スライドレジスタ割付問題の厳密解法

秋川友宏[†] 山下義行[††] 中田育男[†††]

レジスタ#0, #1, ... を #(K+0), #(K+1), ... にいっせいに改名する命令を持つプロセッサを仮定し，いっせいに改名されるレジスタをスライドレジスタとよぶことにする．このようなプロセッサを採用した計算機の1つとしては，筑波大学の CP-PACS があげられる．従来のレジスタにおいては，ループプログラムのレジスタ割付問題は Cyclic Arc Graph の彩色問題として定式化できることが知られているが，スライドレジスタに関してはその割付問題の性質が詳しくは分かっていなかった．本論文では，この問題を巡回セールスマン問題の変形として定式化し，この問題に厳密解法を与える．また，厳密解法で実際に最適解を求め，その求解時間をもとに本厳密解法の実用化の方法を提案する．さらに，本厳密解法を近似解法の評価に応用する．筆者らは先に近似解法を提案・評価しているが，提案時の評価は55%程度の最適性しか保証できていなかった．最適解を利用した新たな評価により，同解法が85%以上の例題に対して最適解を与えていたことが確認できた．

An Exact Algorithm of Slide-Register Allocation Problem

TOMOHIRO HARAIKAWA,[†] YOSHIYUKI YAMASHITA[††] and IKUO NAKATA[†††]

Assume a processor with an instruction which renames registers #0, #1, ... respectively to #(K+0), #(K+1), ... all at once is given, and call these registers slide-registers. CP-PACS in the University of Tsukuba is one of the computers that utilize such a processor. For conventional registers, it is known that the register allocation problem of a loop program can be formulated as the coloring problem of Cyclic Arc Graph. However, for the slide-registers, this allocation problem has not been studied in detail. This paper shows an exact algorithm of this problem by formulating it as a variation of Traveling Salesman Problem. Moreover, based on the solution time obtained by solving problems with this exact algorithm, this paper presents a practical application method of this formulation. Furthermore, this exact algorithm is applied to the evaluation of approximation algorithms. In our recent work, an approximation algorithm was proposed, but our evaluation method there could only assure its optimality as 55%. The new evaluation using the optimal solutions revealed that the approximation algorithm proposed formerly was giving the optimal solution to more than 85% of examples.

1. はじめに

レジスタ番号をまとめて付け換える命令，すなわち，レジスタ#0, #1, ... を #(K+0), #(K+1), ... にいっせいに改名する命令を持つプロセッサを仮定する．ここで，このような改名のための命令を slide K，いっせいに改名されるレジスタを**スライドレジスタ**とよぶことにする．このようなプロセッサを採用した計算機としては，後に詳述する筑波大学計算物理学研究センターの CP-PACS[1] のほか，スライド量 K を1に固定した Cydra 5[2] があげられる．従来のレジスタにおいては，ループプログラムのレジスタ最小化問題は NP 困難な問題であり，Cyclic Arc Graph の彩色問題として定式化できることが知られている[3]が，スライドレジスタにおいては同問題の性質が詳しくは分かっていなかった．本論文は，巡回セールスマン問題というよく知られた問題との類似性に着目してスライドレジスタ割付問題を定式化することで，その性質をより明らかにしようと試みるものである．

巡回セールスマン問題は NP 困難な問題であり，それとの類似性に着目した本論文はスライドレジスタ割付問題に（決定性）多項式時間解法を与えているものではないから，常用するレジスタ割付は何らかの近似

† 筑波大学工学研究科
Doctoral Program in Engineering, University of Tsukuba
†† 筑波大学電子・情報工学系
Institute of Information Sciences and Electronics, University of Tsukuba
††† 図書館情報大学図書館情報学部
University of Library and Information Science

解法にゆずることとなろう．しかし，よく知られた問題との類似性に着目しておくことには既知のヒューリスティックを近似解法に応用するなどの将来面が，定式化に基づく厳密解法そのものには特に重要なループのレジスタ割付のほか，常用する近似解法の設計・評価に応用するなどの実用面が期待できる．本論文では実際に厳密解法で最適解を求め，その求解時間をもとに本厳密解法の実用化の方法を提案する．さらに，本厳密解法を近似解法の評価に応用する．筆者らはこれまでに近似解法を提案している[4]が，提案時には占有レジスタ数の下界となる同時生存変数の最大数 W_{max}[3]との一致判定により評価せざるをえず，10,000 例中 55%程度の例題について最適性を保証できたにすぎなかった．本厳密解法で 10,000 例の最適解をすべて求め，それとの比較による再評価を行ったところ，同解法が 85%以上の例題に対して最適解を与えていたことが確認できた．

2. 基本事項

本章では，超並列計算機 CP-PACS のノードプロセッサを具体例として，slide K 命令を持つプロセッサについて説明する．また，slide K 命令を考慮したレジスタ表現法である Spiral Graph についても説明する．

2.1 スライドウィンドウ・アーキテクチャ

スライドウィンドウ・アーキテクチャは，主として主記憶アクセスのレイテンシを隠蔽するために提案され[5]，超並列計算機 CP-PACS のノードプロセッサ HARP-1E に採用された．HARP-1E は PA-RISC1.1[6] アーキテクチャの浮動小数点数レジスタセットに**スライドウィンドウ機構**を付加したものである（**図 1**）[7]．

HARP-1E のレジスタはグローバルレジスタとローカルレジスタからなる．グローバルレジスタの個数 g は 4，8，12 から選択可能であり，32 から g を減じた個数がローカルレジスタの個数となる．グローバルレジスタは従来のアーキテクチャにおけるレジスタと同様であり，従来のレジスタ割付フレームワークを適用できる．一方，ローカルレジスタは，レジスタウィンドウのオフセットを変更することによっていっせいに改名可能であるという特徴を持つ．以降，このローカルレジスタを，その意味から**スライドレジスタ**とよぶ．HARP-1E は改名のための命令，`FWSTPset`（set Floating Window STart Pointer）命令を持つ（slide $-K$ に相当）．また，`FRPreload` および `FRPoststore`（以下，単に preload，poststore と書く）という命令を持ち，ウィンドウの外側のレジスタに対してもウィンドウオフセットからの相対指定でロード/ストアを行うことができる．

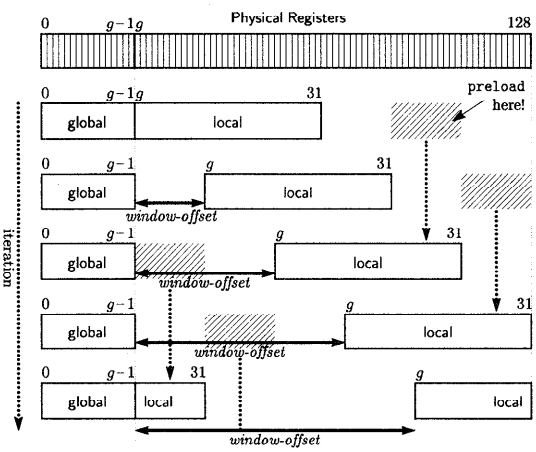


図 1 スライドウィンドウ・アーキテクチャ
Fig. 1 Slide-Window Architecture.

