論　文

並列分枝限定法に対するビジュアリゼーションシステム

大西　克実 †　　榎原　博之 ††　　中野　秀男 †††

A Visualization System for Parallel Branch-and-Bound Method

Katsumi ONISHI†, Hiroyuki EBARA††, and Hideo NAKANO†††

あらまし　分枝限定法は，組合せ最適化問題の解法として適用される計算手法の一つである．ところが問題の規模が大きくなると，この解法の振舞いは複雑になり，実行時間が指数関数的に増加する．このため，分枝限定法の並列化という手法が提案されている．しかし，並列分枝限定法のように複雑な振舞いをするアルゴリズムの問題点を的確に把握し，対策を考えることは非常に困難である．
そこで本論文では，並列分枝限定法のアルゴリズムの振舞いをアニメーション技法を用いて画面上に表示するシステム **Sapal-BB** を実現し，これを使用することで，アルゴリズムの有効性の判断を容易にする．本システムでは，分枝限定法の実行過程で作られる分枝図を動的に表示する．生成される分枝図は全体を一度に見ることが困難なため，3つにレベル分けし表示している．最後に二種類の並列分枝限定法の実現を本システム上で実行し，その結果の比較・検討を **Sapal-BB** の機能を利用して行う．

キーワード　組合せ最適化問題，分枝限定法，並列処理，視覚化

1. まえがき

分枝限定法は，組合せ最適化問題の中でも特に難しい問題の解法に適用される計算手法の一つで，実用的にも有効な手法の一つである [1], [2]．しかし，この分枝限定法を用いても解を得るために必要な計算時間は，問題の規模が大きくなるにつれて指数関数的に大きくなる傾向がある．このために，分枝限定法の並列化という手法が考え出され，いろいろな並列化手法が提案されている [3]〜[5].

分枝限定法では問題を次々と分解し，その過程で得られる与えられた問題についての情報を，次の段階での操作に利用する．このために分枝限定法のアルゴリズムが次の段階で行う操作を，正確に予測することは不可能である．分枝限定法のアルゴリズムの解析は今までにも行われているが [6]〜[11]，それらの解析においては単純化のために多くの仮定が行われている．さらに平均的な効率を予測するものではなく，最悪の場合を想定している場合が多い．

† 大阪大学工学部通信工学科
2–1 Yamadaoka Suita-shi Osaka, 565 Japan.
†† 関西大学工学部情報処理教室
3–3–35 Yamate-cho Suita-shi Osaka, 564 Japan.
††† 大阪市立大学学術総合情報センター
3–3–138 Sugimoto Sumiyoshi-ku Osaka-shi, 558 Japan.

分枝限定法の並列化のように逐次型のアルゴリズムを並列化した場合，並列型のアルゴリズムと逐次型のアルゴリズムが同じ解を出しても，これら二つのアルゴリズムの実行過程はまったく異なることがあり得る．また，並列型にした場合，逐次型では考慮する必要がなかった条件も考える必要がある．これらのことから並列化したアルゴリズムの動作を予測することは困難である．

ところで，最近の新しい研究分野としてアルゴリズムアニメーション（Algorithm Animation）と呼ばれる分野がある [12], [13]．これはアルゴリズムによって問題が解かれる過程を逐次ディスプレイ画面上に表示し，アルゴリズムが実際の問題を解いていく様子を動画（Animation）として見せる技法である．この手法を用いると，紙の上に書かれただけでは理解しにくいアルゴリズムの全体的な流れも容易に把握できるようになる．この考え方を振舞いの複雑な並列分枝限定法に適用すればアルゴリズムの開発に有効な手段を提供できると考えられる．

そこで本論文では，アルゴリズムの開発者が並列分枝限定法の実装を比較・検討する際の支援ツールとして利用することを念頭に置き，上記アルゴリズムアニメーションの手法を適用し，並列分枝限定法の振舞いをワークステーションの画面上に逐次表示するシステ

ム **Sapal-BB**（Simulator and Animator of Parallel Algorithms for Branch-and-Bound method）を実際に作成する．さらに，本システムの有効性について実行結果から検討を行う．

2. 並列分枝限定法

2.1 分枝限定法

分枝限定法（branch-and-bound method）の基本的な考え方は，図1に示したように，直接解きがたい問題に表れる整数変数を固定し，より解きやすい部分問題（partial problems）に分解する．そしてこの部分問題をすべて解くことにより，与えられた問題を解くという考え方である．

部分問題に分解する操作のことを分枝操作（branching operation）とよぶ．可能なすべての分枝操作を行うだけでは単なる列挙法と異なるところがなく効率が良くならない．そこで，不必要な部分問題の生成を抑えるために，整数変数をすべて固定しなくても可能解が求まる場合，あるいは着目している部分問題からは最適解が求まらないことが結論できる場合には，新たな分枝操作を適用しないようにする．この手続きを限定操作（bounding operation）とよぶ．

