TR-0777

並列分枝限定法による混合整数計画問題の解法

川岸　太郎

June, 1992

ICOT

Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan

(03)3456-3191～5
Telex ICOT J32964

Institute for New Generation Computer Technology

並列分枝限定法による混合整数計画問題の解法

川岸 太郎
(財) 新世代コンピュータ技術開発機構 (ICOT)*

概要

組合わせ最適化問題の一つの定式化である混合整数計画問題について，分枝限定法による解法の並列化を行った．分枝限定法における探索木の複数の枝を並列な探索プロセスとして異なるプロセッサに割り当てて仕事を分散させ，また並列プロセス間で情報のやり取りを行って枝刈りを進め，解の探索を速める．ICOT で開発された並列言語 KL1 でプログラムを記述し，並列マシン Multi-PSI 上で速度向上の実測を行った．

Abstract

We discuss a parallelization of a branch-and-bound algorithm to solve the mixed integer programming problems, which is one formulation of combinatorial optimization problems. Sub-problems in a search tree created through the branching process are distributed to different processors, and each sequential search process communicates with others to transmit information on the recently found candidate solutions to make pruning of sub-nodes proceed over all processors. We implemented the parallel search algorithm in KL1 language in the parallel environment Multi-PSI system, and made a measurement of speedup.

1 はじめに

組み合わせ最適化問題を解くための定式化の枠組としては様々なものがあるが，その中で整数計画問題としての定式化は広範囲の問題に適用可能なものである．整数計画問題の解法は，厳密な最適解を求めるものに限った場合にも，それぞれの問題の持つ特殊構造に応じて，探索を速めるための様々な方法がある．このうち問題の制約の構造に依存せずに，広範囲の問題クラスに一般的に有効な方法として分枝限定法がある．

分枝限定法は，与えられた問題が解けるかどうかを試して，解けなかった場合にそれに制約を付加して問題を分割すると言う操作を繰り返し，元の問題を根として木構造に分割して行く方法であり，その過程で発見される部分問題の解によって木の枝刈りをしながら最適解の探索を行う．探索木の葉となるのは，部分問題で解が見つかったものか，解のないことが分かったものである．分枝限定法は探索ヒューリスティックを用いて枝刈りの効率化を狙っているが，全ての探索枝よりも良い値を持つ最適解の探索を行うので，探索問題で一般的に見られるように問題のサイズの増加に対して探索空間は指数関数的に増え，計算量もそれに応じて増えるため，適用できる問題サイズに限界が生じる．

分枝限定法の並列化については [Quinn 90] が MIMD 計算機上での並列アルゴリズムを考え，[Li,Wah 86] はヒューリスティック評価関数の特性によって並列分枝限定法の速度向上を理論的に評価している．Quinn は Travelling Salesman 問題を扱ってハイパーキューブ NCUBE/7 上で実験しているが，そのアルゴリズムは分枝操作で生成される部分問題の一部を必ず別プロセッサに投げるものである．

本研究では混合整数計画問題，すなわち整数変数と実数変数の両方を含んだ線形制約に対して，線形目的関数の最大値あるいは最小値を求める問題を取り上げ，分枝限定法の並列実行により規模の大きな問題を高速に解くことを目標として，並列プログラムを作成し，分散メモリの並列計算機上で実験を行った．ここではプロセッサ間でのデータ転送を減らすために，Quinn の場合とは違って，部分問題が生成

*Parallel Branch-and-Bound Method for Mixed Integer Programming Problems
Kawagishi, Taro
Institute for New Generation Computer Technology

される度に別プロセッサに投げることはせず，探索木の一定の深さで負荷分散を止めるようにした．

並列化は分枝限定法における複数の探索枝を異なるプロセッサに割り振って仕事を分散させる．その場合に並列効果をうまく引き出す上で問題となる点は，(1) 部分探索木の大きさのばらつきの影響を少なくすること (負荷分散)，(2) 逐次の場合の枝刈りを，並列探索でも全プロセッサに渡って同様に行うこと，(3) 問題を分割して並列に同時探索することによって起こる，逐次であったなら避けられていた見込み計算を減らすことが上げられる．