このような機構を用いると，ループ構造のプログラムに対して主記憶レイテンシを隠蔽する目的コードを生成することができる．現在の繰返しにおいては，すでにレジスタにロードされた値を用いて演算を行うとともに，数回先の繰返しで必要となる値を`FRPreload` 命令を用いて将来ウィンドウが移動する位置（図 1 斜線部）にロードしておくのである．

`FWSTPset` 命令はジャンプ命令と同時実行可能であるため，多くの場合はループ末尾にジャンプ命令とともに置かれる．各繰返しの末尾においていくつウィンドウをスライドさせるかは，繰返し 1 回あたりいくつの値をメモリから読み出すか，すなわちループ中の load 命令の数（より正確には，付録のように preload 命令の数）によって決めている[4]．

なお，図 1 のように，スライドレジスタのレジスタ番号は g から始まるのであるが，簡単のため，本論文では 0 から始まるものとし，`SR0`, `SR1`, ... と書く．

2.2 Spiral Graph

以上にみられるスライドレジスタの表現法としては，Spiral Graph がある．その詳細は論文 4) に譲り，概略のみ説明する．

Spiral Graph は，横軸にループ内の命令ステップ番号をとったらせん状のグラフである（**図 2**）．繰返し 1 回あたりのマシンサイクル数を**イテレーション開始間隔**または**イテレーション立ち上げ間隔**（II: Initiation Interval）[8] という．1 マシンサイクルごとに次の命令の実行を開始するプロセッサ（ノンブロッキング型のプロセッサという）においては，II は繰返し 1 回あ

```
do {
0:  load   v1
1:  add    v6, v1 -> v4
2:  mult   v4, v4 -> v5
3:  add    v5, v1 -> v2
4:  mult   v1, v2 -> v3
5:  add    v2, v3 -> v6
6:  store  v2
    slide +1
}
```

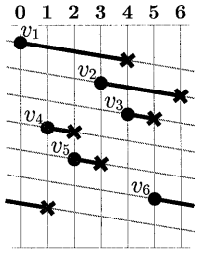


図 2 Spiral Graph
Fig. 2 Spiral Graph.

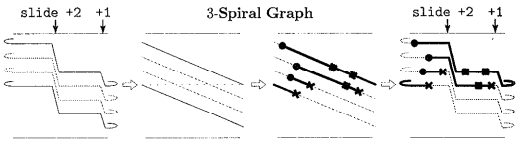


図 3 Spiral Graph の考え方
Fig. 3 The concept of Spiral Graph.

たりの命令ステップ数と同じであるから，本論文ではループ内の命令ステップ数と II を同一視する．グラフの斜線は，後述するように，スライド命令によるレジスタ番号のずれを表している．Spiral Graph 上においては，各々の変数もしくは一時名（以下，単に変数という）のライブレンジは線分で表現する．たとえば，v_1 については，ステップ 0 の load 命令で代入された値がステップ 4 の mult 命令まで保持されなければならない．そこで，$v_1 = [0,4)$ と書く．また，繰返しをまたぐライブレンジは，繰返しのたびに終点にイテレーション立ち上げ間隔 II を加えて表記するものとする．たとえば，v_6 については，ステップ 5 の add 命令で生成された値が次の繰返しのステップ 1 まで保持されなければならないから，$v_6 = [5,8)$ と表す．

Spiral Graph は，ループ 1 周の合計スライド量，すなわち，あるレジスタが次の繰返しでいくつ先の番号になるのかのみを保存したグラフである．先述した理由から通常スライド命令はループの末尾に置かれ，本論文の厳密解法もそのようなループを扱うのであるが，Spiral Graph 自体は，より一般的に，複数のスライド命令を含むループも扱うことができる．複数のスライド命令を含むループにおいて，レジスタ番号の変化のようすを表すと，横軸が途中で何度か折れ曲がったグラフ（**図 3** 最左）になる．この例ではループ 1 周の合計スライド量が 3 であるから，3 重らせん構造を持つ 3-Spiral Graph として表す（図 3 中央左）．以後，K-Spiral Graph の K 本のらせんを，便宜上，上からトラック 1，トラック 2，...，トラック K とよぶ．

Spiral Graph を用いると，レジスタ割付は，この

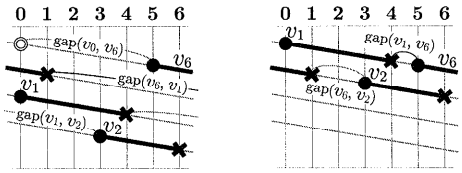


図 4 ライブレンジ間の隙間
Fig. 4 'gap' between live ranges.

K 本のトラックの上に変数のライブレンジを次々と巻き付けていく問題と見なすことができる（図 3 中央右）．すべて巻き付けたなら，スライド命令を考慮してグラフの形状を復元すればよい（図 3 最右）．

Spiral Graph におけるいくつかの用語を定義する．

定義1（**始点，終点**） 変数のライブレンジ $v_i = [s_i, e_i)$ において，s_i $(0 \le s_i < II)$ を v_i の**始点**，e_i $(s_i < e_i)$ を**終点**という． ■

定義2（**開始，終了ステップ**） 変数のライブレンジ $v_i = [s_i, e_i)$ において，$s_i \bmod II$，$e_i \bmod II$ をそれぞれ v_i の**開始ステップ**，**終了ステップ**といい，$\mathrm{SS}(v_i)$，$\mathrm{ES}(v_i)$ と書く☆． ■

定義3（**ライブレンジの長さ**） 変数のライブレンジ $v_i = [s_i, e_i)$ において，$e_i - s_i$ を v_i の**長さ**といい，$\mathrm{length}(v_i)$ と書く． ■

定義4（**ライブレンジ間の隙間**） 変数のライブレンジ v_i と v_j において，$\{\mathrm{SS}(v_j) - \mathrm{ES}(v_i)\} \bmod II$ を v_i から v_j までの**隙間**といい，$\mathrm{gap}(v_i, v_j)$ と書く． ■

図 4 の実線弧から分かるように，隙間は非対称，すなわち $\mathrm{gap}(v_i, v_j) \neq \mathrm{gap}(v_j, v_i)$ である．

2.3 基本方針

以下では，スライドレジスタ割付問題を Spiral Graph を用いて定式化する．コード生成には，大別して命令スケジューリングを行ってからレジスタ割付を行う方法とレジスタ割付を行ってからスケジューリングを行う方法がある[8]が，我々は前者を採用している．これは，スケジューリングが素直に行えるというスライドウィンドウの特徴を重視していることと，スケジューリングが定まらないとレジスタのリネーミング情報が定まらず，レジスタ割付を先に行うのに困難がともなうことによる．

本論文では，命令スケジューリング後の II やスライド量を含む情報をもとに占有レジスタ数を最小化することを考え，占有レジスタ数が最小になるとき，その割付が最適であるということにする．これは，占有

☆ 今後の一般化のため，単に s_i とせず，$s_i \bmod II$ としておく．

$$\text{minimize} \quad \sum_{i=0}^{N}\sum_{j=0}^{N}(c_{ij}\,\xi_{ij}) \tag{1.1}$$

$$\text{subject to} \quad \sum_{i=0}^{N}\xi_{ij}=1, \quad j=1,\ldots,N, \tag{1.2}$$

$$\sum_{i=0}^{N}\xi_{ip}=\sum_{j=0}^{N}\xi_{pj}, \quad p=0,\ldots,N, \tag{1.3}$$

$$y_i-y_j<N(1-\xi_{ij}), \quad i\neq j=1,\ldots,N, \tag{1.4}$$

$$\xi_{ij}\in\{0,1\}, \quad \forall i,\ j, \tag{1.5}$$

$$y_i \quad \text{arbitrary.} \tag{1.6}$$

制約系 1 巡回セールスマン問題の整数計画法での表現
ILP 1 Formulation of Traveling Salesman Problem.

