Vol. 30 No. 4　　情　報　処　理　　Apr. 1989

解　説

3. プログラム設計環境のツール

3.2 分散アルゴリズム設計支援ツール†

都倉　信樹††

1. はじめに

アルゴリズムはそれが実行される実行系としてなにを想定するかで分類される．一台の計算機で実行されるアルゴリズムは**逐次的アルゴリズム** sequential algorithm とよばれる．単にアルゴリズムの研究といえば，普通このクラスのアルゴリズムの研究をさす．その研究内容は，まず，与えられた問題を解くアルゴリズムが存在しうるかどうか(可解かどうか)，可解であれば，できるだけ能率のよいアルゴリズムを見いだすこと，あるいは，その問題を解くアルゴリズムの可解を求めて，問題自体の難しさを明らかにすることなどである．能率のよいアルゴリズムを求めることは，実用上重要であり，計算機科学の主要な課題の一つである．

計算機技術の進歩によって，複数台の処理装置をもつ並列計算機の可能性がみえると，それらの機械で問題をとくための**並列アルゴリズム**の研究がはじまった．そのねらいは高速のアルゴリズムの開発であるが，n 台のプロセッサを用いるなら，効率は一台の場合にくらべ，n 倍になると期待されることになる．この意味で，並列アルゴリズムの研究は逐次形アルゴリズムとの競争にさらされる．

VLSI 技術の進展はさらに従来と違った，高並列度計算機の可能性を開くものであり，それに対応して，**VLSI アルゴリズム** VLSI algorithm や，**シストリックアルゴリズム** systolic algorithm の考えがうまれた．これらアルゴリズムの評価基準で興味深いのは，面積と時間を重視する点である．時間は計算所要時間であり小さいほどよい．また，面積は IC のコストに支配的に影響するため，これも小さいほどよい[1]．

このように，前提とする計算モデルによって，いろいろなクラスのアルゴリズムが考えられるが，ここに述べる**分散アルゴリズム** distributed algorithm は(パケットを用いて通信する広域)計算機ネットワークモデル上のアルゴリズムと考えればよい．計算機を通信線で結合した計算機網はつぎつぎと構築されつつあるが，そのようなネットワーク用のソフトウェアを作るに当たって，分散アルゴリズムが必要となる．分散アルゴリズムはそのようなネットワークの性質を抽象化したモデルのうえで考える．この際の評価尺度としては，交換されるメッセージ数や**理想時間** ideal time などが用いられる[2]．並列アルゴリズムは，同期的に動作する規則的に結合されたプロセッサからなるパラレルマシンモデル上で，考えることが多いのに対し，分散アルゴリズムは，通信線の（パケットの）伝達速度の変動があり，グローバルな同期を仮定しないという意味で，非同期の系であり，ネットワークトポロジも特になにも仮定しないというモデルを通常採用する．このため，並列アルゴリズムでは，容易に解けるような問題でも，かなり複雑な方法をとらねばならないこともある．そのアルゴリズムの開発，デバッグもそう容易ではない．

多くの分散アルゴリズムの論文が発表されているが，厳密な証明の与えられている例は少なく，ときには提案されているアルゴリズムが虫を含んでいることもある．このような状況で，分散アルゴリズムのシミュレータがあれば，その研究に資するところがあると考えられる．そういうシミュレータを開発した例は著者の知るかぎり報告がなく，ここにその作成事例を報告し，その作成の際に突当たったいくつかの問題，主として，アルゴリズムをどう表示すれば分かりやすくなるかという，「アルゴリズムの可視化」の問題と，研究者が自分でパーソナルコンピュータを操作するという前提であるため，できるだけ，操作しやすいという「操作性」の問題について，使用経験をまじえて報告したい．こうして作られたシミュレータはグループメ

† A Tool for Distributed Algorithm Simulation by Nobuki TOKURA (Faculty of Engineering Science, Osaka University).