以下混合整数計画問題の定式化を行い，逐次アルゴリズムを示し，次にそのアルゴリズムの並列化と実験結果について述べ，ここで上げた問題点の分析を行う．

2 問題の定式化

扱う最適化問題は次のものである．

Problem – *ILP*

実数変数 x_j と整数変数 y_j とに関する目的関数

$$z = \sum_{j=1}^{n_1} p_j x_j + \sum_{j=1}^{n_2} q_j y_j$$

を，次の線形制約条件の下で最小化せよ：

$$\sum_{j=1}^{n_1} a_{ij} x_j + \sum_{j=1}^{n_2} b_{ij} y_j \geq e_i,\ i = 1, \ldots, l,$$
$$\sum_{j=1}^{n_1} c_{ij} x_j + \sum_{j=1}^{n_2} d_{ij} y_j = f_i,\ i = 1, \ldots, m,$$
$$x \in R^{n_1},\ x_j \geq 0,\ j = 1, \ldots, n_1,$$
$$y \in Z^{n_2},\ y_j \text{は整数},\ j = 1, \ldots, n_2,$$
$$l_j \leq y_j \leq u_j,\ j = 1, \ldots, n_2,\ l_j, u_j \in Z,$$
$$a_{ij}, b_{ij}, e_i, c_{ij}, d_{ij}, f_i \text{は実数定数}.$$

実際問題では整数変数 y_j の値は 0, 1 のみを取る場合が多いが，ここでは一般的な場合を扱っている．

3 逐次分枝限定法による解法

上で与えた混合整数計画問題を部分問題に分割するために，またその最適解の評価値を得るための準備として，問題 ***ILP*** の連続緩和問題，つまり次の様に整数変数を実数変数に置き換えた問題 ***LP*** を考える．

Problem – *LP*

実数変数 x_j, y_j に関する目的関数

$$z = \sum_{j=1}^{n_1} p_j x_j + \sum_{j=1}^{n_2} q_j y_j$$

を，次の線形制約条件の下で最小化せよ：

$$\sum_{j=1}^{n_1} a_{ij} x_j + \sum_{j=1}^{n_2} b_{ij} y_j \geq e_i,\ i = 1, \ldots, l,$$
$$\sum_{j=1}^{n_1} c_{ij} x_j + \sum_{j=1}^{n_2} d_{ij} y_j = f_i,\ i = 1, \ldots, m,$$
$$x \in R^{n_1},\ x_j \geq 0,\ j = 1, \ldots, n_1,$$
$$y \in R^{n_2},\ y_j,\ j = 1, \ldots, n_2,$$
$$l_j \leq y_j \leq u_j,\ j = 1, \ldots, n_2,\ l_j, u_j \in Z,$$
$$a_{ij}, b_{ij}, e_i, c_{ij}, d_{ij}, f_i \text{は実数定数}.$$

LP の解はシンプレックス法で求めることができ，その整数変数が整数値を取る場合にはそれで解が決定する．そうでない場合には仮に整数変数 y_s の取る値が非整数値 $\bar{y}_s$ だったとして，その両隣の整数を区間境界とする二つの異なる制約条件 $l_s \leq y_s \leq [\bar{y}_s]$, $[\bar{y}_s] + 1 \leq y_s \leq u_s$ を既にある制約に付け加えて，元の問題を二分する．この手続き (分枝操作) を繰り返し行い探索空間を分割して行くことによって，部分問題の制約条件はその度ごとに区間制約が付加されて，区間 $([\bar{y}_s], [\bar{y}_s]+1)$ の曖昧さが除かれ，ある深さに達すると連続解が整数解となるものが得られる．

さて問題空間を上のように分割するだけでは整数値ごとの列挙に過ぎないが，既に整数解が得られていて，ある部分問題がそれより良い目的関数値を与えないことを保証する手立てがあるならば，それを探索せずに，より良い目的関数値を与える可能性のある問題のみを探索すれば良いことになる．混合整数計画問題についてはこの様な枝刈りをする基準として，連続緩和問題の最適解を用いることができる．全ての部分問題について連続解は，その部分問題の実際の整数解よりも良い目的関数値を与えるので，連続解の目的関数値が既に得られている整数解よりも悪い部分問題は最適値を与え得なく，探索不要となることが分かる (限定操作)．

この様にして分岐と，目的関数値による分枝の限定操作を繰り返して最適解を見つけるのが分枝限定法である．分岐して得られる分枝を探索ノードを呼ぶことにする．

3.1 逐次アルゴリズム

Step 0 初期設定
最初の問題 ***ILP*** を ILP_0 として，探索すべきノードの集合 $\mathcal{N}$ を $\{ILP_0\}$ とし，連続緩和問題 LP_0 を解く．整数解が得られたなら Step5 に行き，そうでなかったなら，暫定解 z を ∞ と置き，Step1 に行く．

