

超並列計算機を用いた分散構造最適設計

～最適化，構造解析の同時分散処理

正員 鈴木 克幸* 正員 大坪 英臣*
内藤 祐史**

Distributed Structural Optimization Using Massive Parallel Computer
—Distributed Computation of Optimization and Structural Analysis—

by Katsuyuki Suzuki, *Member* Hideomi Ohtsubo, *Member*
Masashi Naito

Summary

The new computational algorithm for structural optimization that is suitable for massive parallel computers is proposed. Both analysis algorithm and optimization algorithm are paralleled simultaneously, which make it possible to use much more processors than the subdivision number of the structure. The 3 dimensional truss problems were solved as examples. For the parallel computation of analysis, it was shown that when the granularity of the computation for each processing element is small, the efficiency of parallel computation is low, but as the granularity increase, the efficiency also increase. For the parallel computation of the optimization, it was shown that not only the algorithm gives high efficiency, but also it often gives better solution which is closer to global optimal point. Finally both optimization and analysis are computed in parallel, and it was shown that quite high speed up rate can be obtained.

1. はじめに

構造最適設計は、近年汎用ツールなどの登場により急速に設計の現場に普及しつつある。しかし、現在実務においてに用いられているのはほとんどが部品あるいは小規模な構造や単純化したシステムの最適化である。一般に、大規模で複雑な構造物やシステムに対して最適設計を試みる場合、設計変数、状態変数の数が膨大な数になるという問題がある。このため、1回の解析に必要な計算時間の増加はもちろん、最適化においてはその解析を何度も繰り返す必要がありその反復回数の増加によって計算時間は膨大なものとなる。また、一般に構造最適設計の問題は数学的には非凸な問題であり、設計変数の増加に伴い解の多峰性が顕著になり、しばしば真の意味での最適解であるグローバルな最適解とは大きく異なるローカルな最適解に収束してしま

いがちである。

これに対し、筆者らが試みてきた、最適化問題をいくつかのグループの設計変数に分解し、そのグループ内で最適化を行うとともにグループ間の調整を行い全体の最適化を行うという分散最適化手法¹⁻²⁾は、計算時間を大幅に短縮できると共に、複数の設計者がそれぞれの部分の設計を行うと同時に互いの調整を行うという、大規模システムにおける実際の設計環境に適合した手法である。

本論文では、非常に多くの設計変数、状態変数を持った大規模な構造に対する最適化に対して分散最適化を適用し、そのような分散最適化を効率的に行うための超並列マシンにおける計算アルゴリズムを提案し、その計算効率を実証する。さらに、この様な分散最適設計によって、全体を一度に最適化を行うより、グローバルな最適解に近いよりよい解が得られる傾向のある事を示す。

2. 超並列計算機

近年、安く高性能なマイクロプロセッサの出現により、それを大量に使う事によって従来のスーパーコンピュータや大型汎用コンピュータに比べはるかに安価で、そし

* 東京大学工学系研究科船舶海洋工学専攻

** 日本 IBM

原稿受理 平成9年7月10日

秋季講演会において講演 平成9年11月14, 15日

てより高性能な超並列計算機が注目を集めている。しかし、実際には並列計算は重要性が叫ばれているほどには普及を見ていない。これは、並列計算機を用いる際にはある程度マシンのハードウェア、ソフトウェアに対する知識と経験が必要とされることがネックになっている。すなわち、従来の大型計算機では、2倍早いマシンで計算すれば基本的に2倍のパフォーマンスが得られた。ベクトル処理マシンにおいては、その性能を十分に発揮するためにはコーディングにある程度の工夫が必要であったが、従来のソースコードを単純にそのまま用いても、ある程度の性能向上が得られた。しかし、現在並列計算機を使うには、どの処理をどのPE (Processing Element) に割り当てるかを人がプログラム内で指示する必要があり、ソースコードの変更等が不可欠である。例えばHPF (High Performance Fortran) のように、アルゴリズムそのものの変更なしにコンパイラ支援で同値的に並列化を行う方法等も研究されているが、並列計算機を真に有効に使うためにはそれに適した問題の分割と数値計算アルゴリズムが必要である。しかし、有限要素法などで用いられる連立方程式の求解や非線形計画法に代表される最適化手法は、一部分の変化が全体に影響を与えるため、完全な並列計算は不可能である。本研究では、構造最適設計において現れる、最適化、解析の2つの側面に対して並列計算を試み、最適設計を超並列計算機で行うのに適した計算アルゴリズムを提案する。