†† 大阪大学基礎工学部

ンバのアルゴリズムのチェックなどで使用されている．

2. 分散アルゴリズムシミュレータの概要と構成

2.1 背景

当研究室は分散アルゴリズムの研究を進めてきた[3),4)など]．この種の研究は論文として発表するが，論文の内容は，問題の説明，アルゴリズムの提示，その正しさ（正当性）の証明，そして，能率の評価という形が一般的である．逐次的アルゴリズムでも，その正当性の証明は一般にそう容易ではないが，さらに非同期で，任意のトポロジという前提での証明をしなければならないため，相当証明は難しくなる．アルゴリズムの着想をえて，いろいろと検討して正しいという確信をもちながらも，その証明の記述をより分かりやすくするために，何版もの改定を重ねて，長時間の努力の末，論文を投稿するということもある．途中の段階では，アルゴリズムと証明を照らしあわせて検討するのだが，どう動くのかが，かならずしも理解しやすくないために，誤解をすることもあるし，また，ときにはアルゴリズムとの食い違いがあることもある．メンバの苦闘を見ていて，なにかもう少しアルゴリズムの動的なふるまいを分かりやすくする方法はないかと考えるようになり，徐々にシミュレータの作成に焦点があってきた．何回か構想をたて，それを寝かしておいたが別のプログラムのモジュールを流用すれば作れるのではないかという着想が生まれた．そのプログラムは 32 ドットフォントのエディタであった．

2.2 32 ドットフォントエディタ

これは，当研究室で，教材提示システムとして開発改良し，著者の講義中で使用していた CD 4 に関係する．CD 4 では第一，第二水準の漢字しか表示できず，「計算論」の講義で，いくつかの外字を使う必要が生じ，そのためのフォントエディタが必要となり作成した．以前に作成した 24 ドットフォントエディタを 32 ドットに拡大し，外字ファイル名の入力以外はキーボードを使用せず，マウスだけで操作できるように操作性を改良した．32×32 の方眼紙様の画面上で，マウスを移動し，左ボタンをおせばドットが on となり，右ボタンをおせば off になるようにし軽快にフォントを作成できるようにした．その他の編集機能（左右上下白黒の反転，左右上下のシフト，AND, OR, XOR, コピーなど他のフォントとの演算，フォントファイルへの格納，取り出しなど）もメニューのクリックと直接指示などで操作できるようにした．このエディタのマンマシン・インタフェース，モジュールを利用すればかなり使いやすいものになりそうだと考えたことから，シミュレータの構想が具体化した．

2.3 対象アルゴリズム

まず，最初にとりあげたアルゴリズムは，「深さ優先生成木構成問題」を解くアルゴリズムであり，これまで提案されたものにくらべ，広い範囲で優れたアルゴリズムである[3)]．このアルゴリズムは任意のトポロジのネットワーク上で，あるプロセッサをルート（根）と指定したときに，各ノードへ最少の中継回数で至るような経路となるように木を決める問題である．

2.4 最初の問題点

シミュレータを作る上で，検討した最初の問題は，トポロジ情報をどのように入力するかであった．最初の構想では，隣接ノードリストを入力するという考えであり，その場合ノード番号を与え，つづいて，それに隣接するノードの番号を適当な区切記号で区切って入力するという形になる．この入力操作はキー操作が多く使いにくそうという感じがあった．そのとき，フォントエディタの 32×32 の方眼状の部分を，隣接行列を表示するものとみなすことにして，隣接行列をフォントエディット同様，マウスを移動することで入力して，結果もその行列の形で与え，また，どのプロセッサが活動しているかも対応するドットを赤色などで表示すれば，実行状況もよく分かると考えた．

