## 5.1 人工生命のSCM計画問題への応用

### まえがき－開発の経緯

本例は，典型的な「人工生命」の産業への応用例であり，かつ企業の経営そのものに決定的な役割を果たしているものであり，前記の物語の後半で説明されていたシステムのモデルとなったシステムである。

本システムは開発当初から「人工生命」と「クラスタ並列コンピュータ」の全面的な実用化を最大の目標として1990年に開発が開始されたものである。1999年には，石油，食品分野のユーザを中心に100システム以上の適用実績をあげ，1999年度における日本の最も優れたビジネス用のコンピュータシステムとして「ソフトウェア·プロダクト·オブ·ザ·イヤー'99」を受賞した。しかしながら，この時点では，クラスタコンピュータはまだ高価であり，本格的な「クラスタ並列コンピュータ」を導入するユーザの出現には至っていなかった。2000年になると，不況による企業の統廃合が活発化し，この動きにより超大規模のユーザから本システムの採用要求が相次いだ。これに対応するため，私達は，本格的な「クラスタ超並列コンピュータ」で「人工生命」を動作させることを研究の主たるテーマに据えた。この結果，2001年に，全てのCPUを常にほぼ100%で稼動させることのできる「人工生命システム」を完成させた。

本節は，本システムの構成，機能，動作結果を述べるものである。

### 5.1.1 SCM（サプライチェーン管理）実行システムの台頭の背景

近年のブロードバンドインターネットの普及は，処理の分散化を促進する強力な手段である。しかし，その一方で，その大容量，高速データ伝達性に対応できない従来のコンピュータにおいて，これまで存在しなかった「深刻な問題」を発生させる大きな力ともなっている。つまり，通信速度の方がはるかに高速になってしまい，コンピュータの処理性がシステムのボトルネックになり始めたのである。また，従来のメインフレーム，SMP，スーパーコンピュータなどの形態の計算機では，実社会で爆発的に発生している「巨大で複雑な問題」への対応が極めて困難になってきている。以上のような，コンピュータを取り巻く環境が，従来のコンピュータに比べ桁外れに大きい処理性を有する，新しいコンピュータの出現を強力に促している。

### （1）クラスタ並列コンピュータへの期待と問題点

これらの「巨大で複雑な問題」に対する有効な手段としてUNIXやLinuxなどのOSをベースとする小型のSMP（メモリ対照型並列コンピュータ）サーバを要素とする「クラスタ超並列コンピュータ」を採用するケースが急激に増加しつつある。このタイプのコンピュータは膨大なスケールのトランザクション処理，大規模データベースの高速検索，受注・オークションなどの実時間高速処理等比較的シンプルであるが単位時間当たり大きな処理能力（スループット）を要求される処理には大変有効である。また同等な処理性を有するメインフレームやスーパーコンピュータに比べコストパフォーマンスは，はるかに優れており中規模の企業，ユーザであっても導入が容易である。しかし，前述のとおり近年の巨大で複雑な最適化問題への応用を行おうとすると多くの問題点を克服する必要が生じる。

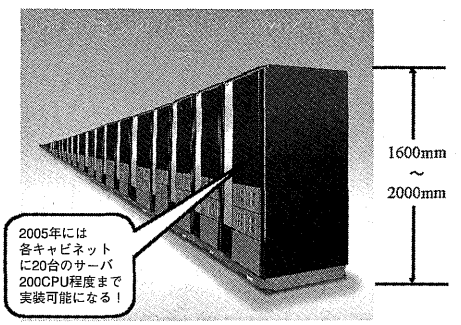



図５－１－１　クラスタ並列コンピュータシステムの外観

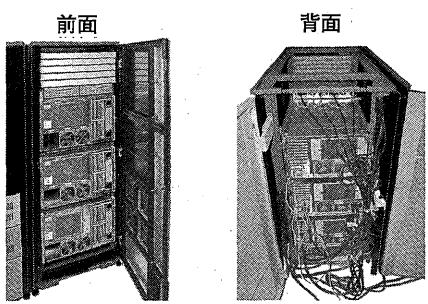



図５－１－２　内部構造（SMPサーバと高速バスの実装状況）

著者らも，ビジネス用コンピュータのベンダとして今後このタイプのコンピュータが主流になることを確信したため，図５－１－１，図５－１－２に示すクラスタ並列コンピュータシステムを開発し，広く提供していくこととした。このタイプのコンピュータにおいては，共有メモリ上に配置される複数のCPUは高速スイッチング機構により全てのCPUが同時にメモリをアクセスした場合でも，またCPU同志で交信した場合でも待ち時間が発生しない。しかし，それぞれ異なる共有メモリに接続されているCPU間の交信は高速バスや高速LANを経由して行わねばならないため大きな交信のための待ち時間が発生してしまう。このため，SMPサーバの台数が多くなるとシステム全体の稼動率が急激に低下するという問題が数多くの事例で指摘されている。これは，クラスタ並列計算機に限らず，あらゆる並列計算機における並列情報処理での本質的な問題点でもある。

**（２）従来の最適化手法**

以上のような問題を抱える並列計算機を用いて「巨大で複雑な最適化問題」を効率よく解くために従来から多くの手法が提案されてきた。例えば最近では遺伝的アルゴリズム（GA）をSMPサーバの特性に応じて高効率化を図る並列GA島モデル，解の摂動に重要な役割を行う温度を適応的に定める温度並列シミュレーテッド･アニーリング（TPSA/AN），あるいはGAとSAを組み合わせた最適化方法の研究などが活発に行われている。

ところがこれらの手法による最適化対象の多くはシンプルなモデルであり，実戦的な産業，ビジネス問題に比較し，その複雑さの度合いを示す制約次元は低いものであった。従来の研究対象の制約次元は高々10程度であり，かつそれらは数式で取り扱うことのできる連続変数によって表現される場合が多い。

これに対し，現在の経済，金融，産業，流通などの実戦分野に多数存在する巨大で複雑なプロセスを最適化しようとする場合，少なくとも2000以上の変数が必要である。そしてこれらの変数の大半は非連続でかつ非線型である。また，最適化のゴールである目的関数自体も多くの変数により構成されるのが一般的である。このような問題を解くには1000を超えるCPUを実装している超並列計算機上で画期的な高い効率で動作する新しい方法が必須となってきた。

### （3）研究の目標 ー「人工生命型クラスタ並列コンピュータ」の実用化

以上に対し，これまで著者らは1980年代中頃から相互結合型ニューラルネットワーク（SAを含む），GA，ATM（非同期処理）等をベースとする新しい方法を数多く提案し成果を得てきた。しかし，これまでの研究では使用するコンピュータのCPU並列規模や共有メモリの実装容量が対象とする問題のサイズに比較し十分ではなかったために，実在する巨大問題への有効な方法の開発には至っていなかった。この結果，著者らは「これを実現するにはアルゴリズムやモデリングなどの論理的研究努力だけではもはや不可能である」と結論づけた。そしてコンピュータのハードウェア構成を含め様々な関連分野の理論，技術を検討し以下の研究方向を定めた。

