Title	繰り返し分割再配置に基づく２次元配置最適化
Author(s)	金子，哲
Citation	
Issue Date	2002-03
Type	Thesis or Dissertation
Text version	author
URL	http://hdl.handle.net/10119/1522
Rights	
Description	Supervisor:金子 峰雄，情報科学研究科，修士

修 士 論 文

繰り返し分割再配置に基づく２次元配置最適化

北陸先端科学技術大学院大学
情報科学研究科情報システム学専攻

金子 哲

2002 年 3 月

修 士 論 文

繰り返し分割再配置に基づく２次元配置最適化

指導教官　金子 峰雄 教授

審査委員主査　金子 峰雄 教授
審査委員　宮地 充子 助教授
審査委員　浅野 哲夫 教授

北陸先端科学技術大学院大学
情報科学研究科情報システム学専攻

010032 金子 哲

提出年月: 2002 年 2 月

目次

図 目 次

表目次

第1章　はじめに

1.1　背景

VLSIは計算機，通信機器を始めとする様々な情報処理，信号処理システムの主要構成要素であり，その製造技術の進歩により伴い，回路規模の増大および微細化が急速に進んでいる．また，低消費電力化，小型化，高速化等の要求とも相伴い，VLSIの設計は非常に困難な問題となってきている．一方，多品種少量生産や設計期間の短縮などの要求も高まり，設計の自動化は重要な課題となっている．

VLSI設計の工程はシステム設計，RTL設計，論理設計，回路設計，そしてレイアウト設計と階層的に大きく分類することができる．このなかでレイアウト設計工程では，チップ上にコンポーネントを配置し，それらの間の配線を行う工程で，多くの場合，チップ面積の最小化や総配線長の最小化が要求される．こうした要求に対して様々な評価に基づく最適化問題がNP困難であることが知られており，大規模な問題を実用的な時間で解くことのできる効率的な手法が求められている．

一般にNP困難と言われている問題に対して，大域的な最適解を探索するアルゴリズムとしては，Simulated Annealing(SA)法等の確率的最適化手法が提案されている．SA法は現在の解からの隣接解の生成と，“温度”と呼ばれる制御パラメータを用いてその解の 受理/棄却 判定を繰り返す手法で，十分な温度スケジューリングを行うことで最適解に到達することが保証されているアルゴリズムである．しかし，最適解を得るためにはアルゴリズム内部で多数の繰り返し処理を必要とするため，その計算時間は膨大なものとなり，大規模な問題を実用的な時間で解くには適していない．SA法はVLSIのレイアウト設計における配置問題に対しても適用されているが，年々，回路規模の増大に伴い，その適用は困難になってきている．

近年，1次元配置問題に対してSA法と同等の解をより高速に得ることが可能なSimulated Quenching(SQ)法[1]が提案された．SQ法はスロットが 幅 W × 高さ1で一列に並ぶスロットアレイをいくつかのサブグループへ分割し，複数のサブグループにまたがるネットに接続するコンポーネントに対してフォース値を与え，サブグループ内でその値に基づいてコンポーネントをスロットへ再配置する，という3つの操作を繰り返すことで解の最小化を行う手法である．

また，[2]では，SQ法の成功が（i）スロットとコンポーネントのサブグループ化,（ii）各サブグループに対する部分問題の構成,（iii）部分問題における再配置手法にあるとの見地から，SQ法の2次元配置問題への拡張が試みられている．そこでは，各ネットの配線

長を厳密に反映させたステップコストを用い，サブグループの一括再配置を実現することにより，総配線長評価では，一般的な実装のSA法とほぼ同等の解を得ることのできる手法を提案している．しかし，このアルゴリズムでは部分問題における再配置手続きの中で枝重み付き2部グラフの最小重み完全マッチングを求めているため，実行に多大な時間を要している．

1.2 提案手法

本稿では上記（i）,（ii）,（iii）及び[2]の結果をふまえ，総配線長評価の下でSAとほぼ同等の解をより高速に得るアルゴリズムを提案する．部分問題における目的関数としてはフォース値に基づいくコスト関数とステップコストに基づくコスト関数のそれぞれに対するいくつかの再配置手法を提案する．また，提案する手法の有効性を確かめるための実験とその結果を報告する．

1.3 本稿の構成

以降，第2章では前準備として2次元配置問題について説明し，その後，SQ法とSA法について説明する．第3章ではフォース値に基づいた手法について提案し，第4章ではステップコストに基づいた手法を提案する．第5章では第4章で得られた結果を元にそのさらなる可能性について試行する．最後に第6章でまとめと今後の課題を述べる．

第2章　準備

2.1　2次元配置問題について

本稿にて取り扱う2次元配置問題において，回路をコンポーネント（component）集合Vとそれらを接続するネット（net）集合$E = \{E_1, E_2, ...\}$で表されるハイパーグラフ$G = (V, E)$でモデル化する．各コンポーネントはスロット（slot）を 幅$W \times$高さHで2次元格子状に並べたスロットアレイSに割り当てるものとする．問題に入力及び出力を以下のようにする．

入力： スロットアレイSとハイパーグラフ$G = (V, E)$

出力： 全てのVについて全てのSへの重複のない割り当て

ある割り当てにおいて，ネットに含まれるコンポーネントをすべて含む矩形のうち，最小のもの，つまりバウンディングボックス（Bounding Box）の半周長をそのネットの配線長（length）とし，全てのネットについての総配線長の最小化を目標とする．$x(v)$，$y(v)$をそれぞれコンポーネントvの配置されているスロットのx, y座標としてネットの総配線長の評価式を表すと次のようになる．

$$\sum_{E_i \in E} \Big((\max\{x(v_j) | v_j \in E_i\} - \min\{x(v_k) | v_k \in E_i\}) + (\max\{y(v_l) | v_l \in E_i\} - \min\{y(v_m) | v_m \in E_i\}) \Big) \tag{2.1}$$

2.2　Simulated Quenching法

ここでは1次元配置問題に対するSQ法について説明する．

2.2.1　1次元配置問題

SQ法で取り扱う1次元配置問題において，回路をコンポーネント集合Vとそれらを接続するネット集合Eで表されるハイパーグラフ$G = (V, E)$でモデル化する．各コンポーネントはスロットを 幅$W \times$高さ1 で一列に等間隔に並べたスロットアレイSに割り当てるものとする．問題における入力及び出力を以下のようにする．

入力： スロットアレイ S とハイパーグラフ $G = (V, E)$

出力： 全ての V について全ての S への重複のない割り当て

各ネットの配線長はそのネットに含まれる最左端のコンポーネントと最右端のコンポーネントの間の距離とし，全てのネットについての総配線長の最小化を目標とする．

2.2.2 SQ 法のアルゴリズム

１次元配置問題に対する SQ 法は,

1. スロットとコンポーネントのサブグループ化：
   スロットアレイ上にピッチ（pitch）p の間隔でカットライン（cutline）を引く．但し，最左端のカットラインの位置をオフセット（offset）o とする．o には 0 以上 p 未満の値をランダムに採用する．隣り合うカットラインの間に存在するスロットとコンポーネントをサブグループ（subgroup）と呼ぶ.
2. フォース値の設定：
   カットラインを跨ぐネットに対し，そのネットの最も 左端/右端 のコンポーネントにそれぞれ $+1/-1$ のフォース値を与える．各コンポーネントごとに全てのネットについて与えられたフォース値を合計したものをそのコンポーネントの総フォース値（total force value）とする．
3. 再配置：
   各サブグループ内で総フォース値に基づいてコンポーネントを昇順にソートしたものを新しい配置とする．

以上の３つの操作を p をスロット幅 W から 2 になるまで $0.03 * p/\log_2 p$ ずつ減少させながら，各ピッチで一定回数ずつ繰り返す手法である．

図 2.1 は $p = 3$ におけるフォース値の例を示している．図において正方形のマスはスロットを表し，その中のアルファベットがふられた小さな四角形はコンポーネントを表している．太線はカットラインを表し，更に円と破線の組み合わせがネットを表す．$net1$ と $net3$ についてはカットラインを跨いで存在しており，その両端のコンポーネントに対してフォース値が各ネットの円の上に示されるように与えられる．これらの値をコンポーネント毎に総和して得られた総フォース値を各コンポーネントの上部に示す．図 2.2 では図 2.1 で得られた総フォース値に基づいて，サブグループ内でコンポーネントをソートしている．

各サブグループ内のコンポーネントの再配置を部分問題として考えると，以下の特徴を有している．

- 部分問題は原問題の忠実な部分問題とはなっていない．再配置のための評価関数がカットラインを跨ぐネットの両端のコンポーネントのみに依存するため，１つのサブグループ内に閉じたネットに対しての配線長は無視されている．

- p が小さくなるにつれて，SQ の部分問題と総配線長を評価関数とする忠実な部分問題との差は縮まっていき，$p = 2$ では等価になる．

- 部分問題を解く高速なアルゴリズムが存在する．

2.2.3　SQ 法のプログラム

SQ 法のプログラムを以下に示す．ここで Loop は同一ピッチでの繰り返し回数を表す．

```
begin
pitch = W;
Do{
    i = 0;
    Do{
        サブグループ生成;
        各コンポーネントについて総フォース値を計算;
        サブグループ内でコンポーネントの並びをソート;
        i++;
    }while(i<Loop);
pitch -= 0.03 × ( pitch/log2(pitch) );
}while(p >= 2.0);
end
```

2.3　2 次元配置問題に対する Simulated Annealing

SA は確率的な最適化手法の一つである．アニーリング（焼き鈍し）とは，固体の温度を十分高くして液状にした後，温度をゆっくり下げることにより，その固体の単結晶を得るという物理的なプロセスのことで，SA では，計算機中でこのアニーリングを擬似的に行うことで，与えられた問題に対して設定されたコスト関数の大局的な最小点を見つけ出す。

2 次元配置問題に対する SA では，次のような動作を温度パラメータ T をある値から十分小さな値になるまで少しずつ T を減少させながら繰り返すことで配線長の最小化を行う．

1. コンポーネントの選択：
   ランダムに 2 つの異なるコンポーネントを選択し，式 2.1 を用いてそれらの割り当てスロットの入れ替え前の評価値 E_0 と入れ替え後の評価値 E_1 の変化量 $\Delta E = E_1 - E_0$ を求める．

2. 入れ替えの受理判定：
   ΔE と温度パラメータ T に基づき次式のような確率 P でその入れ替えを受理する否かの判定をする．

$$P(\Delta E) = \begin{cases} 1 & \text{if } \Delta E < 0 \\ \exp(-\frac{\Delta E}{T}) & otherwise \end{cases} \tag{2.2}$$

2 次元配置問題に対する SA のプログラムを以下に示す．温度 T は Start_temperature から開始する．各温度で Loop 回処理を行った後 T を Decrease の割合で減少させていき，End_temperature になるまで処理を続ける．

```
begin

T = Start_temperature;
while( T > End_temperature ){
    i = 0;
    while( i < Loop ){
        交換前の評価値 E0 を計算;
        ランダムに 2 つのコンポーネントを選択肢割り当てスロットを交換する;
        交換後の評価値 E1 を計算;
        dE = E1 - E0;
        if( dE >= 0 ){
            if( 0 から 1 の間の乱数 > P(dE) ){
                交換を受理せず元に戻す;
            }
        }
    i++;
    }
    T *= Decrease;
}

end
```

