アルゴリズム 43－6
（1995．1．23）

# レジスタ割り当てにおける循環区間グラフの彩色アルゴリズムについて

紀伊 康之　大山口 通夫　太田 義勝
三重大学工学部

レジスタ割り当ては、効率の良いコードを生成するため、コンパイラにおいて重要な最適化技術のひとつである。構造化されたプログラムのレジスタ割り当て問題は、循環区間グラフの彩色問題に還元できる。この問題は一般にＮＰ完全であるが、最近ファットカバー・アルゴリズムと呼ばれる効率の良い発見的アルゴリズムがHendrenら(1992)によって提案された。このファットカバー・アルゴリズムはグラフの最大幅のみの解析に基づいていたが、本稿では最大幅と最小幅の両方の解析に基づくように改善した新しい発見的アルゴリズムを提案する。

# A Coloring Algorithm of Cyclic Interval Graphs for Register Allocation

Yasuyuki Kii　Michio Oyamaguchi　Yoshikatsu Ohta
Faculty of Engineering, Mie University

Register allocation is an important optimization technique in compilers to generate efficient codes. The register allocation of well-structured programs is reducible to the coloring problem of cyclic interval graphs. Although the problem is NP-complete, an efficient heuristic algorithm called fatcover algorithm was proposed by Hendren et al.(1992). In this paper, we improve this fatcover algorithm based on the analysis of the maximum width of cyclic interval graphs alone, and propose a new heuristic algorithm based on the analysis of both maximum and minimum widths.

# 1 はじめに

コンパイラのコード生成部において、効率の良いコードを生成するために重要な最適化技術のひとつとしてレジスタ割り当てがある。

レジスタ割り当てとは、プログラム実行中の各時刻において、どの変数の値をどのレジスタに割り当てるかを決定する処理である。従来、レジスタ割り当ての問題は、干渉グラフと呼ばれるグラフの彩色問題に還元することで解かれてきた。

干渉グラフとは、グラフの各頂点が変数の生存区間[1]に1対1対応し、2つの生存区間が重なる時かつその時のみ、対応する頂点間に辺を結ぶことで得られるグラフである。

またグラフ $G$ の彩色問題とは、$G$ とある正の整数 $K$ が与えられた時に、辺で結ばれた頂点どうしは同じ色で塗らないという条件の下で、$G$ のすべての頂点を $K$ 色以下で彩色できるかどうかを判定する問題である。各色をレジスタと考えれば、この問題はちょうど、区間が重なる変数の値どうしは同じレジスタに割り当てることができない、というレジスタ割り当ての問題に対応している。

干渉グラフの彩色問題は一般にＮＰ完全なクラスに属すため、これまでにさまざまな発見的解法が提案されている[1][2][3]。

しかしHendrenら[4][5]は、構造化されたプログラムのレジスタ割り当ては循環区間グラフの彩色を基に解くことができることを示した。循環区間グラフ表現はループ構造内における変数の生存区間を自然な形で表すことができ、また多重ループやＩＦ文のような構造を持ったプログラムのレジスタ割り当てにも適応することができる。

循環区間グラフの彩色問題も干渉グラフ同様にＮＰ完全であるが、$K$ が定数であるとき、$K$ 色でグラフを彩色できるかどうかを判定する問題は多項式時間で解くことができる[6]。なお一般のグラフでは、3以上の定数 $K$ に対して上記の問題はＮＰ完全となる。

Hendrenらは、循環区間グラフの最大幅の情報を利用した、ファットカバー・アルゴリズムと呼ばれる彩色問題を解く発見的解法を提案した。しかし2.2節で述べるように、循環区間グラフ $G$ の彩色数[2]はそのグラフの最小幅とも密接な関係がある。従って、本稿では、グラフの最大幅と最小幅の両方に着目した $j$-カバーの概念を導入して、$j$-カバーを用いた、彩色問題を解く新しい発見的解法を提案する。また本方法とファットカバー・アルゴリズムの比較を行い、本方法の利点について述べる。

[1] 変数が定義されてから最後に使用されるまでの期間

# 2 循環区間グラフ

この章では、本稿で用いる諸定義と、循環区間グラフの彩色数に関して既に知られている結果について述べる。

## 2.1 諸定義

ループ構造をしたプログラムのコードにおける変数の生存区間（以下単に区間と呼ぶ）は円弧モデルの円弧と見ることができるので、これらの区間から作られる干渉グラフは循環区間グラフと見なせる。