①ミッションクリティカルな実戦的ビジネス現場での要求に答え得る戦略的なクラスタ並列コンピュータシステムの開発

②従来の並列GA，並列SA的な理論よりもむしろ人工生命的立場からのアプローチによる新しい理論，システムの開発

SA，GAは情報処理的には元来逐次実行型処理から発生した理論といってよく，それを並列化するための努力が成されてきたといえる。これに対し，人工生命は誕生の環境そのものがコンカレントに様々な事象が発生する状況であり，かつプログラムに対応する生命体は自律的にかつ並列で動作する。つまり，並列処理に対する適応度が従来の並列情報処理手法に比較して格段に高いことは間違いないからである。

今回，この方針で研究を進めた結果，2000次元をはるかに超える巨大な最適化問題に対して有効に動作する人工生命型最適化システムを得ることができた。このシステムはCPUの実装数が増加するほど，並列効率が100%に近づく。これは従来の並列処理概念を変えるものであるといえる。

### （4）巨大情報処理計算の重要性について

さらに著者らは，従来研究対象とされることがまれであった「巨大計算量のもたらす結果」に大きな興味を有していたため，高速なCPUを複数実装する並列計算機を1000時間以上連続して動作させる実験も行った。この結果，並列計算は高速化をもたらすだけでなく，解の質的改善を行うことを発見した。つまり，同じプログラムを1CPUで動作させた場合と，並列計算機で動作させた場合，計算量が同じであっても並列計算機上の解は1CPUの結果よりはるかによかったのである。この現象は偶発的ではなく何回でも再現する

ことを確認したため，これは「創発（Emergence）」現象の可能性が高いと推察している。

著者らは，開発したコンピュータシステムはハードウェア，ソフトウェアともにハイパー構造を有し，かつプログラムは人工生命として動作するため本システムを「HAL（Hyper Artificial Life computer）」と称し今後広い分野に適用拡大していく予定である。

本研究では，このシステムのうち「人工生命型最適化アルゴリズム」の振る舞いと，その解析結果および評価について述べる。また，問題の複雑さとそれに対する様々な手法との比較結果についても報告する。

### 5.1.2 クラスタ並列コンピュータシステム

#### （1）全体構成

図5－1－3は図5－1－1，図5－1－2に示したクラスタ並列コンピュータの内部構成を示すブロックダイアグラムである。本システムはEIA規格に準拠する外形を有する小型のSMPサーバを基本コンポーネントとしている。標準規格を満足する多くのメーカーのSMPサーバを基本コンポーネントとすることができる。図5－1－2，図5－1－3のSMPサーバは各々4個のCPU，8Gバイトの共有メモリおよび76Gバイトのハードディスクを実装可能である。このサーバを最小構成要素として，これらを高さ1600mmあるいは2000mmの標準ラックに格納している。各サーバは1Gbpsの高速データバスで接続され，全てのサーバが，あたかもひとつの装置のように協調して動作する。

#### （2）ソフトウェア構成

図5－1－4にソフトウェアの構成を示す。複数のSMPサーバはそれぞれUNIX系のOSであるHP－UX，AIXまたはLinuxをソフトウェア階層の最下層に装備している。その上位階層にサブOS，ミドルソフトが位置し，最上層に実業務対応のアプリケーションプログラムが存在する。この構成が各SMPサーバ分だけ存在する。従来のクラスタ並列コンピュータでもこれらはLANなどによりハードウェア的には結合されていたが複数のサーバを協調させて動作させるにはそれらを統括するハードウェア，あるいはソフトウェアが必要であった。すなわち，各SMPサーバが自律的に動作しながらかつひとつの問題を解決するような動きをさせることは困難であった。しかし今回開発したシステムでは図に示すとおり自律的に複数のSMPサーバを協調制御するハイ

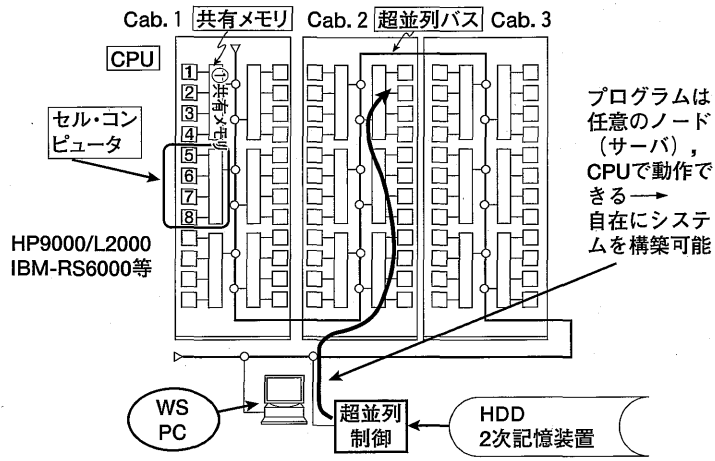



図５－１－３　ブロックダイアグラム

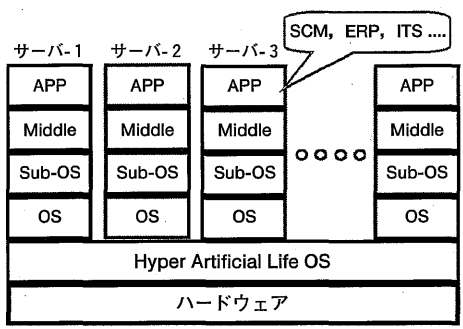



図５－１－４　ソフトウェアの階層概念図

パー人工生命型のOSを有している。ここでハイパー（Hyper）とは各サーバを上位概念で制御する，という意味で使用している。すなわち，複数のOS，複数のCPUを自在に活用することのできる人工生命プログラムの上位制御概念そのものを示している。

### （３）人工生命的動作のための機能

本項では，現行のコンピュータを人工生命型のコンピュータへ進化させるための基本的機能を以下に説明する。

**1）自律制御機能**

従来のクラスタ並列コンピュータでは，各プログラムは各々のサーバに接続されている個々のHDDなどの二次記憶装置に格納されており，必要に応じて共有メモリ上に実行型プログラムが呼び出され動作していた。これに対し本システムでは人工生命プログラムの原個体は並列バス上に接続されている共有ハードディスクに格納される。各人工生命プログラムは自律的に自身が動作可能な複数のCPUを優先順位と共に記憶しており優先順位が高く正常に稼動している場合にそのCPUで動作する。該当CPUが非稼動の場合次の優先順位を有するCPU上で動作できる。すなわち，あるプログラムが有する動作可能な複数のCPUのうち，少なくともひとつのCPUが稼動していれば該当プログラムが有する機能は停止しない。これによりシステム全体の信頼性が飛躍的に改善される。