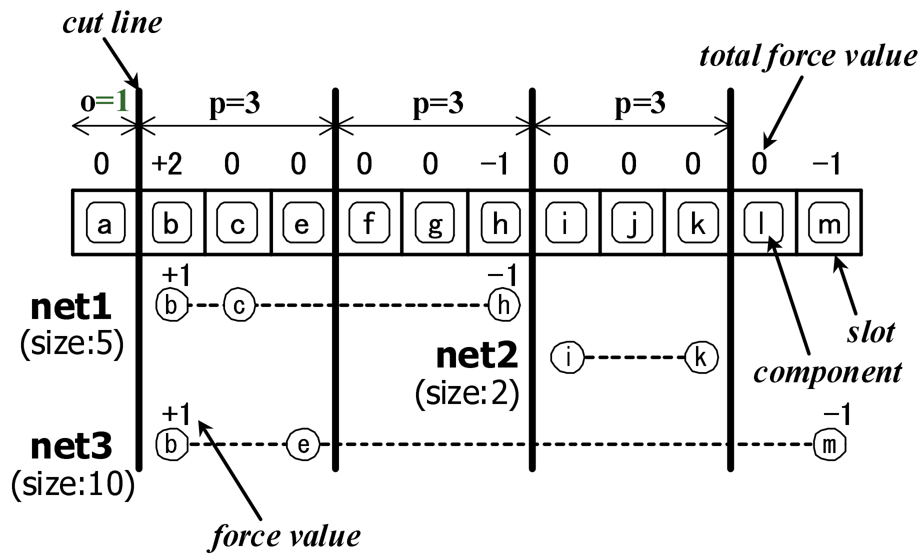


図 2.1: SQ におけるフォース値の与え方

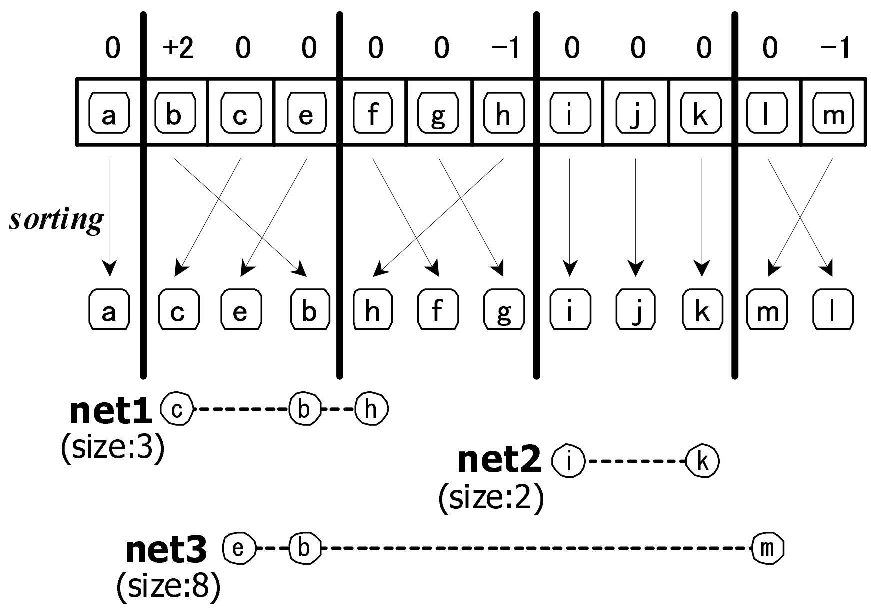


図 2.2: SQ における再配置

第3章　フォースに基づく手法

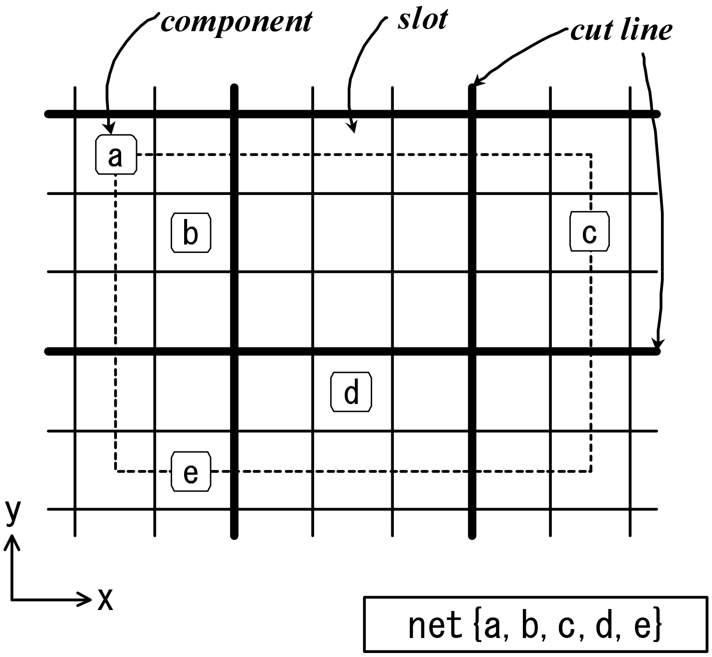
3.1　アルゴリズムの概要

1次元配置問題に対するSQ法を拡張して，2次元配置問題へ適用する．アルゴリズムは，ピッチ p を $\max\{W, H\}$ から1になるまで $0.03 * p/\log_2 p$ ずつ減少させながら，各ピッチで以下の処理を一定回数繰り返し行う．

1. サブグループ化：
スロットアレイ上にピッチ p の間隔で 垂直方向/水平方向 カットラインを引く．但し，最左端のカットラインの位置を水平方向オフセット o_h，最下端のカットラインの位置を垂直方向オフセット o_v とする．o_v，o_h には0以上 p 未満の値をランダムに採用する．カットラインで区切られた領域に存在するスロットとコンポーネントをサブグループと呼ぶ．
2. フォース値の決定：
水平方向に関して，垂直カットラインを跨ぐネットに対し，そのネットに接続し， 左端/右端 のサブグループに属するコンポーネントにそれぞれ $+1/-1$ の水平方向フォース値 を付加する．同様に垂直方向に関しても，水平カットラインを跨ぐネットに対し，そのネットに接続し， 下端/上端 のサブグループに属するコンポーネントにそれぞれ $+1/-1$ の垂直方向フォース値を付加する．
各コンポーネントに対し，全てのネットについて与えられた 水平方向/垂直方向 のフォース値をそれぞれ合計したものを水平方向総フォース値（F_h），垂直方向総フォース値（F_v）とする．
3. 再配置：
各サブグループ内で水平方向総フォース値と垂直方向総フォース値に基づいて再配置を行う．再配置手法として水平方向と垂直方向の再配置を逐次処理するFVS（<u>F</u>orce <u>V</u>alue based replacement by <u>S</u>orting）法と2次元再配置をグリーディに行うFVCH（<u>F</u>orce <u>V</u>alue based replacement with <u>C</u>onvex <u>H</u>ull）法を試みた．

フォース値算出の例として，図3.1を考える．ここで正方形のマスはスロットを，その中のアルファベットがふられた小さな四角形はコンポーネントを，太線はカットラインをそれぞれ表している．今，ネット $\{a, b, c, d, e\}$ に対し，図に示すカットラインが引かれたときの各コンポーネントの 水平方向フォース値/垂直方向フォース値 を図3.1下側に示す．

以降，FVS法とFVCH法のそれぞれにおける再配置手法について述べる．



	a	b	c	d	e
水平方向フォース値	+1	+1	-1	0	+1
垂直方向フォース値	-1	-1	-1	+1	+1

図 3.1: フォース値の例

3.2 FVS 法

FVS 法は再配置を水平方向と垂直方向で逐次処理する手法である．具体的には，図 3.2 のように各サブグループにおいて，同一 y 座標のスロットに割り当てられているコンポーネント集合を考え，各コンポーネントの水平方向総フォース値に従って昇順にソートし，その結果の順にスロットに再割り当てを行う．引き続いて垂直方向についても同様の処理を行う．

3.3 FVCH 法

FVCH 法は各コンポーネントの F_h と F_v の大きさに従ってコンポーネントを 1 つずつ取り出し，サブグループ内の各空きスロットへ割り当てたときの評価の一番高いスロットにそのコンポーネントを割り当てるグリーディな手法である．取り出したコンポーネントについて座標（x, y）のスロットでの評価を $F_h \times x + F_v \times y$ とする．評価値が最大となるスロットはサブグループ内の空きスロットに対する凸包（Convex Hull）[5][6] の境界線

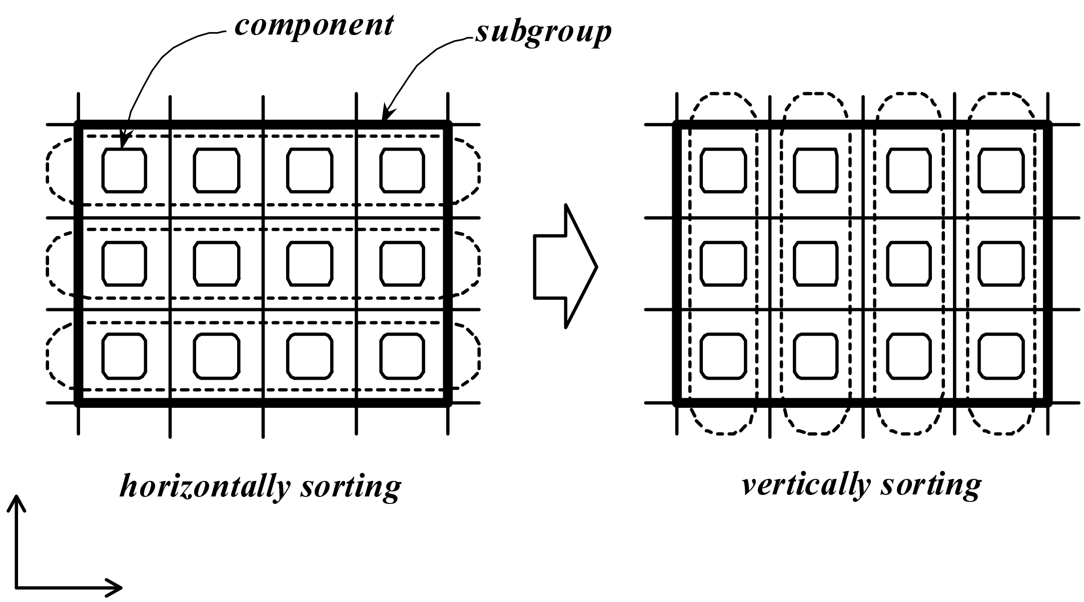


図 3.2: FVS 法での再配置方法

上に存在することになるため，凸包の境界線上の空きスロットの中から評価値が最大となるスロットを探索し，コンポーネントを割り当てる．以下にアルゴリズムを構成する上での方針を詳細に述べる．

3.3.1 再配置先候補スロット

ここでコンポーネントに与えられた F_h, F_v を 2 次元ベクトルとして考え，その指す方向を" 外側" とする．するとサブグループ内のスロットで最も外側となるスロットのコストが最大となる．また，このベクトルに対して直交するベクトル方向に並ぶスロットについては全て同一のコストが課せられることになる．図 3.3 の右側の矢印は $F_h = -3, F_v = 3$ のコンポーネントについて，そのフォース値をベクトルとした $\vec{\mathbf{u}} = (-3, 3)$ を示している．図 3.3 の左側の太線で囲んだ部分はサブグループを表し，$\vec{\mathbf{u}}$ と直交する方向を示す破線矢印はその線上のスロットのコストが等しいことを表している．この場合，サブグループの左上のスロットがコスト最大となり，空きスロットであればコンポーネントはそこに配置されるべきである．そして $\vec{\mathbf{u}}$ と反対方向に向かうにつれてコストは小さくなる．

このような性質からコンポーネントの配置先としてはサブグループの空きスロット上で，各コンポーネントに対して最も外側になりうるスロットのみを取り扱うことにする．そこでサブグループ内の空きスロットについて，それらを平面座標上の点集合とみなし，それらの凸包を構成し，その境界線上にある点，つまり境界線上の空きスロットのみを配置先の候補とする．平面上の点集合の凸包は，その集合の全ての点を含む最小の凸多角形

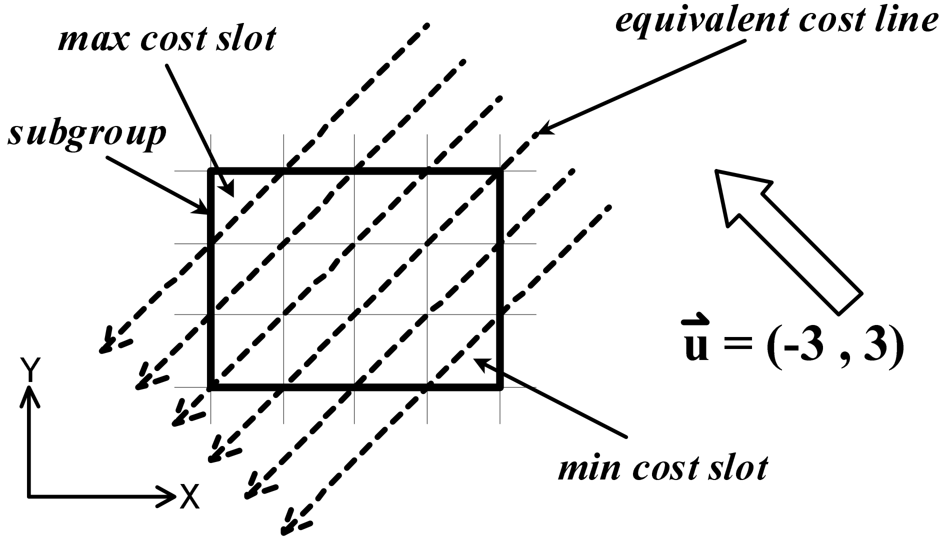


図 3.3: $F_h = -3, F_v = 3$ のコンポーネントに対するスロットへのコスト

であり，点のうちのいくつかが凸多角形をつくり，残りの全ての点はその内部に含まれる．

サブグループ内の空きスロットに対する凸包を図 3.4 に示す．同図において黒丸はコンポーネントの配置が確定しているスロット，白丸は未確定な空きスロットを示す．凸多角形の辺 (side) にあたる部分は複数のスロットで構成され，凸多角形の角 (極値点，extremal) は１つのスロットから構成される．凸多角形の辺には同図の破線矢印のように平面座標上で反時計回りにベクトルを与える．そして，あるコンポーネントに対しては，そのフォース値のベクトルと直交するベクトル $\vec{\mathbf{u}}^{\perp}$ を持つ凸多角形の辺を配置先として選ぶようにする．そのような辺がない場合は，$\vec{\mathbf{u}}^{\perp}$ を挟み込むようなベクトルを持つ 2 つの辺の間の極値点を配置先として選ぶようにする．

3.3.2　再配置先の決定

コンポーネントを取り出した直後にその配置先スロットを決定してしまうと，サブグループ全体でのコストが低下してしまう可能性がある．たとえば図 3.5 のように v_1 について，３つあるコストの等しい配置先スロットの中から S_{11} を配置先として決定してしまうと，v_2 については S_{11} が最適な場所であるにもかかわらずそこには配置することができなくなってしまう．このようなことを防ぐため．配置先候補スロット数とそこへ配置されるコンポーネントの数が等しくなったときに，それらを確定することにする．また凸多角形全体のスロット数と，その時点までに取り出され，かつ，未配置であるようなコンポーネントの数が等しくなったときもそれらを確定する．

