TeXConf 2017（2017 年 10 月 14 日）

Extract PDFmark による PDF ファイルサイズ削減

細田 真道
http://www.trueroad.jp

概要

TEX などで同じフォントが埋め込まれた多数の図 PDF を貼り込むと、結果として生成される PDF には同じフォントが重複して埋め込まれ、ファイルサイズ増加につながる。Ghostscript で処理することにより埋め込み制御もできるが、PDF の開き方やリモートからのリンクが失われる。本稿では Extract PDFmark によって、これらを失わずにファイルサイズの小さい PDF を得る方法を説明する。

1 はじめに

各種の TEX や LATEX などを使って PDF のドキュメントを作る場合、図としてたくさんの小さな PDF を用意して、メインの PDF に貼り付けるようなことが行われる。このとき、図の PDF では同じフォントを使っていることが多いと思われる。

楽譜作成プログラム LilyPond [2] のマニュアルは GNU 公式文書フォーマットである Texinfo [3] 形式のソースファイルを XETEX で処理し、PDF を生成している。楽譜作成プログラムという性格上マニュアルには楽譜の断片を多数含み、これらは LilyPond で PDF として生成したものを図として貼り込んでいる。もちろん、各々の楽譜の断片には同じフォントが使われていることが多い。

一般的に、図 PDF にフォントが埋め込まれていた場合、そのフォントはそのままメイン PDF に埋め込まれる。同じフォントが埋め込まれている複数の図 PDF を貼り込むと、メイン PDF にはフォントが重複して複数回埋め込まれることになりファイルサイズの増加につながる。

図 PDF 作成時にフォントの埋め込み方法を工夫し、メイン PDF を Ghostscript 処理により埋め込み制御をしてファイルサイズを削減することもできる。しかし残念ながら、この処理で Ghostscript は PDF のページモードやリンクの宛先名を残さないため、処理後の PDF は意図したとおりに開かれなくなったり、リモートからのリンクが機能しなくなったりする。

本稿では、この問題を解決する方法として Extract PDFmark [1] を紹介する。Extract PDFmark は Ghostscript 処理前の PDF からページモードやリンク宛先名を pdfmark として抽出することができる。これを Ghostscript 処理へ追加することにより、意図したとおりに開くことができ、リモートからのリンクが機能する、ファイルサイズの小さい PDF を得ることができる。こうした方法を LilyPond のマニュアル PDF [1)] 生成を交えて説明する。

2 PDF

本稿の対象となるのは、楽譜やグラフ、複雑な表など、何らかのフォントを使っている図を TEX などの原稿へ複数貼り込んで生成する PDF ドキュメントである。PDF の機能のうち本稿に関連する部分を述べる。

1) 日本語マニュアルは公式ビルド環境の都合で PDF 版が存在せず HTML 版のみ公式サイトにある。環境を用意すれば図 1 のような PDF 版を生成できる。

2.1　しおり

「しおり」または「ブックマーク」「アウトライン」などと呼ばれる機能は、章・節など文書構造をツリー状に表示し目的の部分へ簡単にジャンプできるようにしたものである（図 1 左の部分）。また、ページモードを指定すると PDF を開いたときに最初から「しおり」が表示されるよう設定することもできる。LilyPond のマニュアル PDF も、Texinfo の機能により「しおり」を作成し、ページモードで PDF を開いたときに「しおり」が表示されるようにしている。本稿 PDF も LaTeX の hyperref パッケージで「しおり」を作成し、開いたときに表示されるようにしている。

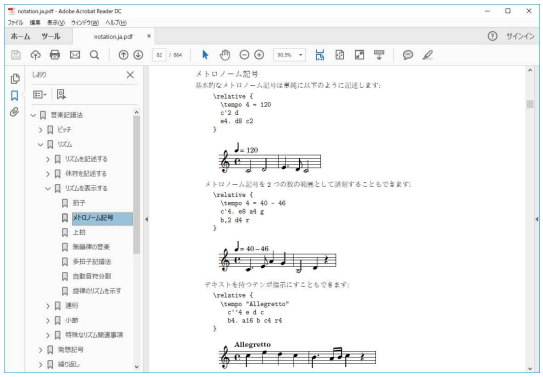

図 1　PDF の例（「しおり」が左に表示されている）

2.2　ハイパーリンク

ハイパーリンクも便利な機能である。PDF でリンクが設定されている場所をクリックすると、設定された場所へジャンプすることができる。リンク先としては URL だけでなく、同じ PDF 内のどこかや他の PDF 内のどこかも指定できる。