**2）自己増殖機能（クローン生成機能）**

人工生命体としてのプログラムは共有バスを用いて，自身が動作しているのとは異なるCPU上に自身と同一のプログラム，すなわちクローンを容易に生成できる。この機能を「自己増殖機能」と称する。人工生命システムたり得る基本的な機能である。

**3）非同期交信機能**

人工生命プログラム同志はそれらの物理的存在位置にかかわらず，非同期に交信を可能としている。これは，従来のメールボックス機能とほぼ同じ機能であり，交信先のプログラムの動作を中断させることなくデータの交換を可能とする。すなわち応用プログラムが，異なるCPU，サーバ，OS上に存在し，物理的には共有バスやLANを介してのみ交信可能な場合でも双方の人工生命体，つまりプログラムには交信のための待ち時間が発生しない。これらの新しい機能は，国際的な並列処理基準に基づくMPI，PVMなどのサブシステムをベースにして開発したHAL-OSが実施している。

以上，「人工生命型最適化システム」開発の準備および機能についての最小限の説明を行った。しかしながらこのクラスタ並列コンピュータは他にも多くの特徴，機能を有している。例えば高いロバスト性，高いスケーラビリティ，高い保守性，飛躍的に高いコストパフォーマンスなどである。

### 5.1.3　巨大スケールの複雑系最適化問題の実際

#### （1）最適化問題のタイプ

最適化問題には多くのタイプが存在することが知られている。代表的なものとして「解が局在するヒューリスティック問題」と，組み合わせ数の爆発が主たる課題となる「NP完全問題」がある。前者はエキスパートシステムやOR手法が得意とする対象である。これらは対象とするプロセスやシステムがブラックボックス的であるにもかかわらず，経験的知識やルールにより比較的短い探索経路で準最適解に近づくことのできる問題である。従来，現実の製造，流通，経営の問題はあまりに複雑に見えるため，この領域の問題だと考えられてきた。

しかしながら，コンピュータの著しい能力向上と，社会・経済・産業の大きな構造の変化がこの概念を変えつつある。すなわち，非常に複雑に見える大きな問題も，それを構成する要素の種類を正確に抽出することにより巨大ではあるが組み合わせ最適化問題と捉える科学的思考に移行しつつある。従来，コンピュータによる設計，計画，制御システムは製造や物流の一部分の限定されたプロセスのみを対象としてきたため，比較的シンプルな物であった。このため経験的な手法を，組み合わせ爆発問題に対して適用した場合も，実用的には疑問が残るものの，ある程度有効な解を得ることは可能であった。

しかし近年は世界的に供給と需要のバランスが大幅に崩れ，厄介なことに供給能力が需要をはるかに上回る状況となり，この結果部分最適化はかえって全体最適化に逆行する状況になってしまっている。

#### （2）全体最適化の典型例であるSCM

ここでは実在の大手食品製造・流通企業の供給連鎖管理（SCM：Supply Chain Management）を実時間で実行する場合を例として取り上げる。SCMとは，顧客からの供給要求（オーダー）を起点にした場合の，（受注）→（調達）→（製造）→（輸送）→（保管）→（配送）の一連のビジネス活動を最適化する問題である。従来は，各プロセスごとの部分的な最適化により全体を最適化するボトムアップ的な経営・管理を実施することで十分であった。しかし，ビジネスを取り巻く環境が一変した現在，部分最適は全体を悪化させることが明らかになってきたため，このような全体最適が必須になってきたわけである。SCMシステムは，供給に要するトータルコストが目的関数でありこれを最小化することがゴールである。図5－1－5の小さな点がユー

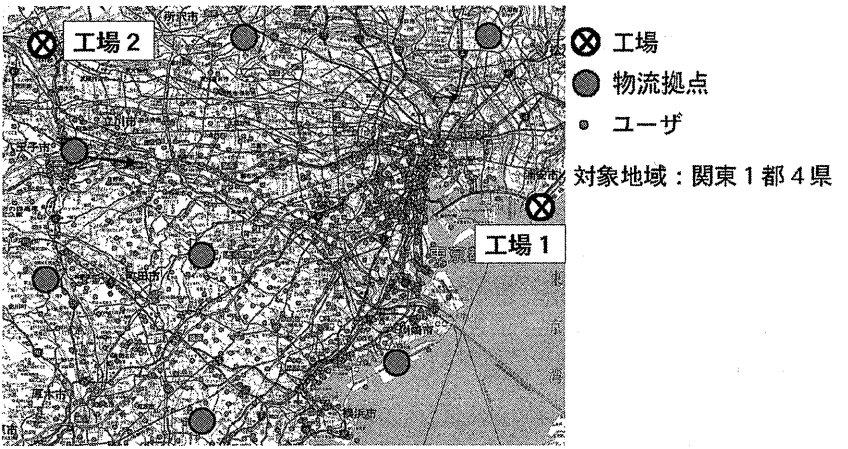



図5－1－5　典型的なサプライチェーン

ザ，大きな円が工場や物流拠点を示している。オーダーの発生，物流における交通状況の変化などは時々刻々変化していくため，計画として極めて高い処理性が求められる。これをミッションクリティカルな環境と称する。具体的な稼動例としては，我が国全域を対象としたエリアに100以上の工場，物流拠点が存在する供給構成で，同時に50000ユーザ，20万品目の供給要求が発生するような巨大な例がすでに存在し稼動している。この場合であっても実際のシステムの計画立案に許容される時間は5分～15分以内である。50,000都市の巡回セールスマン問題（TSP）に2300の制約を加えた問題と考えればよい。現実的には，これに少なくとも200万本の主要道路からなる道路ネットワークが加わるので，組み合わせの数は検討するまでもなく想像を絶する数になる。

図5－1－6は，このプロセス構造をブロックダイアグラムで示したものである。一連の供給連鎖は大別して5つのサブプロセスを含んでいる。すなわち材料調達，製造，輸送，保管，配送である。これらのサブプロセスは相互に強い相関を有しており，全体最適を実現するためには個別に最適化を行うことは避けなければならない。これらの各プロセスの役割，振る舞い，制約を効果的にかつ過不足なく記述するためには，図5－1－6に示した制約ルールを正確にプログラムに表現しなければならない。本例における制約ル

ールの数は2320である。

図５－１－７に，SCM実行システムのシステム構成図を示す。本システムは実時間で動作するシステムであり，時々刻々全国から入力するオーダーを受信処理する受注システム（OES）と，顧客の商品在庫状況から自動補充オーダーを生成する需要予測システム（RPS）と，広域に分散配置された工場，物流拠点の商品在庫をチェックしながら，最もコストが少なくなるような出荷拠点，輸送手段，輸送経路を短時間で決定し出荷指示を行う輸配送計画システム（TMS）と，出荷指示を受け短時間で物品を出荷し在庫量を管理する倉庫管理システム（WMS）により構成されている。いずれのシステムもハードウェアは「クラスタ並列コンピュータ」である。また，OESとTMSは全面