分枝操作も限定操作もなされていない問題は活性部分問題と呼ばれる．これら活性部分問題の中から次に分解すべき部分問題を選び出す操作を探索（search）とよぶ．このようにして探索，分枝操作，限定操作を繰り返していくといくつかの可能解（feasible solution）

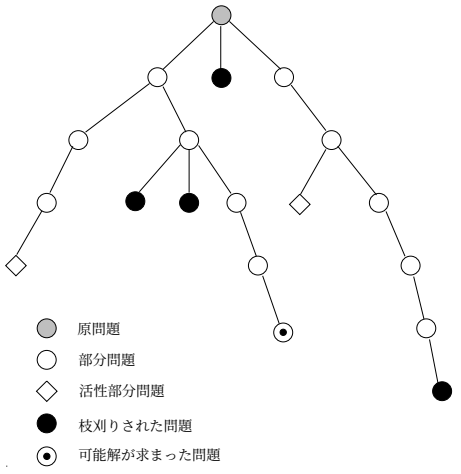

図1 分枝限定法の分枝図
Fig. 1 Branch figure for branch-and-bound method

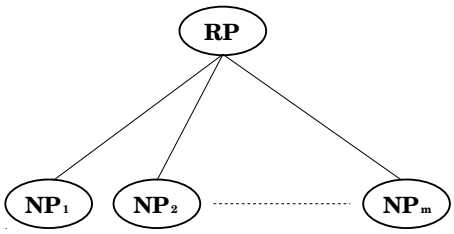

図2 星状結合ネットワーク型マルチプロセッサシステム
Fig. 2 Star-shaped multi-processor system

が得られる．分解の過程を与えられた原問題からたどっていくと，分枝操作の進み方を示す分枝図が得られる（図1）．実行中のある時点で最適と考えられる解を暫定解と言い，すべての部分問題を処理した時の暫定解が与えられた問題の最適解になる．

2.2 並列計算機のモデル

本報告で考える並列計算機の論理的なモデルは，図2に示すような星状結合ネットワーク型マルチプロセッサシステムである．ＲＰ（Root Processor）は親プロセッサを示し，ＮＰ$_i$（i =1,2,...,m）（Node Processor）は子プロセッサを示す．このモデルにおいて，各プロセッサは共有メモリを持たず，専用記憶装置だけを持つとしている．プロセッサ間のデータ伝送はネットワークを通じて行われる．このモデルを実際にはEthetnet で接続された計算機上で UNIX のプロセス間通信の機能を用いることにより実現している．

2.3 並列分枝限定法のアルゴリズム

上述したようなマルチプロセッサシステム上で，並列分枝限定法を実現する方法として，本報告では次のような方法を採用している．但し，視覚化を実現する際には，今回の方法とは異なった並列化手法を採用する場合についても考慮している．

親プロセッサは与えられた問題のデータと解の管理，および子プロセッサへの問題の割当てを行う．子プロセッサは親プロセッサから与えられた活性部分問題を初期問題と見なし，探索と分枝操作，限定操作を繰り返し行う方法である．この方法では，親プロセッサに関しては記憶領域および通信時間が節約でき負荷は軽くなる．しかし，各子プロセッサでは探索終了時間に差が生じるため，空き状態のプロセッサができてしまい，子プロセッサを効率良く利用できなくなる．このため，親プロセッサが部分問題の再割当を行う必要がある．

