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

2008－MPS－72 (3)
2008－BIO－15 (3)
2008／12／17

Linuxディストリビューションの最適構成

田中 秀宗[*1]
東京工業大学 数理・計算科学専攻

福永 アレックス[*2]
東京工業大学 グローバルエッジ研究院

概要

オペレーティングシステム（OS）を配布するときには，CDやDVDなど，限られた容量しか持たない記憶媒体にOSを収める必要がある．そのために，全てのOSの機能の中から，記憶媒体に収まるだけの重要な機能を選択しなければならない．本研究では，記憶媒体に収まるLinuxディストリビューションを構成するために，最適なLinuxパッケージの集合を選択する問題を対象とする．我々は，この問題を整数計画問題に定式化し，一般的な整数計画ソルバーを適用することで，最適なDebian系Linuxパッケージの集合が得られることを実証した．

Optimal Configurations of Linux Distributions

Hidetoki Tanaka[*1]
Dept. of Mathematical and Computing Sciences,
Tokyo Institute of Technology

Alex Fukunaga[*2]
Global Edge Institute,
Tokyo Institute of Technology

Abstract

Operating system distributions (such as Linux distributions) are distributed on media such as CDs and DVDs, which have limited capacity. This means that the amount of functionality that can be provided in a OS distribution is limited by the storage capacity of the distribution medium. We consider the problem of selecting a subset of packages which maximizes the value of an OS distribution. We model this problem as an integer program, and show that standard mixed integer program solvers can be used to design an optimal, Debian-based Linux distribution according to our model.

1 序論

今日，オペレーティングシステム（OS）を構成する機能の肥大化が進んでいる．OSの多くの機能を管理するために，いくつかのソフトウェア及びライブラリをパッケージとしてひとまとめにする手法が，主にLinuxで取られている．パッケージとして，複数のソフトウェアやライブラリをまとめて管理することで，一般のユーザーからは個々のソフトウェアやライブラリが見えなくなり，抽象化されるので，管理が容易になるという利点がある．パッケージの形式としては，現在，Debian[*3]系のLinuxで主に用いられるdeb形式とRed Hat Linux[*4]系のLinuxで主に用いられるrpm形式が主流である．

パッケージ間にはいくつかの関係が存在する．あるパッケージ A が他のパッケージ B の機能を用いている場合，A は B がインストールされていなければ使うことができない．このとき，パッケージ A はパッケージ B に依存しているといい，パッケージ A はパッケージ B と依存関係にあるともいう．また，あるパッケージ A と他のパッケージ B が同じような機能を持っていると，それらの機能が同一システム上で競合してしまうため，A と B は同時にイン

[*1] tanaka7@is.titech.ac.jp
[*2] fukunaga@is.titech.ac.jp

[*3] http://www.debian.org/
[*4] http://www.redhat.com/

```
Package: console-common
Installed-Size: 456
Version: 0.7.69
Depends: debianutils (>= 1.13),
console-data, kbd | console-tools (>= 1:0.2.3dbs-54),
lsb-base (>= 3.0)
Conflicts: console-data (<< 1999.08.29-20)
Size: 131006
```

図1 パッケージメタデータ

ストールすることができない場合がある．このとき，パッケージ A はパッケージ B と競合しているといい，パッケージ A はパッケージ B と競合関係にあるともいう．

これらパッケージ間の関係は，Debian の場合，図1のような形式で記述される．このように，パッケージ間の関係を記述したデータをメタデータと呼ぶ．このメタデータは，バージョン 0.7.69 のパッケージ console-common は，バージョン 1.13 以上の debianutils, console-data, kbd あるいはバージョン 1:0.2.3dbs-54 以上の console-tools，およびバージョン 3.0 以上の lsb-base に依存していて，バージョン 1999.08.29-20 未満の console-data と競合することを示している．

様々な機能を含んだ OS を配布するためには，CD や DVD などの記憶媒体に，複数のパッケージを収めなければいけない．しかし，多くの場合，全てのパッケージを限られた記憶容量に収めることは不可能である．そのため，記憶媒体に収める一部のパッケージを取捨選択することが必要となる．各パッケージには依存関係や競合関係が存在するので，それらの関係を満たしたパッケージ群を選ばなければいけない．また，記憶媒体に収められたパッケージ群は，できるだけ重要なパッケージの集合でなければいけない．これらの条件を満たすパッケージ群を求める問題は，ナップサック問題からこの問題への多項式時間帰着の存在が容易に示せるので，困難な問題であることがわかる．

本研究では，各パッケージに対して重要度を設定し，インストールされるパッケージの総容量が与えられた記憶媒体の容量を超えないという条件の下で，インストールされるパッケージの重要度の和が最大となるような，パッケージ群を求めることを目標とする．ここで，各パッケージに対する重要度は，OS を配布する側が自由に設定することができるが，実験では Debian Popularity Contest[*5] で公開されている，各パッケージをインストールしたユーザー数を利用した．この目標を達成するために，パッケージ間の関係を制約式に置き換え，目標とする問題を整数計画問題に変換し，この変換した整数計画問題を，一般的な整数計画ソルバーを用いて解くことで，与えられた記憶容量に収まるパッケージ群を求めることができることを示す．