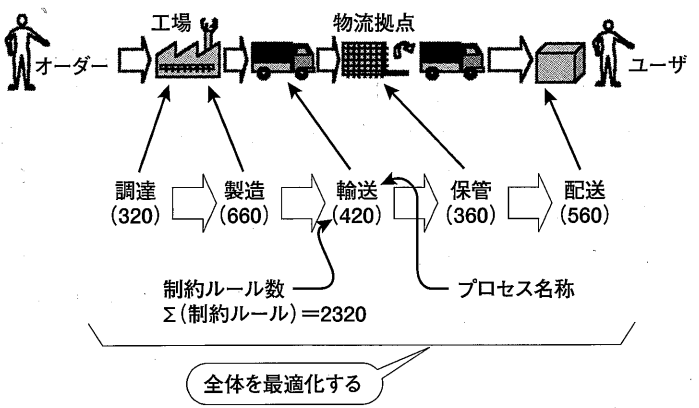



図５－１－６　供給連鎖のブロックダイアグラム

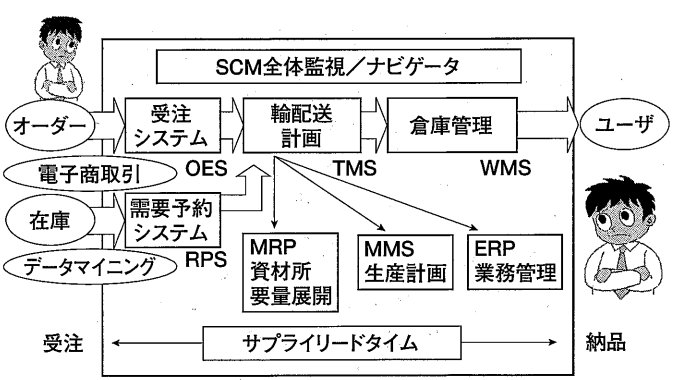



図５－１－７　SCM実行システム構成図

的に「人工生命構造」のソフトウェアで動作している。

### （3）巨大問題の次元

これまで説明してきた問題の複雑さを「次元」という単位を導入して定義する。対象プロセス全体をPとするとPは図５－１－６に示すように，制約ルールが各サブプロセスごとに存在し，総計で2320存在する。したがって

P＝f（n1，n2，n3，n4，n5，……，n2320）

と表現できる。現実的には道路ネットワークの膨大な次元を加える必要があるがここでは説明の簡素化のため省略する。各niはi番目の制約／ルールを表現しているが，各々単位が異なり，また数式で表現できるとは限らない。また，数式で表現できる場合でもほとんどが非線型でかつ非連続である。またこの問題の目的関数Fも同様に以下のように定義される。

F＝f（m1，m2，m3，……，mmax）

ここで，mmax≦2320，（m1，m2，m3，……，mmax）⊆（n1，n2，n3，……，n2320）である。すなわち，目的関数FはPを構成する要素の部分集合で構成される関数である。

この問題の最適化とは上記全ての制約を満足しながら目的関数Fの値を最大，または最小にする計画である。したがって図５－１－８の概念図に示すように，2320の制約同心円に囲まれた可能解空間の中に最適解が存在する，

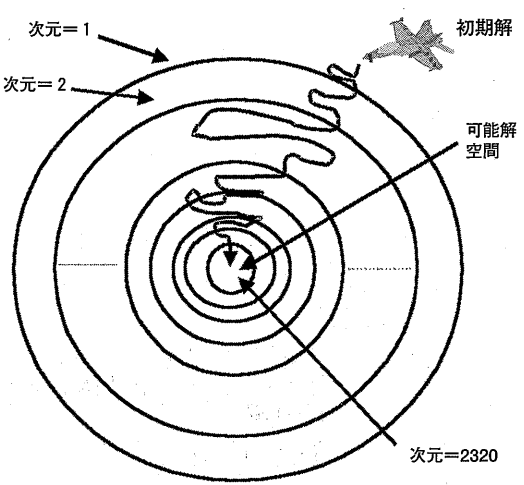



図５－１－８　巨大で複雑な最適化問題の概念図

と考えることができる。最適化の過程は，図５－１－８の探索飛行機が上記可能解空間に到達し，かつその中から最適値を選び出すプロセスであるといえる。なお，図５－１－８では各制約が明確なラインで示されているが，実際は他の制約との相関により曖昧領域や許容領域を有している。結局，この制約＝変数の数が対象問題の複雑さを示す重要な要素になっており，本論文ではこれを「次元」と称する。

### ５.１.４　従来の最適化手法とその課題

最適化に関する研究は情報処理の中心的位置を占めるものである。このため，様々な方法が提案されてきた。以下近年の研究を取り上げ，それらを本研究で対象とする新しい複雑な問題に適用した場合についての課題について検討していく。

#### （１）並列SA（シミュレーテッド・アニーリング）

SAは相互結合型ニューラルネットワークのボルツマン・マシンと等価な方法であり熱力学系の焼きなまし原理を用いた最適化方法として古くから知られている。この方法を用いた場合，理論的に最適解が必ず得られることが最大の長所である。著者らも1980年代には100サイトを超える実際の製造現場における生産計画の最適化へ応用を試みていた。しかし，実際の問題への展開においては，当時のコンピュータの能力限界とあいまって，極めて処理性の低いことが最大の障害となった。その主たる理由のひとつは，当時主流のSAアルゴリズムは逐次実行型の手順を基本としていたからである。これにより問題の要素数が増大すると，解を得るまでの時間は級数的に増加していく。このため，このモデルと等価な相互結合型ニューラルネットワークで並列化を試みたが，結果は処理手順が煩雑になるだけで劇的な処理性の改善からは程遠いものであった。

ところが，1980年代後半に入ると従来の逐次処理型のSAに対しCoranoが解の摂動近傍を温度とエネルギー関数の景観をもとに適応的に決めるという概念を導入して並列化を図った。近年でもこの方法に関連した研究が散見される。しかし，本研究で対象としている巨大で複雑な実社会への問題への適用は以下の点において困難であると思われる。すなわち，温度並列SA研究の主題が解受理のための温度制御理論や方法にやや偏ってしまい，対象モデルの急激な巨大化と複雑化という点への検討が十分ではないと考えられる。