2.5 シミュレータの中心部

対象としたアルゴリズムは，あるプロセッサが一つのメッセージを受け取るごとに，そのメッセージの内容と，現在のプロセッサの状態によって，どのように，プロセッサの内部状態を更新し，また，他のプロセッサへどのようなメッセージを送るかが指定されている．そこで，各プロセッサごとに，メッセージキューを用意し，メッセージが到達すれば，このキューの最後につけくわえる．また，プロセッサの動作を行うときは，一つのメッセージを取り出して，その内容をみて，状態の更新を行う．この部分は手続き algorithm として作成し，その引数は，どのプロセッサかをさすプロセッサ番号と，メッセージキューにある先頭のメッセージをさすポインタである．各プロセッサのもつ局所的なデータは一括したレコード形の変数に格納している．メッセージは，その種別と二つのパラメータをもつ．このアルゴリズムは隣接する節

点のいくつかに同一内容のメッセージを送信するという形である．この同じ内容のメッセージはヒープ領域上にとり，受信バッファにはそれへのポインタのみを格納するという方法をとっている．

シミュレートしたいアルゴリズムを Pascal で記述する．適当なデータ構造を定義し，拡張子 .def のファイルに与え，また，アルゴリズム本体は拡張子 .alg のファイルに与える．これらの部分はシミュレータから分離しており，別のアルゴリズムの記述と容易におきかえできる．ただし，Pascal なので，コンパイルは一括して行う．

シミュレーションの方法として，一ステップずつ動作を確認しやすいように，シングルステップモードを用意した．マウスをクリックするごとに，一ステップずつ進む．そしてそのとき，受信したメッセージの内容とどこから受信したかを表示し，受信したノード番号とその内部状態の，受信前後での変化のようすを表示する．また，送信されるメッセージの内容を表示する．内部状態の中に，集合型の変数がいくつかあり，この表示に当初困った．すなわち，要素をすべて列挙すると画面上で改行が必要になる．要素数に変動が多いと次の表示の際一部が残ったりして，見苦しい．そこで，できるだけ短く表示するために {2, 3, 4, 7, 10, 11, 12, 13} という集合なら，2..4, 7, 10..13 というように表示することとした．これで，かなり見やすくなった．しかし，その後，考えが変わった．集合変数がどう変化するかが関心事であり，変化を強調して出力する方がよいと考えた．そこで，N がはじめ { } であり，そこに要素，3, 4 が加わったとき，

N [ ]+[3, 4]

(逆に，要素が減ったときは，－で表す.) というような表現法にかえた．このように内容をどう表示すればよいかということは，目的にあった工夫を必要とする．

2.6 その後の改良

最初の版を試作し，研究室の分散アルゴリズムのグループのメンバに見せて，意見をきいた．いろいろの希望や提案があったが，それらを参考にしながら，新しい版の構想をねっていった．以下，多くの版を作ったが，三つの代表的な版について，簡単に説明する．なお，この三つの版をイ，ロ，ハとよぶ．

イでは，メッセージを受け取って動作するプロセッサ一つの詳細な情報が表示されるだけであるが，イからロへの移行では，できるだけ，いつも全部のノードとメッセージのようすが一覧できるように改良した．また，メニューを右に常時表示し，いろいろのサービス間をスムースに移動できるようにすることをめざした．また，結果の生成木はできるだけ木のイメージになるように表示する工夫をしたが，あまり感覚的にはしっくりこなかった．美的に木を描くというのはしばしばとりあげられるテーマであるが，どうもあまり満足できないような気がする．結局ロからハの構想にうつることとなった．ロからハへの移行は，次の点が大きな違いである．