レジスタ数が物理レジスタ数以下であるときに最適とする条件に比べて厳しく，奇妙に思えるかもしれない．我々がこのように定めているのは，効果的なレジスタ割付の結果レジスタに十分な空きができれば，将来的にこの情報をスケジューラにフィードバックすることによってループ・アンローリングなどの技法を適用してさらに高速なコードを生成できる可能性があり，またHARP-1Eにおいてはスライドレジスタの節約でグローバルレジスタ空間を広くあけることで，外周ループの変数をより多くレジスタ上に保持することが可能になるためである．

以下，巡回セールスマン問題と対比させながら，スライドレジスタ割付問題の定式化を行う．

3. 1-Spiral Graphと巡回セールスマン問題

本章では，まず簡単な場合として，1トラックのSpiral Graphに限定して話を進める．はじめに巡回セールスマン問題について簡単に説明し，次に，1-Spiral Graphへのレジスタ割付問題と巡回セールスマン問題との類似性について述べる．

3.1 巡回セールスマン問題

頂点集合（virtex set）$V=\{v_1,\ldots,v_N\}$ と基点（depot）v_0 に対し，v_i から v_j への移動コスト c_{ij} がたとえば**図5**の表のように与えられているとする．このとき，基点を出発して各頂点を1度ずつ訪問し，再び基点に戻るような閉路のうち，移動コストの和が最小になるものを求めよという問題を考える．この種の問題は，**巡回セールスマン問題**とよばれている[9]．図5のように $c_{ij}\neq c_{ji}$ であるものは，とくに**非対称な巡回セールスマン問題**とよばれる．

訪問順を総当たり法で調べてみると，この例題では $v_0\to v_1\to v_2\to v_3\to v_4\to v_0$ の順に巡回するのが最適であることが分かる．総コストを計算するために

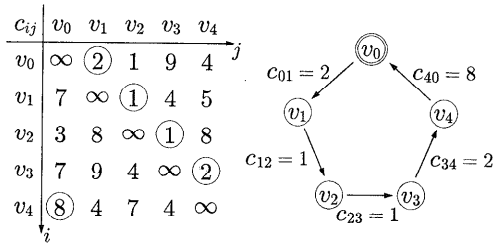


図5 巡回セールスマン問題としての最適解（Total cost = 14）
Fig. 5 Optimal solution for a Traveling Salesman Problem.

0-1変数 ξ_{ij} を考え，v_i の直後に v_j を訪問するとき1，そうでないとき0と定義する．図5中，○のついているところは $\xi_{ij}=1$，そうでないところは0である．総コストは $\sum_{i=0}^{N}\sum_{j=0}^{N}\xi_{ij}\,c_{ij}$ で計算でき，この場合，その値は14である．

巡回セールスマン問題は，整数計画法を用いて**制約系1**のように表現され，この制約系を解くことによって最適解を得ることができる[10]．一般に，整数計画法における制約式は左辺を定数とし，両辺の関係は $\leq$, $=$, $\geq$ のいずれかを用いて記述するのが普通であるが，当座は読みやすさを優先し，正式な記述は最後にゆずる．

式(1.1)が目的関数，(1.2)から(1.6)までが制約条件である．式(1.2)は，どの頂点もちょうど1回だけ訪問を受けることを示す．式(1.3)は，セールスマンがある頂点に到着したら，次はその頂点が出発点となることを示す．式(1.2)，(1.3)によって，移動コストの表の各列，各行に1つずつ○がつくことが保証される．しかし，これだけでは所望の解を得ることはできない．たとえば，**図6**のように○がついた場合には，v_3，v_4 が基点を出発するルートに含まれないからである．巡回セールスマン問題では，基点を含む閉路

c_{ij}	v_0	v_1	v_2	v_3	v_4
v_0	∞	(2)	1	9	4
v_1	7	∞	(1)	4	5
v_2	(3)	8	∞	1	8
v_3	7	9	4	∞	(2)
v_4	8	4	7	(4)	∞

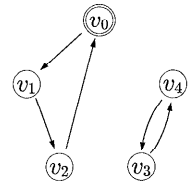


図 6 割当問題としての最適解（Total cost = 12）
Fig. 6 Optimal solution for an Assignment Problem.

（cycle）を**巡回路**（tour），基点を含まない閉路は一般には**部分巡回路**（subtour）とよばれる．巡回セールスマン問題の解はただ1つの巡回路をなしていなければならないから，得られた解に部分巡回路が含まれないようにしなくてはならない．式 (1.4) はこのための条件式であるが，詳しくは文献 10) にゆずる．式 (1.5) は整数条件である．

3.2 1-Spiral Graph と巡回セールスマン問題

図4を再び眺めよう．図4右が図4左のライブレンジを並べ換えたものであることから，1-Spiral Graph におけるレジスタ割付問題は，与えられたライブレンジを1本のトラック上へ適当な順序で巻き付け，占有レジスタ数を最小化せよという問題であることが分かる．これは与えられた都市を適当な順序で訪問し，移動距離を最小化せよという巡回セールスマン問題そのものである．スライドレジスタ割付問題を巡回セールスマン問題として解くためには，各ライブレンジ間について，頂点間の移動コスト c_{ij} に相当するものを定義しておく必要がある．便宜上，これをライブレンジ間の**連結コスト**とよぶことにする．連結コスト c_{ij} としては，ライブレンジ間の隙間 $\mathrm{gap}(v_i, v_j) = \{\mathrm{SS}(v_j) - \mathrm{ES}(v_i)\} \bmod II$ をそのまま利用することができる（たとえば，図2と定義 4 より $\mathrm{gap}(v_1, v_2) = 6$ であるから，$c_{12} = 6$ とする）．図4から，隙間の総和が最小になる配列のとき，占有レジスタ数もまた最小化されることが分かる☆．

ライブレンジ間の隙間の総和を最小化するといっても，ただ闇雲に巻き付けていくわけにはいかない．たとえば，図2において v_6 の開始ステップは5であることより，v_6 をトラック先頭に巻き付けるにあたっては，5ステップぶんの隙間をあけ，v_6 の開始ステップの正当性を保証しなければならない．そこで，トラック先頭に v_j を配置するときには，無条件に $\mathrm{SS}(v_j)$ の

☆ 図4の実線弧から分かるように隙間は非対称であるから，これは非対称な巡回セールスマン問題である．

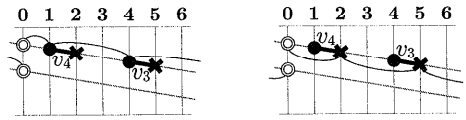


図 7 始点基準（左），終点基準（右）によるコスト定義
Fig. 7 Cost definition according to s_i (L) and e_i (R).

隙間をあけることと定める．トラック先頭のライブレンジ v_j の開始ステップが保証されれば，その終了ステップから $\mathrm{gap}(v_j, v_{j'})$ ステップの隙間をあけて次のライブレンジ $v_{j'}$ を配置する要領で，後続のライブレンジの開始ステップも保証することができる．

以上の定義は，基点に相当する仮想的なライブレンジ $v_0 \notin V$ を考え，さらに $\mathrm{ES}(v_0) = 0$ であると見なすと，定義 4 の v_0 に対する拡張として記述できる．

定義4′（ライブレンジ間の隙間） 変数のライブレンジ $v_i, v_j \in \{v_0\} \cup V$ に対して，v_i から v_j までの隙間 $\mathrm{gap}(v_i, v_j)$ を

$$\mathrm{gap}(v_i, v_j) \equiv \begin{cases} \{\mathrm{SS}(v_j) - \mathrm{ES}(v_i)\} \bmod II & \text{if} \quad (v_i \neq v_j) \\ \infty & \text{if} \quad (v_i = v_j) \end{cases}$$