### （2）並列GA島モデル

本手法は，最も進化したGAのマナーに基づく手法といえる。この方法はクラスタ並列コンピュータの特徴をうまく活かしたもので，基本コンポーネントとしてのSMPサーバをひとつの独立した「島」とみなし，この島内では基本的には従来のGAと同等の処理を行うが，島同志では非同期に解の交換を行う。同一の共有メモリに接続されているCPU間の交信時間は無視し得るのでGAの基本的手順である遺伝的操作，淘汰，選択などにおける交信アイドル時間は生じない。さらに島同志の交信は非同期に行われるのでその交信により発生する待ち時間も無視することができる。

このようにCPU間の交信効率についてはよい結果が得られるが，従来の研究においては対象モデルの次元が小さいため，個体評価つまり染色体から表現型を生成する時間の分散が引き起こす大きな待ち時間の追放に関する検討が十分ではないと思われる。すなわち2000を超えるような高次元の問題に本方法を適用した場合，各島の内部での淘汰，選択時に大きな待ち時間が発生することになる。また島モデルはSMPサーバの構成に大きく影響を受けるため，ハードウェアを変更する都度ソフトウェアを変更する必要があると思われる。著者らの求める最適化方法のゴールはCPUの数やサーバの構成などハードウェアやOSの制約がなく，かつあらゆる無駄時間が排除される方法を実現することである。

## 5.1.5　巨大問題が引き起こす並列処理の本質的課題

これまで，複雑問題の次元の定義を行い，その後，従来の最新の最適化方法を巨大問題解法に適用した場合の課題について述べた。この課題は，従来の手法の欠点を述べたのではない。全く新しい問題のクラスに適用したときに生ずるであろう問題点を推測したものである。対象問題の次元が社会，経済，産業分野の激変によりインフレ的に膨張してしまったのである。つまり，これまでの最適化理論の適用範囲を大きく超えているといってもよい。

本研究が対象としている問題を計算機のプログラムとして記述しようとすると非常に多くの処理命令を有するものとならざるを得ない。著者らが現在多くのユーザに納入しているシステムでは，最適化部分を除いたプロセスの記述部分の容量はC言語などの高級言語で20万行を超えている。20万行という数値はひとつの対象を記述するソフトウェアとしては巨大なもので，著者

らはデバッグとフィールド試験に10年の歳月と膨大な開発費用を費やしてきた。このように大きなモデルになると当然処理時間が増大してゆくのであるが，並列コンピュータ上での問題点は絶対的な処理時間の増大ではなく，複数回処理を実行させた場合の処理時間の不均一性にある。つまり，処理時間の大きなばらつきの度合いである。従来のGAにおける表現型生成の処理がこれにあたる。この処理は各個体の持つ染色体ごとに独立して処理し得るので並列計算が可能であり，従来の並列GAの多くはこの部分の並列化を主たるポイントとしている。

表１は従来の並列GAを，全く同一の能力を有する複数のCPU上で処理させた場合の実験結果を示したものである。表１中のグラフの縦軸は問題の複雑さの指標である染色体の要素数を示しており横軸はそれに対する平均処理時間と分散，および従来の方法と今回開発したシステムでの並列システム全体稼動率を示している。問題の複雑さは制約の数，染色体の要素の数，プログラミング上の分岐数などの多くの要因が引き起こすものであるが，ここでは検証の容易さから染色体の要素数を選択している。表１から，要素数が少ない場合は表現型計算に要する人工生命体毎の並列処理時間は平均値に近く，ばらつき度合いを示す分散が小さいことが分かる。しかし，要素数が増大していくにつれ平均値も大きくなっていくがそれにつれて分散値も急速に増大している。例えば１個の染色体に対する表現型を生成するのに要する時間は0.1秒の場合もあれば100秒を要する場合もある，ということである。これは，染色体の構成により，処理命令数が大きく変動していることを示している。

表１　複雑さによる並列効率の変動

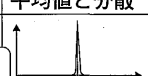
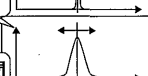
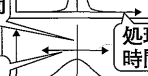
| 複雑さ<br>#of orders | 処理時間の<br>平均値と分散 | 並列GAによる<br>並列効率 | 人工生命よる<br>並列効率 |
|---|---|---|---|
| 100＞ |  | 95% → | 95% |
| 500 |  | 55% ↗ | 96% |
| 1,000 |  | 8% ↗ | 98% |
| 10,000 |  | 3% ↗ | 99% |


図５－１－９　並列GAの動作タイミング

これを従来の並列GAで処理させると図５－１－９のタイミングチャートに示すとおり淘汰，選択等全体を観察して世代を更新する処理において大きな同期待ち時間が発生していることが分かる。近年，企業や自治体の急速な統合廃統合化により増加しつつある巨大事業体の染色体を構成する要素数は10000を超えるケースが大半である。ここで要素数とは，１日のオーダー件数，製品ロット数，あるいは顧客数を指している。このような状況で並列GAを並列コンピュータで動作させるとCPUの稼動効率は３％以下になってしまう。つまり1000CPUを用いた場合でも30本のCPUをフルに稼動させた場合と同程度の処理性しか得られない。

このように対象問題が巨大化した場合GAにおける淘汰，選択のように全体を観察しなければならない処理を含む方法を用いると，計画問題に限らず通常の業務処理や設計問題などの，ほとんどの現実の問題では，同期による大幅な効率低下を避けることができないという問題が発生する。

### ５.１.６　人工生命型並列最適化システムの開発

これまで述べてきた内容を要約すると，巨大な問題を現実的な時間で解くためには，

（１）膨大な次元を有する対象プロセスの表現方法の開発

（２）同期処理等により発生する並列効率低下回避の方法の開発

を行う必要がある。以下，これらを含む人工生命型の最適化方法について説明する。

これまで，従来のいくつかのアルゴリズムを簡単に説明した。しかしこれらはハードウェア構成に依存すること，同期処理を含むため高次元問題において並列効率が大きく低下する，という問題があった。そこで著者らは新しいモデルの具体的な開発目標として以下を定めた。

1）並列コンピュータの構成等ハードウェアに一切依存しないこと(汎用性)

2）CPUの数が大きくなる程クラスタ全体の並列効率が100%に近づくこと（高効率性）

3）CPUの数を増減させてもソフトウェアの変更が不要であること（スケーラブル）

4）処理途中に一部のCPUやサーバが停止しても機能が停止しないこと(高信頼性)

5）逐次型の処理では到達し得ないレベルの解に到達できること(創発性)

以上を実現するため研究当初はGA，SAをベースに検討していたが，これらの考え方だけでは上記目標を満足することは困難であることが分かった。そこで，一般的には産業応用品やビジネス問題の解決手段として取り上げられることがまれであった人工生命理論と方法に着眼した。

この理論はカオス，コンピュータグラフィックス（CG），フラクタル，ゲーム理論，ロボティックス等の分野との相互作用により広範囲の展開を見せていた。この結果それ以前には見られない様々な研究結果が得られたが，コンピュータそのものの前進に対する貢献と，現実の経済，産業等の実戦分野への貢献は見るべきものが少なかったといえる。特にビジネス分野におけるミッションクリティカルな分野への応用はきわめて少なかった。このような状況にもかかわらず，著者らは並列処理を含む従来のコンピュータをベースとする方法では対応が困難であった巨大な問題に対し人工生命理論/システムは極めて有効なものと考えた。