以上のことを踏まえた上での配置先の確定に至るまでの動作は，次のようになる．

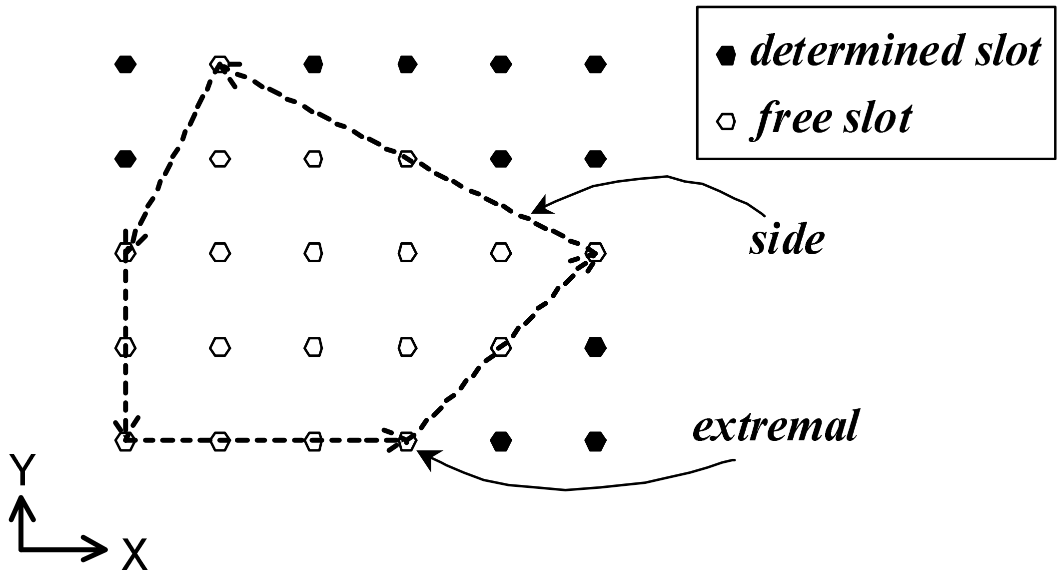


図 3.4: 空きスロットに対する凸包

1. 再配置先候補スロットの記録
   コンポーネント v_i について，未確定スロット中でコスト最大のスロットを探索し，配置先候補スロット集合 $C(v_i)$ として記録し 2. へ進む.
2. 確定可能コンポーネント及びスロットの確認
   $v_1 \sim v_i$ の中で未確定のコンポーネントの集合を Q とする．Q のある部分集合 Q' に対して，
   $$|Q'| = |\cup_{v' \in Q'} C(v')|$$
   のとき Q' 内のコンポーネントの配置先を確定し，3. へ進む．上の式を満たす Q'（$\neq \emptyset$）が存在しないときは 1. へ戻って次のコンポーネントについて調べる.
3. 配置先候補スロットの更新
   $v \in Q \backslash Q'$ について配置先候補 $C(v)$ を更新し，2. へ戻る.

上記のようなアルゴリズムに従うとコンポーネント v の再配置先候補スロットは

- 未確定スロット集合に対する凸多角形の辺上にあるスロット全て.
- 未確定スロット集合に対する凸多角形の極値点上の１つのスロットのみ.

となり，配置先を確定するコンポーネント集合 Q' の選び方としては

- Q の中で同一配置先候補を持つコンポーネントの全て（図 3.6）.
- Q の全て（図 3.7）.

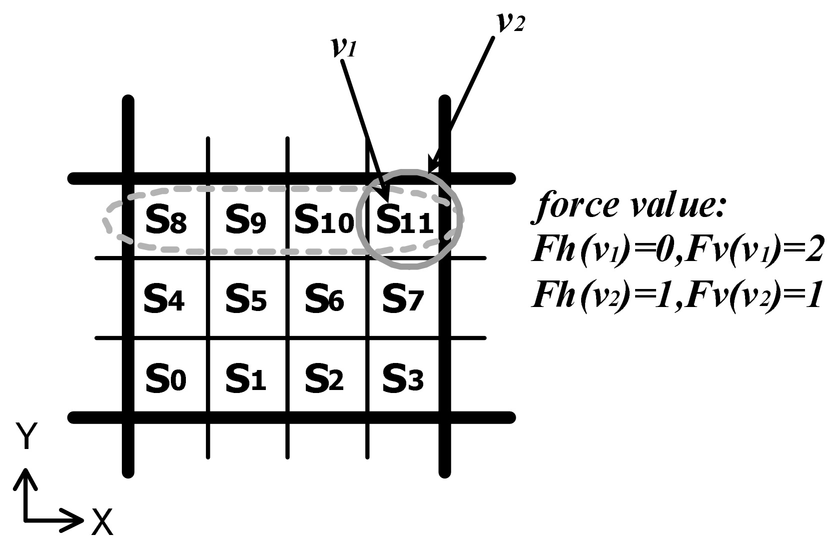


図 3.5: コストが低くなる例

コンポーネントを確定後は確定済みスロットを除いて，空きスロットで凸包を再構成し，さらに残りのコンポーネントについて配置先を調べていく．

3.3.3 安定性 (スタビリティ) について

[2] によると安定性を考慮した再配置はその効果が大きい．安定性の考慮とはコストの等しいコンポーネントが複数ある場合，なるべく元の配置を崩さないように再配置し，一旦収束した解が悪くならないようにすることである．

まず，コンポーネントのソート段階で，F_h, F_v の等しいコンポーネント同士についてはもともとサブグループの外側にあったものが優先されるようにする．そこでソート時のみ，各コンポーネントのフォース値を $F'_h = F_h + stb_h, F'_v = F_v + stb_v$ のように修正してからソートを行った．スタビリティ値 stb_h, stb_v は各方向において，コンポーネントの元の座標値からサブグループの中心の座標値を引いたものを用いる．また，コンポーネントの配置を確定するときは，コンポーネントの元のスロットから確定する凸多角形の辺のスロットに垂線を下ろし，その交点の位置を辺のベクトル方向に合わせてソートし，コンポーネントをスロットの端から確定していく．図 3.8 の例ではコンポーネントはベクトルの始点方向のスロットから v_4, v_5, v_2, v_3, v_1 の順番で確定していく．

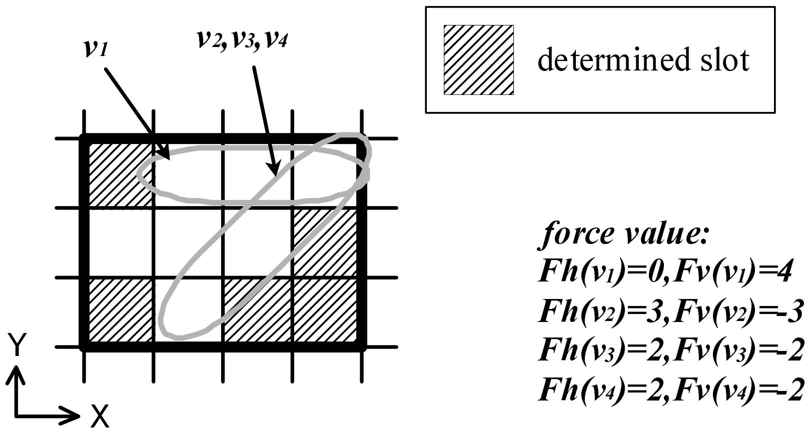


図 3.6: $Q' = \{v_2, v_3, v_4\}$ となる例

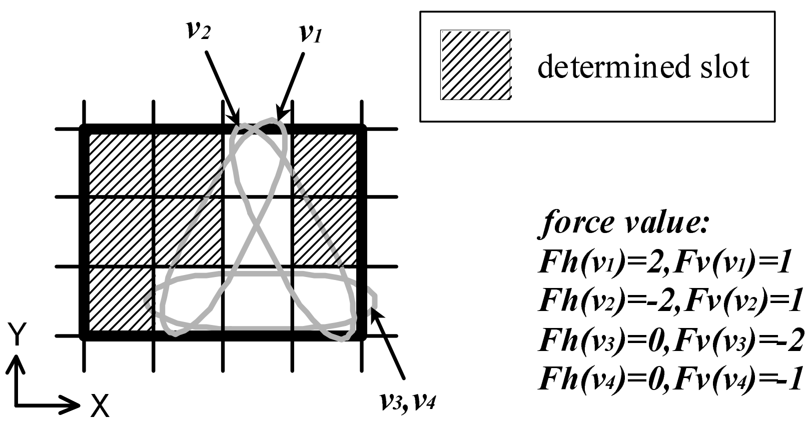


図 3.7: $Q' = \{v_1, v_2, v_3, v_4\}$ となる例

3.3.4　コンポーネントの選択順序

コンポーネントはなるべくコストの大きくなるように $max\,\{|F_h|, |F_v|\}$ の順にソートした結果を $v_1, v_2, ..., v_n$ として用いた.

以上のように空きスロット集合に対する凸包を利用して再配置を行う方法が FVCH である．図 3.9 に FVCH の再配置処理をフローチャートにしたものを示す.

3.4　実験と考察

FVS 法と FVCH 法の性能を確かめるため，実験を行った．実験に用いた 2 つのハイパーグラフのサイズを表 3.1 に示す．ここで各ハイパーグラフはネットの次数が 2 から 20 の

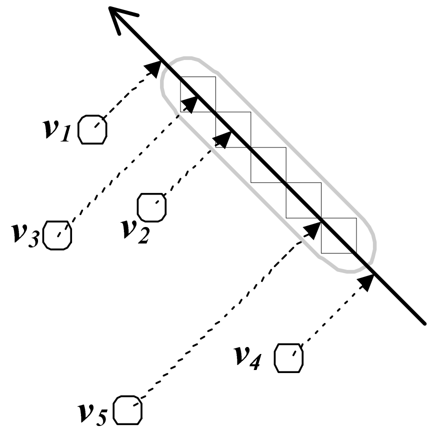


図 3.8: 元の配置を考慮した再配置場所の決定

間に分布し，5 端子ネットの個数が一番多くなるような指針に基づき乱数で生成した．またスロットアレイは data30 では $W = H = 30$，data100 では $W = H = 100$ とした．各ハイパーグラフの端子数とネット数の関係を図 3.10，図 3.11 に示す．

表 3.1: ハイパーグラフのサイズ

name	# comps	# nets
data30	900	500
data100	10000	6000

data30，data100 それぞれに対する SA 法，FVS 法，FVCH 法の実験結果を表 3.2，3.3 に示す．ここで，#loops は各ピッチでの繰り返し回数を示す．

図 3.12，3.14 は総配線長と実行時間の関係を図示したものである．また，図 3.13，3.15 に data30，data100 での配線長の変化を示す．

FVS 法の総配線長結果は，各ピッチでの繰り返し回数を増やしても，SA 法と比較して劣っている．この理由として，

1. 再配置において水平方向と垂直方向を逐次処理している．
2. 原問題の総配線長最小化に対し，部分問題では，フォース値に基づいた間接的な目的関数を用いている．

の 2 点があげられる．一方，FVS 法と FVCH 法との比較では，最終的な総配線長においては性能差がみられない．

以上より，SA 法に対して提案手法の総配線長の性能が劣っている原因は，部分問題の目的関数の不適切さにあると考察される．

表 3.2: data30 に対するフォース値に基づく手法での実験結果

method	#loops	total length	runtime (sec.)
SA	—	9682	2439
FVS	1	11193	2
	10	10692	9
	50	10616	42
	100	10584	85
	300	10494	252
FVCH	1	10844	12
	10	10640	110
	30	10662	330
	50	10611	550

表 3.3: data100 に対するフォース値に基づく手法での実験結果

method	#loops	total length	runtime (sec.)
SA	—	388866	40195
FVS	1	425483	33
	10	406791	290
	50	402544	1391
	100	400876	2768
FVCH	1	414583	244
	5	412820	1168
	10	413061	2379
	15	412801	3476