**循環区間グラフ**とは、グラフの各頂点がある円の各円弧に1対1対応し、円弧 $I_i$ と $I_j$ が重なる時かつその時のみ、対応する頂点 $v_i$ と $v_j$ に辺を結ぶことで得られるグラフのことである。例えば図1(a)のグラフは図1(b)の対応する円弧モデルが存在するので循環区間グラフである。

以後、特に断ることなく循環区間グラフとその円弧モデル表現を混同して用いる。即ち、循環区間グラフを円弧の有限集合と定義する。また、円弧を区間と呼ぶ。

**表記 2.1** プログラムコードの各命令をプログラム実行の単位と考え、各命令に順に時刻1から $M$ までの番号を振る。その時変数の生存区間 $I$ はその開始点（時刻）$t$ と終了点（時刻）$t'$ に対応する番号で表す（$1 \leq t, t' \leq M$）。即ち、$I=[t, t')$ と書いて端点 $t'$ を含まない $t$ から $t'$ までの区間を表すとする。

例えば、命令 $i$ で定義され命令 $j$ で最後に使用される区間は $[i, j)$ で表すことができる。またレジスタ割り当てにおいては各区間の重なり方のみに興味があるので、開始点も終了点もない時刻は取

[2] $G$ を彩色するのに必要な最小の色数

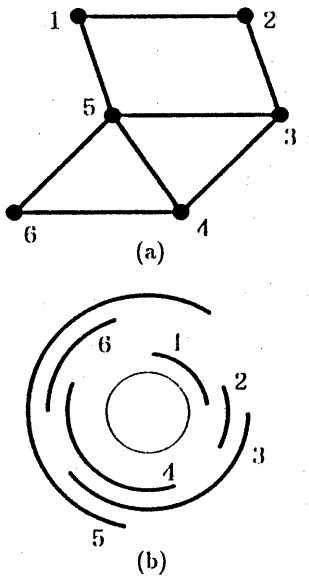


図 1: 循環区間グラフ(a)とその円弧モデル(b)

り除いて考えることができる。よって時刻の数 $M$ は区間数 $m$ の 2 倍以下と仮定して一般性を失わない。以後最大時刻を $M$ で表す。

**定義 2.2** 時刻 $t$ が区間 $I=[t_1, t_2)$ にカバーされているのは、

1. $t_1 < t_2$ の時、$t_1 \leq t < t_2$
2. $t_1 \geq t_2$ の時、$t_1 \leq t \leq M$ または $1 \leq t < t_2$

のいずれかの時である。

**定義 2.3** 2 つの区間 $I_1$ と $I_2$ が重なるとは、$I_1$ と $I_2$ の両方にカバーされるある時刻 $t$ が存在することである。

**定義 2.4** 時刻 $t$ における循環区間グラフ $G$ の幅とは、$t$ をカバーする $G$ の区間の数である。また $G$ の幅の最大値を $W_{max}(G)$、最小値を $W_{min}(G)$ と書く。

**定義 2.5** 循環区間グラフ $G$ に対して、$W_{max}(G)$ となるような時刻を $G$ のファットスポットと言い、$G$ のすべてのファットスポットの集合を Fatspots($G$)で表す。同様に $W_{min}(G)$ となるような時刻を $G$ の最小スポットと言い、$G$ のすべての最小スポットの集合を Minspots($G$)で表す。

**表記 2.6** 循環区間グラフ $G$ の彩色数を CN($G$)と表記する。

## 2.2 循環区間グラフの彩色数

循環区間グラフの幅と彩色数の関係については、次の結果が既に知られている。

**定理 2.7** ([4][5]) 任意の循環区間グラフ $G$ の彩色数 CN($G$)に関して、以下の不等式が成立する。
$W_{max}(G) \leq \mathrm{CN}(G) \leq W_{max}(G) + W_{min}(G)$

このように循環区間グラフの彩色数はそのグラフの最大幅と最小幅の両方に関係がある。

# 3 ファットカバーと $j$-カバー

この章では、まず Hendren らが[4][5]で示したファットカバーの定義とファットカバー・アルゴリズムの概要について述べる。次に、本稿で導入する $j$-カバーの定義を与え、$j$-カバーを用いて循環区間グラフの彩色数を特性化する。