ハードウェアとしては東京大学大型計算機センターに導入された日立製作所の超並列計算機SR 2201を用いる。SR 2201は分散メモリ型マシンであり1個あたりの演算性能が0.3 GFLOPSのPE (Processing Element) を32~2048個(東京大学のものは1024個)搭載している。また、3次元クロスバネットワークにより高いプロセッサ間通信能力を備えている。

並列処理で用いられるプログラミングモデルには主にData ParallelとMessage Passingの2種類がある。Data Parallelは繰り返し文などの同じ演算を異なるデータに対して各プロセッサで行う場合に、Message Passingは各プロセッサで異なるジョブを実行し、ジョブ間でデータを受け渡ししながら計算を行う場合に用いられる。後述する本論文の分散アルゴリズムは、Data Parallelのように単純に繰り返し文を並列化するというアルゴリズムではないので、ここではプログラミングモデルとしてMessage Passingを用いる。実際のプログラミングに際してはMessage Passingのプログラムモデルをサポートする通信インターフェイスとしてMPI (Message Passing Interface) を用いた。MPIは多くの並列計算機ベンダーが標準化に参加しており、汎用性が高い。

3. 並列処理アルゴリズム

3.1 最適化問題の並列処理

まず、設計変数、制約条件を互いに関係の低いものどうしに分解を行う。この論文では、文献1)と同様に構造物を部分構造に分解し、それぞれの部分構造に関連した設計変数と制約条件のグループに分けるというアプローチをとる。また、多領域の問題に対してはそれぞれの問題のローカルな設計変数、制約条件のグループに分解すると考えられる。部分構造間の調整には、モデル調整法³⁾を用いる。設計変数を、システム全体に関わるグローバルな設計変数 y とローカルなサブシステム i にのみ関わる x_i に分解する。目的関数は

$$F = f_0(y) + \sum_{i=1}^n f_i(x_i, y)$$

のように、 y のみに依存する項と、 y と x_i にのみ依存する項の和として表現できる場合を考える。さらに、制約条件も y のみに依存する制約条件

$$g_0(y) \leq 0$$

と、 y と x_i にのみ依存するもの

$$g_i(x_i, y) \leq 0$$

に分解できるとする。逆に言えば、複数のサブシステムに関係している設計変数は全てグローバルな設計変数と考える。このようにすれば、サブシステム i の問題を、他のサブシステムの設計変数、制約条件を考慮に入れずに独立に解くことができる。

本論文では、全体の最適化をFig. 1のようなフローによって行う。まず、分解した各サブシステムをそれぞれ最適化する。そして、その設計変数情報に基づいて、グローバルな設計変数に対して最適化を行う。各サブシステム内の最適化手法としてはDOT⁴⁾による可能方向法を用いた。こ