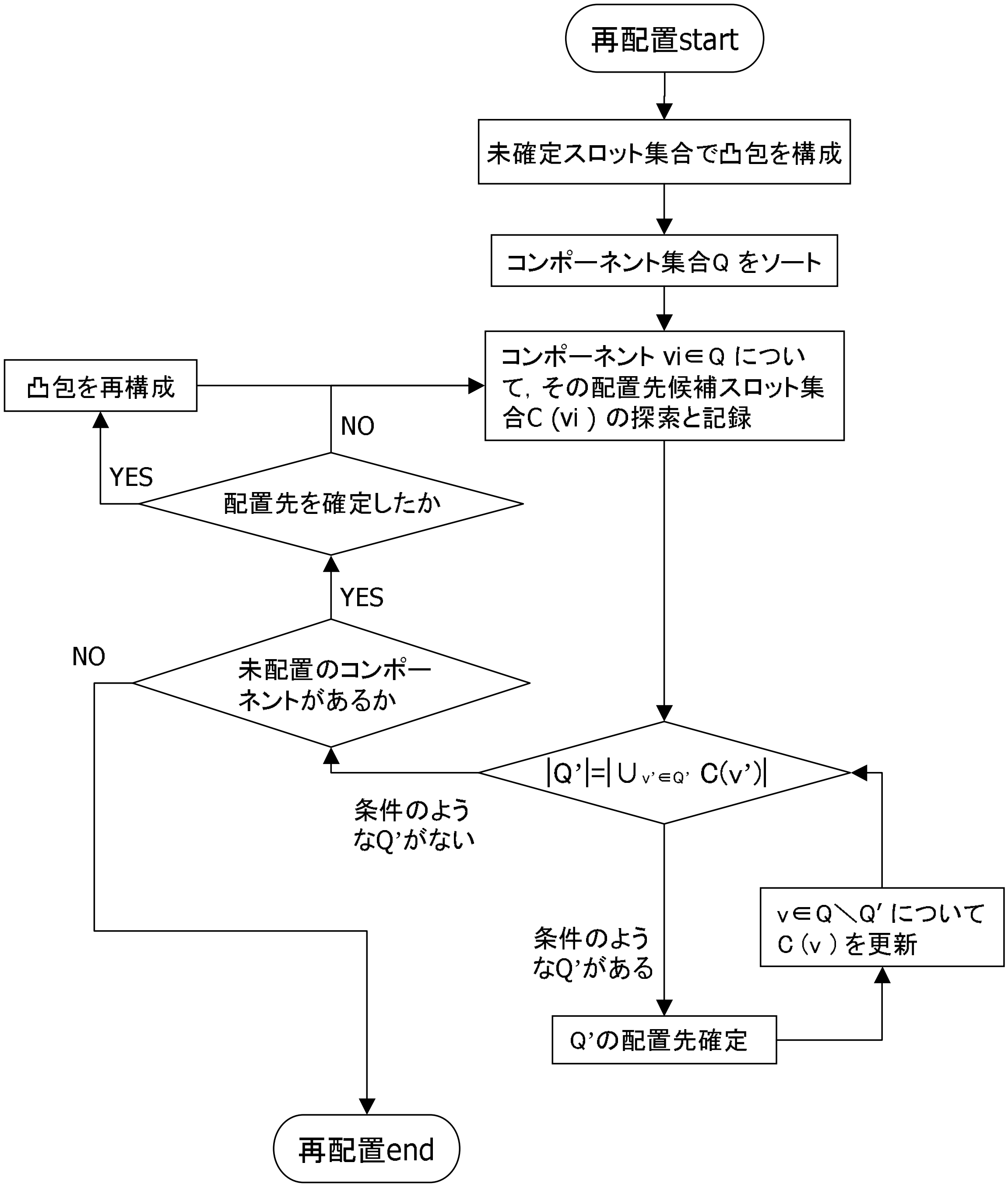


図 3.9: FVCH の再配置処理でのフローチャート

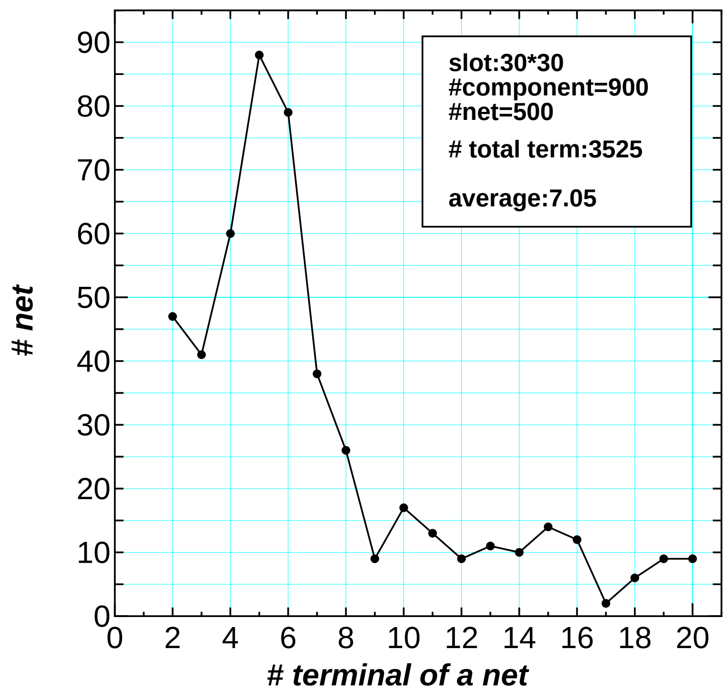


図 3.10: data30 の端子数とネット数

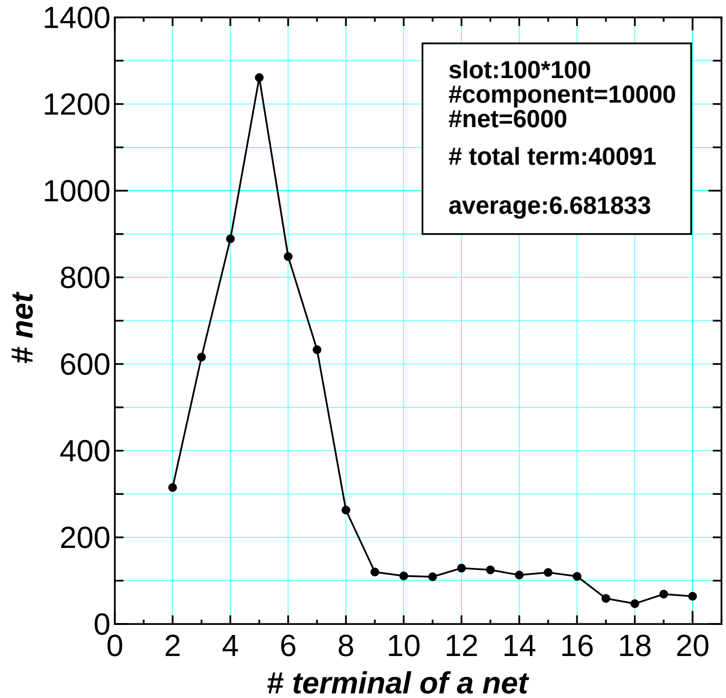


図 3.11: data100 の端子数とネット数

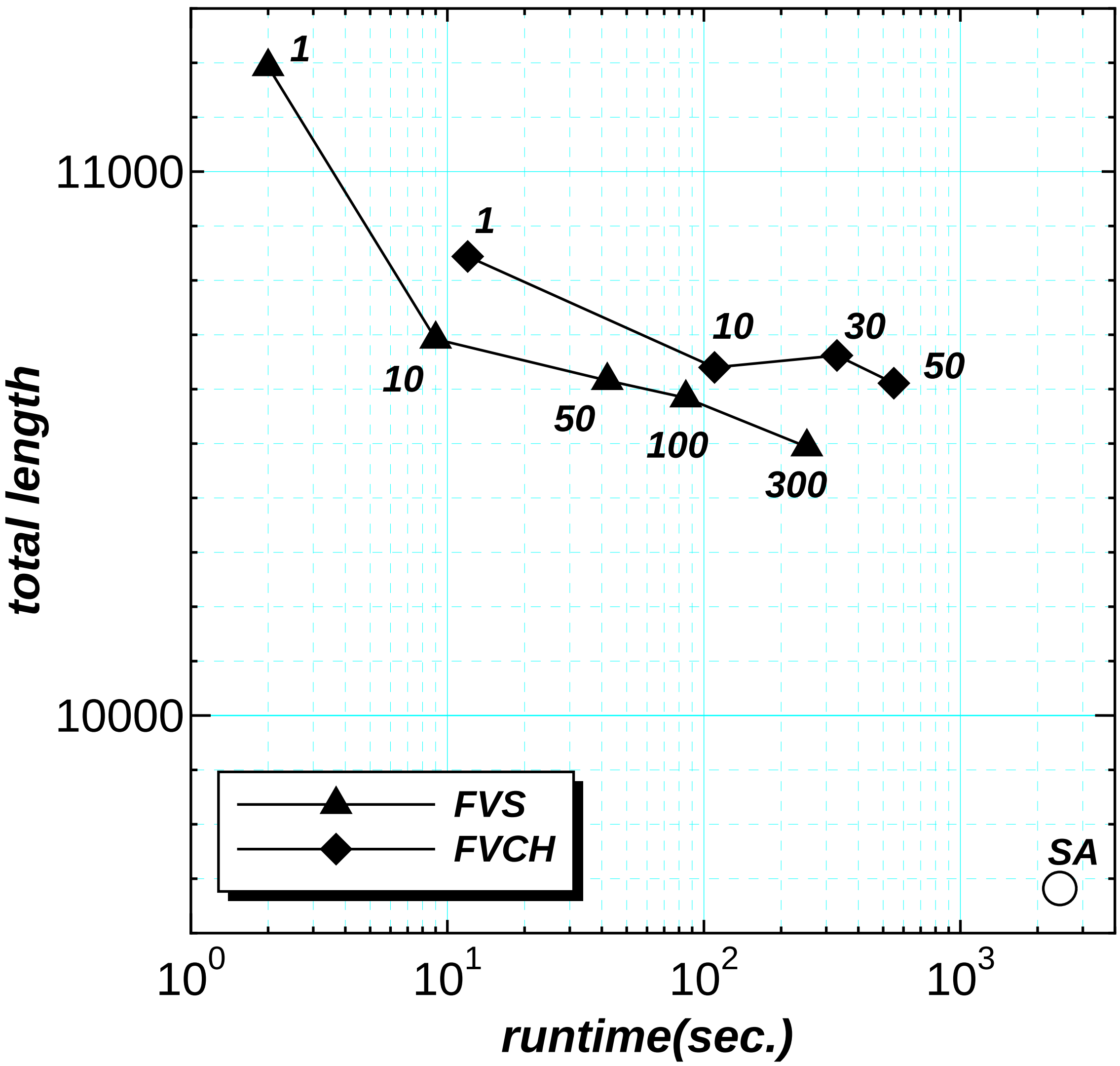


図 3.12: data30 に対するフォース値に基づく手法での総配線長と実行時間の関係（各点に付された数は#loops を表す）

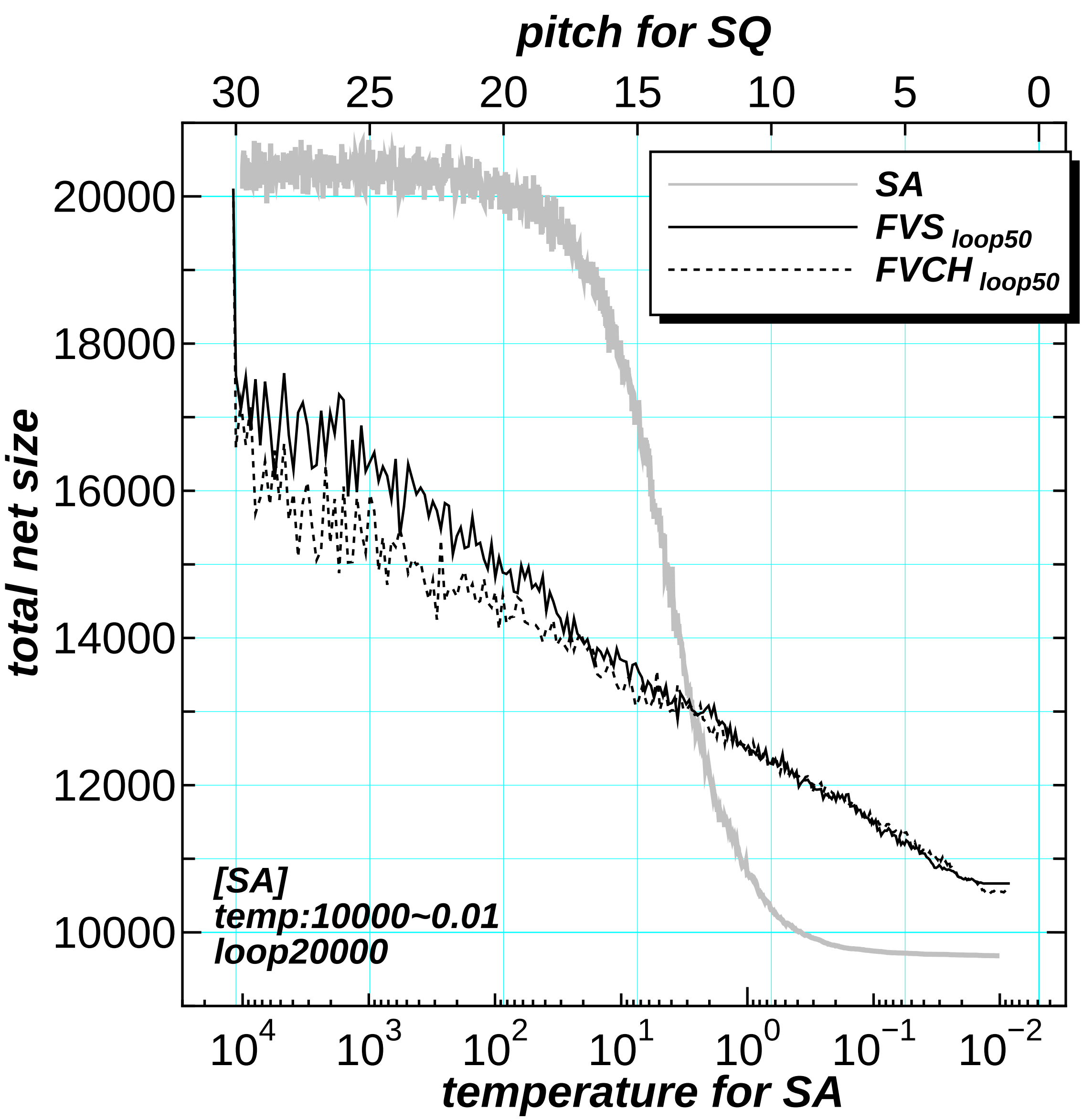


図 3.13: data30 に対するフォース値に基づく手法での総配線長の結果

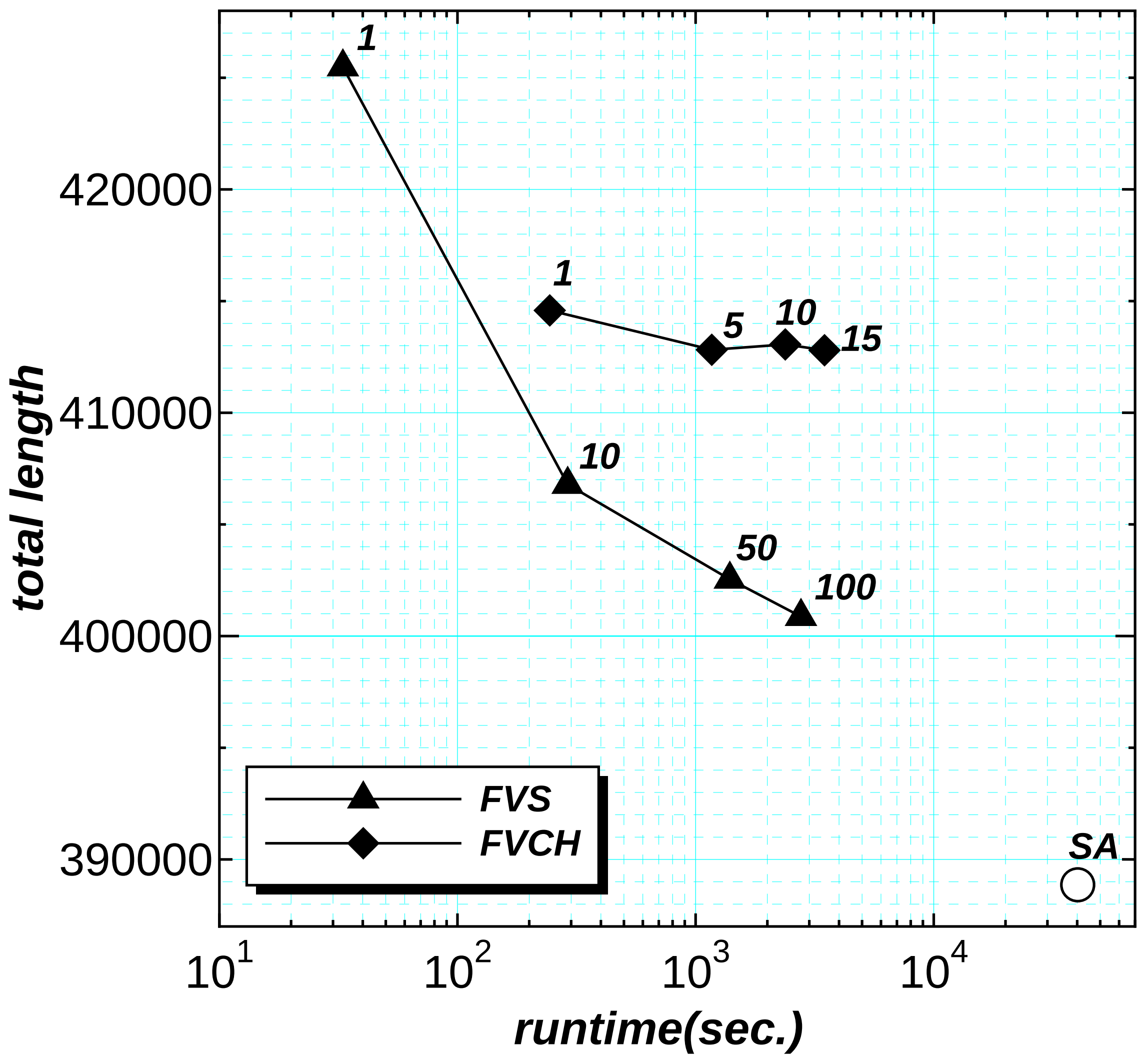


図 3.14: data100 に対するフォース値に基づく手法での総配線長と実行時間の関係（各点に付された数は#loops を表す）

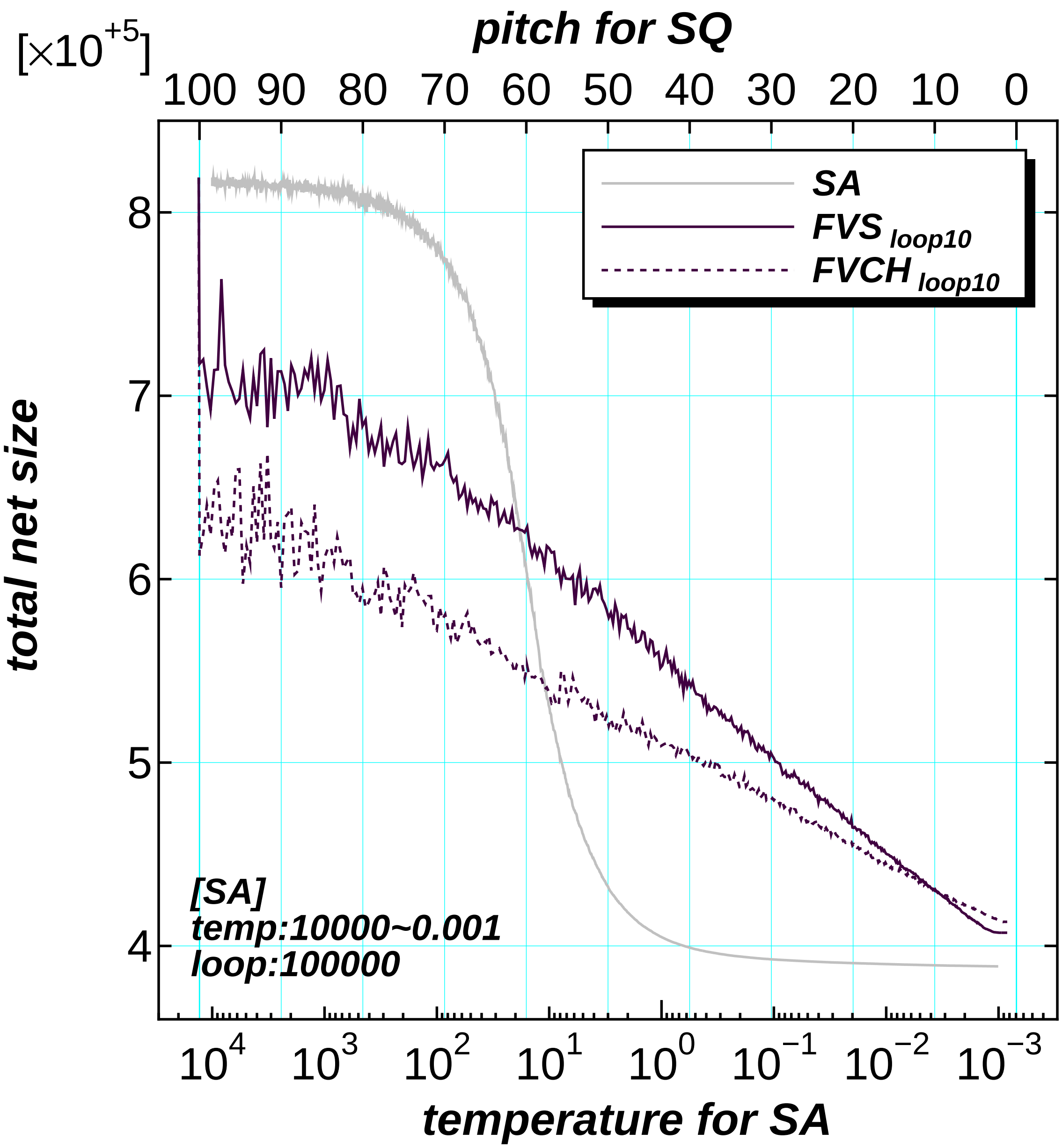


図 3.15: data100 に対するフォース値に基づく手法での総配線長の結果

第4章　ステップコストに基づく手法

4.1　ステップコストについて

再配置の対象としているサブグループ g に含まれるコンポーネント c のスロット s でのステップコストを以下のように定義する．

1. c に接続するネット n に対し，n に接続するコンポーネント中から g に含まれるものを全て取り除いたときのネットの水平方向の長さ l_{h0} と垂直方向の長さ l_{v0} を求める．
2. c を s に割り当てたときの n の配線長の l_{h0} からの水平方向の増分，l_{v0} からの垂直方向の増分をそれぞれ $C_h(c,s,n)$，$C_v(c,s,n)$ とする．
3. c に接続するすべてのネットに対し，$C_h(c,s,n)$，$C_v(c,s,n)$ それぞれを合計したものを c を s に割り当てたときのステップコスト $SC_h(c,s)$，$SC_v(c,s)$ とする．

ステップコストの例として，図 4.1（a）の灰色で示したサブグループ g における再配置を考える．g に含まれるコンポーネント a に対して，ネット $\alpha = \{a,b,c,d,e\}$ についてのコスト $C_h(a,s,\alpha)$，$C_v(a,s,\alpha)$ を図 4.1（b）に示す．

4.2　アルゴリズムの概要

[2] では再配置をサブグループに含まれる全てのコンポーネントを全てのスロットに割り当てる問題として考え，ステップコストに応じた枝重み付き 2 部グラフの最小重み完全マッチングを解き，解を得ていた．その結果，総配線長に関して SA とほぼ同等の解が得られる反面，実行時間は SA よりも長くなってしまっている．この手法の各繰り返しでの計算量を評価すると，コンポーネント数を m，ネット数を n として，$O(m^3+mn)$ となり，高速化のためにはより小さな計算量の再配置アルゴリズムが必要である．そこで，第 ?? 節の再配置で水平方向と垂直方向を逐次処理しても性能が劣化しないという結果をふまえ, 水平方向と垂直方向の逐次処理で最小重み完全マッチング [3][4] を用いて再配置を行う SCM（<u>S</u>tep <u>C</u>ost based replacement by <u>M</u>atching）法を試みた．また，SCM 法を観察し，その性質を用いた発見的手法である SCS（<u>S</u>tep <u>C</u>ost based replacement with <u>S</u>lope）法を試みた.

以下に SCM 法と SCS 法のそれぞれにおける再配置手法の概要を示す.

4.2.1 SCM 法

SCM 法では再配置を水平方向と垂直方向の逐次処理で実現し，各方向では 2 部グラフの最小重み完全マッチングを解くことで実現する．水平方向，垂直方向それぞれのコンポーネント c のスロット s への割り当てのコストはそれぞれ $SC_h(c,s)$，$SC_v(c,s)$ を用いる．各繰り返しでの計算量は $O((\sqrt{m})^3 \times \sqrt{m} + mn) = O(m^2 + mn)$ となる．