## 3.1 ファットカバー

**定義 3.1** 循環区間グラフ $G$ の部分グラフ[3] $F$ が次の 2 つの性質を満たす時、$F$ を $G$ のファットカバーと呼ぶ。

1. $F$ のすべての区間はお互いに重ならない。
2. $\forall t_f \in \mathrm{Fatspots}(G)$, $t_f$ をカバーする区間が $F$ 内に存在する。

定義より $F$ は 1 色で彩色でき、かつ $F$ を $G$ から取り除いたグラフ $G-F$ の最大幅は $G$ の最大幅 $-1$ となる。Hendren らが提案した発見的な彩色アルゴリズムであるファットカバー・アルゴリズムは、次のように与えられる。

ファットカバー・アルゴリズム ([4][5])

1. $G$ のファットカバー $F$ を求める。
2. $F$ を 1 色で彩色して取り除き、$G$:=$G-F$ とする。
3. 1へ戻る。

[3] より正確には、区間の集合である $G$ の部分集合

なお、このアルゴリズムのステップ1でファットカバーが求まらなかった場合は、残りのグラフ $G$ を単純に（$W_{max}(G)+W_{min}(G)$ 以上の色を使って）彩色する。またこのアルゴリズムの時間計算量は $O(m*W_{max}(G))$である[4]（但し、$m$ は $G$ の区間数とする）。

### 3.2 $j$-カバー

定義3.1から明らかなように、ファットカバーはグラフの最大幅のみに着目しており、その最小幅は考慮していない。しかし定理2.7で述べたように、グラフの彩色数と最小幅には密接な関係があり、アルゴリズムのステップ1で、グラフの最小幅をも減らすファットカバーを優先して取り除くことができれば、アルゴリズムの改善が期待できるであろう。そこで本稿では新たにグラフの最大幅と最小幅の両方を考慮した $j$-カバーを定義する。

**定義 3.2** 循環区間グラフ $G$ の部分グラフ $F$が次の3つの性質を満たす時、$F$を $G$ の $j$-カバーと呼ぶ（但し、$1 \leq j \leq W_{min}(G)$とする）。

1. $F$は $j$色で彩色できる。
2. $\forall t_f \in \mathrm{Fatspots}(G)$, $t_f$をカバーする区間が $F$内に存在する。
3. $\exists t \in \mathrm{Minspots}(G)$, $t$をカバーする $j$本の区間が $F$内に存在する。（この時 $t$ を $F$の基準最小スポットと呼ぶ）

$j$-カバーの定義より、1-カバーはグラフの最小幅をも減らすファットカバーになっていることが分かる。

また $j$-カバーをさらに拡張した$(i,j)$-カバーを次のように定義する（但し、$1 \leq i \leq j \leq W_{min}(G)$とする）。$G$ の部分グラフ $F$が、$j$-カバーの条件1, 3と、次の条件

2′ $G$ のすべての時刻 $t$ に対して、$t$ の幅 $w$が $w > W_{max}(G) - i$ を満たすならば、$t$ をカバーする$(w - (W_{max}(G) - i))$本の区間が $F$内に存在する。

を満たす時、$F$を$(i,j)$-カバーと呼ぶ。

$(i,j)$-カバーとは直観的に言えば、

1. $W_{max}(G-F) = W_{max}(G) - i$
2. $W_{min}(G-F) = W_{min}(G) - j$

となるような $j$色で彩色できる $G$ の部分グラフ $F$のことである。

**表記 3.3** 区間 $I_1, \ldots, I_k$を含む $j$-カバーを、$j$-カバー$(I_1, \ldots, I_k)$と書く。

### 3.3 循環区間グラフの特性化

次の定理は前節で定義した$(i,j)$-カバーと、循環区間グラフ $G$ の彩色数 $\mathrm{CN}(G)$の関係を示している（但し、～は否定を表す）。

**定理 3.4** 任意の循環区間グラフ $G$ に対して、次の関係が成立する。（但し、$n = W_{max}(G)$, $k = W_{min}(G)$とする。）

1. $\mathrm{CN}(G) = n \Leftrightarrow \exists(k,k)$-カバー
2. $\mathrm{CN}(G) = n + i \Leftrightarrow$
   $\sim \exists(k-(i-1),k)$-カバー $\wedge\ \exists(k-i,k)$-カバー
   （但し、$1 \leq i \leq k-1$）
3. $\mathrm{CN}(G) = n + k \Leftrightarrow \sim \exists k$-カバー