と定義する． ■

以上のコスト定義に基づいて制約系1を解き，その並びに従ってレジスタ数を算出すれば占有レジスタ数を最小化できる．しかし，図4を再び眺めると，ライブレンジ自身の長さも含めて連結コストを定めておけば，トラック上の連結コストの総和を II で割るだけで占有レジスタ数を簡単に算出できることが分かる．そこで，占有レジスタ数が R 個のとき，連結コストの和が $R \cdot II$ になるようにコストの定義を与えておく．**図7**のように，ライブレンジ自身の長さを含めたコストの測り方には，始点間のステップ数を測る方法（図7左）と，終点間のステップ数を測る方法（図7右）の2種類が考えられるが，本論文では終点基準のみ説明することとし，始点基準については省略する．

定義5（連結コスト） 変数のライブレンジ $v_i, v_j \in \{v_0\} \cup V$ に対して，v_i から v_j までの**連結コスト** c_{ij} を

$$c_{ij} \equiv \begin{cases} \mathrm{SS}(v_j) + \mathrm{length}(v_j) & \\ \qquad \text{if} \quad (v_i = v_0) \wedge (v_j \in V)\ , & \\ \mathrm{gap}(v_i, v_j) + \mathrm{length}(v_j) & \\ \qquad \text{if} \quad (v_i \neq v_j \in V)\ , & \\ \{(II - \mathrm{ES}(v_i)) - 1\} \bmod II + 1 & \\ \qquad \text{if} \quad (v_i \in V) \wedge (v_j = v_0)\ , & \\ \infty \qquad \text{if} \quad (v_i = v_j) & \end{cases}$$

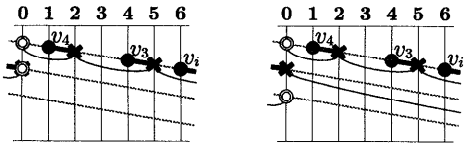


図 8 トラック末尾のライブレンジが 0 を終了ステップに持つ場合
Fig. 8 In case a track ends with v_i s.t. $\mathrm{ES}(v_i) = 0$.

と定義する. ■

定義 5 において, v_i と v_0 の連結コスト c_{i0} が $\{II - \mathrm{ES}(v_i)\} \bmod II$ でなく $\{(II - \mathrm{ES}(v_i)) - 1\} \bmod II + 1$ である理由は, $\mathrm{ES}(v_i) = 0$ のときを考えると明らかになる. 前者, 後者による定義をそれぞれ**図 8** の左右に示した. 前者の定義では, $\mathrm{gap}(v_i, v_0) = 0$ となるため, 占有レジスタ数 2 に対してコストが $1 \cdot II$ になってしまっている (図 8 左). 後者の定義では, $\mathrm{gap}(v_i, v_0)$ の範囲を $1 \leq \mathrm{gap}(v_i, v_0) \leq II$ にすることで, 占有レジスタ数 R に対する連結コストの和が $R \cdot II$ になるようにしている (図 8 右).

定義 5 は一見複雑に見えるが, v_0 を便宜上次のように定義すると, 実は非常に簡潔に書ける.

定義6 (基点 v_0) 変数のライブレンジの集合 $V = \{v_1, v_2, \ldots, v_N\}$ の他に仮想的なライブレンジ v_0 を考え,

$$v_0 = [II - 1,\ II)$$

と定義する. ■

このように定義すると, $\mathrm{ES}(v_0) = II$ より

$$\begin{aligned} c_{0j} &= \mathrm{SS}(v_j) + \mathrm{length}(v_j) \\ &= \{\mathrm{SS}(v_j) - II\} \bmod II + \mathrm{length}(v_j) \\ &= \{\mathrm{SS}(v_j) - \mathrm{ES}(v_0)\} \bmod II + \mathrm{length}(v_j) \\ &= \mathrm{gap}(v_0, v_j) + \mathrm{length}(v_j) \end{aligned}$$

さらに, $\mathrm{SS}(v_0) = II - 1$, $\mathrm{length}(v_0) = 1$ より

$$\begin{aligned} c_{i0} &= \{(II - \mathrm{ES}(v_i)) - 1\} \bmod II + 1 \\ &= \{(II - 1) - \mathrm{ES}(v_i)\} \bmod II + 1 \\ &= \{\mathrm{SS}(v_0) - \mathrm{ES}(v_i)\} \bmod II + \mathrm{length}(v_0) \\ &= \mathrm{gap}(v_i, v_0) + \mathrm{length}(v_0) \end{aligned}$$

であるから, 結局, 定義 5 は以下のように整理される.

定義5′ (連結コスト) 変数のライブレンジ v_i, $v_j \in \{v_0\} \cup V$ に対して, v_i から v_j までの**連結コスト** c_{ij} を

$$c_{ij} \equiv \mathrm{gap}(v_i, v_j) + \mathrm{length}(v_j)$$

で定める. ■

定義 5′ のコスト定義のもとで制約系 1 を解くと, 最小レジスタ数を与えるレジスタの巻き付け順が得られ, 総コストを II で割れば最小レジスタ数が得られる.

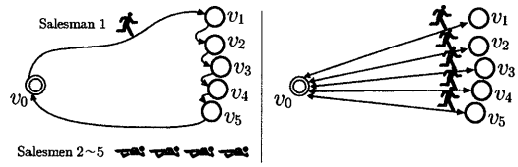


図 9 MinSUM 型 (左) と MinMAX 型 (右) の最適解
Fig. 9 MinSUM (L) and MinMAX (R) solution.

4. Spiral Graph と巡回セールスチーム問題

ここまで 1 トラックの Spiral Graph に限定して話をすすめてきたが, ここで話を一般化して, K トラックの Spiral Graph を扱う.

4.1 巡回セールスチーム問題

セールスマンが 1 人でなく, K 人いる問題を考える. この問題は Traveling Salesmen Problem あるいは Multiple Traveling Salesmen Problem などとよばれる☆. 我々は, これに**巡回セールスチーム問題**という訳語をあてる. 巡回セールスチーム問題には, MinSUM 型と MinMAX 型の 2 つの目的関数がある.

基点を出発してすべての都市をまわり, 再び基点に戻るセールスチームを想定する. **図 9** は, チームの 5 人のセールスマンが 5 つの都市を分担して訪問する例である. MinSUM 型の目的関数は, 燃費節約というモデルにたとえることができる. 各セールスマンが配送車を使うものとして, その燃費を節約するには 5 人の移動コストの総和を最小にすればよい. したがって, 図 9 のような都市の配置であれば, 1 人がすべての都市を担当し, 残りの 4 人は休むというのが最適になる. 一方, 燃費でなく, 配送を完了するまでの時間を節約しようとするのが MinMAX 型の目的関数である. このためには, 総コストは犠牲にして, すべての都市を 5 人のセールスマンで均等に分担して回収する, すなわち 5 人の移動コストのうち最大のものを最小化するのが最適になる. 通常巡回セールスチーム問題といえば前者をさすが, 本論文では後者を扱う. 理由は後述する.

制約系 2 は, MinSUM 型の巡回セールスチーム問題の整数計画法による記述である[11]. なお, ξ_{ijk} は, **セールスマン k が** v_i の直後に v_j を訪問するとき 1, そうでないとき 0 と定義された 0-1 変数である.