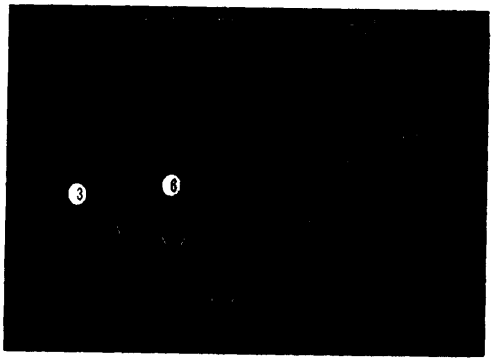
まず，(1) 操作性の向上．ある作業段階にあるときは，その作業名が黄色で画面右上に表示され，それが完了すると白色にかわるとともに，次に選択可能な操作がポップアップメニューで現れる．段階的に標準的に選ばれる道筋のものは，つねにカーソルの近くにあり，マウスでのポイントをしやすくしている．誘導的メニューなので，ユーザには操作上記憶すべきことがらは最小限でよい．

(2) トポロジをそのままグラフ表示する．入力も自分の好きな位置でマウスをクリックするとその位置にノードがおかれる．二つのノードを選択して，左クリックすると，その間にエッジがひかれる (取消しも可能)．このため，以前の隣接行列方式とくらべ，可視性が増した．また，メッセージはエッジ上に小さい白い切片として，動的に存在を表示する．また，ノードの状態は色わけして常時表示する．また，結果も生成木の場合は，その木を赤色の線で表示するので，直接結果が分かる．

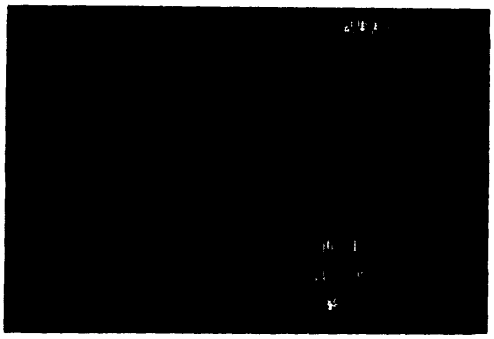
(3) 理想時間算定モードを追加した．

(4) 一度テストランして，問題のある例題について追試したり，おもしろい動作のものを見つけたときデモ用に保存するなど，トポロジ情報をファイルに保存し，また，再ロードして，もとのトポロジをそのまま再現することができる．このことで研究の効率も向上した．また，おなじトポロジの例題をいくつかの別のアルゴリズムに与えて比較研究することにも使える．このように，単に一度シミュレートするだけでなく，何回も繰り返して同じ例題についてシミュレートしたり，ある動作のトレースを記録したり，気にいった例題のトポロジを記録するなどのシミュレートの外まわりの機能がアルゴリズムのデバッグや開発に大きな効果があることを強調しておく必要があろう．

このように，ロからハへの移行では，アルゴリズムの可視化と，シミュレーションの操作性についてかな



(a) 連続実行中の画面の例.
ノードはその状態によって，色わけされており，メッセージキューのようすは枝上の白い切片で表示されている．右半分には，現在のメッセージの処理がどのように行われたかの詳細な情報が表示されている．



(b) 結果表示画面の例.
計算結果を表示している．同じ連結成分の属するノードを同じ色の枝でつないでいる．また，次に選択可能な操作がポップアップメニューに現れている．「再実行」は同じトポロジについての再実行，「はじめへ」は，トポロジの変更をして，実行するための選択肢である．

図-1

り大きな改善ができた．

画面の典型的な状態を**図-1** に示す．グラフがあり，ところどころのプロセッサに小さい白い切片がみえるが，これはここにメッセージが到着しているという表示である．シングルステップ動作のときは，任意の切片をクリックするとそのメッセージを受信したとして動作する．この機能により，任意の動作順序を指定できる．また，その途中から連続実行モードに移行することも中断することもできる（右クリップでポップアップメニューをだし選択）．連続実行モードでは，まず例題についてひととおり動かして結果をみるというために，ある決まったメッセージ選択基準で切片を選んで自動的にシミュレートしていく．各プロセッサは状態によって色わけされており，アルゴリズム動作状態が把握しやすくなっている．右の方にはあるメッセージの処理に関する必要な情報がすべて表示されている．たとえば，プロセッサの番号，どこからどういうメッセージを受け取り，内部状態はどう変化するか，どういうメッセージを送信するかを表示している．これを追ってみていけば，アルゴリズムの誤りは容易にみつけられる．また，実行中に交信されたメッセージ数などの評価データも画面に表示している．理想時間計算モードでは，ある時刻に発信されたメッセージは次の時刻にすべて処理されるという形でシミュレートする．時間は右上に表示される．また，計算結果は，それぞれの問題に合致した形でグラフ上に表示するので，直接的に判断可能である．ポップアップメニューはマウスのある位置に調整して現れ，標準的な動作のときはマウスはほとんど移動せずクリックすればよい．