また、PDF 外部からリンクできるように、名前付きアンカーとして機能する named destination（本稿では宛先名と呼ぶ）を設定できる。例えば、節に宛先名を設定しておくと、PDF 相互間でリンクする場合や HTML から PDF へリンクする場合など PDF 外部（リモート）から節へジャンプするリンクを作ることができる。

LilyPond のマニュアル PDF は「学習マニュアル」や「記譜法リファレンス」など、用途ごとに複数の PDF に分かれており、Texinfo の機能でそれぞれのノード（節）に宛先名が設定され、PDF 間相互にリンクが張られている。本稿 PDF も hyperref パッケージにより URL にリンクを指定してあるほか、参考文献の引用番号や図表の参照番号から参照先へジャンプできるようにしている。また、節はもちろん、図表や参考文献などにも宛先名が設定されており、リモートからこうした場所を指定したリンクを作ることができる。

2.3　フォント

PDF にはドキュメントで使用しているフォントを埋め込むことができる。これにより、PDF 作成者が指定したフォントが存在しない環境でも、PDF 作成者の環境と同様の表示ができる。この時、特に日本語を含む CJK フォントはグリフ数が多いため、フォントを丸々フルセットで埋め込んでしまうと PDF ファイルサイズが大きくなってしまう。そこで通常は PDF 内で使用しているグリフのみを抜き出して埋め込む、サブセット埋め込みをすることが多い。

また、逆にフォントを埋め込まずファイルサイズを小さくすることもできる。この場合、PDF 作成者が指定したフォントの情報をもとに、PDF ビュアーが環境中のフォントから同一のフォント、もしくは同じような表示が望める代替フォントを探し出して使うことになる。ファイルサイズは小さくできるが、必ずしも PDF 作成者の環境と同様の表示とならないことがあるので、基本的にはフォントを埋め込む方法がとられる。

3　LilyPond

LilyPond [2] はソースファイル（テキストファイル）をコンパイルすることで、楽譜の PDF などを生成することができる楽譜作成プログラムである。ソースファイルの例を以下に示す。

```
\relative
{
  \clef treble
  \key e \major
  \time 4/4
  \tempo "Allegro"

  \partial 8 e''8 |
  gis gis gis fis16 e b'4. b16 a |
}
```

このソースから生成した楽譜を図 2 に示す。この図は実際に LilyPond で生成した PDF を LuaLaTeX の原稿に貼り付けたものである。その他に図 3, 4 にも楽譜の例を示している。これらによって、本稿の PDF そのものが複数の図を貼り付けた PDF となり、Extract PDFmark によるファイルサイズ削減のサンプルとなっている。

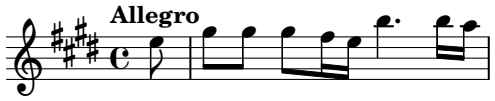

図 2　楽譜の断片（ヴィヴァルディ「春」より）

4　Texinfo

Texinfo [3] は GNU 公式文書フォーマットであり、楽譜作成プログラム LilyPond を含め GNU 関連プロジェクトのドキュメント形式として広く使われている。Texinfo 形式で書かれたファイルを 1 つ用意すれば HTML や PDF をはじめとする様々な形式のドキュメントを出力することができる。HTML などを出力する際にはスクリプトなどで変換処理が行われるが、PDF を出力する際には TeX が使われる。この際には LaTeX ではなく plain TeX が使われるが、図を張り付けた PDF を作成するワークフローは基本的に LaTeX と同様のため、本稿では基本的に Texinfo と各種 LaTeX を区別しないこととする。以下に、Texinfo 形式ファイル冒頭部分の抜粋を例として示す。

```
\input texinfo-ja.tex

@documentencoding UTF-8
@documentlanguage ja

@settitle 吾輩は猫である
@afourpaper

@titlepage

@title 吾輩は猫である
@author 夏目漱石

これは日本語Texinfoファイルのサンプルとし
...
```

1 行目で Texinfo のコマンド等を定義する plain TeX 用マクロを読み込み、2 行目から@で始まるコマンドを用いてマークアップしていくようになっている。