Step 1 分岐ノードの選択
$\mathcal{N} = \emptyset$ ならば Step5 に行く．

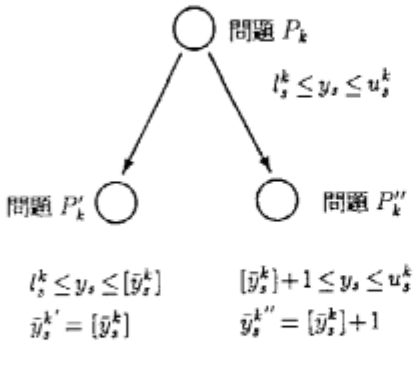

図 1: ノード (部分探索問題) の分岐

$\mathcal{N} \neq \emptyset$ ならば，ヒューリスティックスに従って次に分岐すべきノード ILP_k を $\mathcal{N}$ から選ぶ．

Step2に行く．

Step 2 分岐変数を選択して分岐する

ヒューリスティックスに従って，ILP_k において分岐操作を行う整数変数 y_s を選び，それに関して分岐を行う．得られるノードを $ILP_{k'}$，$ILP_{k''}$ とする．

Step3に行く．

Step 3 二つの分岐を連続緩和

連続緩和問題 $LP_{k'}$ と $LP_{k''}$ をシンプレックス法で解く．Step4に行く．

Step 4 二つの分岐を探査

分岐 $ILP_{k'}$ に対する $LP_{k'}$ が解を持たないか，あるいはその解 $z_{k'}$ が暫定解より良い値を与えない，つまり $z_{k'} > \bar{z}$ であった場合は，分岐を停止する (限定操作)．

解 $z_{k'}$ に対して，それを与える点 $(\bar{x}^{k'}, \bar{y}^{k'})$ の $\bar{y}$ が整数で，しかもこれまでの暫定解より良い値を与える，つまり $z_{k'} < \bar{z}$ であった場合は，$\bar{z} = z_{k'}$，$\bar{x} = \bar{x}^{k'}$，$\bar{y} = \bar{y}^{k'}$ とする (暫定解の更新)．

$(\bar{x}^{k'}, \bar{y}^{k'})$ が整数解でなく，しかも暫定解より良い値を与える場合は，分岐ノードの集合に付け加えて，$\mathcal{N} := \mathcal{N} \cup \{ILP_{k'}\}$ とする (さらに探索すべき分岐)．

$ILP_{k''}$ についても同じことを行い，Step1に行く．

Step 5 最終操作

$\bar{z} \neq \infty$ ならば，暫定解 $(\bar{x}, \bar{y})$ を最適解とする．

$\bar{z} = \infty$ ならば，問題 ILP は解を持たない．

3.2 分岐のためのヒューリスティックス

逐次的な探索過程において，探索木の辿り方を決定する重要な要素として次のものがある：

1. 分岐操作で生成されている部分問題 (ノード) の集合からどれを優先して次の分岐操作を行うか．
2. 分岐操作を行う変数の選択．

これらの選択は，全探索空間の内のできるだけ小さな部分の探索で最適解が見つけられる様になされることが望ましい．本研究では，OR においてこのために効果的と考えられているヒューリスティックスの最良の一つ ([今野，鈴木 82]) を用い，並列アルゴリズムを構成する基礎とした．ここでは簡単にそれを説明する．

部分問題の選択

深さ優先則と評価値優先則の両方を合わせた方法を用いる．まず分岐過程で各整数変数 y_j に対して擬似コスト $p_{up}(j)$，$p_{dn}(j)$ を計算する．これは連続緩和問題の解の近傍で，その整数変数を増加，あるいは減少させた時の連続最適解の増加率である．次に各ノードのヒューリスティック評価値 $h(ILP_k)$ を計算する．これは

$$h(ILP_k) = \bar{z}_k + \sum_{j=1}^{n_2} \min\{p_{up}(j)(1-f_j), p_{dn}(j)f_j\},$$
$$f_j = \bar{y}_j^k - [\bar{y}_j^k],$$

で定められる．

ノード ILP_k が $ILP_{k'}$ と $ILP_{k''}$ に分岐したとする．

n-i. 左右の分岐の少なくとも一方が未分岐である時には，二つの中から評価値 $h(ILP)$ の小さな方を次の分岐ノードに選ぶ (深さ優先)．

n-ii. 左右の分岐の両方が分岐停止の時には，

- a. まだ暫定整数解が求まっていなければ，最も最近に分岐したノードを選ぶ (深さ優先)．
- b. 既に暫定整数解が求まっていれば，最も評価値の良いノードを選ぶ (評価値優先)．