また定理3.4に関連した結果として、次の定理がある。

**定理 3.5** 任意の循環区間グラフ $G$ に対して、次のことが成立する。

1. $\exists 3$-カバー $\Leftrightarrow \exists i$-カバー
   （但し、$3 \leq i \leq W_{min}(G)$）
2. [ある最小スポットをカバーするある2区間 $I_i$と $I_j$に対して、$\sim \exists 2$-カバー$(I_i, I_j)$ ]
   $\Rightarrow \mathrm{CN}(G) \geq W_{max}(G) + 2$
3. [ある最小スポットをカバーするある区間 $I$に対して、$\sim \exists 1$-カバー$(I)$ ]
   $\Rightarrow \mathrm{CN}(G) \geq W_{max}(G) + 1$

## 4 彩色アルゴリズム

この章では、新しい彩色アルゴリズムの概要を示す。これは定理3.5の結果を用い、1-カバー, 2-カバー, 3-カバーの3つのカバーを求めることで循環区間グラフの彩色をする発見的なアルゴリズムである。

[4]但し、4.2節で述べる前処理の 1から 3の情報が既に与えられていると仮定した場合の時間計算量である

## 4.1 彩色アルゴリズム

循環区間グラフを彩色する新しい発見的アルゴリズムを図2に示す。なお、循環区間グラフは区間の集合 $\{I_1, I_2, \ldots, I_m\}$ として与えられるものとする。アルゴリズム中の前処理と、1-カバー, 2-カバー, 3-カバーのアルゴリズムについては後述する。また本アルゴリズムの解説は次章で述べる。

```
前処理;
Col:=0;
while W_min(G) ≥ 1 do
  begin
    if ∃1-カバー then
      begin G から 1-カバーを取り除く;
            Col:=Col+1 end
    else if ∃2-カバー then
      begin G から 2-カバーを取り除く;
            Col:=Col+2 end
    else if ∃3-カバー then
      begin G から 3-カバーを取り除く;
            Col:=Col+3 end
    else begin
      Col:=Col+W_max(G) + W_min(G);
      goto 99 end
  end;
Col:=Col+W_max(G);
99:
```

図 2: 彩色アルゴリズム

(但し、時間計算量の関係から、本アルゴリズム中の 1-カバー, 2-カバーは、基準最小スポットをカバーする区間についてのみ 1-カバーが存在するかを調べ、また、基準最小スポットをカバーする1組の2区間についてのみ 2-カバーが存在するかを調べている。)

## 4.2 前処理

本アルゴリズムの while ループ内での処理を簡単にするため、あらかじめ次の情報を求めておく。

1. 昇順にソートされた Fatspots($G$)
2. 区間の開始点での昇順ソート系列
3. 区間の終了点での昇順ソート系列
4. ある最小スポット $t$ と、$t$ をカバーする区間の集合
5. $\forall t_f \in$ Fatspots($G$), $t_f$ をカバーする区間の集合

これらの情報は $O(m \log m)$ で求めることができる。なお、上記の4で求めた最小スポットは、各カバーの基準最小スポットとして利用する。また一般性を失うことなく、1から3で求めた Fatspots($G$) と区間の開始点・終了点での昇順ソート系列はこの基準最小スポットの次の点を基準としてソートされていると仮定する。

## 4.3 1-カバーアルゴリズム

1-カバーアルゴリズムは、基準最小スポットをカバーする各区間 $I$ に対して、図3の 1-カバー($I$)アルゴリズムを実行し、ある $I$ に対して *true* が返ってくる時 *true* を返し、すべての区間に対して *false* が返ってくる時 *false* を返す。なお、アルゴリズム中の s-p($I$)と e-p($I$)はそれぞれ、区間 $I$ の開始点と終了点を返す関数とする。

```
手続き 1-カバー(I);

  Iをマークする; I_x:=I;
  while next_fatspot(I_x) ≠ ⊥ do;
    begin
      fs:=next_fatspot(I_x);
      mark_interval(I_x, fs);
      I_x:=choice_interval(fs);
      if I_x=⊥ then return false
    end;
  return true;
```