組合せ最適化問題の例としては，次のような０－１

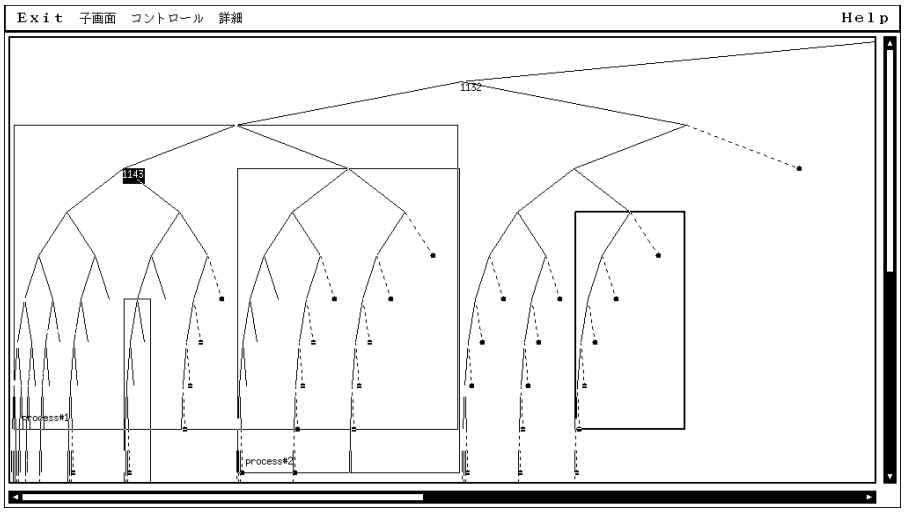

図 3 親画面の描画例
Fig. 3 An example view of parent level

ナップサック問題を対象として取り上げている．この問題は ”有限集合 $\mathbf{U} = \{1,2,\ldots, \mathbf{i}, \ldots,\mathbf{n}\}$ の各要素が，非負整数のパラメータ $\mathbf{a_i}$ および $\mathbf{c_i}$ を持つとき，ある正整数 $\mathbf{b}$ に対して $\sum_i \mathbf{a_i} \cdot \mathbf{x_i} \leqq \mathbf{b}$ なる制約条件下で，目的関数 $\sum_i \mathbf{c_i} \cdot \mathbf{x_i}$ を最大にする０－１分枝変数（branching variable）$\mathbf{x_i}$ を求める問題 ”と定義される．

3. Sapal-BB の概要

１章で述べたように，本研究では，ＴＡＮＧＯ，ＺＥＵＳなどのツールで実現されているアルゴリズム・アニメーション技法を並列分枝限定法に適用し，アルゴリズムの開発支援ツールとして利用することを目標としている．これまでにも，一般的な並列アルゴリズムの評価・開発支援ツールに関しては，さまざまなシステムが提案されている [16], [17], [19]．これらのツールでは，並列計算機を構成する各ホストの負荷情報およびホスト間の通信状況を図的に表示し，負荷が特定の計算機に集中するような不具合，もしくは通信遅延により並列処理が停滞する状況などを理解しやすく示している．

前節で述べたようなアルゴリズムを実装した並列分枝限定法を実行している並列計算機に対してこのようなツールを利用した場合，並列処理が停滞するような通信は，親プロセッサが部分問題の再割当を行う場合だけであり，それ以外のほとんどの時間に関して，子プロセッサの処理が滞ることはない．

並列分枝限定法の検討・評価を行う際に，はじめに考慮されなければならない点は，不必要な分枝木上での探索を行っていないかどうかである．このような場合，一般的な並列アルゴリズムの評価・開発のために利用されるようなツールでは不充分である．

そこで，本研究では，ＴＡＮＧＯ，ＺＥＵＳなどのツールで実現されているようにアルゴリズム内部で利用されるデータ構造を視覚化する方法を採用する．２章で述べたように分枝図は，分枝限定法において探索の進み方を保持する内部データ構造であると見なせる．**Sapal-BB** では並列分枝限定法のシミュレータの実行と同時にこの分枝図を動的に視覚化する．

本システムは，各プロセッサによって並列に探索が行われる様子だけではなく，枝刈りや部分問題の再割当が行われる状況，さらに数値情報も含めて使用者に理解しやすく示し，実行中のアルゴリズムの理解を助ける事を目標に設計を行った．分枝限定法の実行中に構成される分枝図は規模が大きくなる傾向があり一枚の画面中に表示するだけでは，有効な情報を読み取ることが難しくなる事が予想される．そこで **Sapal-BB** では X window System で提供されるマルチウィンドウ環境を利用し，複数の表示画面を用意することで描画される分枝図から情報を読み取り易くする．分枝図全体を概観するための画面，各子プロセッサが探索を行っている周囲を見る画面，数値情報も含めて細かく

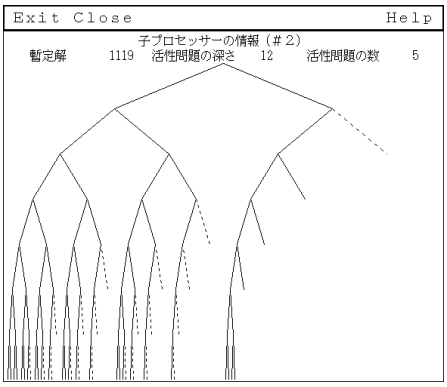

図 4　子画面の表示例
Fig. 4　An example view of child level

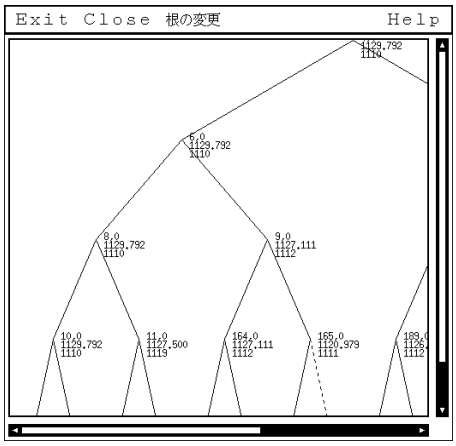

図 5　孫画面の表示例
Fig. 5　An example view of grandchild level

分枝図を見る画面を用意する．また，暫定解の時間的な更新の様子を表示する画面も用意している．以下では，このように **Sapal-BB** で用意されている複数の画面の機能について述べ，本システムの構成，実行例に基づく評価についても述べる．