分岐変数の選択

ノード ILP_k で次に分岐する変数を選ぶ場合には，

v-i. まだ暫定整数解が求まっていなければ，$(\bar{x}^k, \bar{y}^k)$ で整数値を取らない y_j^k に対して，評価値の左右の増加量の差が最大のものを選ぶ，つまり

$$\max_j \{|p_{up}(j)(1-f_j) - p_{dn}(j)f_j|; \quad f_j \text{は非整数}\}$$

の値で計った優先順位に従って選ぶ．

v-ii. 暫定整数解が既に求まっていれば，$(\bar{x}^k, \bar{y}^k)$ において整数値を取らない y_j^k に対して，評価値の左右の増加量の小さい方が最大のものを選ぶ，つまり

$$\max_j \{\min\{p_{up}(j)(1-f_j), p_{dn}(j)f_j\}; \quad f_j \text{は非整数}\}$$

の値による優先順位に従って選ぶ．

4 分枝限定法の並列化

本研究ではICOTで開発された並列マシンMulti-PSIとその上で利用できる並列言語KL1を用いて，上で述べた分枝限定法の並列化を行った．Multi-PSIは16台あるいは64台のプロセッサが2次元メッシュ状にネットワーク結合された分散メモリの並列マシンである．データ転送速度はどのプロセッサ間でも同一である([Taki 88])．

並列化の方針としては，分枝過程で生成される複数の分枝を異なるプロセッサに割り当てて並列に探索を進め，それぞれの逐次探索プロセスは互いに枝刈りのための情報を交換し合って，全体としての枝刈りを進める．この場合探索空間をなるべく均等に並列プロセスに割り振って良い負荷分散を行うことと，逐次アルゴリズムが行っていた枝刈りを各探索プロセスを通じて同時に行うことが重要となる．

次にこれらを具体的にどの様に実現したかを述べる．

負荷分散

探索木の分岐に伴って生成される探索プロセスを異なるプロセッサに分配する際に，分岐一回毎に別プロセッサに探索プロセスを投げていたならばデータ転送コストが掛かり過ぎるので，全プロセッサに仕事が行き渡ったなら，それぞれの仕事を一つのプロセッサで実行する方が良いと考えられる．

そこでまず上のアルゴリズムの探索木において一定の深さ d まで分岐して，そこで生成される 2^d 個(それまでに枝刈りされなかった場合)の子問題を複数のプロセッサに投げる．これらの子問題に属する部分探索木の大きさは予め知ることはできないが，ここではサイクリックに順次異なるプロセッサに分散し，それぞれのプロセッサは複数の子問題を割り当てられる．従って負荷分散は静的に行われる．そしてそれぞれのプロセッサ内では，それ以降の深さの探索を逐次に行う．現在は子問題の生成を一台のプロセッサで行っている．

枝刈りのための探索ヒューリスティックス

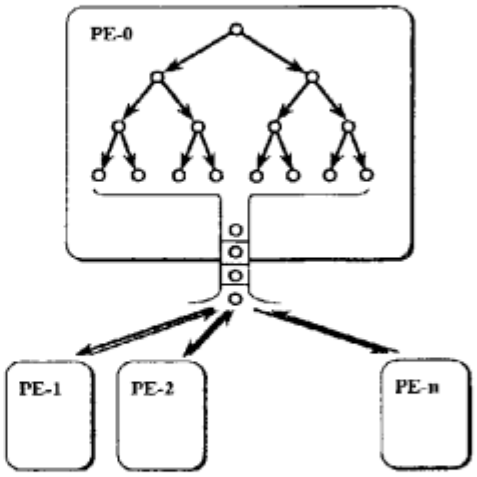

図 2: 並列プロセスの生成と負荷分散

並列に実行する場合に一つ一つのプロセッサには複数の子問題が割り当てられる訳であるが，それらに対して枝刈りによる探索の効率化を計るために，逐次アルゴリズムのノード選択戦略および分枝変数選択戦略をそのまま採用し，それぞれのプロセッサ内でこれを用いた逐次探索を行う．その戦略は3.2で述べたように深さ優先と評価値優先の双方を用いるが，それぞれのノードの持つ評価値に従った評価値優先戦略を用いる部分に関しては，KL1言語が提供しているプライオリティ制御機能を用いて実現した．つまり複数個ある探索プロセスたちに対して，それ自身のノードの評価値に従ったゴール・プライオリティを割り当てて，優先度の高いプロセスから順に実行させている(KL1言語ではプログラム中の任意のゴールに対して，実行優先度を予め割り当てることができる [Chikayama 88])．