式 (2.*) が式 (1.*) に対応している. 式 (2.1) が目的関数, 式 (2.2) から (2.5) までが制約条件である. 式

☆ 定着した訳語はないようであるが, それぞれ, 巡回セールスメン問題, 巡回マルチセールスマン問題というところであろう.

$$\text{minimize} \quad \sum_{k=0}^{K}\left(\sum_{i=0}^{N}\sum_{j=0}^{N}(c_{ij}\,\xi_{ijk})\right) \tag{2.1}$$

$$\text{subject to} \quad \sum_{k=1}^{K}\sum_{i=0}^{N}\xi_{ijk}=1, \quad j=1,\ldots,N, \tag{2.2}$$

$$\sum_{i=0}^{N}\xi_{ipk}=\sum_{j=0}^{N}\xi_{pjk}, \quad k=1,\ldots,K, \;\; p=0,\ldots,N, \tag{2.3a}$$

$$\sum_{j=1}^{N}\xi_{0jk}=1, \quad k=1,\ldots,K, \tag{2.3b}$$

$$y_i-y_j<N\left(1-\sum_{k=1}^{K}\xi_{ijk}\right), \quad i\neq j=1,\ldots,N, \tag{2.4}$$

$$\xi_{ijk}\in\{0,1\}, \quad \forall i,j,k, \tag{2.5}$$

$$y_i \quad \text{arbitrary.} \tag{2.6}$$

制約系 2 MinSUM 型巡回セールスチーム問題の整数計画法での表現
ILP 2 Formulation of MinSUM Multiple Traveling Salesmen Problem.

$$\text{minimize} \quad z_0 \tag{3.1a}$$

$$\text{subject to} \quad \sum_{i=0}^{N}\sum_{j=0}^{N}(c_{ij}\,\xi_{ijk})\le z_0, \quad k=1,\ldots,K, \tag{3.1b}$$

(2.2)〜(2.6) に同じ (3.2)〜(3.6)

制約系 3 MinMAX 型巡回セールスチーム問題の整数計画法での表現
ILP 3 Formulation of MinMAX Multiple Traveling Salesmen Problem.

(2.2) は，どの頂点も，ちょうど **1 人のセールスマン**によって，ちょうど 1 回だけ訪問を受けることを示す．式 (2.3a) は，セールスマンがある頂点に到着したら，次はその頂点が出発点となることを示す．式 (2.3b) は 1 人のセールスマンが 2 つ以上の巡回路を受け持たないことを保証する．式 (2.4) は部分巡回路除去のための条件であり，すべての閉路が基点 v_0 を含むことを保証する．式 (2.5) は整数条件である．

我々の扱う問題は MinMAX 型であるから，MinSUM 型の表現をもとに，その表現を**制約系 3** に導いておく．ここで，式 (3.1b) は，実は目的関数の一部である．式 (3.1b) は，$k=1,\ldots,K$ のそれぞれについて z_0 が $\sum_{i=0}^{N}\sum_{j=0}^{N}(\xi_{ijk}\,c_{ij})$ 以上になるようにし，式 (3.1a) は z_0 を最小化しようとするから，z_0 は $\sum_{i=0}^{N}\sum_{j=0}^{N}(\xi_{ijk}\,c_{ij})$ の最大値に等しくなる．すなわち，式 (3.1a)，(3.1b) で

$$\text{minimize} \quad z=\max\left\{\sum_{i=0}^{N}\sum_{j=0}^{N}(\xi_{ijk}\,c_{ij}) \;\middle|\; k=1,\ldots,K\right\}$$

という最大コスト最小化の目的関数を表す．残りの制約式の意味は制約系 2 と同様である．

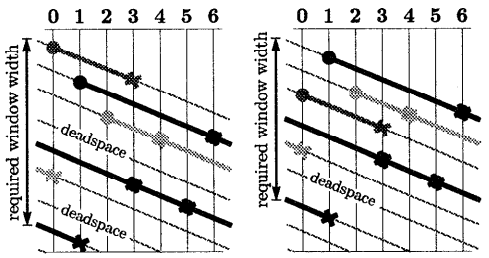


図 10 K トラックの Spiral Graph におけるデッドスペース
Fig. 10 Dead spaces in K-Spiral Graph.

なお，図 9 からも明らかなように，MinMAX 型の巡回セールスチームにおける全員の移動コストの総和は，一般に MinSUM 型のそれよりも大きい．

4.2 K-Spiral Graph と巡回セールスチーム問題

まず，K トラックの Spiral Graph と巡回セールスチーム問題との関係について述べる．**図 10** は 3 トラックの Spiral Graph であるが，各トラック上のライブレンジ配列を 3 人のセールスマンそれぞれの巡回路と見なせば，これは巡回セールスチーム問題である．そのため，巡回セールスチーム問題の制約系を用いてレジスタ割付を求めればよいのであるが，K-Spiral Graph において最適解を求めるにあたっては 2 つの

$$\text{minimize} \quad \sum_{k=1}^{K}(z_1 - II \cdot \delta_k) \tag{4.1a}$$

$$\text{subject to} \quad \sum_{i=0}^{N}\sum_{j=0}^{N}(c_{ij}\,\xi_{ijk}) = z_k, \quad k = 1, \ldots, K, \tag{4.1b}$$

$$z_k \leq z_{k-1}, \quad k = 2, \ldots, K, \tag{4.1c}$$

$$(2.2)\sim(2.6) \text{ に同じ} \tag{4.2}\sim(4.6)$$

$$\delta_k \leq z_1 - z_k, \quad k = 1, \ldots, K, \tag{4.7a}$$

$$\delta_k \in \{0, 1\}, \quad k = 1, \ldots, K. \tag{4.7b}$$

制約系 4 スライドレジスタ割付問題の整数計画法での表現
ILP 4 Formulation of Slide-Register Allocation Problem.

課題がある．その 1 つは K 本のトラックのコストがなるべく均等になるようにすること，もう 1 つは K 本のトラックのコストが（広義の）単調減少列となるようにすることである．図 10 は 3 トラックの Spiral Graph であり，コストが II, $2 \cdot II$, $3 \cdot II$ のトラックを含んでいる．このように，特定のトラック上にライブレンジが集中すると，トラック上に巻き付けたとき，間に無駄な空きレジスタを生じてしまう（図 10 左は占有レジスタ数はたしかに 6 であるが，その間の空きレジスタのため，幅 8 のレジスタウィンドウが必要となる）．そこで，以下の K-Spiral Graph における議論では，間の空きレジスタを含めた占有レジスタ数を**占有ウィンドウ幅**とよび，これを最小とする割付を**最適解**とよぶ．トラックのコストをなるべく均等にするには MinMAX 型の制約系を用いるのがよいが，それでも不均等を避けられなかった場合，それらのトラックはコストの大きい順に並べるのがよい．この理由は図 10 の左右を比較すれば自明である．これらの議論から，K-Spiral Graph におけるスライドレジスタ割付問題は**制約系 4** のように定式化できる．式 (4.2)～(4.6) の意味は，制約系 2 および制約系 3 と同様である．

式 (4.1a) が目的関数であり，式 (4.1b)，式 (4.1c) は目的関数の一部である．式 (4.1b) は，トラック k のコストを z_k で表すことを宣言している．目的関数および δ_k の意味については後で説明する．式 (4.1c) は K 本のトラックのコストを（広義の）単調減少列にするための条件式，式 (4.7a)，式 (4.7b) は δ_k の性質を決定するための条件式である．δ_k は $z_k < z_1$ のとき，かつそのときに限り 1 になる 0-1 変数であるが，δ_k がそのような性質を持つことは，式 (4.7a) および式 (4.1c) から簡単に示せる．z_k と z_1 との大小関係，およびそのときの δ_k の値について考えてみると，次の場合が考えられる．

- $z_k = z_1$
  式 (4.7a) より，$\delta_k = 0$ でなければならない．
- $z_k < z_1$
  $\delta_k = 0$, 1 の双方が式 (4.7a) を満たす．ところが，目的関数の部分式 $z_1 - II \cdot \delta_k$ について考えてみると，この部分式の値は $\delta_k = 0$ のとき z_1，$\delta_k = 1$ のとき $z_1 - II$ であり，δ_k が 0 であるとすると目的関数 (4.1a) は最小化されない．したがって，δ_k は 1 に束縛される．