4. Sapal-BB の機能

4.1 親　画　面

親画面には分枝図全体が描かれる．その例を図 3に示す．

- 実際に画面上に描画される領域は，分枝図全体の一部であるがスクロールバーを利用し画面上に描画される領域を変更することが可能である．
- 図中の点線は，枝刈りされた問題を表している．四角の枠は子画面が表示している範囲を表している．また，どのプロセッサが処理した部分問題であるかを理解しやすくするために，各プロセッサごとに処理した枝を異なる色に設定することが可能である．
- 図中の数字は子プロセッサから新たに親プロセッサに報告された可能解を表しており，その位置によってその可能解が得られる深さなどを知ることができる．さらに，その時点での暫定解は，白黒を反転させて強調している．
- 分枝図が描画されている領域の任意の点をマウスでクリックすると孫画面が表示され，分枝図中には孫画面が表している領域を示す四角形が描かれる．
- “**子画面**” メニューを使用すると各プロセッサに対して用意される子画面の表示・非表示を切り替えられる．
- “**コントロール**” メニューには “シミュレータ実行停止” と “シミュレータ実行継続” の二つのサブメニューがある．メニューを選択することにより，シミュレータの実行を制御することが可能である．
- “**詳細**” メニューからは，暫定解の変化を示すグラフや，各プロセッサが処理した部分問題の数を調べるための表を表示できる．

4.2 子　画　面

子画面には各々のプロセッサが処理中の問題を葉とする分枝図の一部が描かれる．この画面は新たな活性問題ができた場合や，枝刈りが生じた場合に書き換えを行う．しかし，パラメータの設定により，常に書き換えをさせるのではなく，数回に一度書き換えをさせることも設定により可能である．描画例を図 4に示す．

- 画面中には，この画面がどのプロセッサに対応しているかを示すラベルがある．その下に各プロセッサごとの情報が示される．一つ目はそのプロセッサが現在知っている暫定解であり，二つ目は現在表示されている葉の分枝図全体中における深さである．最後の数字はそのプロセッサが現在管理している活性部分問題の数である．

4.3 孫　画　面

孫画面にも分枝図の一部が描かれる．この画面は親画面の分枝図上をマウスでクリックすることで表示され，実際に孫画面で観察可能な領域は親画面の分枝図上に描かれた四角形で表される．この四角形は子画面の領域を表す四角形と区別するためより太く，別の色で描かれる．描画例を図 5に示す．

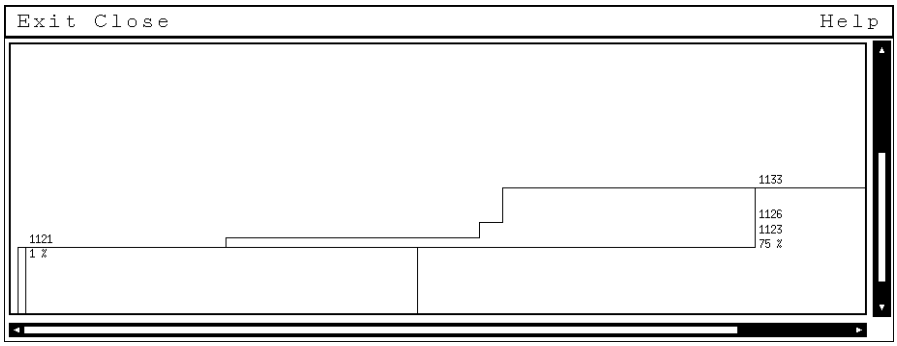

図 6　暫定解の変化を表すグラフ
Fig. 6　An example view of the incumbent value graph

● 孫画面は親画面と同様に，観察可能な領域すべてを表示はしない．表示されていない領域はスクロールバーを用いたり，孫画面全体の大きさをウィンドウマネージャのリサイズ機能を使って変更することで表示できる．

● 各節点の脇に振られた数字の意味を以下に示す．

一段目：“，”で区切られた二つの数字がある．第一の数字をたどることによりどのように変数の固定が進んできたかについて知ることができる．第二の数字はその問題が枝刈りされた順序を表している．この数字が“０”である節点の問題は枝刈りがなされていないということを表す．

二段目：該当する部分問題の上界値を表す．

三段目：該当する部分問題の下界値を表す．

● “**根の変更**”メニューを選択することにより孫画面に表示されている部分木の根を変更することが可能である．サブメニュー“**上へ**”を選ぶと根がひとつ上の部分木を再描画し，親画面中の観察可能領域を示す四角形も上に移動する．“**左下へ**”，“**右下へ**”も同様に動作する．