大域データの転送

探索領域を異なるプロセスに分けて仕事を分散させたとしても，一つのプロセスで得られる探索情報を他のプロセスで生かして枝刈りを促進することができなかったならば，最終的に最適解を得るまでに，他のプロセスからの情報があったおかげで探索しなくて済んだ領域まで探索しなければならなくなり，探索の高速化を鈍化させたり逆に遅くなったりする影響が出る．

そこで並列に走っている複数個の逐次探索プロセスの間で，暫定整数解を大域データとして転送し合い，できるだけ最新の暫定解によって枝刈りを行うようにする．つまり枝刈り情報をなるべく早い段階

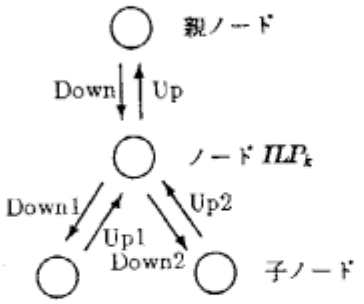

図 3: ノード間の暫定解報告ストリーム

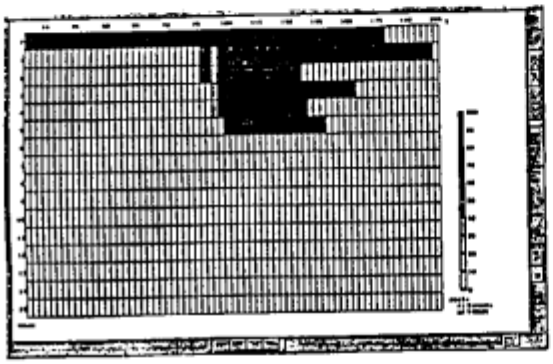

図 4: PE の実行状況の時間推移 (ParaGraph)
横軸：時間，縦軸：プロセッサ番号

で別プロセッサに転送し，並列なプロセス間に渡って枝刈りを行うことが重要になる．これは大域データを転送するゴールの優先度を，実際に探索を行うゴールよりも高くすることによって実現している．

5 実験結果

上に述べた並列アルゴリズムを KL1 プログラムで実現し，混合整数計画問題の例題であるジョブショップ・スケジューリング問題を解いて実験した．4 ジョブ 3 機械の問題に対して，プロセッサ台数の増加による速度向上，最適解を得るまでに探査した探索ノード数を示し，さらに並列プログラム実行解析ツール ParaGraph による測定結果を示す (図 4，図 5)．

プロセッサ台数	1	2	4	6
速度向上	1.0	1.5	1.9	2.3
探索ノード数	201	218	233	334
枝刈りノード数	101	95	99	109

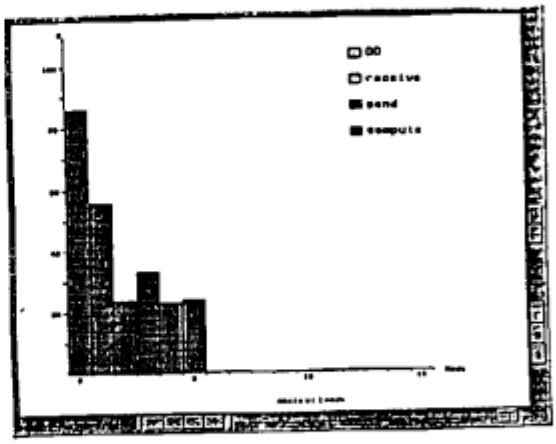

図 5: プロセッサの稼働率 (ParaGraph)
横軸：プロセッサ，縦軸：稼働率

まずこの例題に関しては，最初に分岐する準備としてシンプレックス法で LP 問題を解く部分に 15.7% の計算コストを費やしている (逐次の場合) ので，並列効果が得られるのは残り 84.3% であり，速度向上は高々 6 倍と見積もられる．一般的に問題サイズに対して探索空間の大きさは指数関数的に増えると考えられ，一方シンプレックス法のコストは平均的にサイズの線形オーダーで増大する．従って問題サイズを大きくすれば，初期のシンプレックス法に要する時間は全体の探索に要する時間に比べて十分小さくなるため，並列効果も増大することが期待できる．