4.2.2 SCS 法

隣接する 2 つのスロットでのステップコストの差分に注目して近似的に再配置を行なう手法である．左端/下端 のスロットから順にコンポーネントを割り当て，注目するスロットでのステップコストの傾きが最大のコンポーネントをそのスロットに配置する．例えば図 4.2 において，コンポーネントの配置先を S_iから S_{i+1} に換えたとき，コンポーネント a の方がコンポーネント b よりもステップコストが増加するので 1 スロット分のステップコストの増分を傾きで表し，その大きい方，a をまず S_i に配置し，b は S_{i+1} に配置するようにする．

$SC(c,s)$ を注目する方向でのステップコスト $SC_h(c,s)$ または $SC_v(c,s)$ とし，$\Delta_{i,j} = SC(c_j, s_{i+1}) - SC(c_j, s_i)$ を定義する．今，SCM 法のある一方向での再配置において，隣接する 2 つのスロット s_i，s_{i+1} 及びそれぞれに割り当てられたコンポーネント c_j，c_k に関して次の補題が成り立つ．

補題 1 $\Delta_{i,j} \geq \Delta_{i,k}$

証明 SCM 法の一方向での再配置における最小重み完全マッチングのコストを D とする．ここでスロット s_i に割り当てられたコンポーネント c_j と s_{i+1} に割り当てられたコンポーネント c_k の割り当て先を交換すると，そのときのコスト D' は,

$$
\begin{aligned}
D' &= D - SC(c_j, s_i) - SC(c_k, s_{i+1}) \\
&\quad + SC(c_j, s_{i+1}) + SC(c_k, s_i) \\
&= D + \{SC(c_j, s_{i+1}) - SC(c_j, s_i)\} \\
&\quad - \{SC(c_k, s_{i+1}) - SC(c_k, s_i)\} \\
&= D + \Delta_{i,j} - \Delta_{i,k}
\end{aligned}
$$

ここで $\Delta_{i,j} < \Delta_{i,k}$ とすると，D が最小であることに矛盾する．よって，補題が成り立つ．
□

SCS 法では，スロットを 左端/下端 から順に考え，そのときにまだスロットに割り当てられていないコンポーネントの中から，次のスロットとのステップコストの変化量（傾き）がもっとも大きいものを割り当てるアルゴリズムを用いて再配置を行う．この手法の各繰り返しでの計算量は $O((\sqrt{m})^2 \times \sqrt{m} + mn) = O(m\sqrt{m} + mn)$ である．

この手法について，次の定理が成り立つ．

定理 1 ピッチ p が 2 以下のとき，SCS 法は SCM 法と等価である．

4.3 実験と考察

SCM 法，SCS 法の性能を確かめるため，実験を行った．入力ハイパーグラフにはフォース値に基づく手法に対する実験と同じく data30 と data100 を用いた．

data30，data100 それぞれに対する SCM 法，SCS 法の実験結果を表 4.1，4.2 に示す．それぞれの表には比較のため，SA 法と FVS 法の結果も示す．図 4.3，4.5 は総配線長と実行時間の関係を図示したものである．また，図 4.4，4.6 に data30，data100 での総配線長の変化の様子を示す．

SCM 法では総配線長に関して，SA とほぼ同等の解を得ることができた．これは，最適化処理を逐次処理で行っても解の質が劣化しないことを示している．また，フォース値に基づく手法である FVS 法と比較して，ステップコストの有効性が確認される．これは忠実な部分問題と採用した部分問題の目的関数の差が小さくなったことによると考えられる．しかし，実行時間は SA と比較して，最悪 1.5 倍程度要する結果となった．

次に SCS 法について，SA 法や SCM 法と比較すると，総配線長性能はほぼ同等である．SCS 法での各繰り返しでは必ずしもコストが最小の割り当てが求められるわけではない．それは図 4.6 の SCS 法と SCM 法の途中結果に差があることからもわかる．しかし，最終的な結果ではほとんど差がないことから，部分問題の性質として，最適化過程の終りの方（$p = 2 \sim 1$ 付近）では忠実な部分問題との差が小さいことが重要であると考えられる．