4.4　その他の画面

その他の画面としては，各プロセッサにおいて暫定解が改善されていく様子を表すグラフを表示できる．描画例を図 6に示す．

図中の４桁の数字は可能解の値を表している．また“%”で表された数字は，横軸の左端がシミュレーションが始まった時，右端がグラフを表示させた時と考えた場合の経過時間の割合を表している．縦軸は暫定解の値を表している．

このグラフから，シミュレーションが始まってから可能解が見つけられるまでの経過時間を知ることができる．また，今回構成したシミュレータでは，可能解の値を他のすべてのプロセッサに対して，親プロセッサを通して報告するようにしているので，その値がすべてのプロセッサに伝わるために必要な時間についても推測することができる．

5.　Sapal-BB システムの構成

5.1　全 体 構 成

Sapal-BB は以下の図 7に示したように，アニメーションを行うプロセス（アニメータ）と，親プロセッサおよび子プロセッサを実行するプロセス群からなるシミュレータの二つの部分から構成されている．これらのプログラムはすべて UNIX 上でＣ言語を用いて作成している．ＧＵＩ（Graphical User Interface）を実現するためのツールとしては，X Window System と Motif Widget Set を用いている．通信を実現するための手段としては UNIX 上の Socket 機能を使用している．

X Window System では，ワークステーション上の入出力装置はＸサーバと呼ばれるプロセスが管理している．ユーザがプログラム中から画面に描画を行いたい場合，Ｘサーバに対してＸプロトコルを用いて通信を行うことで描画を要求する．

Sapal-BB では図 7に示したように，Ｘサーバと分枝限定法を実行するシミュレータの部分とは，アニメータによって，完全に分離されている．アニメータとＸサーバはＸプロトコルを用いてイベントの通知や描画要求などの通信を行う．シミュレータは，アニメータに対して，アルゴリズムが処理した仕事についての情報を報告する．このため，シミュレータのプロ

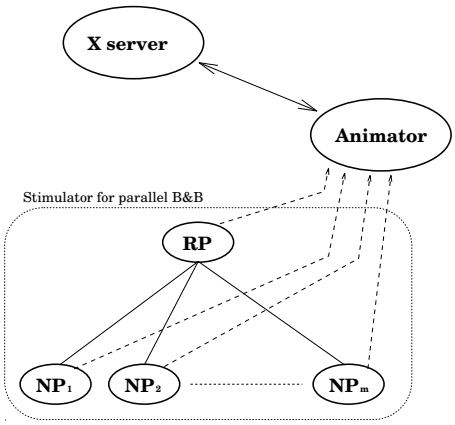

図 7　**Sapal-BB** の構成
Fig. 7　Composition of **Sapal-BB**

グラム中では，アニメータに必要なメッセージを送るだけで充分になる．すなわちシミュレータのプログラム中に，Xlib もしくは X Toolkit 等のプログラミングで必要になる関数の呼出しを記述する必要がない．

シミュレータからアニメータへの通信はソケットで構成した通信路を用いている．このため，図 7で示したすべてのプロセスを，別々の計算機で動作させる事も可能である．アニメータとシミュレータの間では，シミュレータの停止を実現するためにシグナル機能と呼ばれる UNIX に用意されているソフトウェア割り込みの機能も利用している．

5.2　シミュレータの構成

Sapal-BB で構成したシミュレータは親プロセッサと，１個以上の子プロセッサからなる構成である．子プロセッサの数についてはプログラムのコンパイル時に指定できるが，６章の例では視覚化する問題の規模や視覚化した時の見やすさなどを考えて子プロセッサ数を３台としている．

上述したように各プロセッサからアニメータに通信するときには，直接ソケットにそれらの情報を書き込むのではなく，通信用のサブルーチン群をあらかじめ用意し，それらを分枝限定法のアルゴリズム中に埋め込むことで実現している．このためアニメーションを行わないシミュレータが必要な場合，通信用のサブルーチン呼び出しと初期化時に行っているアニメータとの接続手続きをコンパイラのプリプロセッサ機能を用いて無効にすれば良い．

描画用のサブルーチンとしては，暫定解の変更・活性部分問題の数・部分問題の状態（活性・枝刈り・再割当）・処理中の部分問題の深さ・実行終了をアニメータに知らせる機能を用意している．

5.3　アニメータの構成

アニメータは，シミュレータ用のすべてのプロセッサとの接続を確立すると，アニメーション用ルーチンの実行を開始する．X Toolkit を用いたプログラムで必要になる初期化を行った後，Xサーバからのイベントを受け付ける状態になる．このままではシミュレータからの描画要求を受け付けられないため，X Toolkit で用意されている WorkProc 機能を利用する．この機能はXサーバからのイベントの通知がない時に，あらかじめ登録しておいたルーチンを処理するようにする機能である．これを用いてシミュレータの各プロセッサとアニメータとの通信路である複数のソケットを UNIX の select 機能を用い常に監視するようにし，メッセージがあれば，それに応じて適切な描画を行う．シミュレータから送られてくる各メッセージには，タイムスタンプを付加し，描画の順序に矛盾が生じるような時間的な問題が起きないように考慮している．