2 関連研究

本研究と関連のあるプロジェクトとして，Tucker らによる Opium [2] と，Mancinelli らによる EDOS project [1] が挙げられる．Opium は，ユーザーがパッケージの追加や削除を行う際に，既にインストールされているパッケージ群と無矛盾なパッケージ群を求め，インストールやアンインストールを行う，パッケージ管理システムである．それに対して，EDOS project で構成されている debcheck は，パッケージを配布する側に立って，依存関係や競合関係によって，インストールすることができない，壊れたパッケージが配布されていないか調べるためのツールである．これらは，本研究と同じく，パッケージ間の依存関係や競合関係を解決するために，関係を論理式や制約式に変換し，整数計画ソルバーや SAT ソルバーなどでそれらの解を見つける手法を用いている研究である．

これら関連研究は，少数のパッケージに関する整数計画問題や充足可能性問題を解くものであったが，本研究では，全てのパッケージに関する整数計画問題を解くこととなる．本研究では，このような大規模な整数計画問題に対しても，同様の手法で解を得ることができることを実証する．

3 定式化

この章では，パッケージに付随するさまざまな情報について述べ，それらに形式的な定義を与える．

全てのパッケージの集合を $P = \{p_1, \ldots, p_n\}$ とする．P に関する依存関係の集合を，$D \subseteq$

[*5] http://popcon.debian.org/

```
Package: A
Version: 2.0
Depends: B (>= 1.0) | C, D (>> 0.1)

Package: B
Version: 1.5
Depends: E
Conflicts: F

Package: C
Version: 3.0
Depends: D, F

Package: D
Version: 0.3
Depends: E | F

Package: E
Version: 2.1
Depends: B

Package: F
Version: 0.5
Conflicts: A (<< 1.0)
```

図 2　パッケージ間の関係

$\{(p,\{d_1,\ldots,d_k\})|\ p\in P,\ \{d_1,\ldots,d_k\}\in\mathcal{P}(P)\}$ で表す．D の 1 つの要素 $(p,\{d_1,\ldots,d_k\})$ は，パッケージ p をインストールする場合には，パッケージ $d_1,\ldots,d_k$ のうち最低 1 つはインストールされていなければいけないことを表している．また，C に関する競合関係の集合を，$C\subseteq\{(p,c)|\ p,c\in P\}$ で表す．C の 1 つの要素 (p,c) は，パッケージ p をインストールする場合には，パッケージ c はインストールされていてはいけないことを表している．なお，ここではパッケージのバージョンは考えないものとする．具体的には，各関係に付けられたバージョンとパッケージに付けられたバージョンに矛盾があればその関係を削除し，矛盾が無ければその関係を残す．図 2 の例では，パッケージ A のパッケージ B への依存関係は残るが，パッケージ F のパッケージ A との競合関係は削除される．

これらの 3 つ組 $R=(P,D,C)$ をディストリビューションと呼ぶ．図 2 の例では，

$$P=\{\mathtt{A},\mathtt{B},\mathtt{C},\mathtt{D},\mathtt{E},\mathtt{F}\}$$
$$D=\{(\mathtt{A},\{\mathtt{B},\mathtt{C}\}),(\mathtt{A},\{\mathtt{D}\}),(\mathtt{B},\{\mathtt{E}\}),(\mathtt{C},\{\mathtt{D}\}),(\mathtt{C},\{\mathtt{F}\}),(\mathtt{D},\{\mathtt{E},\mathtt{F}\}),(\mathtt{E},\{\mathtt{B}\})\}$$
$$C=\{(\mathtt{B},\mathtt{F})\}$$

あるディストリビューションに対して，そのディストリビューションのパッケージの一部がインストール可能であることを，次のように定義する．

定義 1 (インストール可能性)．ディストリビューション $R=(P,D,C)$ において，パッケージの集合 $I\subseteq P$ がインストール可能であるとは，任意のパッケージ $p\in I$ に対して，

(1) $(p,S)\in D\Longrightarrow d\in I$ となる $d\in S$ が存在
(2) $(p,c)\in C\Longrightarrow c\notin I$

を満たすことをいう．

次に，本研究が対象とするディストリビューション最適構成問題を定義する．

定義 2 (ディストリビューション最適構成問題)．ディストリビューション $R=(P,D,C)$，サイズ関数 $s:P\to\mathbb{Z}_{\geq 0}$，重み関数 $w:P\to\mathbb{Z}_{\geq 0}$ および最大容量 M が与えられたとき，条件