5　TeX で図を貼り込む

5.1　図のフォント

各種 TeX / LaTeX で図を使った（貼り付けた）PDF を作成するには、何らかの方法で作成した図のファイルを TeX のソースファイルで取り込む（LaTeX なら`\includegraphics` を使う）ことになる。この際、グラフなどの文字を含んだ図は PDF ファイルとして用意することが多くなっていると思われる。通常、特に何も考慮せずに文字を含んだ PDF を作成すると、使用しているフォントのサブセットが埋め込まれた PDF が生成される。サブセットを埋め込む方法は、様々な環境で同一の表示ができつつファイルサイズを抑えることができるため、この PDF が最終目的物であれば正しい選択だといえる。

この方法で図 2 の PDF を生成したときのフォント情報を Poppler [4] 付属の pdffonts を利用して表示したものを表 1 に示す。emb 列 sub 列ともに yes になっているので、フォントが埋め込まれておりサブセットであるということがわかる。また、name 列のフォント名にサブセットであることを示す接頭辞が付いている。

```
name                                type              encoding          emb sub uni object ID
----------------------------------- ----------------- ----------------- --- --- --- ---------
AUVQXI+Emmentaler-20                Type 1C           Custom            yes yes no       8  0
RDUTAJ+TeXGyreSchola-Bold           Type 1C           WinAnsi           yes yes no      10  0
```

表 1　「春」PDF フォント情報（オプション無指定、サブセット埋め込み）

しかし、こうした PDF を図として TEX ソースファイルで大量に取り込んだ場合、TEX が出力するメイン PDF には、各々の図 PDF に埋め込まれていたフォントがすべてそのまま埋め込まれてしまう。複数の図で同じフォントが使われていることも多いと思われるが、重複は解消されず、そのまま同じフォントが複数回埋め込まれてしまい、ファイルサイズの増加につながってしまう。

5.2　フォント重複の解消

この問題を解消するには、図 PDF 作成時のフォント埋め込み方法を工夫し、TEX が出力した PDF を Ghostscript で処理してフォントの埋め込みを制御する方法がある。制御の方法としては大きく 2 つのアプローチがある。

5.2.1　フルセット（非サブセット）埋め込み

図 PDF に埋め込まれているフォントを、サブセット埋め込みではない、非サブセット、つまり全グリフをフルセットで埋め込むようにするアプローチである。これによって図 PDF のサイズは増え、この図を張り付けた TEX 出力 PDF も非常に大きなサイズになってしまう。しかし、この PDF では、同じフォントはすべてフルセットで埋め込まれた全く同じものになるので、後から重複しているものを取り除いて統合することができる。さらに、統合後に使われているグリフのみを含んだサブセット化をすれば、必要なフォントの必要なグリフが 1 つだけ埋め込まれている状態にできる。これにより、中間ファイルのサイズは大きくなってしまうものの、最終ファイルの大きさを減らすことができる。

このアプローチは、中間ファイルのサイズが大きくなるというデメリットがあるものの、比較的フローが単純でわかりやすい。しかし、うまくいっているように見えても問題を抱えていることがある。まず、正しいフルセット埋め込みの PDF を作成することが困難である。PDF 生成エンジンとしてもよく使われる Ghostscript は最新バージョンの 9.22rc2 でもフルセット埋め込みに問題がある。表面的にはフルセット埋め込みができたように見えるが、実際にはすべてのグリフを含んでいない「バグ」がある [5]。

さらに、フォントを統合する方法が困難である。従来、こういったフォントの統合およびサブセット化には Ghostscript が広く使われてきた。Ghostscript によるフォント統合は、バージョン 9.16 まではそのまま動作するが、バージョン 9.17 から `-dPDFDontUseFontObjectNum` オプションが必要となった。そして、このオプションはバージョン 9.21 までは動作するが、9.22rc1 で削除されてしまった [6]。gs-devel メーリングリストでの議論によると、そもそも重複するフォントを削除する動作は意図していたものではなく、この動作により文字化けを起こすことがあるので廃止した、というような説明が読める[2)]。

また、フォント統合を目的の一つとしている pdfsizeopt [7] というツールがあるが、グリフ数 256 を超えるフォントの処理に未対応である [8]。和文フォントはもちろんグリフ数 256 を超えるし、欧文フォントもギリシャ文字やキリル文字のグリフを含んでいるなどグリフ数が 256 を超えるフォントが多く、実質的に使用できない。

そして、何らかの方法でフォントの統合ができたらサブセット化する。これには Ghostscript を使うことができる。いずれにせよ、フルセット埋

2)　議論の中で、MuPDF なら重複フォント削除ができるという情報もあったが、そもそも正しいフルセット埋め込み PDF が用意できないので試すことができない。

name	type	encoding	emb	sub	uni	object ID
TeXGyreSchola-Bold	Type 1	WinAnsi	no	no	no	12 0
Emmentaler-20	Type 1	Custom	no	no	no	8 0
Emmentaler-20	Type 1	Custom	no	no	no	14 0
Emmentaler-20	Type 1	Custom	no	no	no	10 0