6.　実　行　例

以下では，２０個の０－１変数を持ち，各変数に対する係数は，特定の範囲の中からランダムに選んだナップザック問題を解いた場合の実行結果を例として考える．２章でも述べたとおり，今回実装したような並列化を行った分枝限定法では，初期に割当てた部分問題の終了時間が子プロセッサごとに異なるため，部分問題の再割当を行わなければ，空き状態の子プロセッサができることになり，プロセッサの利用効率が悪くなる．

再割当を行う部分問題の選び方としてはさまざまな方法が考えられるが，再割当を適切に実行しない場合，次々と親プロセッサと子プロセッサの間で部分問題の再割当が起こり，通信に費やす時間が多くなる．これは計算機の有効利用という点から好ましくない．この章では，再割当のための部分問題を選ぶときに，“分枝図上で根に近い部分問題から選ぶ方法（幅優先割当）”，“部分問題の下解値がもっとも小さい問題を選ぶ方法（最良下解値優先割当）” という二つの方法を **Sapal-BB** の終了時の親画面から比較する．

図 8は幅優先割当の実行結果，図 9は最良下解値優先割当の場合の実行結果である．どちらの結果も分枝図全体の一部であるが，同じ深さの部分を表示してい

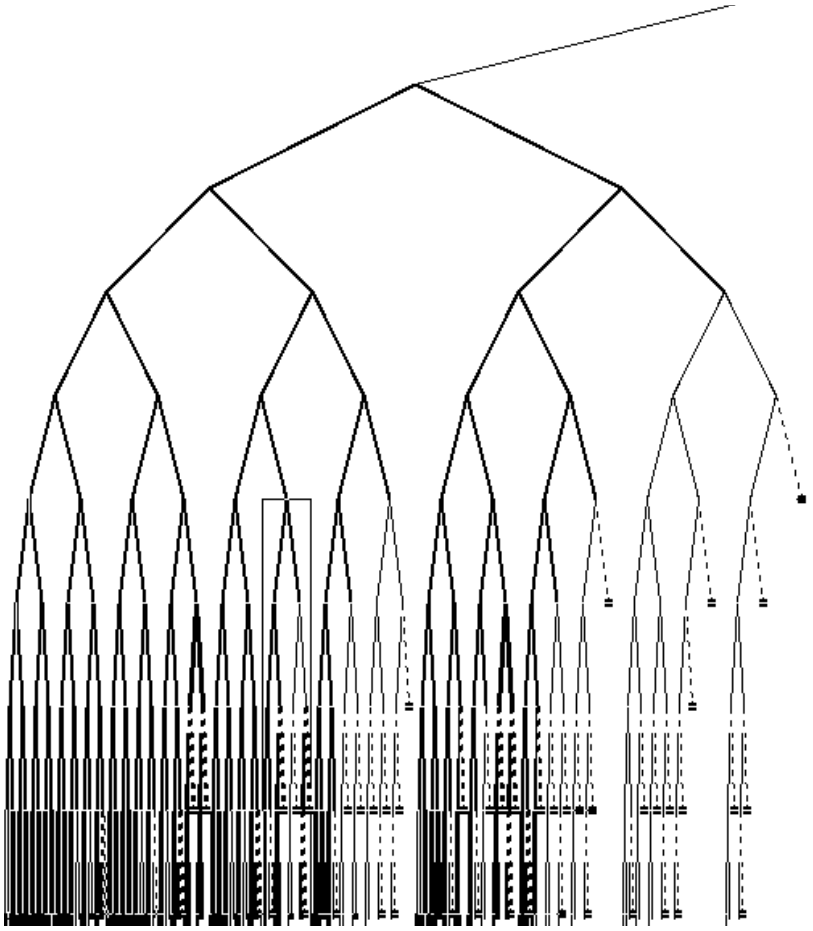

図 8　幅優先割当の場合の結果
Fig. 8　Result of breadth first assignment

る．実際に実行するときには，カラーディスプレイ上に表示し，部分問題を表す分枝木上の枝を処理したプロセッサに応じて色分けすることでどの部分木をどのプロセッサが処理したかを表示できる．ここでは，処理したプロセッサに応じて枝の太さを変更している．

図 8と図 9を比べると，最良下解値優先割当を取っている場合は，部分問題の再割当が分枝図上で右下の部分に見られるが，幅優先割当を取った場合右上の部分からも再割当が行われることが確認できる．分枝図上で下に位置する問題はより変数の固定が進んでいるために，上の方に位置する問題に比べて，それ以降の探索が早く終わることが多い．探索が早く終わると子プロセッサは次の部分問題を要求するから，再割当の回数が多くなる．このことから，各プロセッサを部分問題の処理に集中させるためには，幅優先割当を行うほうがよいことがわかる．