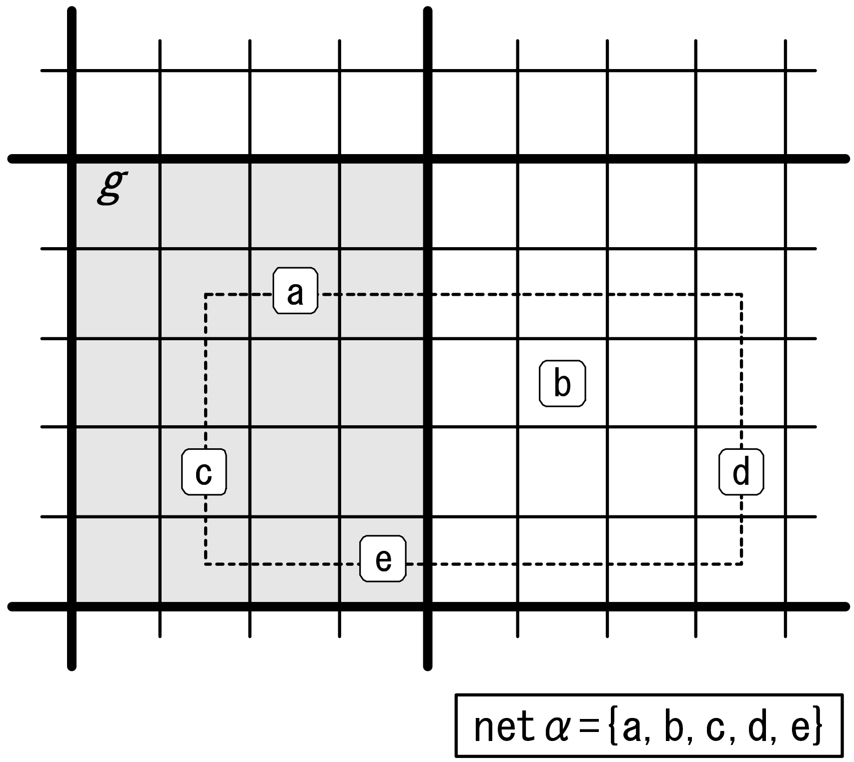
また，SCS 法の実行時間は各ピッチでの繰り返し回数が等しい SCM 法と比較して，data30 で 1/12，data100 で 1/18 程度になっている．これは，SCM 法の各繰り返しでの計算量 $O(m^2 + mn)$ に対して SCS 法の計算量が $O(m\sqrt{m} + mn)$ であることに因るものである．但し SCS 法で SCM 法と同程度の解を得るには，SCM 法よりも多少大きな繰り返し回数が必要である．

表 4.1: data30 に対するステップコストに基づく手法での実験結果

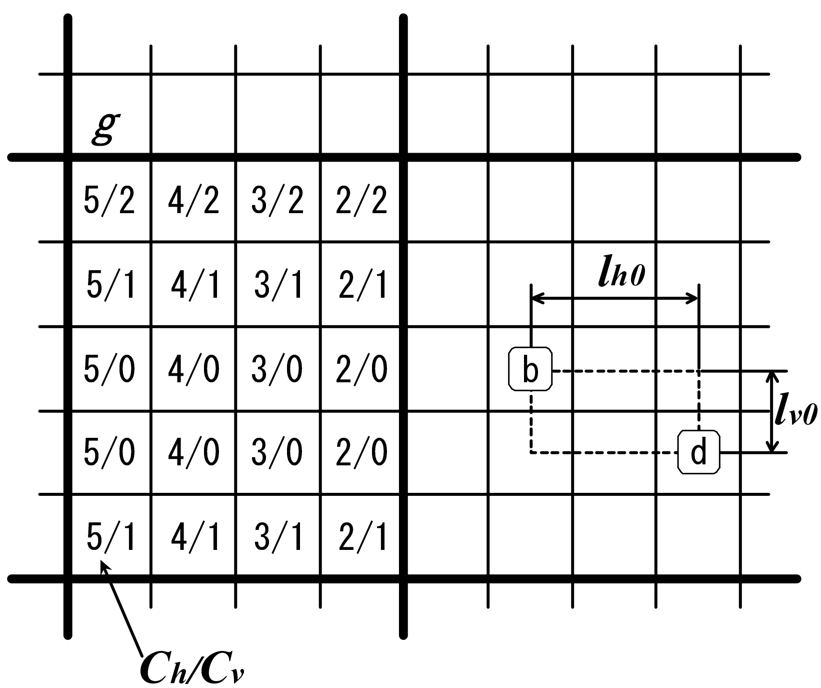
method	#loops	total length	runtime (sec.)
SA	—	9682	2439
FVS	300	10494	252
SCM	1	10273	22
	10	9948	213
	50	9861	1061
	100	9831	2120
SCS	1	10168	3
	10	9976	20
	50	9951	99
	100	9930	168
	200	9912	335

表 4.2: data100 に対するステップコストに基づく手法での実験結果

method	#loops	total length	runtime (sec.)
SA	—	388866	40195
FVS	100	400561	2768
SCM	1	401194	2426
	10	391380	24583
	20	389184	48525
	25	388880	60935
SCS	1	396933	139
	10	392177	1290
	20	391203	2569
	25	390752	3212
	30	390699	3850



(a)



(b)