表 2　「春」PDF フォント情報（非埋め込み）

め込みによるアプローチは、表面的には正常に動作したように見えても、どこかで何らかの文字化けや、その他の問題が発生している可能性があるので注意しなければならない。

5.2.2　非埋め込み

図 PDF をフォント非埋め込みにするアプローチである。これにより、その図を貼り付けた TEX 出力 PDF も、図で使われているフォントについて非埋め込みとなる。もちろんこのままでは表示環境によって異なる表示となってしまう。例えば LilyPond の場合、音符もフォントで表現しており、通常の環境には音符フォントはインストールされておらず、代替できるようなフォントもないため、音符の表示ができなくなってしまう。そこで、TEX 出力 PDF へ必要なフォントを埋め込む処理を行う。このアプローチは中間ファイルのサイズが小さいままでフローを進めることができ、大量の図を貼り付けたドキュメントであってもディスク容量の消費量を抑制できる。

この方法で図 2 の PDF を生成したときのフォント情報を表 2 に示す。emb 列 sub 列ともに no になっているので、フォントが埋め込まれていないことがわかる。name 列に接頭辞もついていない。Emmentaler-20 フォント（音符フォント）が 3 回登場するが、これは後でフォントを統合するため 3 種類の異なるエンコーディングを定義して使用しているからである[3)]。

埋め込み処理には Ghostscript を使うことができるが、使用するフォントそのものを何らかの方法で Ghostscript へ渡す必要がある。基本的には、特定のディレクトリへフォントファイルを置いたり、Ghostscript の設定ファイルを編集したりする必要があるなど、非常に煩雑であり、ドキュメント生成を自動化することも困難である。また、Ghostscript は OTC フォント[4)]を直接ロードすることができないため、そのままでは OTC フォントが使えないという問題もある。

一方、Ghostscript はフォントが埋め込まれたファイルを読み込むことができるので、これを利用してフォントを渡す方法がある。フォントが埋め込まれた PostScript ファイルと同じ形式で、フォントリソースのみが書かれているファイルを一時的に用意し、PDF 処理時に PDF とともに Ghostscript へ入力ファイルとして渡す方法である。この方法であれば OTC フォントも中身の CFF を抽出して、フォントリソースとして書き出すことによって利用可能である。特定のディレクトリへファイルを配置する必要もなく、設定ファイル編集も不要なので自動化にも都合が良い。LilyPond には、このためにフォントリソースを別途出力するオプション `-dfont-export-dir` があり、マニュアル PDF はこの方法を採用し、`make doc` で自動的に生成される。また、本稿 PDF もこの方法を使っている。

いずれにしても、Ghostscript へフォントを渡さなければならないので、フルセット埋め込みのアプローチと比較すると複雑であることは否めない。また、非埋め込みのアプローチでは、図を作成したときのフォントの種類やエンコーディングなどによって、最終的に出力した PDF で文字化けやその他の問題が発生する可能性があるので、注意が必要であることには変わりはない。

3)　こうしないと楽譜の断片毎に都度生成された異なったエンコーディングが使われ、後で統合できなくなることがある。

4)　OpenType/CFF Collection フォント

5.3 Ghostscript で失われるもの

いずれのアプローチであっても Ghostscript は重要な役割を果たしている。具体的には TEX 出力 PDF を Ghostscript で処理する、というフローを経る必要がある。しかし、Ghostscript は元々の PDF に存在した情報のうち、いくつかを保持することができず失ってしまう。

一つはページモードの情報である。例えば hyperref パッケージで「しおり」を作成し、PDF を開いたときに「しおり」が開かれた状態になるよう設定したとしても、その PDF を Ghostscript で処理すると「しおり」が開かれない状態に変わってしまう[5)]。もっと深刻なのは、宛先名の情報である。どこかの PDF ドキュメントから別の PDF ドキュメントの特定の項目へジャンプするリンクを貼ったとしても、Ghostscript で処理をすると飛び先が意図した項目の場所ではなく、ドキュメントの先頭ページになってしまう。PDF 相互間のリンクでなくても、宛先名をアンカーとして指定した URL が常にドキュメントの先頭を開くようになってしまうようになる。