この例では，終了時の画面のみの比較を行った．本システムでは，実際には並列分枝限定法の実行過程における解の探索状況を時間を追って概観できる．また，各部分問題毎の数値情報などを詳細に調べることもできるため，アルゴリズムの評価・検討に有効に活用できる．これは，アルゴリズム・アニメーションにおける内部データの視覚化を，並列分枝限定法において分枝図の動的な視覚化として実装したためである．今回の実行例は，０－１ナップザック問題であるが，他の組合せ最適化問題に対する分枝限定法を検討する際にも本システムで用いた手法は有効であると考える．

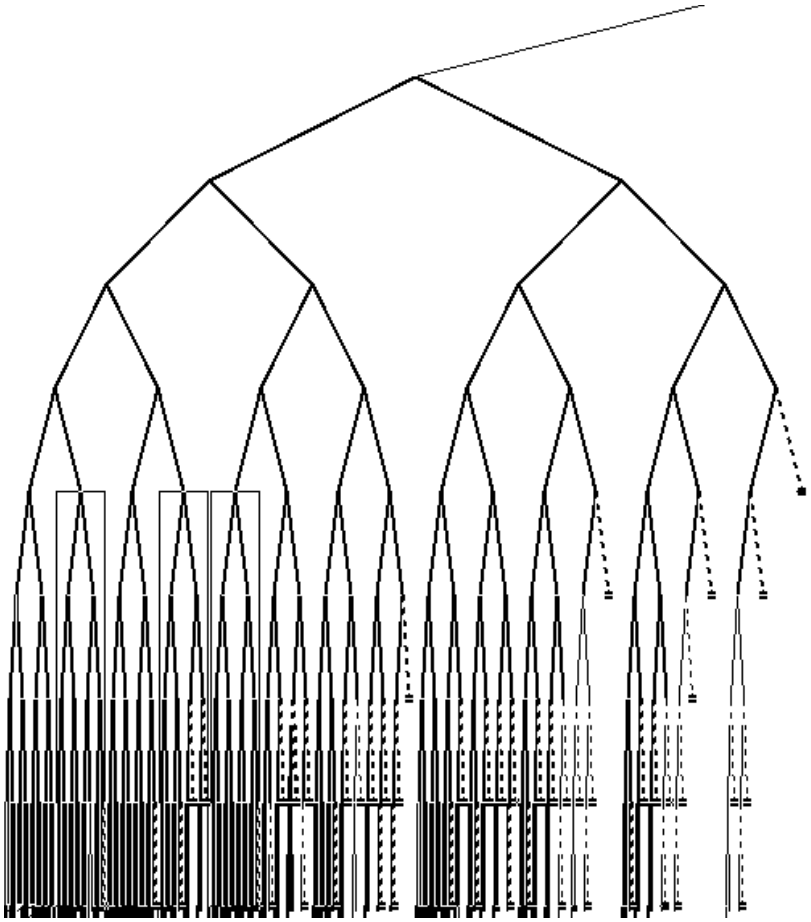

図 9　最良下解値優先割当の場合の結果
Fig. 9　Result of best first assignment

7. むすび

Sapal-BB では，並列分枝限定法の視覚化を０－１ナップザック問題を例として取り上げ実現した．

視覚化を実現するために，分枝限定法のシミュレータ本体から，アニメーションの要求を直接表示装置に行うのではなく，それらに関する機能はシミュレータから分離するようにした．具体的には，アニメータとシミュレータを UNIX のプロセスのレベルで分離した．このめ今回行った並列化とは異なるさまざまな種類の並列分枝限定法に対してもアニメーションを行うことが可能になっている．

シミュレータに視覚化の機能を付け加えるための変更をできるだけ少なくするためにアニメーション用のサブルーチン群を用意した．このため，シミュレータ用のプログラム中に関数呼出しを入れるだけで視覚化が可能となった．

描画される分枝図は，全体を大きく見るものから，具体的な数値を乗せた図まで親，子，孫と３レベルの画面を用意した．親画面中に子，孫の表示領域も示したので相互間の関係も把握できる．

Sapal-BB では一度実行が始まると終了まで使用者は指示ができないというものではなく，対話的にシミュレータの停止と再実行を指示することが可能である．しかし，アニメーションの表示の制御にはほかにも速度を任意の時点で変更したり，前の状態に戻すなどの制御もある．このようなより高度な要求に対しても考慮する必要がある．

さらに今後は，アニメータの機能を充実させるとと

もに，このツールの機能を使用して実際に並列分枝限定法のアルゴリズムを評価・開発することが考えられる．実際にアルゴリズムの開発に用いれば，その過程でこのツールの問題点や有効性などがより明らかになると考える．