2.7 よりよいマンマシン・インタフェースをめざして

以上のような基本的な方向とともに，細かい手直し，調整はかなり必要であった．そういう作業の特質の例になりそうなものをいくつか紹介する．ノードは円を書いて，その中に数字でノード番号を表示している．その番号は，テキスト画面でなく，グラフィック画面に書きこんでいる．理由は，テキスト画面へ数字を書きこむ方が容易であるが，文字の表示位置は 80×25 の位置に限定されている．そのためテキスト画面で数字をかく方法の場合は，マウスでここにノードをおきたいと思ってクリックした位置から，少しはずれた位置に円を書くこともある．その食い違いはわずかといえばわずかであるが，人間には違和感を与える．そこで，どの位置をクリックされても，それを中心とした円を書き，しかも，その中にピッタリと数字がおさまるようにしようと考えた．そのためには，任意のドット位置に数字を書く方法が必要である．そのためにフォント ROM を読み取ることを計画した．その方法でプログラムは完成したが使ってみると，フォントの読み出しと画面表示のためのフォントアクセスが競合して，画面がフラッシュをたいたような感じになりよい印象を与えないと判断して，しばらく使ったのち，この方法は放棄した．結局，フォント ROM からの読み出しをファイルからの読み取りに変更して解決したのである．

また，しばしば困ったのが，マウスカーソルの表示

されているときに画面を書くとカーソルが残るとか，画面に残るクズ情報である．書くことはやさしいが，タイミングよくうまく消すことの難しさを味わった．全画面クリアなどのプリミティブはあるが，全画面を消すと，見ている人間には連続性がなくなり，あまりよくない．連続性をたもちつつ不必要な部分をうまくタイミングよく消すという問題がいつもつきまとった．

マウス操作については，比較的単純な規則をつくった．すなわち，メニューの選択，ノードやエッジの選択指示はすべて左クリックを用い，作業の変更（ポップアップメニューよび出し）は，右クリックとしている．そして，ユーザには，「できるだけ左クリックで操作し，なにか違うことをしたくなれば，右クリック」とだけいって，プログラムの入ったフロッピィディスクを渡しているが，問題なく使っている．操作法のマニュアルなどがなくて使えるシステムという目標もたてていたが，ほぼ達成されたようである．ただし，渡すときに一度実演して説明はしている．

シングルステップの際のメッセージの選択は，ロでは大きな正方形のカーソルが画面に現れるので，選択は難しくない．ハでは，メッセージを表示する小さい切片をポイントする必要があり少し操作しにくいのは事実である．（実際には，その切片そのものでなく，その近傍をクリックしても感応するようにしているが，小さいことに変わりはない）．なお，小さい対象をマウスで能率よくポイントする方法については当研究室で検討しているが[5]，そこで有望とされている方法は今回のプログラムには組みこんでいない．というのは，その実験はマウスドライバを改造して行っているが，このプログラムの稼働する環境とは違い移植作業を要するからである．

また，画面の配色も難しい問題である．アナログRGBの場合（4096色から16色）の場合は，比較的落着いた配色ときめ細かい色わけが可能であるが，デジタルRGBの場合（8色）は，背景と字のコントラストを必要とすることを考えると，組合せもかなり限定され，配色もどきつさを感じる．