図 3: 1-カバー($I$)アルゴリズム

但し、各関数（手続き）は以下のような処理を行うものとする。

- next_fatspot($I_x$)
  e-p($I_x$)≤ $t$ <s-p($I$)を満たす最小のファットスポット $t$ を返す。$t$ が存在しない時は⊥を返す。
- mark_interval($I_x, fs$)
  [ s-p($I$), e-p($I_x$) )と重ならない $fs$ をカバーする各区間 $I'$にマークし、$I'$に $I_x$へのポインタをつける。
- choice_interval($fs$)
  $fs$ をカバーするマークされた区間で、終了点が最小なものを返す。そのような区間が存在しない時は⊥を返す。

また 1-カバーが存在するかどうかだけでなく、1-カバー($I$)が存在する時は、$I_x$から $I$ までポイン

タをたどることで、容易に1-カバーを得ることができる。

最後に1-カバーアルゴリズムに関する2つの補題を示す。

**補題 4.1** グラフ $G$ の区間 $I$に 1-カバー$(I)$が存在する時かつその時のみ、手続き 1-カバー$(I)$は *true* を返して停止する。また、手続き 1-カバー$(I)$の出力として得られる区間系列は 1-カバーである。

**補題 4.2** 1-カバーアルゴリズムの時間計算量は $O(m * W_{min}(G))$である。

## 4.4 2-カバーアルゴリズム

2-カバーアルゴリズムでは、基準最小スポットをカバーする2区間 $I_1$, $I_2$を1組選び（s-p$(I_1) \leq$ s-p$(I_2)$とする)、2-カバー$(I_1, I_2)$が存在するかどうかを調べる。その概要を図4に示す。このアルゴリズムにおいては、各区間は $C_1$と $C_2$のマーク（もしくはその両方）がついているかどうかの情報と、それぞれのマークについて着目ファットスポットと呼ぶ情報を持っていると仮定し、$C_i$（$i = 1, 2$）の着目ファットスポットを $C_i$_$fs$ と表記する。また後で2-カバーを得るため、$C_1$と $C_2$用の2つのポインタも用意しておく。

但し、各関数（手続き）は以下のような処理を行うものとする。

- mark$(I, C, bfs)$
  $I$に $C$のマークをつけ、$C$_$fs$:=$bfs$ とする。
- next_fatspot$'(I_x)$
  e-p$(I_x) \leq t <$s-p$(I_1)$を満たす最小のファットスポット $t$を返す。$t$が存在しない時は⊥を返す。
- mark_interval_$C_1(I_x, fs)$
  $fs$ をカバーする各区間 $I'$に対して、
  1. $I'$が[ s-p$(I_1)$, $I_x$の $C_2$_$fs+1$ )と重ならず、かつ $I'$にまだ $C_1$のマークがついていないなら、mark$(I', C_1, fs)$を実行し、$I'$の $C_1$用のポインタが $I_x$をさすようにする。
  2. $I'$が[ s-p$(I_2)$, e-p$(I_x)$ )と重ならず、かつ $I'$にまだ $C_2$のマークがついていないなら、mark$(I', C_2, fs)$を実行し、$I'$の $C_2$用のポインタが $I_x$をさすようにする。

手続き 2-カバー$(I_1, I_2)$;

```
mark(I1, C1,基準最小スポット);
mark(I2, C2,基準最小スポット);
I1, I2で終了点の大きい方を Ixとする;
fs:=next_fatspot'(Ix);
if fs = ⊥ then return true
fs をカバーする各区間 I'に対して、
  if I'と I1が重ならない then mark(I', C1, fs);
  if I'と I2が重ならない then mark(I', C2, fs);
Ix:=next_interval(Ix);
if Ix=⊥ then return false

while next_fatspot'(Ix) ≠ ⊥ do;
 begin
   fs:=next_fatspot'(Ix);
   if Ixに C1のマークがついている then
     mark_interval_C2(Ix, fs);
   if Ixに C2のマークがついている then
     mark_interval_C1(Ix, fs);
   Ix:=next_interval(Ix);
   if Ix=⊥ then return false
 end;
return true;
```

図 4: 2-カバー$(I_1, I_2)$アルゴリズム

- mark_interval_$C_2(I_x, fs)$
  mark_interval_$C_1(I_x, fs)$の"$C_1$"と"$C_2$"を 入れ換えたもの。

