- 提案アルゴリズムを PostScript ユーザ空間で実装した場合、TrueType フォントを PostScript 言語で処理するための処理時間がもっとも遅延に影響している。
- 処理の遅延は最初にフォントを読み込む際にしか発生しないが、簡単な事務文書を低解像度で 1 ページのみ表示する場合などには無視できない遅延である。

今回は印字処理における提案アルゴリズムの適用を目指し、PostScript ユーザ空間で全ての処理を行なったが、モニタ表示を念頭に置いた場合、PostScript 言語による実装はそれ自体の処理の重さがあり、最適ではない。提案アルゴリズムの実装として、より汎用的なものが期待される。また、最新の Adobe-Japan1 規格書の中で、いくつかの OpenType 字形置換機能と Adobe CID の対応づけが明記された。ESP Ghostscript-8.15.4 では縦書き字形を抽出するための OpenType 字形置換機能を使っているのみだが、jp78, jp83, jp04 等の置換機能の利用も検討すべきである。

文　献

[1] Adobe Systems Inc.: "Adobe Technote 5078: Adobe-Japan1-6 Character Collection for CID-Keyed Fonts", Adobe Systems Inc., San Jose (2008).
http://partners.adobe.com/public/developer/en/font/5078.Adobe-Japan1-6.pdf.

[2] Adobe Systems Inc.: "Adobe Technote 5094: Adobe CJKV Character Collections and CMaps for CID-Keyed Fonts", Adobe Systems Inc., San Jose (1998).
http://partners.adobe.com/public/developer/en/font/5094.CJK_CID.pdf.

[3] R. Cook and K. Lunde: "ISO/IEC JTC1/SC2/WG2/IRG IRG N1468: Recommendation For IRG To Use IVD Collections", IRG (2008).

[4] 安岡："IVD のダブリ".
http://slashdot.jp/~yasuoka/journal/426907.

[5] Adobe Systems Inc.: "Adobe Technote 5012: The Type42 Font Format Specification", Adobe Systems Inc., San Jose (1998).
http://partners.adobe.com/public/developer/en/font/5012.Type42_Spec.pdf.

[6] suzuki toshiya, T. Yamada, M. Yamato and H. Suzuki: "Emulation of adobe cid resources by cjk truetype fonts", Document Numerique, **9**, 3-4, pp. 17–43 (2006).