最後に，マンマシン・インタフェースにおいて，重要なのは応答時間が適正かどうかということである．遅過ぎるとユーザはイライラするし，早過ぎては，人間がついていけないという感じをもつ．この点についても可能なかぎりいろいろの調整を行うとともに，シングルステップなどのコントロールの仕方にいくつかの工夫をしてみた．

以上，この一群のプログラム作成の目的は分散アルゴリズム研究の支援であるが，そのプログラムもユーザが使いやすいと思うものでないと，単にシミュレートだけではなかなか受入れてもらえないということも付け加えておくべきであろう．

2.8 別のアルゴリズムのシミュレータ

別のアルゴリズムへの転用は比較的容易で，アルゴリズムの記述を見ながら直接 Pascal 化していく．一つのアルゴリズムの組込みに，約 4〜5 時間の作業というところである．あとは，そのデバッグであるが，今回はアルゴリズムの考案者がいろいろのテストデータについて，実行しチェックしてくれた．シングルステップで観察していくと，どの部分で異常な動作をしているかが簡単につかまえられるので，デバッグはしやすい．アルゴリズムの動きは可視化してあり，直接的に把握できるので，異常もつかみやすい．ただ，アルゴリズムの外側のインタフェース部分などは通常のプログラムと特に変わらない．

これまで，幅優先生成木のアルゴリズムのシミュレータにはじまり，その終了をはやくしたもの，素朴なアルゴリズムの3種のアルゴリズム，切断点決定アルゴリズム，橋決定アルゴリズム，2連結成分決定アルゴリズムなど[6]を作成した．いずれも当研究室で作成したアルゴリズムであり，シミュレータで確認してある．これ以外にも，適用可能なアルゴリズムは多数あり，順次必要なものからシミュレータを作成する予定である．

3. 分散アルゴリズムシミュレータの評価

3.1 使 用 経 験

このシミュレータはアルゴリズムの考案者や共同研究者に使用してもらっているが，これによっていくつかのアルゴリズムの虫が発見できたという点で当初の目的を果たしえたともいえる．また，同一グラフについて，アルゴリズムの比較実験もはじめているが，オーダ的に優位のアルゴリズムがある特定のクラスのグラフについては，若干能率が悪いという例もみつけており，より詳細な研究の糸口にもなると期待している．まだ，始めて間もないが，予想以上に研究上有効な手段となりつつある．

また，教育用の使用にも期待できる．著者はパソコンを教室にもちこんだ講義を試行しているが，その中でシミュレーションをやってみて，説得力の大きいこ

と，学生の反応のよいことを経験した．講義の際の問題は大型の表示装置が必要ということであるが，パソコン自体は高機能化しており，講義の補助として活用できる段階に至っている．そのためのソフトウェアはすべて自作であるが，その労力の大きさを語ると自分もやってみようという教官はまずおられない．もっとよい教材作成支援システムをつくる必要がある．その要素技術となる「アルゴリズム，あるいは，計算の可視化」「提示技術，マンマシン・インタフェースの高度化」などについて，このような例をとおして，検討を進めているところである．ただ，いろいろの工夫を盛りこんでも，それが果たしてどの程度よいといえるのかという「評価」も難問である．教育学の専門家にたずねても，教育効果というのは単に短期間で計測できるものでもないし，すべきでもない．学生が後年になって，よかったというのでも効果があったといえるのだと言われると，この種の研究に不安を感じてしまう．ただ，機会あるごとに研究者の方に見ていただいているが，「こういうシステムをほしかった」とか「作りたかった」という方も多く，関心をもっていただいた．改善点などを熱心に議論していただいた．重要な点をいくつかあげる．

（1） アルゴリズムを現在は Pascal で書き直しているが，直接入力できるようにする方がよいのではないか？ たしかにそのとおりであるが，アルゴリズムの記述言語が確定すればそうすることは困難ではない．現段階ではアルゴリズムもいろいろのものが出てくるところであまり固定できる段階でない．Pascal への翻訳はきわめて容易で，そのことが研究の障害になるほどではない．