以上より，δ_k は $z_k < z_1$ のとき，かつそのときに限り 1 になる 0-1 変数であることが示せた．

これらの意味をふまえて，目的関数について説明する．基本的には，各トラックのコストをなるべく均等にするように，制約系 3 のような MinMAX 型の目的関数とすればよいが，制約系 4 では間の空きレジスタを含めた占有ウィンドウ幅が計算できるようにしてある．式 (4.1c) から，図 10 右のようにトラックはコストの大きい順に並んでいる．各トラックのコスト z_k は，定義 5′ によって，II の倍数になるように定義されていることに注意されたい．間の空きレジスタを含めてレジスタ数を計算するには，トラックのコストが最大コスト z_1 に等しいものとそうでないものに分け，後者のコストを $z_1 - II$ と見積もればよい．式 (4.1a) の部分式 $z_1 - II \cdot \delta_k$ は $z_k = z_1$ のとき z_1，$z_k < z_1$ のとき $z_1 - II$ になるから，式 (4.1a) を最小化すれば間の空きレジスタを含めたレジスタ数，すなわち占有ウィンドウ幅が最小化されることになる．

以上により，定義 5′ のコスト定義のもとで制約系 4 を解くことによって，スライドレジスタ割付問題の最適解が与えられることが示された．

5. 実　　験

本章では，以上で定式化したスライドレジスタ割付問題を数理計画パッケージに投入し，線形緩和を用い

$$\text{minimize} \quad \sum_{k=1}^{K}(z'_1 - \delta_k) \tag{5.1a}$$

$$\text{subject to} \quad \sum_{i=0}^{N}\sum_{j=0,\, i\neq j}^{N}(c_{ij}\,\xi_{ijk}) - II \cdot z'_k = 0, \quad k = 1,\ldots,K, \tag{5.1b}$$

$$z'_{k-1} - z'_k \geq 0, \quad k = 2,\ldots,K, \tag{5.1c}$$

$$\sum_{k=1}^{K}\sum_{i=0,\, i\neq j}^{N}\xi_{ijk} = 1, \quad j = 1,\ldots,N, \tag{5.2}$$

$$\sum_{i=0,\, i\neq p}^{N}\xi_{ipk} - \sum_{j=0,\, j\neq p}^{N}\xi_{pjk} = 0, \quad k = 1,\ldots,K, \;\; p = 0,\ldots,N, \tag{5.3a}$$

$$\sum_{j=1}^{N}\xi_{0jk} = 1, \quad k = 1,\ldots,K, \tag{5.3b}$$

$$y_i - y_j + N\sum_{k=1}^{K}\xi_{ijk} \leq N-1, \quad i \neq j = 1,\ldots,N, \tag{5.4}$$

$$\xi_{ijk} \in \{0,1\}, \quad \forall i,j,k, \tag{5.5}$$

$$y_i \quad \text{arbitrary}, \tag{5.6}$$

$$z'_1 - z'_k - \delta_k \geq 0, \quad k = 1,\ldots,K, \tag{5.7a}$$

$$0 \leq \delta_k \leq 1, \quad k = 1,\ldots,K. \tag{5.7b}$$

制約系 5 実験に用いた制約系
ILP 5 Formulation used for the experiments.

る分枝限定法によって実際の例題を解く．まずは実際的な例題として，Livermore Fortran Kernel（LFK）を用いて問題サイズと厳密解法の求解時間との関係を調べた．本章の前半ではこの結果について報告する．

本厳密解法の直接的な適用対象は比較的小規模な問題に限られる．これは，本論文の手法がスライドレジスタ割付問題に（決定性）多項式時間解法を与えるものではないことによる．サイズの大きな問題に対しては近似解法を用いることになるのであるが，この近似解法の精度を評価したり，厳密解をもとに近似解法を精練するうえでも厳密解法は有用である．筆者らは論文 4) において Slide Coloring Algorithm および Short Bridge Algorithm の 2 つの近似解法を提案し，自動生成した 10,000 個のループプログラムを例題にその性能評価を行ったが，提案時は例題の厳密解を求めることが困難であったために正確な評価ができていない[4]．今回，本厳密解法を利用して例題の最適解をそれぞれ求め，論文 4) の近似解法を再評価する実験も行った．本章の後半ではこの結果について報告する．

5.1 実験条件

実験は，IBM RS/6000（Power2 160 MHz，主記憶 512 MB）上で，数理計画パッケージ CPLEX 6.0 を用いて行った．CPLEX は混合整数計画問題を分枝限定法で解くことができる．

実際に投入した制約系は，**制約系 5** である．式 (5.*) が式 (4.*) に対応している．式 (5.1a)，(5.1b)，(5.7b) が変更されているが，それ以外は制約系 4 の式を数理計画法の書式にかなうように単に変形したものである．式 (5.7b) では，整数変数を少なくするために δ_k の定義を 0-1 変数から連続変数に変更しているが，このことは制約系の意味自体には影響を及ぼさない．この説明は明らかなので省略する．式 (5.1a)，(5.1b) では，z_k のかわりに z_k を $1/II$ にした z'_k を用いている．z_k は II の倍数しかとらないから，z_k が II の倍数でない場合の検討を省くことで，分枝操作を少なくすることができる．また，z_k による定義では求まった目的関数値を II で割って占有ウィンドウ幅を求めていたが，z'_k による定義ではこの操作は不要である．

5.2 LFK における厳密解法の求解時間

本節では，論文 4) で用意した LFK のライブレンジ集合 55 例を用いて，厳密解法の求解時間を調べる．論文 4) で用いた実験用コンパイラは preload 命令を出力するため，付録に示す preload を考慮したコスト定義を用い，55 個の例題のそれぞれに対して制約系 5 の係数 c_{ij} を決定して CPLEX に投入した．

実験結果を図 **11** に示す．横軸には制約系の大きさ

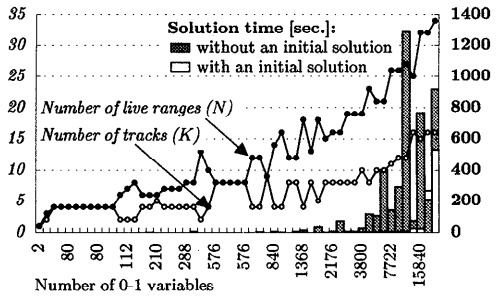


図 11 問題サイズと求解時間
Fig. 11 Problem size and solution time.

を表す指標として系に含まれる整数変数の数☆をとり，その小さい順に55個の例題を並べてある．黒い棒グラフは求解時間を示している（白い棒グラフについては後述する）．このうち最も長くかかったものは1,288.38秒（21′28″38）であった．

21分半というレジスタ割付時間は一般の用途には長過ぎるが，分枝限定法には1つ実行可能解が見つかると以後の限定操作を効果的に行えるという性質があるため，初期解を投入することでより早く最適解を求めることが可能である．初期解は実行可能解でありさえすればよいが，限定操作への寄与という面からは精度の良い近似解であることが望ましい．そこで，近似解法の1つであるShort Bridge Algorithmを前段に利用し，得られた解を初期解として投入した．図11の白い棒グラフは，投入した近似解の精製を試みた場合の求解時間を示したものである．このうち最も長くかかったものは530.37秒（8′50″37）であり，初期解を与えない場合に比べておおむね早く解きおわっている．大きい方から2番目の例題は初期解を与えたことによってかえって時間がかかっているが，これは，この例題の初期解として与えた近似解が不適切であり，最適解にくらべて3レジスタ分冗長だったことによる．また，最長の求解時間を与えた例題では，近似解が最適解より2レジスタ分冗長であった．これらのことから，初期解として与える近似解はそれなりに精度の良いものでなければならないことがうかがえる．