謝辞　本論文の執筆に当たって適切な御指導・御助言を頂いた，大阪大学通信工学科，前田肇教授に深く感謝の意を表します．

文　献

[1] 茨木俊秀：“組合せ最適化問題”, 産業図書（1983）．
[2] 西川衛一, 三宮信夫, 茨木俊英：“ 最適化”, 岩波書店（1982）．
[3] Catherine Roucarirol：“ Parallel branch and bound algorithms - an overview”, Parallel and Distributed Alogorithm, M.Cosnard et al.(Eds), pp.153-163, North-Holland（1989）．
[4] H. Kitagami and H.Hara H. Yamanaka and T. Miyazaki：“Performance Evaluation for Parallel Mixed-Integer Linear Programming System”, IEICE Tech.Report, **91**, 130, pp.167-172（1991）．
[5] A. Taudes and T. Netousek：“ Implementing branch-and-bound algorithms on a cluster of workstations - A survey, some new results and open problems”, Proc. Parallel Algorithms and Transputers for Optimization, pp.79-102（1990）
[6] T.H.Lai and S.Shani：“ Anomalies in parallel branch-and-bound algorithms”, Comm.ACM., **27**, 6, pp.594-602（1984）．
[7] B.W.WAH and C.F.Yu：“ Stochastic modeling of branch-and-bound algorithms with best-first search”, IEEE Trans. Software Eng., **SE-11**, pp.922-934（1985）
[8] G.j.LI and B.W.WAH：“ Coping with Anomalies in Parallel Branch-and-Bound Alogorithms”, IEEE.Trans.Comput., **C-35**, 6, pp568-573（1986）．
[9] M.J.Quinn：“ Analysis and implementation of branch-and-bound algorithms on a hypercube multicomputer “, IEEE Trans. Comput., **C-39**, pp384-387（1990）．
[10] Myung K. Yang, Chita R. Das：“ Evaluation of a Parallel Branch-and-Bound Algorithm on a Class of Multiprocessors”, IEEE Trans. Parallel and Distributed Systems,, **5**, 1, pp74-86（1994）．
[11] R.Kan and H.Trienekens：“ A simulation tool for the performance evaluation of parallel branch and bound algorithm”, Math.Program., **42**, 2, pp.245-271（1988）．
[12] M.H.Brown：“ Exploring Algorithms Using Balsa-II”, IEEE.Trans. Comput., **21**, 5, pp.14-36（1988）．
[13] M.H.Brown：“ Algorithm Animation”, The MIT Press, London（1988）．
[14] M.H.Brown：“ ZEUS: A system for algorithm animation ”, Proc. of the IEEE'91 Workshop on Visual Languages, pp.4-9(Oct. 1991).
[15] J.T.Stasko：“ Understanding and Characterizing Software Visualization Systems”, Proc. of the IEEE'92 Workshop on Visual Languages, pp.3-10(Sep. 1992).
[16] M.T.Heath, J.A.Etheridge：“ Visualizing the Performance of Parallel Programs”, IEEE Software, **8**, 5, pp.29-39(Sep. 1991).
[17] T.J.LeBlanc, J.M.Mellor-Crummey, R.J.Fowler：“ Analyzing Parallel Program Executions Using Multiple Views”, Journal of Parallel and Distributed Computing”, **9**, 2, pp.203-217(Jun. 1990).
[18] C.E.McDowell, D.P.Helmbold：“ Debugging Concurrent Programs”, ACM Computing Surveys, **21**, 4, pp.593-622(Dec. 1989).
[19] B.Tourancheau, X.Vigouroux, M.G.VanRiek：“ The Massively Parallel Monitoring System a truly parallel approach to parallel monitoring”, Proc. of the Workshop on Performance Measurment and Visualization, pp1-17(Oct. 1992).

（平成 7 年 4 月 21 日受付，8 月 4 日再受付）

大西　克実　（学生員）

平 4 阪大・工・通信卒．現在，同大学大学院後期課程在学中．組合せ最適化問題の解法に関する研究に従事．

榎原　博之　（正員）

1958 年生．1982 年阪大・工・通信卒．1987 年同大学大学院博士（通信）課程修了．同年阪大工学部助手．1994 年関西大工学部専任講師となり現在に至る．組合せ最適化問題，計算幾何学，並列アルゴリズム等の研究に従事．工博．情報処理学会，IEEE，ACM 各会員．

中野　秀男　（正員）

昭 45 阪大・工・通信卒．昭 50 同大学大学院博士（通信）課程修了．同年阪大通信工学科助手．平 3 阪大助教授．平 7 大阪市立大学教授となり現在に至る．離散最適化問題の近似解法の評価，情報セキュリティ，ソフトウェア工学等の研究に従事．工博．情報処理学会，日本 OR 学会，ソフトウェア科学会，IEEE，ACM 各会員．