図 4.1: コンポーネント a のネット α に対する C_h, C_v

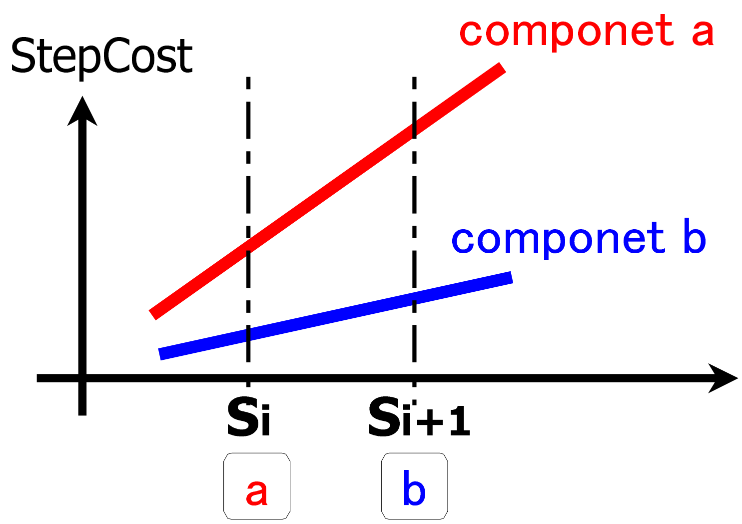


図 4.2: SCS 法での隣接する 2 つのスロット S_i, S_{i+1} における配置

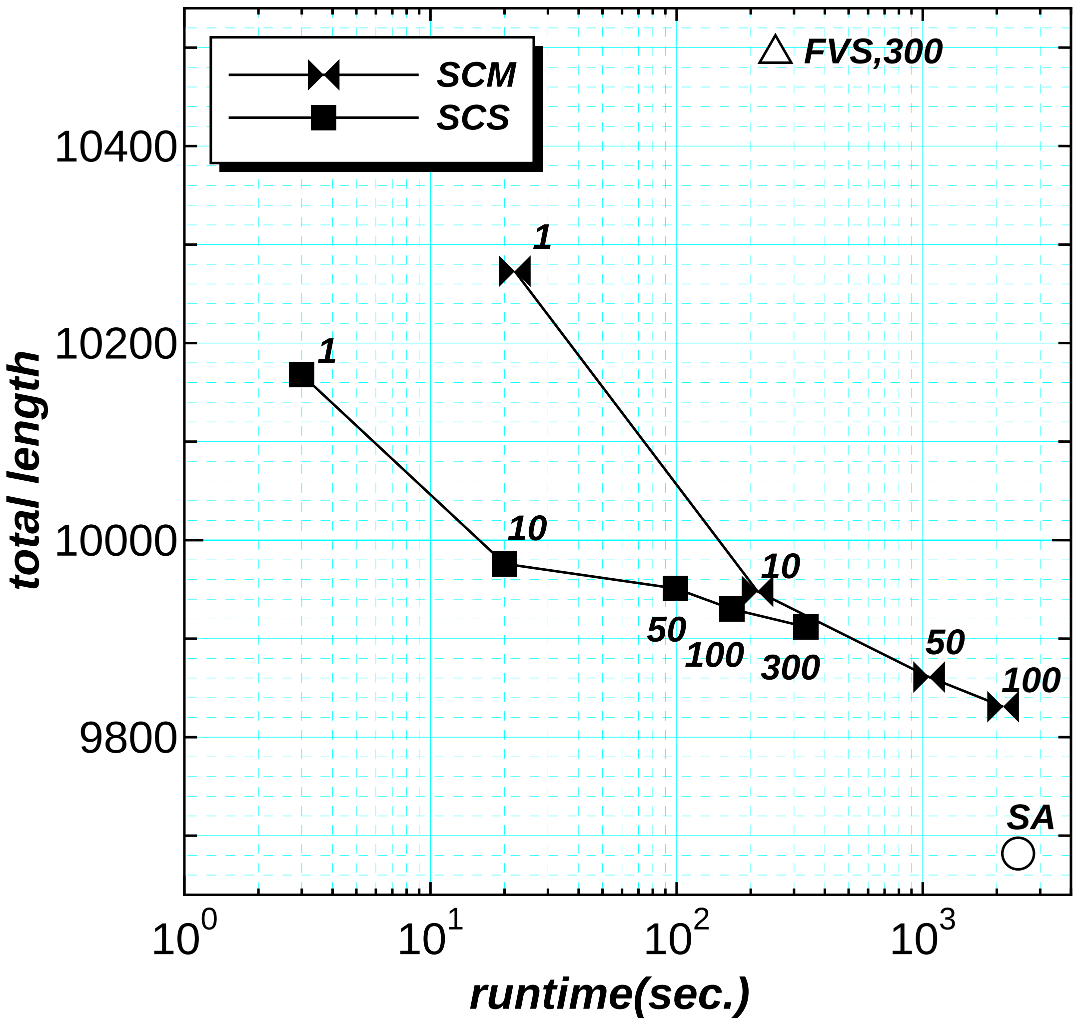


図 4.3: data30 に対するステップコストに基づく手法での総配線長と実行時間の関係（各点に付された数は#loops を表す）

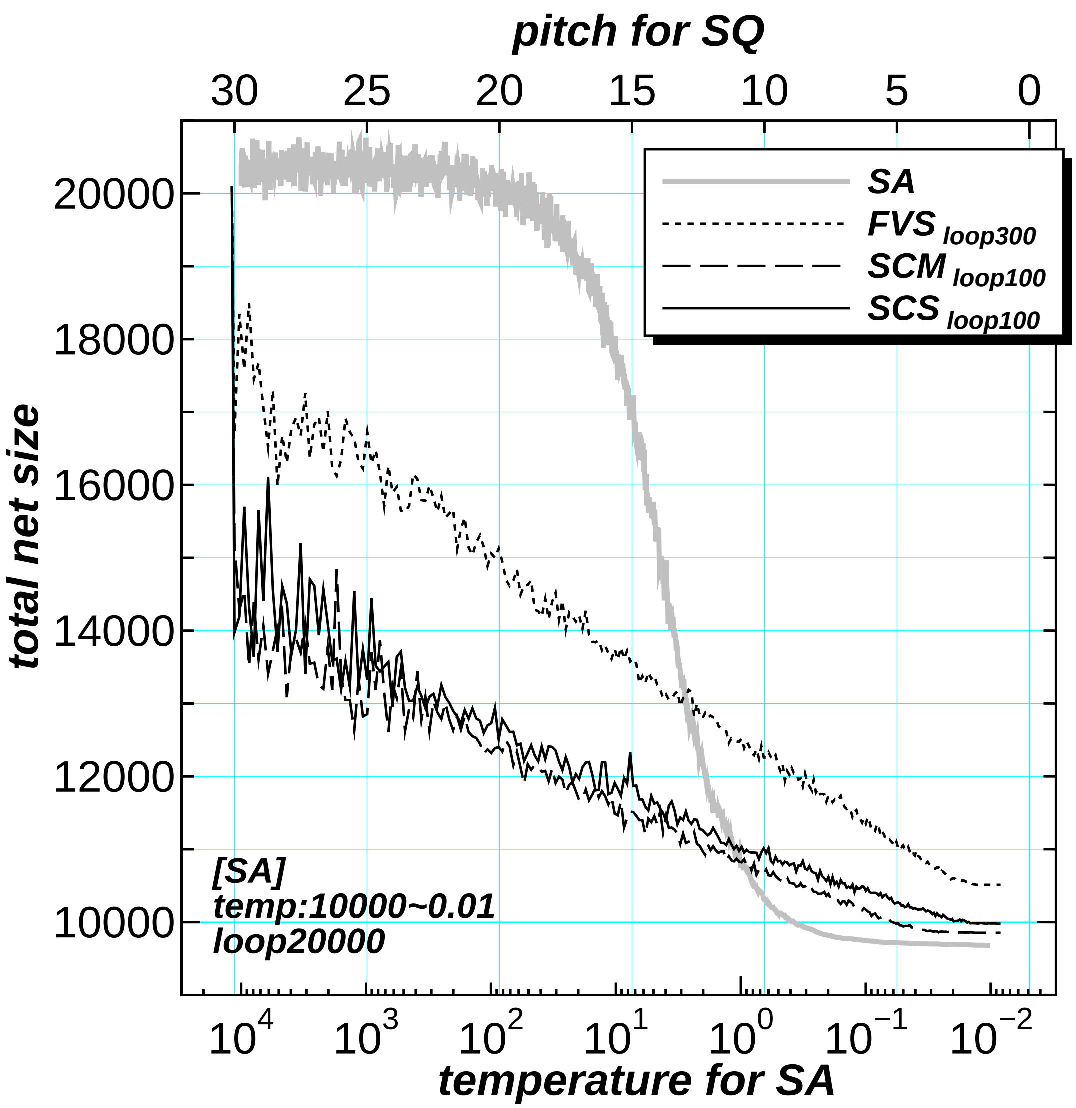


図 4.4: data30 に対するステップコストに基づく総配線長の結果

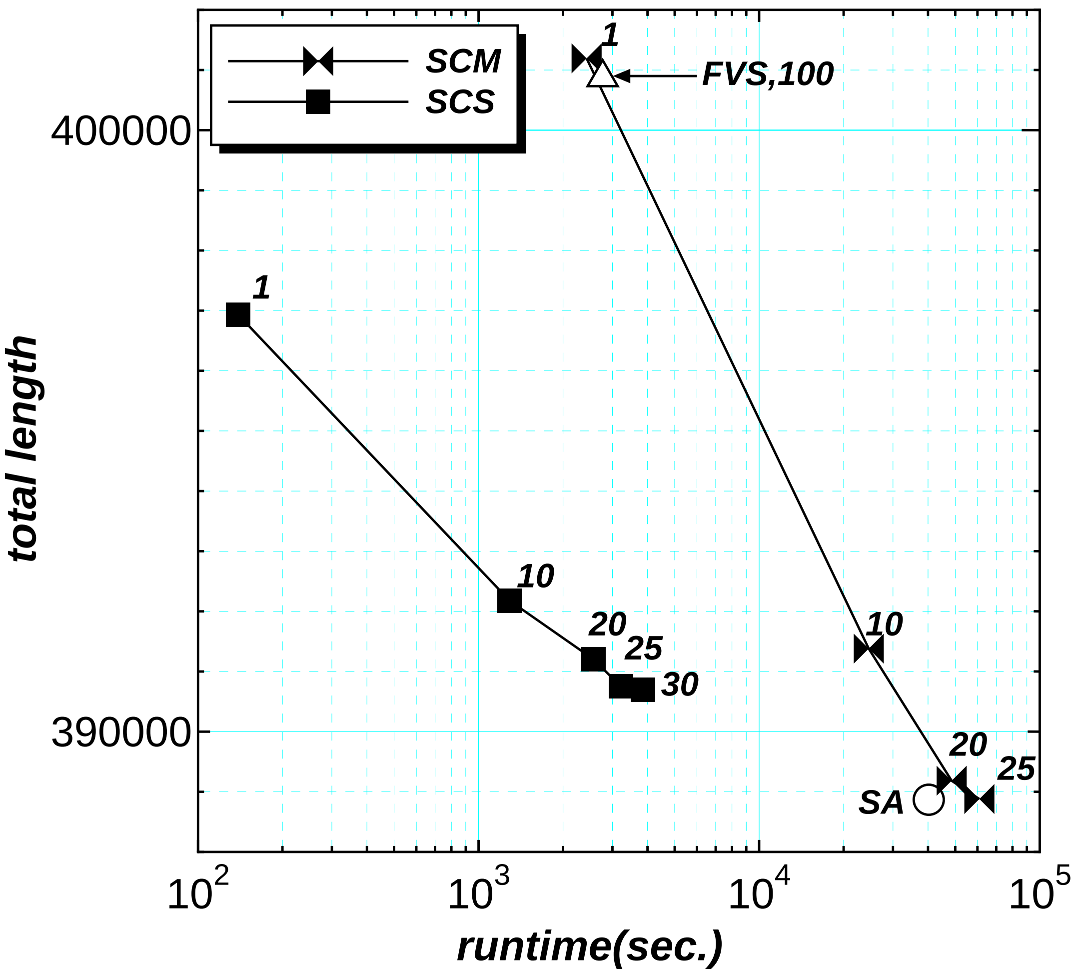


図 4.5: data100 に対するステップコストに基づく手法での総配線長と実行時間の関係（各点に付された数は#loops を表す）

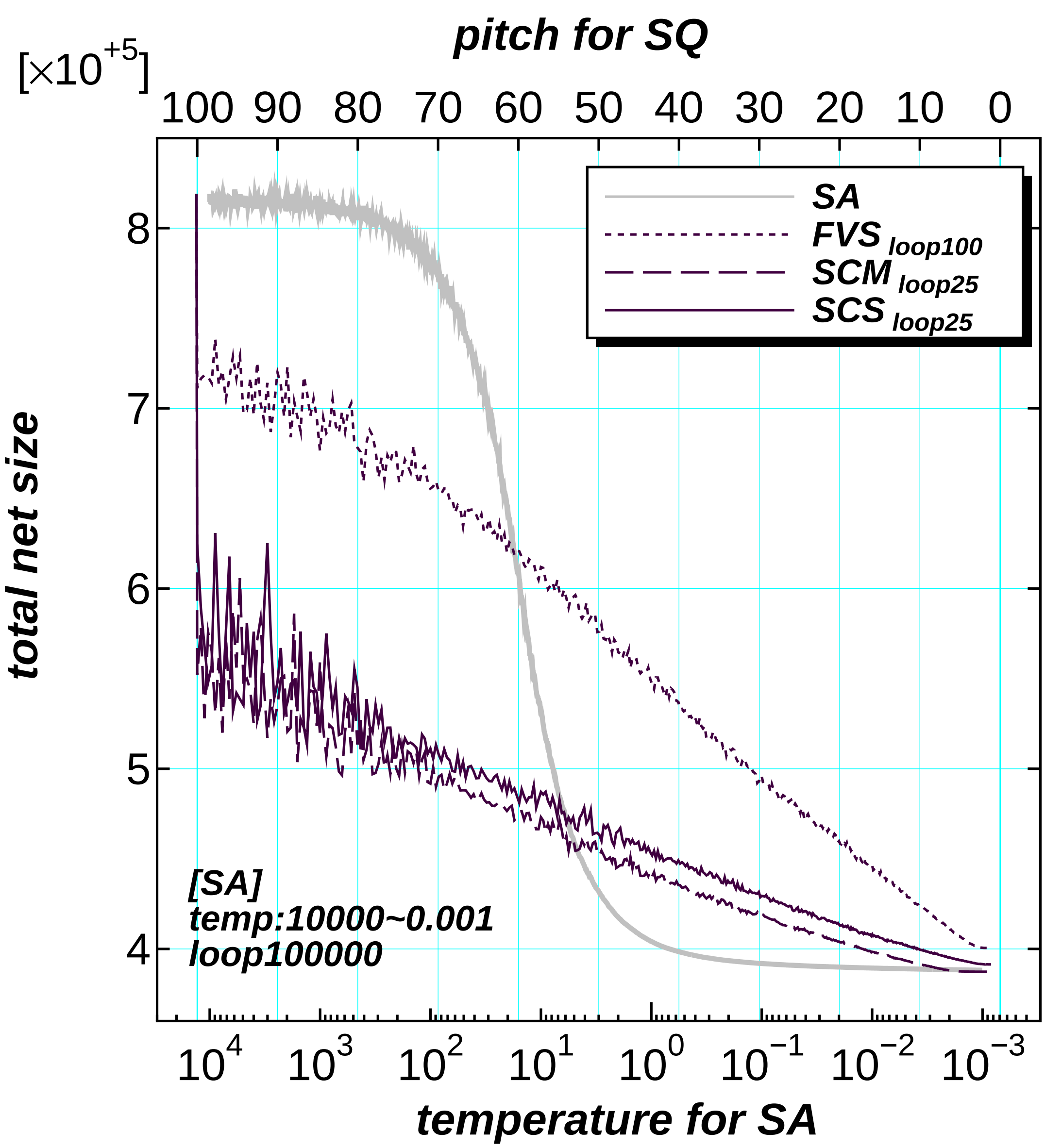


図 4.6: data100 に対するステップコストに基づく総配線長の結果

第5章　SCS法のさらなる展開

SCG法において良好な結果が得られたので，同手法のさらなる可能性を模索する．そこでまず傾きのつけ方を改良することで配線長の長いネットの数を最小化することを試みる．

5.1　部分問題に対する目的関数の改変

SCG法では，SCM法において各ネットに対してつけていたステップコストを隣接する２つのスロット間でのステップコストの変化量（傾き）とした値で表し，再配置領域での全ネットのその値の合計を各スロットごとに比較することによって配置を決めていた．その際，傾きは 水平方向/垂直方向 にそれぞれ１スロットずれるにつれて１ずつであったが，ここではSCG法と同様のやり方で，傾きの大きさを変化させることでネットの重要度に差をつけ，変化の様子を見る．

5.2　SCSv2a法

ステップコストの変化量を傾きで表した値を次のように改変する．対象としているサブグループにおいて，あるネットについて 水平/垂直 いずれかの方向でステップコストを求めたときにコスト最大となるスロットにコンポーネントを配置したときの配線長を求め SUB_{MAX} とする．また，入力となるスロットアレイ全域に渡るネットの長さを $WHOLE_{MAX}$ とする．このときにステップコストの傾き（$slope_{v2}$）を次のように計算することにする．

$$slope_{v2} = \left(\frac{SUB_{MAX}}{WHOLE_{MAX}}\right)^{\alpha} \tag{5.1}$$

ここで α は0以上の数とする．$\alpha = 0$ の場合は傾きは常に1となり，SCG法のときとなんら変わりはない．α が大きくなるにつれ SUB_{MAX} の大きなネットをより重視するようになる．この手法をSCSv2a法と呼ぶことにする．例として図5.1にdata100に対する α の違いによる各配線長ごとの傾きを示す．

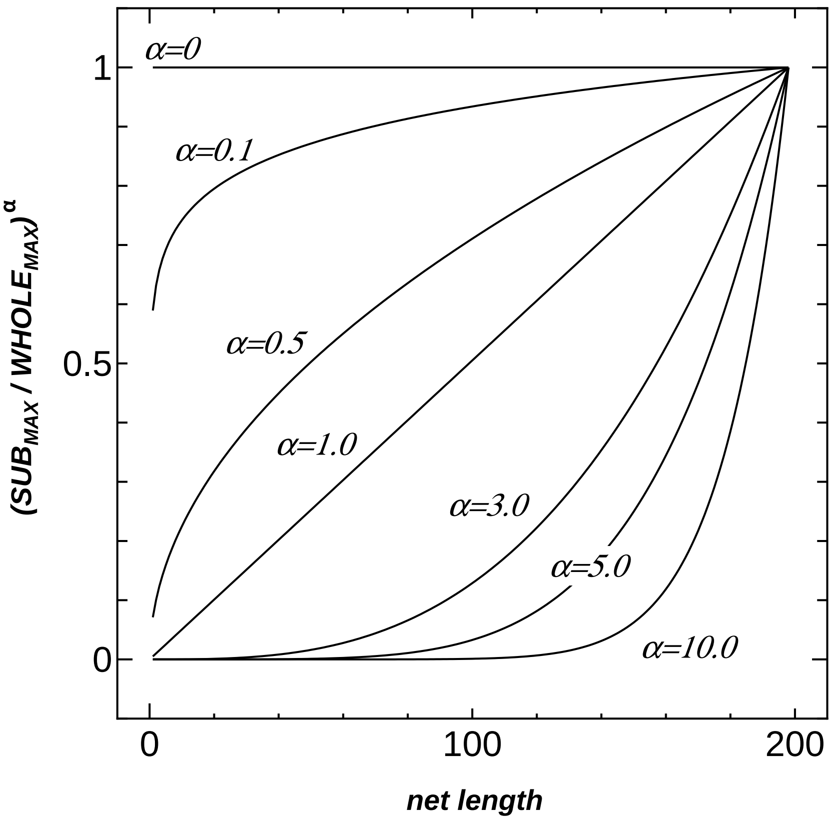


図 5.1: data100 に対する α ごとによるネットの傾きの違い

5.2.1 SCSv2a 法の結果

data100 に対して SCSv2a 法を実行した．α の違いによる最終結果について表 5.1 に示す．ここで各ピッチでの繰り返し回数は 25 回とした．また図 5.2 は最終結果について，横軸にネットの長さをとり，縦軸にその長さまでのネットがいくつあるかを示したものである．

表 5.1 を見ると α の値が大きくなるにつれて総配線長は増加している．分散（variance）の値は α の値が大きくなるにつれて小さくなっている．図 5.2 を見ると α の増加につれて長さ 80 から 100 付近のネットの数が増加していることが見て取れる．

5.3 SCSv2b 法

続いて SCSv2a 法にしきい値 Th をもうけ，配線長が Th より大きいのネットについては式 5.1 で傾きをつけ，Th 以下の場合には式 5.2 での h を傾きとしてつけることにする．

表 5.1: α の違いによる SCSv2a 法の最終的な配線長の結果

α	total length	average	variance
0	390834	65.1	2235.9
0.1	406308	67.7	1246.6
1	417870	69.6	958.9
5	473411	78.9	392.0
10	491264	81.8	309.2

表 5.2: data100 に対する $\alpha = 10$ での Th の違いによる最終結果

Threshold	total length	average	variance
—	491264	81.8	309.2
80	487231	81.2	333.4
100	461335	76.9	755.8
120	412532	68.8	1464.8

$$h = \frac{1}{Th} \int_1^{Th} \left(\frac{SUB_{MAX}}{WHOLE_{MAX}} \right)^{\alpha} dSUB_{MAX} \tag{5.2}$$

この方法を SCSv2b 法と呼ぶことにする．

また，SA 法についても SCSv2b 法と同様の傾きとしきい値を用いて試してみた．この場合，SA 法では傾きをそのまま用いて計算することができないので，式 5.1 と式 5.2 をそれぞれ積分した値を各ネットの長さとして用いる．つまり，評価に用いる各ネットの長さ Len を式 5.3 の Len' のようにし，これを SAv2 法として実験を行なった．

$$Len' = \begin{cases} \frac{1}{\alpha+1} \cdot \left(\frac{Th}{WHOLE_{MAX}} \right)^{\alpha} \cdot Len & (\text{if } Len \leq Th) \\ \\ \frac{1}{\alpha+1} \cdot \frac{Len^{\alpha+1}}{(WHOLE_{MAX})^{\alpha}} & (otherwise) \end{cases} \tag{5.3}$$

5.3.1 SCSv2b 法と SAv2 法の結果

SCSv2b 法について data100 を用いて，各ピッチでの繰り返し回数を 25 回，$\alpha = 10$，Th は 80，100，120 について試した．結果を表 5.2 と図 5.3 に示す．