6 Extract PDFmark

Ghostscript 処理で失われる情報を保持するため Extract PDFmark [1] を利用できる。

6.1 インストール

Extract PDFmark は Debian 9 stretch や Ubuntu 17.04 Zesty Zapus で extractpdfmark パッケージとなっており、その他 Linux ディストリビューションでもパッケージ化されているものがある。また、Cygwin でもパッケージになっている。こうした環境ではパッケージを使って簡単にインストールできる。その他の環境ではソースからインストールする必要があるが、Autotools で作成されているので比較的簡単にできると思う。

6.2 仕組み

Extract PDFmark の仕組みを述べる。

6.2.1 pdfmark

pdfmark [9] は、PDF の機能を PostScript の中に記述するためのものである。例えば、PDF を開いた際に「しおり」が開かれるようにするページモード設定は、以下のようになる。

```
[ /PageMode /UseOutlines /DOCVIEW pdfmark
```

このような pdfmark を含んだ PostScript ファイルを Ghostscript で PDF へ変換すると、生成された PDF は開いた際に「しおり」が開かれるようになる。同様に宛先名についても pdfmark で設定できる。

6.2.2 抽出

PDF の仕様は公開されているので、TEX 出力 PDF からページモードの情報や宛先名の情報を読み取ることが可能である。Extract PDFmark は Poppler [4] ライブラリを使ってこれらを読み取って pdfmark の形式で出力するものである。コマンドライン引数に対象となる PDF ファイルを指定すると標準出力へ pdfmark を出力する。

6.2.3 Ghostscript の入力

Ghostscript は一度に複数の入力を取り扱うことができ、入力ファイルごとに形式が PostScript なのか PDF なのか個別に判断する。そこで、pdfmark のみ記述された PostScript ファイルと PDF ファイルをともに与えることで、入力になっている PDF に設定されている機能とは関係なく、もう一つの入力である pdfmark の機能が適用された PDF を出力することができる。

これを利用することにより、ページモードや宛先名が失われる Ghostscript であっても、これ

5) 「しおり」の内容は保持される。

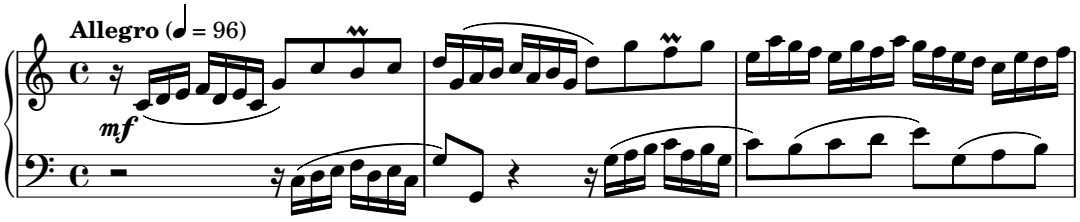

図 3　楽譜の断片（「インヴェンション第一番」より）

らを保持したまま処理をすることができるようになる。具体的には、まず、Ghostscript 処理前の TEX 出力 PDF から、Extract PDFmark を利用してページモードと宛先名を pdfmark 形式で抽出する。次に、Ghostscript 処理へ抽出した pdfmark と TEX 出力 PDF を一緒にかける。そうすると、TEX 出力 PDF で指定されていたページモードと宛先名が、そのまま保持された状態で Ghostscript 処理された PDF を得ることができる。

6.3　使い方

フォント非埋め込みのアプローチで、フォントリソースのみが含まれている PostScript ファイルを `fonts/*.font.ps` へ置いたとすると、以下のような使い方となる。

```
$ extractpdfmark TeX出力.pdf > 抽出pdfmark.ps
$ gs -q -dBATCH -dNOPAUSE -sDEVICE=pdfwrite \
    -sOutputFile=最終.pdf \
    fonts/*.font.ps \
    TeX出力.pdf 抽出pdfmark.ps
```

6.4　注意事項

Ghostscript 9.19 までは、一部の宛先名を正しく扱うことができない [10] ため、英数字以外の名前を扱うには Ghostscript 9.20 以降が必要となる。Texinfo はノード名がそのまま宛先名になるので、この制限に触れてしまう可能性が高い。

非埋め込みの図 PDF を使うアプローチで、LilyPond が生成する図 PDF に TrueType フォントを使うと文字化けを起こすことがある。欧文フォントでは WinAnsi エンコーディング範囲外のグリフ（例えばギリシャ文字など）が文字化けするし、和文フォントでは全体が文字化けする。そのため、LilyPond ではフォント非埋め込み PDF を出力するオプション `-dgs-neverembed-fonts` が指定されても、あえて TrueType フォントは埋め込むようにしている。