(1) I は R においてインストール可能
(2) $\sum_{p\in I}s(p)\leq M$

の下で，目的関数 $\sum_{p\in I}w(p)$ を最大化する $I\subseteq P$ を求める問題を，ディストリビューション最適構成問題と呼ぶ．

サイズ関数 s が各パッケージのサイズ，重み関数 w が各パッケージの重要度を表している．最大容量 M 以下のサイズで，インストールされるパッケージの重要度を最大化するような，インストールされるパッケージの集合を求めるのがディストリビューション最適構成問題である．

4　モデリング

前章で定義したディストリビューション最適構成問題を，整数計画問題へ変換する．ディストリビューションを $R=(P,D,C)$，$P=\{p_1,\ldots,p_n\}$ とする．パッケージ p_i に対して，変数 $x_i\in\{0,1\}$ を

$$x_i=1\iff p_i\text{ がインストールされる}$$

と定める．

次に，制約条件を定める．依存関係 $(p_i,\{p_{j_1},\ldots,p_{j_k}\})\in D$ は，論理式 $x_i\to x_{j_1}\vee\cdots\vee x_{j_k}$ を意味する．これを制約式の形に直すと，$1-x_i+x_{j_1}+\cdots+x_{j_k}\geq 1$ となる．すなわち，

$$-x_i+x_{j_1}+\cdots+x_{j_k}\geq 0 \qquad (1)$$

が依存関係 $(p_i, \{p_{j_1}, \ldots, p_{j_k}\})$ に関する制約式となる．同様に，競合関係 $(p_i, p_c) \in C$ は論理式 $x_i \rightarrow \neg x_c$ を意味する．これを制約式の形に直すと，$1 - x_i + 1 - x_c \geq 1$ となる．すなわち，

$$-x_i - x_c \geq -1 \qquad (2)$$

が競合関係 (p_i, p_c) に関する制約式となる．最大容量 M に関しては，

$$s(p_1)x_1 + \cdots + s(p_n)x_n \leq M \qquad (3)$$

が，制約式となる．式 (1)(2)(3) がディストリビューション最適構成問題における制約式となる．

これら制約条件の下で，目的関数

$$w(p_1)x_1 + \cdots + w(p_n)x_n$$

を最大化する $(x_1, \ldots, x_n)$ を求めれば，

$$\{p_i \mid x_i = 1,\ 1 \leq i \leq n\}$$

が求める I となる．

5 実験

5.1 実験内容

4 章で行ったモデリングを用いて，実際にディストリビューション最適構成問題を解く実験を行った．基となるディストリビューションには Debian を用い，重み関数として，各パッケージをインストールしたユーザー数を利用した．また，公式に公開されている構成と比較するため，最大容量として，ネットワークインストール用の CD イメージの容量[*6] 127MB，完全な CD イメージの容量[*6] 553MB，完全な DVD イメージの容量[*6] 4.23GB を用いた．整数計画ソルバーには，glpk 4.28 および CPLEX 10.1 を利用した．

5.2 結果

5.1 節で述べた実験を行い，最大容量，公式に公開されている構成の重み合計と実験による重み合計との比，glpk の実行時間，CPLEX の実行時間をそれぞれ記録した．このときの結果を表 1 に示す．

表 1 実験結果

容量	既存の構成との重み比	実行時間 (glpk)	実行時間 (CPLEX)
127MB	2.45	16.2h	11.0s
553MB	1.32	15.5h	7.7s
4.23GB	1.05	21.4h	14.2s

どの場合も，実時間内で結果を出力することができ，出力されたパッケージ群の重み合計は既存の構成の重み合計より高い値となった．

6 結論

本研究では，ディストリビューション最適構成問題を対象とした．この問題は NP 困難な問題であるが，自然なモデリングを行い，整数計画問題へ変換することで，この問題を実時間内で解くことができることを示した．また，出力されたパッケージ群の重み関数の合計は，一般に配布されているディストリビューションのパッケージ群の重み関数の合計より高い値となった．

このようなモデリングを行い，整数計画問題を解くことで，より重要度の高いパッケージ群を含んだインストール CD を構成できることが本研究によって示された．現在，Linux のインストール CD は，手作業でパッケージの取捨選択を行っているが，より重要度の高いパッケージ群を選定するため，本研究のような手法を用いることを提案する．

参考文献

[1] F. Mancinelli, et al., Managing the Complexity of Large Free and Open Source Package-Based Software Distributions, Proc. of the 21st IEEE/ACM Int. Conference on Automated Software Engineering, pages 199-208, 2006.

[2] C. Tucker, et al., OPIUM: Optimal Package Install/Uninstall Manager, 29th Int. Conference on Software Engineering, pages 178-188, 2007.

[*6] ここで用いた容量の値は，CD (DVD) の全容量から，OS のインストーラー等の容量を除いた，CD (DVD) 内のパッケージの総容量である．