表 5.2 を見ると Th を大きくすることで総配線長が小さくなっていき，分散値はおおきくなっている．また，図 5.3 をみると Th 付近でネット数が急激に増えていることがわかる．

SAv2 法については data100 を用いて，$\alpha = 10$，$Th = 100$ で行なった．結果を表 5.3 と図 5.4 に示す．

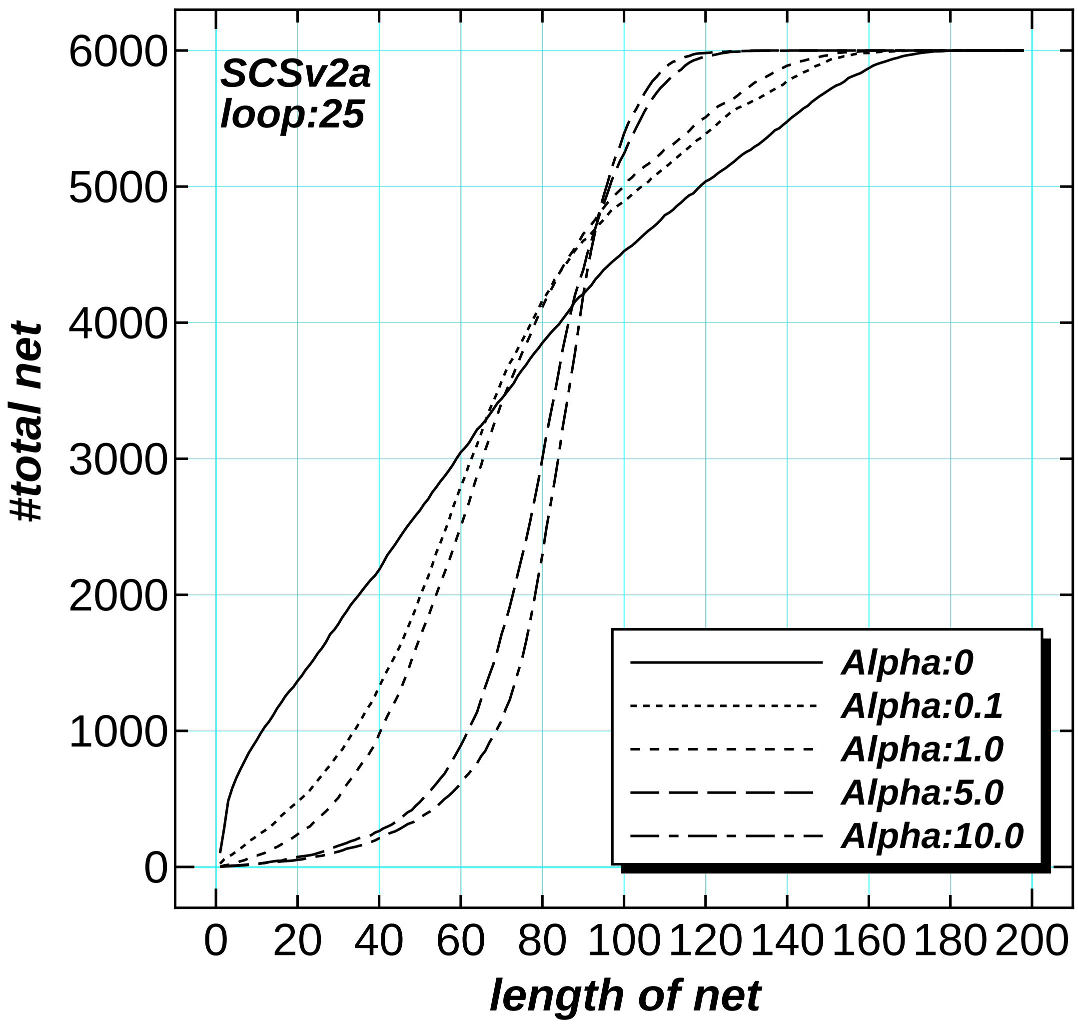


図 5.2: data100 に対する SCSv2a 法のネット数の変化の様子

表 5.3: data100 に対する SAv2 法の $\alpha = 10$ での最終結果

Threshold	total length	average	variance
—	511778	85.3	214.3
100	519954	86.7	350.7

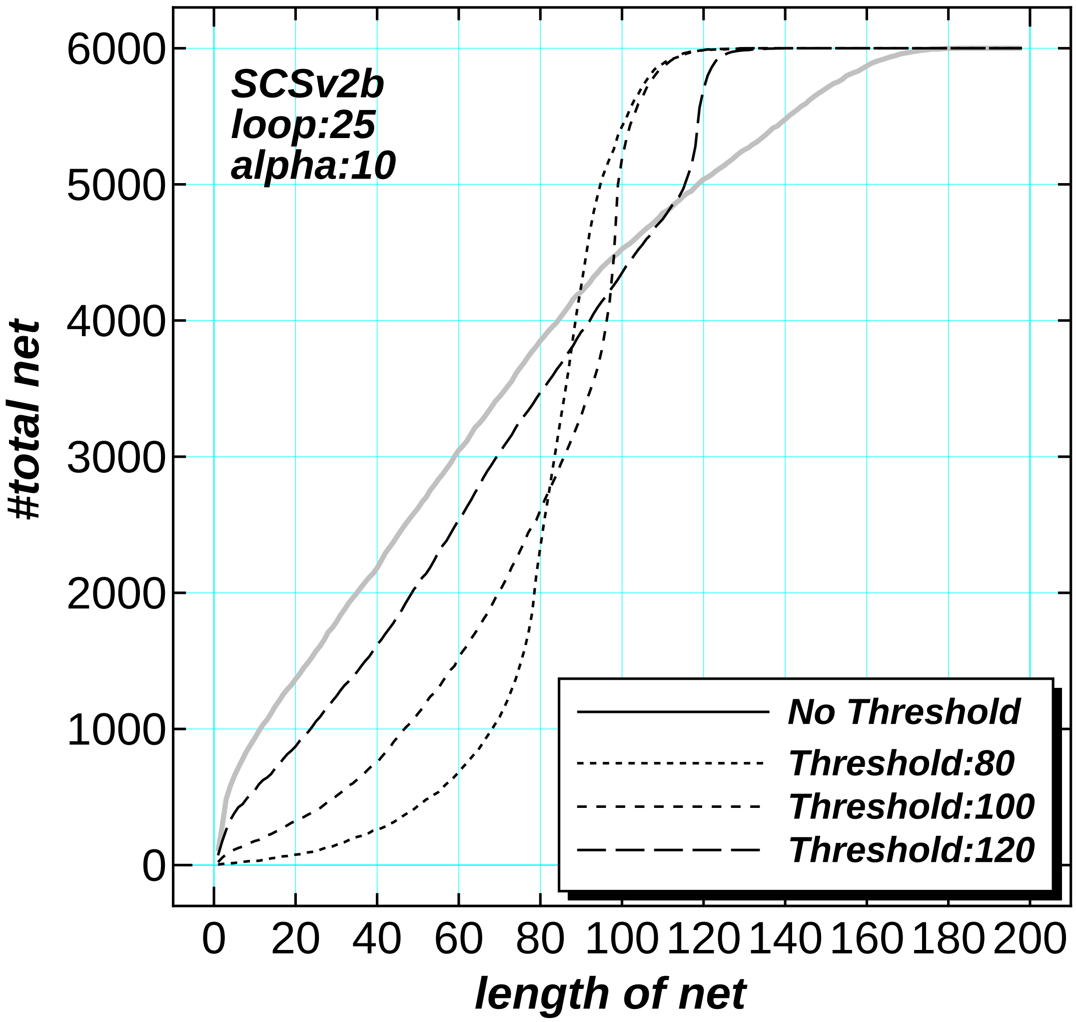


図 5.3: data100 に対する $\alpha = 10$ での Th の違いによるネット数の変化の様子

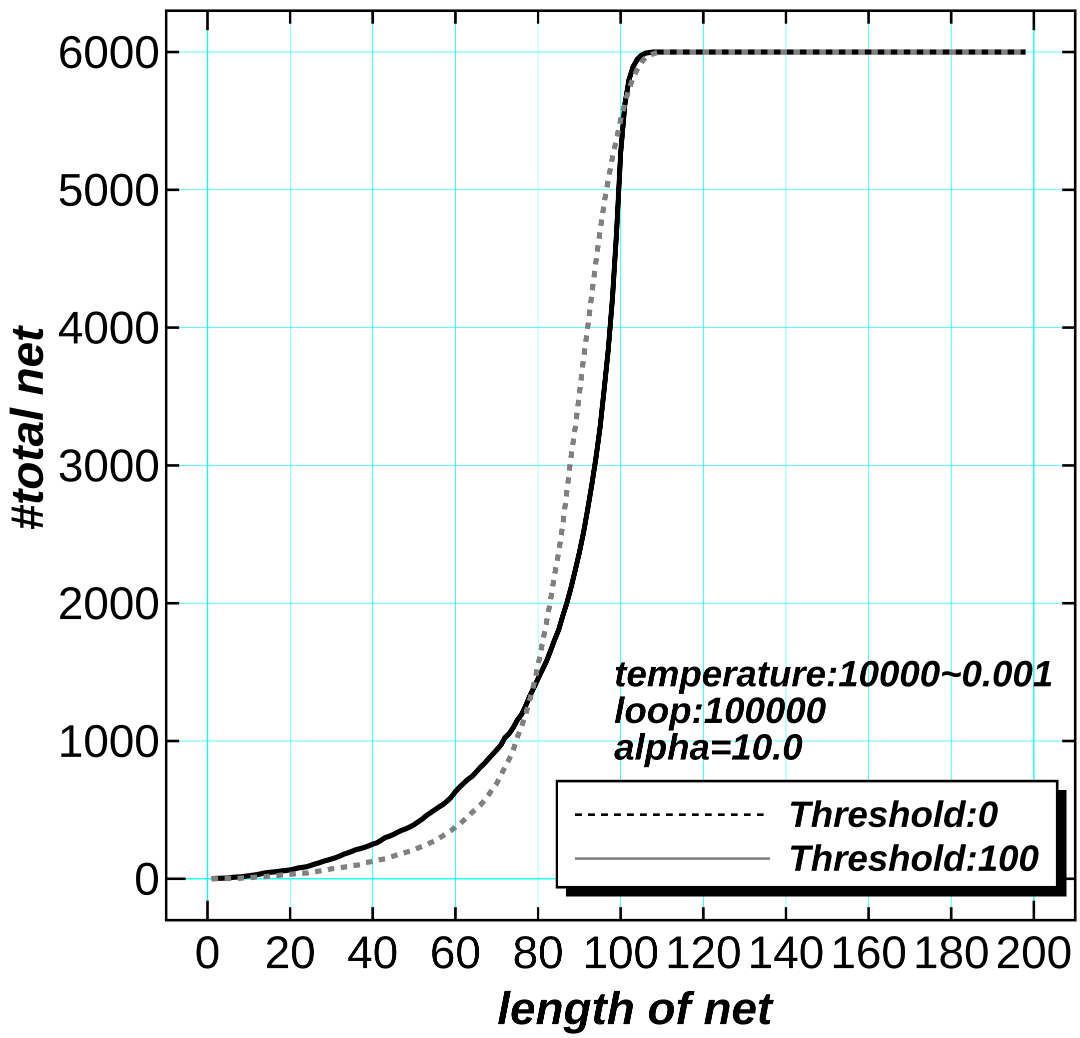


図 5.4: data100 に対する SAv2 法の $\alpha = 10$ でのネット数の変化の様子

第6章　まとめ

6.1　結論と今後の課題

本稿では2次元配置問題を解くアルゴリズムとしてFVS法，FVCH法，SCM法，SCS法を考案し，実験により，これらの最適化能力を確認した．特にSCS法はSAとほぼ同等の解を10 ∼ 20倍程度高速に求めることができ，SCS法の2次元配置問題への有効性を示すことができた．

SCS法の各繰り返しでの計算量は$O(m\sqrt{m} + mn)$であり，主要項はmnである．これは全ネットの総配線長を求めるのに必要な計算量である．よって，今後の課題としては更なる高速化にあたりネットの配線長の効率の良い求め方を考案する必要がある．また，部分問題とその解法を見直すことにより，更なる高速化を実現することがあげられる．

SCSv2法，SCSv2b法については各コンポーネント対してのステップコストの傾きをネットのサイズによって変化させることによって，特定の長さのネットだけが増加するような配線長結果を得ることができた．しかし，これらの手法に関してはまだ試行の段階で評価が不十分である．

謝辞

本研究を進めるにあたり，終始適切なご助言と暖かいご指導をいただいた北陸先端科学技術大学院大学 金子 峰雄教授，同 田湯 智助手，同 高島康裕助手，マイクロアーク株式会社 村田 洋氏，同 佐藤 眞司氏，そして研究室の学生の皆さんに深く感謝いたします．

参考文献

[1] Shinji Sato, “Simulated Quenching:a New Placement Method for Module Generation”, Proc.ICCAD,pp.538-541,1997.

[2] 平間孝廉, 高島康裕, 佐藤真司, 金子峰雄, “Simulated Quenching 法に基づく 2 次元配置最適化手法”, 電子情報通信学会, 信学技報,VLD2000-135,pp.7-12,2001.

[3] Christos H. Papadimitriou, and Kenneth Striglitz, “Combinatorial Optimization Algorithm and Complexity”, Prentice-Hall,Inc.,1982.

[4] Thomas Lengauer, “Combinatorial Algorithms for Integrated Circuit Layout”, John Wiley & Sons Ltd,1990.

[5] Ketan Mulmuley, “Computational Geometry:An Introduction Through Randomized Algorithms”, Prentice-Hall,Inc.,1994.

[6] Mark De Berg, Marc Van Kreveld, Mark Overmars, and Otfried Schwarzkopf, “Computational Geometry:Algorithms and Applications” Springer-Verlag Telos,2000.