7　おわりに

本稿では Extract PDFmark によって、ファイルサイズの小さい PDF を得る方法を説明した。TEX などで同じフォントが埋め込まれた多数の図 PDF を貼り込むと、結果として生成される PDF には同じフォントが重複して埋め込まれ、ファイルサイズの増加につながる。Ghostscript 処理により埋め込み制御ができるが、PDF の開き方やリモートからのリンクが失われる。Extract PDFmark により、これらを失わずにファイルサイズの小さい PDF を得ることができる。

LilyPond のマニュアル PDF は、2016 年 11 月リリースのバージョン 2.19.51 までサブセット埋め込みの図 PDF を使用していたが、翌月リリースの 2.19.52 から Extract PDFmark を採用し非埋め込みの図 PDF を使用している。これにより PDF 生成に要するディスク容量は 4.6 GB から 3.4 GB へ減少、マニュアル PDF の合計サイズも 280 MB から 104 MB と半減以下にすることができた。英語版の記譜法マニュアル (notation.pdf) のサイズは 36 MB から 9.9 MB と 1/3 以下になっている [11]。

なお、各ツールのオプションや動作は本稿執筆時点のものである。特に LilyPond は最新バージョン 2.19.65 を用いたが、Ghostscript 9.22rc1 の `-dPDFDontUseFontObjectNum` 廃止に端を発する議論を受け、次期 2.20 に向けオプションや

一連の動作を見直している最中であり、今後変更される予定である。もともと、フォント重複によるファイルサイズ増を抑える方法としては、フルセット埋め込みによるアプローチの方が有名で有力だと考えていたが、本稿執筆中の 2017 年 9 月にこのような大きな動きがあり、まさにひっくり返ってしまった感が強い。一方で議論を通じて様々な知見が得られ大変勉強になったし、本稿に最新の有用な情報を盛り込むことができたと考えている。この議論は gs-devel メーリングリストや lilypond-devel メーリングリストのアーカイブで読むことができる [12][13]。

最後に、本稿関連ファイルを文献 [14] で公開している。ソースファイルや最終 PDF だけでなく、図 PDF など中間ファイルも用意しているので、参考にしていただけると幸いである。

（2017 年 9 月 30 日提出）

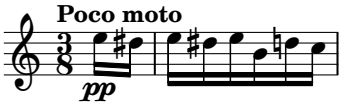

図 4　楽譜の断片（「エリーゼのために」より）

参考文献

[1] Extract PDFmark.
https://github.com/trueroad/extractpdfmark.
http://www.ctan.org/pkg/extractpdfmark.

[2] 楽譜作成プログラム LilyPond.
http://lilypond.org.

[3] GNU 公式文書フォーマット Texinfo.
https://www.gnu.org/software/texinfo/.

[4] Poppler.
https://poppler.freedesktop.org/.

[5] gs-devel メーリングリスト 2017 年 9 月より.
https://ghostscript.com/pipermail/gs-devel/2017-September/010015.html.

[6] gs-devel メーリングリスト 2017 年 9 月より.
https://ghostscript.com/pipermail/gs-devel/2017-September/009988.html.

[7] pdfsizeopt.
https://github.com/pts/pdfsizeopt.

[8] Type1CConverter and Type1CGenerator with fonts with more than 256 glyphs.
https://github.com/pts/pdfsizeopt/issues/14.

[9] pdfmark Reference.
http://kb2.adobe.com/jp/cps/511/511364.html.
http://www.adobe.com/content/dam/Adobe/en/devnet/acrobat/pdfs/pdfmark_reference.pdf.

[10] Ghostscript bug 696974.
https://bugs.ghostscript.com/show_bug.cgi?id=696974.

[11] Issue 5000: Add using Extract PDFmark for document building.
https://sourceforge.net/p/testlilyissues/issues/5000/.

[12] gs-devel メーリングリスト 2017 年 9 月アーカイブ.
https://ghostscript.com/pipermail/gs-devel/2017-September/date.html.

[13] lilypond-devel メーリングリスト 2017 年 9 月アーカイブ.
http://lists.gnu.org/archive/html/lilypond-devel/2017-09/index.html.

[14] TeXConf 2017 files.
https://github.com/trueroad/tr-TeXConf2017.