（2） 実行の制御の仕方は，現在は一ステップごとに任意のプロセッサとメッセージを選択することと，ある一定の方法で自動的に実行する方法と，理想時間を計算するための実行方法と3通りを備えているが，もっといろいろの実行方法が考えられるのではないか？ たとえば，よく質問され，また，議論のあるのは，非同期の系であるから，非決定的な実行過程を作らねばならないのではないかという問題である．各状況で，実行可能なものを非決定的に選択することはなんら難しくはない．必要ならそのような実行制御はすぐに導入できる．しかし，それはアルゴリズムのデバッグという点からは必ずしも必要はない．デバッグは再現性が必要であることと，全体のようすを見ている操作者がその勘を発揮して，できるだけ対象アルゴリズムにとって厳しい状況を作れればよい．非決定的に偶然性をあてにしてデバッグするよりも，あるアルゴリズムについて，特にクリチカルになりそうな状況を設定し，そこで意識してテストする方がよいと思われる．現在はシングルステップで任意の状況を作ることができるが，やや人間の介入が多く必要であるので，それを自動的に行う方法についていくつかの案を考えうる．たとえば，できるだけ輻輳した状況を作るため，メッセージをできるだけ多く少数のノードに集中する，させるとか，一部のノードへの遅延があたかも非常に大きくなかなかメッセージが届かないという状況を作るなどである．これらは大局的な状況を判断して，制御する機構を導入することであり，もう少し経験をつんで適当なものを導入するのがよいと考える．現在の3通りは一応基本的に必要な要求に答えるものであり，デバッグという観点からは効果があったといえよう．

（3） 表示内容をもっと自由に選択できる方がよい．たしかに，現在はすべての情報を表示しているが，ユーザの関心のあるものを選択できるようにするのがよいと考えられる．

（4） 多くの論文は最悪値評価のみを議論しているが，計算機を用いた平均値評価に興味を示す研究者が何人かおられた．筆者もその点に興味がある．これは高性能の計算機を駆使して計算することになろう．

3.2 アルゴリズムの可視化

計算機のグラフィック機能の向上と計算速度の向上で可視化技術が進展しつつある．その技術をアルゴリズムの開発や教育に応用しようという研究は米国をはじめ各所で進められつつある[7),8)]．アルゴリズムの動きをそのアルゴリズムにぴったりの方法で動的に見せるなら，教科書と黒板の講義から脱皮した分かりやすいものになるであろう．筆者も一部講義の中で試行しているが，適切に準備された場合きわめて効果的であるといってよいと考えている．ただ，アルゴリズムの動作を画面に表示したから，それで可視化が成功したといえるかというとそれほど簡単ではない．やはり目的にもっとも合致した内容を人間のもつモデル（計算機の実行レベルからすると抽象度が高い）に変換して示すことが重要であり，また，速度の適切さも重要である．可視化をしたといっても，画面にでている情報が一体なになのかが分かりにくく，動きがあってもこれはなにをしているのかが分からないとか，実行があまりに遅いというものでは効果は乏しいであろう．教

育用の場合は単に一つの実行例を見せるというのでは不足で，うまくいかない例，その理由，別の方法，それらの評価など講義で伝えたいことに対応する材料を十分に用意する必要がある．その意味でまだまだ研究も心要であるし，可視化をするための支援ツール，環境の整備も重要な問題であると考えている．上に評価は難しいと述べたが，正しく評価される評価法や仕組も必要であろう．