以上より，厳密解法を実用で使う際は，できるだけ精度の良い近似解法を前段に用い，問題の重要度に応じたタイムリミットを設けて厳密解法による精製を試みるのが実用的と考えられる．タイムリミット t はオプションなどで与えるものとし，デフォルトを0秒としておけば，通常のコンパイルでは近似解法のみが使われることになるから，この機構は従来のコンパイラの上位拡張と見なすことができる．時間をかけるに値する特に重要なループでは，その重要度に応じて t を大きくすればよい．この場合，厳密解法はタイムリミットで停止するか，それ以前に停止するかのいずれかである．分枝限定法には時間の経過とともにより良い実行可能解が次々に見つかっていくという性質があるため，前者の場合には（最適解であることの保証はないものの）少なくとも初期解と同じかそれ以上の精度の解が得られていることになり，後者の場合には最適解が得られていることになる．

5.3 厳密解法による近似アルゴリズムの再評価

本節では，本厳密解法を用いて近似解法の最適性を再評価する．論文4)では，スライドレジスタ割付の近似解法としてSlide Coloring AlgorithmおよびShort Bridge Algorithmを提案し，ループプログラム自動生成ツールによって生成した10,000例のループプログラムを用いてその評価を行った（生成されたプログラムの例や分布などについては論文4)を参照されたい）．論文4)では，近似アルゴリズムの最適性はレジスタ数の下界である W_{max} 値[3),4)]との比較によって評価している．総当たり法で10,000例のサンプルすべてに対して最適解を求めることは事実上不可能であるが，$W_{max} \leq$ 最適解 $\leq$ 近似解 の関係を利用すれば，$W_{max} =$ 近似解 のときははさみうちの定理により近似解の最適性を保証できる寸法である．しかしこの方法では，$W_{max} \neq$ 近似解 であるときに，近似アルゴリズムが冗長な割付をしているのか，W_{max} が下界として過小評価であるのかを判断することができない．そのため，論文4)ではSlide Coloring Algorithmで42%，Short Bridge Algorithmで55%の最適性を保証するにとどまったが，本厳密解法を用いてある程度効率的に最適解を求め得るなら，はさみうちの定理によらず，つぶさに近似解法の最適性を評価できるはずである．そこで，10,000例の例題すべてについて本厳密解法を適用し，近似解法の再評価を試みた．期間的な制約から，Short Bridge Algorithmによる近似解を初期解として与えたうえ，10分のタイムリミットを設けて本厳密解法を試み，求めた最適解をもとに近似解法の再評価を行った．その結果を述べる前に，本厳密解法の求解時間について簡単に触れる．**表1**は，横方向に時間，縦方向に変数のライブレンジの数をとり，求解時間の分布をまとめたものである．変数のライブレンジの数が多くなるほど，求解数のピークが時間の長い方にシフトしていることが分かる．最下行に，そ

☆ 制約系5の整数変数は0-1変数 ξ_{ijk} $(i \neq j)$ だけであるから，その数は $N(N+1) \cdot K$ である．

表 1 厳密解の求解時間
Table 1 Solution time by the exact method.

Number of Liveranges	Solution time [sec.] ~ 0.1	~ 0.5	~ 1	~ 5	~ 10	~ 30	[min.] ~ 1	~ 3	~ 5	~ 10	~ ∞
1 ~ 15	2,500	1,125	55	58	10	19	2	2	3	5	52
16 ~ 30	82	354	655	1,558	505	292	75	76	24	39	333
31 ~ 45	0	32	11	15	214	697	399	209	18	20	296
46 ~ 60	0	0	0	2	0	1	25	118	51	25	43
Solved	2,582	1,511	721	1,633	729	1,009	501	405	96	89	724
examples	**2,582**	**4,093**	**4,814**	**6,447**	**7,176**	**8,185**	**8,686**	**9,091**	**9,187**	**9,276**	**10,000**

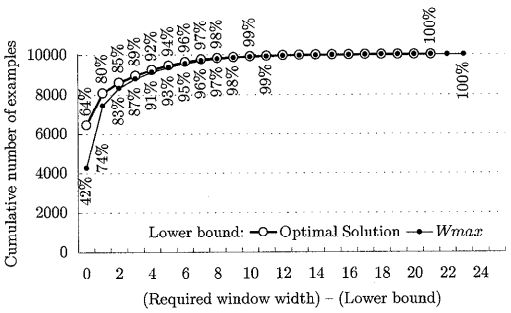


図 12 スライド彩色アルゴリズムの最適性
Fig. 12 Optimality of Slide Coloring Algorithm.

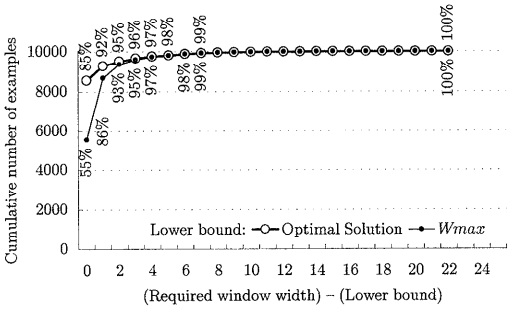


図 13 Short Bridge Algorithm の最適性
Fig. 13 Optimality of Short Bridge Algorithm.

の時間までに解けた例題数を累積で示した．3 分以内に 9 割以上，10 分のタイムリミット内では 9,276 題が解けている．

近似解法の再評価結果を**図 12** および**図 13** に示す．横軸は近似解の要したレジスタ数から下界として厳密解もしくは W_{max} を引いたものである．この値が 0 のときはただちに最適性が保証されるから，厳密解による評価より Slide Coloring Algorithm では 64%，Short Bridge Algorithm では 85%が最適解であり，W_{max} との比較による以前の結果が過小評価であったことがいえる．下界のとり方による近似アルゴリズムの最適性評価の違いを**表 2** にまとめた．724 題の未解決問題について最適性の判定が保留されているため 7%強のゆらぎがあるが，厳密解法によって，より的確に最適性の評価が行えていることが分かる．

実験室で厳密解法を試みることの利点は，近似解法の最適性評価にとどまらない．下界にはつねに過小評価の可能性があるため，たとえばその値が 6 だとしても，幅 6 のウィンドウに収めようと近似解法の改良を試みることが無意味な場合がある．厳密解法によれば，最小ウィンドウ幅とともに具体的な割付例も得られるため，これらを参考にした近似解法の精錬も可能になる．

表 2 近似アルゴリズムの最適性
Table 2 Optimality of approximation algorithms.

Lower bound for the evaluation	Algorithm	
	Slide Coloring	Short Bridge
W_{max}	42% ~	55% ~
Optimal solution	64 ~ 72%	85 ~ 93%

6. ま と め

本論文では，スライドレジスタの割付問題を，巡回セールスマン問題に類似した問題として整数計画法で定式化した．このことにより，GA などの巡回セールスマン問題の既知の解法をスライドレジスタ割付へ応用することも期待できる．また，厳密解法の応用例として実際的な問題を解く一方で，近似解法の評価や精練にも応用可能であることを示した．

謝辞 巡回セールスマン問題をはじめとする数理計画の分野をご指導いただいた筑波大学社会工学系の山本芳嗣教授に感謝したい．また，超並列計算機 CP-PACS の開発，製作に協力いただいている（株）日立製作所に謝意をあらわしたい．CP-PACS プロジェクトは文部省科学研究費補助金（創成的基礎研究）07NP0401 による．