また1-カバーの時と同じく、2-カバー$(I_1, I_2)$が存在する時は、ポインタをたどることで、容易に2-カバーを得ることができる。

最後に2-カバーアルゴリズムに関する2つの補題を示す。

**補題 4.3** グラフ $G$の区間 $I_1, I_2$に 2-カバー$(I_1, I_2)$が存在する時かつその時のみ、手続き 2-カバー$(I_1, I_2)$は *true* を返して停止する。また、手続き 2-カバー$(I_1, I_2)$の出力として得られる区間系列は 2-カバーである。

**補題 4.4** 2-カバーアルゴリズムの時間計算量は $O(m * W_{max}(G))$である。

## 4.5 3-カバーアルゴリズム

3-カバーアルゴリズムでは、基準最小スポットをカバーする $k(=W_{min}(G))$本の区間 $I_1, \ldots, I_k$に対してまず、$k$-カバー$(I_1, \ldots, I_k)$が存在するかどうか

を調べる。定理3.5の1より、$k$-カバーが存在するなら3-カバーが存在する。$k$-カバー$(I_1,\ldots,I_k)$を求めるアルゴリズムを図5に示す（$H$:=$\{I_1,\ldots,I_k\}$とする)。但し、このアルゴリズムの例外として、

- $H$内にある区間$I$が存在し、$I$は$H-\{I\}$内のすべての区間の真部分区間となっている

を満たす時、$G$の区間$I'$が、$H-\{I\}$内のすべての区間と重なり、かつ$I$とは重ならないなら、$I'$に特別な$M'$のマークをつける。$H$と、互いに重なる$M'$のマークのついた2区間では$k$-カバーを構成できない。よってこのような区間系列は除外しなければならず、図5とは異なるアルゴリズムが必要であるが、ここでは省略する。

手続き $k$-カバー$(I_1,\ldots,I_k)$;

$H$内のすべての区間をマークする;
$H$内の区間で終了点最大の区間を$I_x$とする;
$old\_fs$:=基準最小スポット;
**while** next_fatspot$''(I_x) \neq \perp$ **do**;
  **begin**
    $fs$:=next_fatspot$''(I_x)$;
    mark_interval$'(old_fs, fs)$;
    $I_x$:=choice_interval$'(fs)$;
    **if** $I_x=\perp$ **then return** *false*;
    $old\_fs$:=$fs$;
  **end**;
**return** *true*;

図 5: $k$-カバー$(I_1,\ldots,I_k)$アルゴリズム

このアルゴリズムで用いる各関数（手続き）は以下のような処理を行うものとする。

- next_fatspot$''(I_x)$
  c-p$(I_x)\leq t <$s-p$(I)$を満たす最小のファットスポット$t$を返す（但し$I$は$H$内で開始点が最小の区間)。$t$が存在しない時は$\perp$を返す。
- mark_interval$'(old_fs, fs)$
  $fs$をカバーし、$old\_fs <$ s-p$(I')$を満たす各区間$I'$が、$H$内の少なくとも1つの区間と重ならないなら、$I'$をマークする。
- choice_interval$'(fs)$
  $fs$をカバーするマークされた区間で、終了点が最大な区間$I$を返し、$I$を出力する。そのような区間が存在しない時は$\perp$を返す。

また3-カバーが存在するかどうかだけでなく、3-カバーが存在する時は、$H$内の3区間と手続き$k$-カバー$(I_1,\ldots,I_k)$の出力から容易に3-カバーを得ることができる。

最後に3-カバーアルゴリズムに関する2つの補題を示す。

**補題 4.5** グラフ$G$に3-カバーが存在する時かつその時のみ、手続き$k$-カバー$(I_1,\ldots,I_k)$は*true*を返して停止する。また、手続き$k$-カバー$(I_1,\ldots,I_k)$の出力から得られる区間系列は3-カバーである。

**補題 4.6** 3-カバーアルゴリズムの時間計算量は$O(m)$である。

## 5 比較

前節で示した本アルゴリズムとHendrenらが提案したファットカバー・アルゴリズムとの比較結果を表1に示す。ここで循環区間とはループの繰り返しの間でも生きている区間のことである。また本アルゴリズムの時間計算量に関しては、補題4.2, 4.4, 4.6より求められる（但し$m$は入力されたグラフ$G$の区間数)。