そこで，生命の持つ優れた能力をコンピュータ上で得るため，以下のような新しい最適化アルゴリズム，システムを開発した。具体的なアルゴリズムは以下のとおりである。

〈手順1〉初期の人工生命体AL0を生成する。これは一般的なプログラム起動と同様である。

〈手順2〉AL0は論理的に隣接する稼動率の低いCPUを探す。

〈手順3〉稼動率が低いCPUが存在する場合AL0は自身のクローンAL1を生

成する。

〈手順４〉各人工生命体ALiは進化処理マナーに従って最適化処理を開始する。

〈手順５〉最適化途中で近傍CPU上のほかの人工生命体と非同期に交信を行い，相手の進化状況が自身より優れていると判断した場合は遺伝的操作を自身の染色体に対し実施する。ここでの遺伝的操作は突然変異とクロスオーバである。

〈手順６〉自身の最適化に変化があった場合，隣接するCPU上の人工生命体にその結果を非同期で通知する。

〈手順２〉～〈手順６〉を繰り返す。

〈手順７〉最適化終了シグナルを受信した場合他生命体の最良結果データを共通２次記憶装置から読み出し，自身の結果と比較して自身がベストである場合はその結果を最適解として出力する。

上記において「隣接CPU」とは，論理的な位置関係を示しており例えばカルテシアン関係などを指している。また，クローンの生成および他生命体との交信は前述のMPI，PVMを活用したHAL－OSが効率よく実施する。特に交信はメールボックス方式を基本とする方法の開発により，実通信は完全な非同期となり，これによりCPU間の通信による効率低下は皆無に近いシステムとなった。

このアルゴリズムは淘汰，選択等システム全体の同期を必要とする処理を一切含まない。この点が従来の進化的計算方法との決定的な違いであり，こ

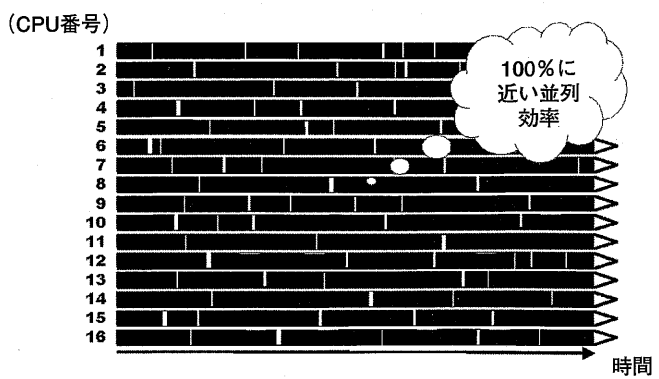



図５－１－10　人工生命システム動作タイミングチャート

れが画期的効果を引き出す大きな要因となった。

**（1）動作結果**

著者らは前述の図5－1－9に示したとおり，従来の一般的な並列GAをクラスタ並列コンピュータ上で動かした結果，淘汰・選択処理における同期により大きな待ち時間が発生していることを発見した。

一方，図5－1－10は開発したシステムを前述のSCM問題に適用し動作させた場合のタイミングチャートである。ランダムに人工生命体どおしの交信が行われているが，待時間はほとんど発生していないことが分かる。図5－1－10内に見られるアイドル時間はCPU間交信以外の要因で発生している微小なものである。またクラスタ全体での同期も全く発生していない。また，新たにCPUを追加した場合であっても自律的に新しい生命体が生成され，最適化処理に加わるよう動作する。逆に，処理途中でSMPサーバ，あるいは特定のCPUが停止した場合であっても，結果が正しく得られるように動作する。

このように並列効率という観点から評価すると，人工生命型システムは処理資源をフルに活用することができ，非常に重要である。

### 5.1.7　実データによるビジネスレベルでの実証実験

上記の動作状況の調査で新しいシステムはCPUの数にかかわらず並列効率が100%に近い値を得ることが可能であることは実証できた。しかし，本研究の最終目的は最適化能力の飛躍的向上にある。そこで，著者らは，新しい人工生命システムの最適化能力に関する性能，機能を検証するため，実在のSCMデータを用いて以下の実証実験を行った。

**（1）実験の環境**

**1）並列コンピュータ構成**

図5－1－1，図5－1－2，図5－1－3に示した構成を有するクラスタ並列コンピュータのうち下記を用いた。

(a) サーバ：H9000 or H9000シリーズL2000　SMPサーバ×4 sets
(b) CPU　：PA8500－RISC（440MHz）×4 ways×4 sets＝16CPUs
(c) 処理能力：Spec_int_rate95＝1186/CPU
(d) OS　：HP－UX11.0（UNIX）×4 sets

**2）使用データ**

実際の食品製造会社の，日々の実データを採用した。

(e) オーダー数／日：6日間の実データ，370～960顧客から約3000品目

(f) 供給拠点　　：2工場，7物流拠点（デポ）

(g) 使用可能車両数：500台

(h) 使用道路数　：約200万本，30分ごとの静的平均速度データを使用

**3）初期解**

最適化における初期解の質は非常に重要な要素である。ここでは実験の客観性を重視してユーザが使用していた既設の最適化システムの解を最適解として採用した。既設のシステムはヒューリステックなルールに基づいて開発されており，かつ十分実用に耐え得るチューニングが付されているものである。

**4）目的関数**

実際のシステムでは，(稼動時間最小)，(走行距離最小)，(総コスト最小)など複数の目的関数を自由に選択することができる。しかし，本論文では最も次元の高い（総コスト最小）の結果を重点的に検証していく。(総コスト)Fcは次のように定義される。

$$
\begin{aligned}
Fc &= \Sigma(\text{製造コスト}) + \Sigma(\text{流通コスト}) + \Sigma(\text{営業コスト}) \\
&= (\Sigma(\text{調達コスト}) + \Sigma(\text{加工コスト}) + \Sigma(\text{製造コスト})) + (\Sigma \\
&\quad (\text{輸送コスト}) + \Sigma(\text{保管コスト}) \\
&\qquad + \Sigma(\text{配送コスト})) + \Sigma(\text{営業コスト}) \\
&= f(c1, c2, c3, c4, \cdots\cdots\cdots, c320)
\end{aligned}
$$

Fcはこのように，プロセスを記述するための多くの変数からなる関数となっており，本例では300次元を超えている。すなわち，320の変数のうち少なくともひとつが変化するとFcが変化することになる。間接的には2000以上の変数全てが次元を形成しているのは繰り返し述べているとおりである。このようにFcは320次元の曲面体を示しており，その表面は極めて変化の激しいものになっている。