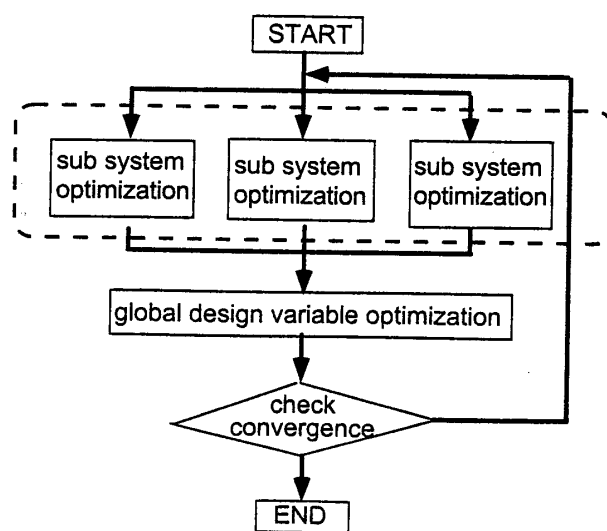


Fig. 1 Distributed Computation of Optimization Problem

ここで、点線で囲まれた各部分構造の最適化は各部分で独立に行うことができ、並列計算機のそれぞれの PE に割り当てることができる。

3.2 解析の並列処理

最適設計においては、設計変数を変化させながら何度も構造解析を繰り返す必要があり、解析に要する時間の短縮は直接最適設計に要する時間の短縮に直接結びつく。そこで、この構造解析を並列処理により計算することを試みる。線形弾性な構造物の静的解析は線形連立方程式を解くことによって行われるが、この解法には大きく分けて直接法と反復法がある。一般に並列処理に適しているのは反復法で、また、最適化に用いる感度解析を差分的に行う場合には、設計変数をわずかに変化させての再解析を何度も行う必要があるため、その意味でも反復法が有利である。

反復法として対角項成分を前処理行列として用いた前処理付き共役勾配法 (PCG 法) を用いる。PCG 法で $Ax=b$ を解く時のアルゴリズムは以下になる。

- (1) x_0 の設定
- (2) $r_0 = b - Ax_0$
- (3) $z_0 = M^{-1}r_0$, $p_0 = z_0$, $k=0$
- (4) $\alpha_k = \frac{r_k^T z_k}{p_k^T A p_k}$
- (5) $x_{k+1} = x_k + \alpha_k p_k$
- (6) $r_{k+1} = r_k - \alpha_k A p_k$, $z_{k+1} = M^{-1}r_{k+1}$
- (7) $\beta_k = \frac{r_{k+1}^T z_{k+1}}{r_k^T z_k}$, $p_{k+1} = z_{k+1} + \beta_k p_k$
- (8) 収束判定 $k=k+1$ (4)にもどる

このアルゴリズムの中で、全体の自由度が大きくなってくるとともに、計算量が最も増えてくるのは(4)と(6)に現れる行列 A とベクトル p_k の積である。そこで、この計算を並列処理するために、Fig. 2 のように剛性行列を部分構造に分割し、それぞれの計算を PE に割り当てていくこととする。これ以外の部分も同じ考え方で部分構造ごとに複数の PE に割り当てていくことは可能であり、それによってより一層の並列化を行うことができるが、ここではそれは行わない。

また、感度を計算する場合の構造解析を、部分構造のみの解析で行うというアプローチも考えられる。すなわち、

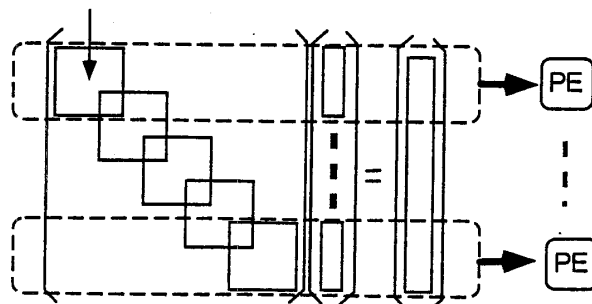


Fig. 2 Distributed Computation of Structural Analysis

部分構造に対して境界条件として全体解析における変位または荷重を与え、部分構造内の設計変数をわずかに変えて部分構造の解析を行うことによって近似的な構造解析を行うことができる (Fig. 3)。この解析の簡略化による計算効率の向上は劇的であり、各部分の最適化と解析を部分構造内でローカルに行えるため、並列処理に非常に適しているが、解析の正確さに問題がある。感度解析においてはそれほど高い解析精度は要求されないが、4章で示すトラスの断面積の最適化問題に対してこの様な方法を実際に試みた所、感度の解析精度の問題により最適化問題の収束が思わしくなかった。部分構造での自由度が比較的大きくなり、境界での自由度に比べてない部の自由度が大きくなればこの様な方法も可能であると思われるが、ここでは次節に述べるような別の計算モデルを用いて最適化、解析の並列処理を行う。