図 4の ParaGraph から分かるように，プロセッサに仕事が与えられている間は稼働率が良いが，各探索プロセスの終了時間のばらつきによって並列効果が低下していることが分かる．枝刈りを伴うアルゴリズムについては，探索が進むに連れて残っている探索プロセスの数は減るので，各プロセッサが受け持つ部分木のサイズは不均衡になって行く傾向がある．現在はこれに対して対処していないが，探索の後半で仕事のなくなったプロセッサが生じた際にそこで仕事の要求を出して，まだ稼働しているプロセッサで生成される探索プロセスを新たに暇なプロセッサに分配することによって，これを改善することが考えられる．

図 5からデータ転送には殆ど時間が掛かっていないことも分かる．

また上の表を見ると，プロセッサ台数の増加に伴って最終解に達するまでの探索ノード数が増加していることが分かる．これは複数プロセッサに分散された探索プロセスが同時並行で走るために，逐次に探

索していた場合ならば枝刈りされていたはずのノードの探索が進んでしまうことによるものであり，この見込み計算の割合は問題の種類と探索空間の形によって決定される．

また表には上げなかったが，静的負荷分散を行う探索木の深さを変えて実験した結果，負荷の均一化を狙って深さを大きくしても仕事の粒度は揃わず，プロセッサに割り当てられる仕事には依然としてばらつきが生じるため，負荷の均衡状態には変化がさほど見られないことが分かった．つまり探索空間の枝刈りが進むにつれて，木の一部だけに探索が集中する傾向がある．従って枝刈りがある程度の頻度で発生する問題については，初期の負荷分散はあまり深くないところで行い，上で述べた様な，暇になったプロセッサからの要求に応じて部分木をプロセッサに投げる方策を取るのが有効性を持つと予想される．

6 まとめ

混合整数計画問題を解くための分枝限定法について，探索枝を複数プロセッサへ静的に分散して並列化を行い，分散メモリ並列マシン上で速度向上を実測した．初期に動的負荷分散を行うことよりは，探索が進んで枝刈りによって起こる各部分問題の大きさの不均衡を動的に分散することによって，速度向上を改善する余地があることが分かった．

次にこれからの課題として，(a) 浮動小数点演算が現在の環境では使えないために，実験できる問題規模が限られているが，この演算が行える並列マシン PIM で実験し，より規模の大きい問題を試みる，(b) プロセッサの平均稼働率を上げるために，暇になったプロセッサからの要求で負荷を動的に分散してみる，(c) 異なる種類の整数計画問題を試して見込み計算の発生量を計る，(d) 異なる探索ヒューリスティックで実験する，と言うことが考えられる．

謝辞

本研究を行うに当たり，研究の機会を与えて下さった相場亮 ICOT 第 4 研究室長代理，有益な助言を下さった東芝システム・ソフトウェア技術研究所の永井保夫氏および ICOT 第 7 研究室の市吉伸行氏，清慎一氏に感謝します．

参考文献

[Benichou et al 71] M.Benichou, L.M.Gauthier, P.Girodet, G.Hentges, G.Ribiere, O.Vincent, Experiments in Mixed-Integer Linear Programming, *Mathematical Programming* 1 (1971) 76-94.

[今野，鈴木 82] 今野，鈴木編. 整数計画法と組み合わせ最適化，日科技連,1982 年.

[沖 他か 89] 沖廣明，瀧和男，清慎一，古市昌一. マルチ PSI における並列詰め碁プログラムの実現と評価，並列処理シンポジウム JSPP 1989, 351-357.

[Taki 88] K. Taki, The parallel software research and development tool: Multi-PSI system, in *Programming of Future Generation Computers*, North-Holland, 1988, pp.411-426.

[Ueda-Chikayama 90] K.Ueda and T.Chikayama. Design of the Kernel Language for the Parallel Inference Machine, *The Computer Journal*, Vol.33, No.6, December 1990, pp.494-500.

[Chikayama 88] T. Chikayama et al. Overview of Parallel Inference Machine Operating System (PIMOS), *Proceedings of the International Conference on Fifth Generation Computer Systems 1988* Vol.1, pp.230-251.

[Quinn 90] Michael J. Quinn. Analysis and Implementation of Branch-and-Bound Algorithms on a Hypercube Multiprocessor, *IEEE Trans. on Computers*, Vol.39, No.3, March 1990, pp.384-387.

[Li,Wah 86] Guo-Jie Li and Benjamin W. Wah. Coping with Anomalies in Parallel Branch-and-Bound Algorithms, *IEEE Trans. on Computers*, Vol.35, No.6, June 1986, pp.568-573.