**5）巨大計算量**

今回用いたクラスタ並列コンピュータはすでに多くの企業や研究所で日々の業務に使用されているものである。使用したRISC－CPUは最新型のものにはやや劣るが依然として高速性を有しておりミッションクリティカルな業務に耐え得るものである。今回の研究では，人工生命モデルの動作をミクロ的に解析するために，CPUの数は1から16としたが，本システムは現実的な制約を考慮しても1000CPU程度までは大きな問題なく動作させ得るものである。

論理的には10,000を超えるCPUであっても並列効率が落ちないことは確実であるが，コスト，スペースなどの点で，現状での実験は困難である。

さて，著者らは，並列計算の本質的なポイントとして巨大計算が生み出す結果に着眼した。限られた資源で実験を行わねばならないので並列コンピュータをできるだけ長時間動作させることにした。本報告においては最大1,000時間＝3,600,000（秒）まで最適化動作を行わせた。この結果1CPU当たりの整数演算命令数は4.2京（4,200,000,000,000,000）命令に達することに成った。これだけ多量の計算を行った場合，従来の情報処理とは異なった結果が得られるのではないかという期待があったからである。

### 5.1.8 結果と考察

#### （1）最適化能力の検証1

図5－1－11，12は，ある1日のデータに対する実験結果である。図5－1－11は最適化開始から約4時間の詳細を，図5－1－12は1000時間までの長期動作結果である。本図は縦軸がFc:総コスト（万円），横軸が最適化処理経過時間で単位は図5－1－11では（分），図5－1－12では（時間）である。

実験対象システムは以下のとおりである。

1）SAモデル：1CPU上での基本的な構造を有するSAであり，温度Tはマニュアルにてチューニングして固定したものである。
2）GAモデル：世代内の個体数を16とし，1CPUで逐次的に動作させた。
3）AL－1CPU：開発したシステムを1CPUで逐次的に動作させた。
4）AL－16CPU：開発したシステムを16CPUで並列動作させた。上記と全く同一のプログラムである。

初期解については，各モデルに対する公平性を保持するため，全てのモデルに既設システムの結果を与えた。ここでは，総コストが915万円であった。

#### （2）並列人工生命モデルの動作検証

まず16CPU上でのALモデルの動作結果を検証する。実用上の処理時間要求値である15分における解は746万円であり，金額にして169万円の低減，初期値に対し18.5%の改善を実現している。1年の有効稼動日を300日とすると年間で約5億円の供給コストを実現することになる。最適化システムの能力として15%以上の改善率は一般的に非常に大きな数値である。改善率の評価を行う場合，比較対象となる初期解の水準が問題になるが，ここでは実際に運

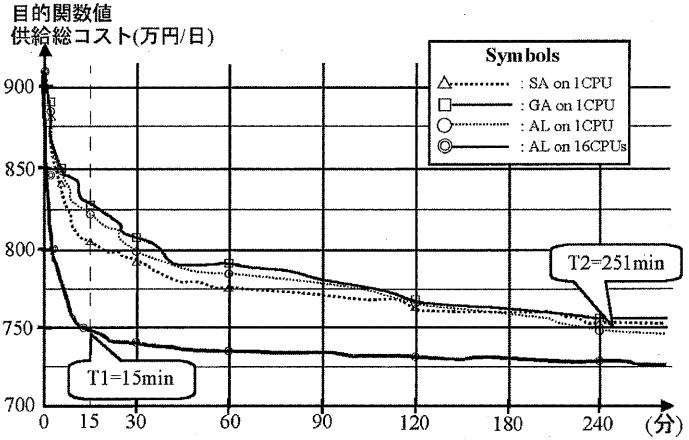



図５－１－11　実験結果１

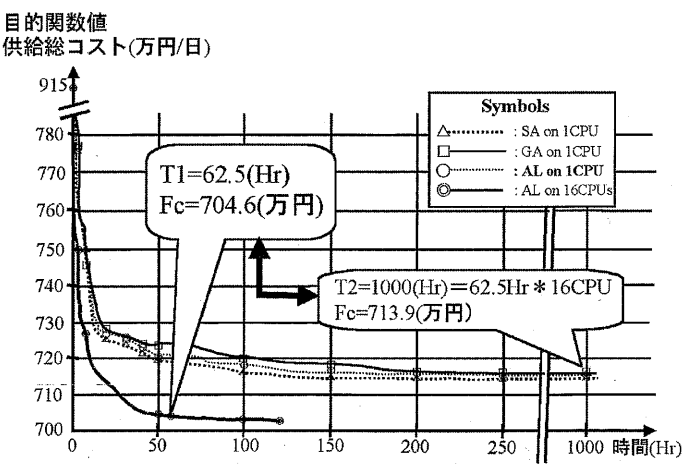



図５－１－12　実験結果１－創発の発生

用している既設システムの値を用いているので，あいまいさはない。

著者らもこの実験結果を最初に得たとき，あまりの大きな改善率に対し実験手順やデータの誤りがあるのではないかと考えた。そして，入力データ，出力データを繰り返し検証したが誤りは認められなかった。結果として，「制約が2000を超える複雑な最適化問題」に従来の経験的方法を適用することは非効率であることを確認することとなった。

ちなみに，複雑さを大幅に低減するケースとして，９つの供給拠点を，1ヵ所にした実験を行った結果，本システムによる最適化改善率は高々５％であった。つまり，問題がシンプルな場合は従来の経験的な最適化方法でも十分活用できるということである。

**（３）他モデルとの比較**

図５－１－11の検証について述べる。上記と同様，15分経過後の他モデルの動作結果を解析する。SAでは105万円，11.5％の削減，GAでは89万円，9.7％の削減，１CPUでのALは91万円，9.9％の削減であった。以上により並列計算機上のALは非常に効率よく最適化を行っていることが分かった。

次にCPUの並列度数と最適化の処理性の関係を検証する。処理性の比較については様々な尺度がある。ここでは処理時間要求値の15分におけるAL－16CPUの場合の解である746万円の水準に到達するために，他のモデルがどれくらいの時間を要するのかを調べ，比較検討する。

１CPU上における３種類のモデルの最適化状況を分析すると，計画開始後２時間ほどはそれぞれ特徴的な動きを見せるが２時間を過ぎると，ほぼ相似の動きになっていることが分かる。したがってここでは，並列処理による最適化効率の向上を検証することが主たる目的であるので，1CPU上でのALの結果に絞り検討する。AL－1CPUが746万円に達するのは251分である。したがって16CPUの並列処理化は251/15＝16.7倍の高速化を達成していることになる。