3.3 プログラム開発面からみて

この一群のプログラム作成の経験はソフトウェア工学での知見の再認識であったり，自分の作成技術の反省であったりした．作成は turbo Pascal v. 3 で行い，PC 9801 UV 2，VM 21，UX 2 などで稼働し，マウスを必要とする．いくつかのモジュールを作成し，それらが共通に利用できたので，大量のプログラムではあったがもっぱら週末の時間と冬休みをあてて比較的短時間に作成できた．この経験から学んだことを一点のみにしぼってあげるなら次の点になる．シミュレーション機能そのものだけなら，アルゴリズム部分が 200 行程度で，多分全体でも 500 行くらいで作れると思われるがポップアップメニューや，グラフィック表示など可視性の向上のためにプログラムはどんどんふくれあがっていく．マンマシン・インタフェースをよく考えたソフトウェアは一桁複雑になるという説がある．今回の例ではソースの行数はそれほどではない（ハで 3300 行程度）が，こまかい手直し，チューニングの作業では非常に労力を要した．その意味では，できあがったソースの大きさより，開発努力で比較することも忘れてはならない．このような可視化とか，高度のマンマシン・インタフェースについての知見を集約し開発技法を確立していくことが重要であると痛感した．

4. むすび

プログラム作成というのは，いつまでも難しく深みのあるものであると思う反面，200 行のプログラムでも苦労して作っていた頃に比べ，やはり機械の性能，コンパイラやプログラム環境がよくなったのだということを感じる．個人的には，これまで，計算機は学生の研究使用を優先しており，自分で使うことはなかったが，パーソナルコンピュータの時代になり，教官にも計算機が使える時代になったということが大きいという感慨をもっている．教育研究にパーソナルコンピュータが道具として有用に使えるようになってきており，われわれの研究教育あるいは生産活動を支援する各種の支援系や環境がますます重要になるであろう．その意味でも本特集の種々の環境の普及が期待される．

まだ，検討すべきことも多々あるが，目標とした点は予想以上といってよいほどに達成された．著者の現在の関心事は「アルゴリズムの可視化」「高度マンマシン・インタフェース」の追求にある．また，講義での計算機の使用は引き続いて行う予定で，現在はそのための改良システムの構想段階である．こういうシステムは紙上で説明しても，感覚はつかめないものであり，説明も紙数の関係で意を尽せないが，多くの方の感想を集約することが一つの客観的な評価にもつながると考えており，感想などを聞かせていただけるとありがたい．評価のためのディスクも準備しているので，関心のある方はご連絡いただきたい．最後に，この研究の発表の機会を与えていただいた本特集の担当委員大場充氏に感謝し，また，ご協力いただいた方方，実際に見て種々のご意見を聞かせてくださった方方にも感謝します．

参考文献

1) 都倉信樹：VLSI アルゴリズムおよび面積時間複雑度，情報処理学会誌 Vol. 23, No. 3, pp. 176-186 (1982).
2) 萩原兼一：分散アルゴリズム入門，bit 4-26 (1988-4).
3) 朴 政鎬，増沢利光，萩原兼一，都倉信樹：幅優先生成木構成問題の効率のよい分散アルゴリズム，電子情報通信学会論文誌 J 71-D(7), pp. 1176-1188 (1983-7).
4) 増沢利光，萩原兼一，都倉信樹：辺故障を考慮したある分散アルゴリズム，電子情報通信学会論文誌 Vol. J 69-D, No. 10, pp. 1394-1405 (Oct. 1986).
5) 西中芳幸，辻野嘉宏，都倉信樹：座標入力のためのポイント手法の提案とその評価，情報処理学会，文書処理とヒューマンインタフェース研究会 16-5 (1988. 2).
6) 朴 政鎬，増沢利光，萩原兼一，都倉信樹：ネットワークの連結性に関するいくつかの問題を解く分散アルゴリズムについて，電子情報通信学会コンピュテーション研究会 COMP 88-61 (1988. 11).
7) Brown, Marc H.: Algorithm Animation, The MIT Press (1988).
8) Visual Programming 特集，Computer 18 (8) (Aug. 1985).

（昭和 63 年 12 月 2 日受付）