循環区間グラフ$G$が（$W_{max}(G)+W_{min}(G)$）以下の色で彩色できることは定理2.7で既に述べたが、本アルゴリズムは$j$-カバー（$j$=1,2,3）を求めて取り除くことにより、この最悪値を減らすことに成功している。即ち、1つのファットカバーの除去は最悪値$W_{max}(G)+W_{min}(G)$を1色で1減らすのに対して、本アルゴリズムの1-カバーの除去は1色で最悪値を2減らすことを可能にしている。

また本アルゴリズムで$G$から2-カバーや3-カバーを取り除く場合、それぞれ2色と3色を使って、それらのカバーを取り除いた残りのグラフの最大幅は共に$G$の最大幅−1にしかならないが、それによって$G$をその彩色数CN$(G)$で彩色することを不可能にしているわけではない。例えば、1-カバーが存在しない時は、定理3.5より、CN$(G)$ $\geq W_{max}(G)+1$であり、従ってもし2-カバーが見つかるならばそれを除去した後でも最適に彩色する可能性が残されている。

ファットカバー・アルゴリズムでは、ファットカバーが求まらない場合は残りのグラフを単純に彩

| | ファットカバー・アルゴリズム | 本アルゴリズム |
|---|---|---|
| アルゴリズムの基本的な方法 | 循環区間に着目し、ファットカバーを繰り返し求める。 | ある最小スポットをカバーする区間に着目し、1-カバーを繰り返し求める。 |
| カバーが求まらない時 | 残りのグラフをその（最大幅＋最小幅）以上の色で単純に彩色する。 | 2-カバー又は3-カバーを求める。見つかればそれを取り除き、その後再び1-カバーを求める。3-カバーが存在しない時は、残りのグラフをその（最大幅＋最小幅）の色を使って最適に彩色する。 |
| 時間計算量 | $O(m * W_{max}(G))$ | $O(m * W_{max}(G) * W_{min}(G))$ |

表 1: ファットカバー・アルゴリズムと本アルゴリズムの比較

色してしまうが、本アルゴリズムでは2-カバーまたは3-カバーを求めて取り除くことで、再び1-カバーを求める機会を与えている。

さらに本アルゴリズムでは、3-カバーが存在しない時にはそのグラフを最適に彩色することができる（定理3.4)。またグラフの最小幅が0になった時も、最適に（$W_{max}(G)$色で）彩色できる。

以上より、本アルゴリズムの最大時間計算量はファットカバー・アルゴリズムよりも$W_{min}(G)$倍増えてはいるが、多くの場合に、より少ない色数で彩色できるという点で、改良されていると言える。

## 6 おわりに

本稿ではHendrenらによって与えられたファットカバーの概念を拡張して、グラフの最大幅と最小幅の両方を考慮した$j$-カバーを定義し、$j$-カバーを用いて循環区間グラフの彩色数を特性化できることを初めて明らかにした。次に、この特性化に基づく新しい発見的な彩色アルゴリズムを提示し、Hendrenらが提案したファットカバー・アルゴリズムとの比較を行い、本アルゴリズムの利点を示した。今後の課題としては実際にこのアルゴリズムをコンパイラに実装し、さらに詳細な性能評価を行うことが上げられる。

## 参考文献


[1] G. J. Chaitin. *Register Allocation and Spilling via Graph Coloring.* Proc. of the ACM SIGPLAN '82 Symp. on Compiler Construction, SIGPLAN Notices, 17(6):pp. 98–105, June 1982.

[2] F. Chow and J. Hennessy. *Register Allocation by Priority-based Coloring.* Proc. of the ACM SIGPLAN '84 Symp. on Compiler Construction, SIGPLAN Notices, 19(6):pp. 222–232, June 1984.

[3] F. Chow and J. Hennessy. *The Priority-Based Coloring Approach to Register Allocation.* ACM TOPLAS, 12(4):pp. 501–536, October 1990.

[4] L. J. Hendren et al. *A Register Allocation Framework Based on Hierarchical Cyclic Interval Graphs.* LNCS 641, pp. 176–191, October 1992.

[5] L. J. Hendren et al. *A Register Allocation Framework Based on Hierarchical Cyclic Interval Graphs.* Technical Report of McGill University, Canada, May 1993.

[6] M. R. Garey et al. *The Complexity of Coloring Circular Arcs and Chords.* SIAM Journal on Algebraic and Discrete Methods, 1(2):pp. 216–227, June 1980.