参考文献

1) 中澤喜三郎，中村 宏，朴 泰佑：超並列計算機CP-PACSのアーキテクチャ，情報処理，Vol.37, No.1, pp.18–28 (1996).
2) Beck, G.R., Yen, D.W.L. and Anderson, T.L.: The Cydra 5 Minisupercomputer: Architecture and Implementation, *J. Supercomput.*, Vol.7, No.1/2, pp.143–180 (1993).
3) Hendren, L.J., Gao, G.R., Altman, E.R. and Mukerji, C.: *A Register Allocation Framework Based on Hierarchical Cyclic Interval Graphs*, LNCS, Vol.641, pp.176–191, Springer-Verlag (1992).
4) 秡川友宏，添野元秀，山下義行，中田育男：スライドウィンドウを考慮したレジスタ割付，情報処理学会論文誌，Vol.39, No.9, pp.2684–2694 (1998).
5) 位守弘充，中村 宏，朴 泰佑，中澤喜三郎：スライドウィンドウ方式による疑似ベクトルプロセッサ，情報処理学会論文誌，Vol.34, No.12, pp.2612–2623 (1993).
6) Hewlett Packard: *PA-RISC 1.1 Architecture and Instruction Set Reference Manual* (1990).
7) 日立製作所：ハードウェア・オペレーティング・マニュアル SR2201 並列プロセッサ.
8) Lam, M.: Software Pipelining: An Effective Scheduling Technique for VLIW Machines, *Proc. SIGPLAN '88 Conf. on PLDI*, pp.318–328 (1988).
9) 山本芳嗣，久保幹男：巡回セールスマン問題への招待，現代人の数理 12，朝倉書店 (1997).
10) Miller, C., Tucker, A.W. and Zelmin, R.A.: Integer Programming Formulation of the Traveling Salesman Problem, *J. ACM*, Vol.7, No.4, pp.326–329 (1960).
11) Christofides, N., Mingozzi, A. and Toth, P.: Exact Algorithms for the Vehicle Routing Problem, Based on Spanning Tree and Shortest Path Relaxations, *Mathematical Programming*, Vol.20, No.3, pp.255–282 (1981).

付 録

preload/poststore 命令への対応

スライドウィンドウ・アーキテクチャはスライドレジスタを活用するための preload/poststore 命令も特徴に持つ．ここでは，ごく簡単にではあるが，これらの命令も視野に入れた定式化について述べる．筆者らは，論文 4) において，Spiral Graph によるレジスタ割付フレームワークを preload 命令に対応させるためには，preload を考慮したコストの定義を与えるだけでよく，アルゴリズムには特に手を入れる必要のないことを示した．本論文の定式化は Spiral Graph の考え方に基づくものであるので，やはりコスト定義の変更だけで preload/poststore 命令に対応させることができる．

一般の命令ではレジスタウィンドウ内のレジスタしか操作対象にできないが，preload/poststore 命令はウィンドウ外のレジスタも操作対象にできる．そのため，ライブレンジの記述を，ウィンドウ内になくてはならない区間 $[s_i, e_i)$ と，ウィンドウ内になくてもよい前後の区間とに分けて考えられるように拡張する．

定義7（生成点，消滅点） 変数のライブレンジ $v_i = [s_i, e_i)$ を，次の条件で 4 項からなる表現 $v_i = [c_i, s_i, e_i, d_i)$ に拡張する．

$$c_i \equiv \begin{cases} s_i\text{から遡った preload 命令の発行時刻} \\ \qquad (v_i \text{ が preload 変数のとき}) \\ s_i \quad (v_i \text{ が preload 変数でないとき}) \end{cases}$$

$$d_i \equiv \begin{cases} s_i\text{から数えた poststore 命令の発行時刻} \\ \qquad (v_i \text{ が poststore 変数のとき}) \\ e_i \quad (v_i \text{ が poststore 変数でないとき}) \end{cases}$$

c_i, d_i をそれぞれ変数 v_i の**生成点**，**消滅点**とよぶ．また，$c_i \bmod II$, $d_i \bmod II$ をそれぞれ $\mathrm{CS}(v_i)$, $\mathrm{DS}(v_i)$ と書き，v_i の**生成ステップ**，**消滅ステップ**とよぶ． ■

定義6′（基点 v_0） 仮想的なライブレンジ v_0 を，

$$v_0 = [II - 1,\ II - 1,\ II,\ II)$$

に拡張する． ■

定義3′（ライブレンジの長さ） 変数のライブレンジ $v_i = [c_i, s_i, e_i, d_i)$ において，

$$\mathrm{prelength}(v_i) = s_i - c_i,$$
$$\mathrm{length}(v_i) = e_i - s_i,$$
$$\mathrm{postlength}(v_i) = d_i - e_i$$

と定義する． ■

定義4″（ライブレンジ間の隙間） 変数のライブレンジ $v_i, v_j \in \{v_0\} \cup V$ に対して，v_i から v_j までの隙間 $\mathrm{gap}(v_i, v_j)$ を

$$\mathrm{gap}(v_i, v_j) \equiv \begin{cases} \{\mathrm{SS}(v_j) - \mathrm{DS}(v_i)\} \bmod II \\ \qquad \text{if} \quad (v_i = v_0) \wedge (v_j \in V) \\ \{\mathrm{CS}(v_j) - \mathrm{DS}(v_i)\} \bmod II \\ \qquad \text{if} \quad (v_i \neq v_j \in V)\ , \\ \{\mathrm{CS}(v_j) - \mathrm{ES}(v_i)\} \bmod II \\ \qquad \text{if} \quad (v_i \in V) \wedge (v_j = v_0) \end{cases}$$

と定義する． ■

定義5″（連結コスト） 変数のライブレンジ v_i, $v_j \in \{v_0\} \cup V$ に対して，v_i から v_j までの**連結コ**

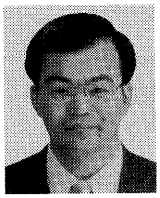
スト c_{ij} を

$$c_{ij} \equiv \text{postlength}(v_i) + \text{gap}(v_i, v_j) + \text{length}(v_j) + \text{prelength}(v_j)$$

で定める． ■

以上の定義のもとで制約系4を解くと，preload/poststore 命令を含むループプログラムに対する最適解が得られる．なお，これらの定義は，両命令を含まないプログラムに対してもそのまま用いることができる．

(平成 10 年 12 月 25 日受付)
(平成 11 年 7 月 1 日採録)

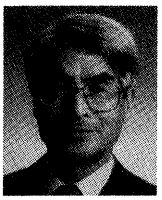


秡川 友宏（学生会員）

1970 年生．1995 年筑波大学第 1 学群自然学類卒業．同年同大学大学院工学研究科入学．計算機アーキテクチャ，コード最適化等に興味を持っている．日本ソフトウェア科学会学生会員．

山下 義行（正会員）

1959 年生．1982 年大阪大学理学部物理学科卒業，日立マイクロコンピュータ・エンジニアリング（株）入社．1986 年退社．1987 年筑波大学大学院博士課程工学研究科電子・情報工学専攻入学．1989 年退学，東京大学大型計算機センター助手．1992 年筑波大学電子・情報工学系講師．1995 年～同助教授．工学博士．プログラミング言語，コンピュータ・グラフィクスの研究に従事．日本ソフトウェア科学会会員．



中田 育男（正会員）

1935 年生．1958 年東京大学理学部数学科卒業．1960 年同大学大学院修士課程修了．1960～1979 年（株）日立製作所中央研究所システム開発研究所勤務．1979～1997 年筑波大学電子・情報工学系教授．1997 年～図書館情報大学図書館情報学部教授．理学博士．筑波大学名誉教授．プログラム言語，言語処理系，ソフトウェア工学等に興味を持っている．著書「コンパイラ」（産業図書），「基礎 FORTRAN」（岩波書店），「コンパイラ」（オーム社）．日本ソフトウェア科学会，電子情報通信学会，ACM，IEEE 各会員．