しかしながら，最適化効率の測定と評価は，その方法と，環境条件において，客観性を欠いていることがある。著者らも，この結果だけで最適化能力の改善度合いを評価するのはやや困難ではないかと考えた。何故なら，どの実験ケースにおいても，４時間程度の動作では目的関数値が収束したようには判断できなかったからである。つまり，巨大で複雑な問題に対し，現存の最速のCPUを用いたとしても数時間程度の実験で最適化度合いを評価するのでは検証として不十分である，と考えた。

**（４）巨大情報処理のもたらすもの**

図５－１－12は，図５－１－11に示した実験と同条件で，処理時間をできるだけ長くし，最適解が収束するのかを検証するため，約1000時間＝約42日間連続で動作させた結果を示したものである。計算量的には1000CPUを１時間動作させた場合と同一になる。

まず，1CPU上でのSA，GAおよびALの結果は約200時間経過後には（総コスト）は715万円まで低減し，それ以降800時間はほとんど変化しなかった。SAの結果は微量であるがわずかに改善が見られた。温度をうまく制御すると確かに最適化能力は微量だが向上するものと思われる。以上の結果からいえるのは，高速のCPUを用いた場合であっても，少なくとも200時間ほど連続して処理を実行しなければ最適化の収束状態が分からないだけではなく，どの段階にあるのかさえ判断ができない，ということである。つまり，1000世代×16個体程度の進化処理での結果は不安定であり客観的で再現性のある結果として評価するのは困難と考えられる。一方非常に多量の処理をした後の結果には以下のとおり注目すべき点が2つ存在する。

**1）巨大計算結果1－高い最適化能力**

最も興味深いことは，長時間の処理を行うと解はどこまで改善されるか，という点である。図5－1－12を観察すると，120時間後の16CPUでのALの動作結果では704万円までコストが低減されているのが分かる。初期解は915万円であったので，結局23.0%の改善が達成されたことになる。最適化実験において1/4もの改善が行われるのは驚くべきことである。この結果が示すことは，現代の実戦的問題が極めて複雑化し，従来の方法での解決が非常に困難になってきているという点である。著者らはこの点に「並列コンピュータ」と「人工生命システム」の重要性があると考えている。本例では23%の改善を達成しているが，この結果を得るために約120時間を要している。しかし，ミッションクリティカルな現場ではこれを15分以内に得る必要がある。従来のコンピュータと手法でこれを実現することは困難でありほぼ不可能と考えている。しかし人工生命ベースの並列コンピュータでは

（120時間）÷（15分）×（16CPU）＝7680CPU

という処理能力のハードウェアを用意することで達成できる可能性があるといえる。

**2）巨大計算結果2－「創発」の発生**

1CPU上でのALの処理結果と，16CPU上でのALの処理結果を比較する。1CPUでの1000時間の処理量は16CPUでは約62時間に該当する。1CPUで1000時間後の総コストは714万円であるのに対し，16CPUでの62時間後の結果は704万円である。同一プログラムを同一時間だけ動作させたにもかかわらず結果が全く異なっており，並列処理のほうが優れている。つまり，解の質の改

善が発生したといえる。著者らにとって，これは全く予想外の結果であった。

以上の実験は合計6回行ったが，同じ結果，つまり「創発」が100%再現することを確認した。1 CPUでの処理，すなわち逐次実行型の処理ではSA，GA，AL他いかなる処理方法をもってしても714万円付近に解が収斂してしまい，1000時間までの処理結果推移を観察すると1000時間をはるかに超える時間をかけたとしても704万円の質まで向上させることは困難ではないかと推定される。したがって，並列処理は単なる処理速度の向上だけでなく，逐次実行型の処理では到達し得ないレベルへの質的改善を実現しているということになる。この現象は，計算機上での情報処理における「創発」である可能性が高いといえる。「創発」とは個々の生命体，すなわちプログラムの論理的な仕組みからだけでは創造し得ないが，それらが数多く集結することで，個々の能力だけでは過去に発生したことのない新しい現象が発生したことを指すからである。

現在，私達はさらに並列度を上げることで，有限時間内において，絶対的な処理量をできるだけ大きくすることを計画している。16並列で確実に「創発」が発生するなら，100並列，1000並列では一層大きな「創発」が容易に発生するものと期待できるからである。

### (5) 巨大情報処理が「創発」を産みだす必要条件

今回の研究において「創発」を発生させる必要条件については，下記のとおり実証実験で推定することができるようになった。

1) 近傍CPU上の人工生命体同志は頻繁に相互作用を行うこと
2) 相互作用の結果は，全体システムに瞬時に波及すること
3) 相互作用は非同期に行われ，相互の処理を一瞬たりとも停止させないこと

今回の人工生命モデルを16CPUのクラスタ並列マシンに実装した場合，何回実験を繰り返しても，同一の水準の目的関数値を有する解を得ることができた。しかしながら，それらの「染色体」から生成される表現型としての実際のSCM計画は毎回異なるものであった。この原因は「クローン生成」「相互作用」などの処理実行時に発生する，ハードウェアを構成する半導体素子の微小な動作のズレから生ずる「時間的なゆらぎ」であるものと推定される。そうでなければ，何回再現テストをしても解の形は同じになるはずだからである。逆にこの「時間的なゆらぎ」が高い最適化能力を産み出しているのではないだろうか？　著者らは今後の研究の重心を「人工生命体の相互作用メ

カニズム」に移してゆく予定である。

### 5.1.9　今後の課題

今後の課題として，1000を超えるCPUを実装した超並列コンピュータで提案したシステムを動作させ質的改善の度合いを検証していく。また「創発」発生のメカニズムの詳細解析を行い，汎用的な「創発型コンピュータ」の基本的理論を確立していくことを推進してゆく。「創発型コンピュータ」とはCPUなどの物理的素子の能力を超える処理能力を有する情報システムと等価である。言い替えれば100個のCPUを実装しているコンピュータが101個分以上の処理を安定して行い得るコンピュータである。１個のCPUの実装密度と動作クロックの限界が完全に姿を現わしている現在，情報処理分野での最も重要なゴールのひとつであると考えるからである。

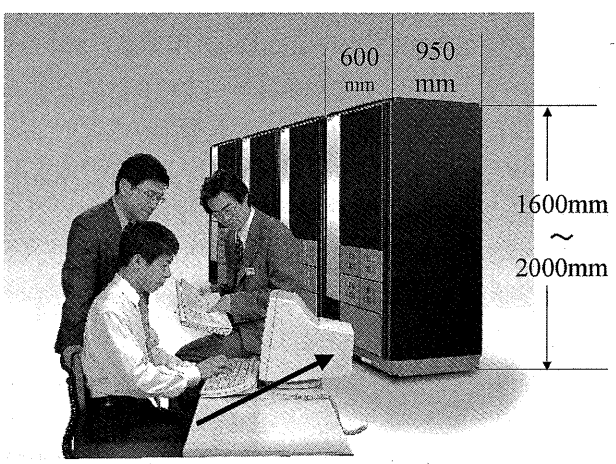



図５－１－13