3.3 最適化、解析の同時並列処理

一般に、前述の最適化の分散処理も解析の分散処理も、構造を小さく分解して1つのブロックの計算量が非常に小さくなってしまうと、計算の粒度（他の部分と独立に計算可能な演算の大きさ）が小さくなり、並列化の効率が落ちることが知られている。すなわち、いくら超並列計算機において多くの PE が利用可能でも、それに合わせて構造を細かく部分分割するのは必ずしも得策ではない。そこで、ここでは3.1節で示した最適化の並列処理と、3.2節で示した解析の並列処理を同時に行うことによって、構造の分割をあまり多くせず、より多くの PE を利用することを考える。例えば、Fig. 4 のように構造を5つの部分構造に分割した場合、5つの PE をそれぞれの構造の最適化に割り当て、さらにそれぞれが5つの PE を構造の解析用に持っているというように、最適化と構造解析をそれぞれ縦横のクロスバネットワークに割り当てることにより、 $5 \times 5 = 25$ 個のプロセッサを用いて計算を行うことができるようになる。この計算モデルによって、少ない分割でもより多くのプロセッサを使った効率の高い計算を行うことが可能になる。

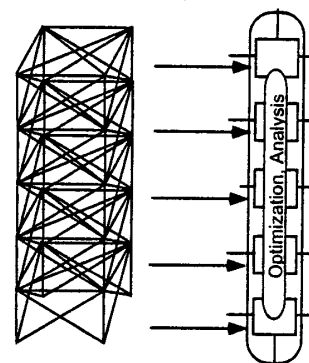


Fig. 3 Distributed Computation of Optimization and Structural Analysis (Model 1)

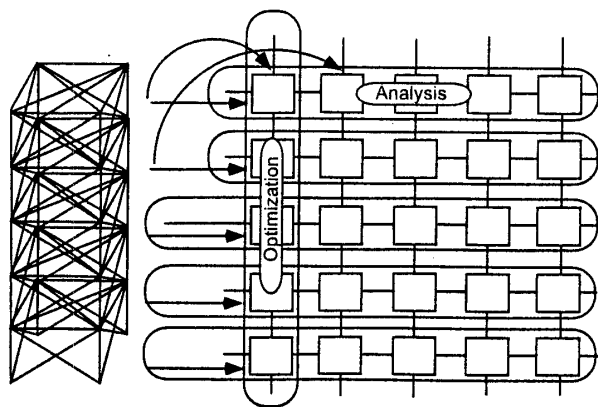


Fig. 4 Distributed Computation of Optimization and Structural Analysis (Model 2)

4. 例 題

4.1 問題設定と計算効率の評価

例題として、3次元トラスの解析、最適設計を行った。最適化の設計変数としては各部材の断面積を用いた。この場合は、グローバルな設計変数は存在しない。目的関数として全体の構造重量、制約条件として各部材の応力の上下限、および座屈条件を課した。

並列計算の効率の指標として、以下の2つを用いる。

$$S = \frac{T_1}{T_n} \quad (\text{スピードアップ率})$$

$$E = \frac{1}{n} \frac{T_1}{T_n} \quad (\text{並列化効率})$$

ただし、1プロセッサで計算を行った場合の時間を T_1 、 n プロセッサで計算を行った場合の時間を T_n としている。

4.2 解析の並列計算結果

まず、解析の並列処理に関する計算効率を考える。Fig. 5に示すようなトラス構造に対して、3.2節で説明した並列処理を行った。構造はすべてのケースで5分割し、5つのPEを用いて計算を行った。Table 1に結果を示す。粒度の小さい場合の並列化効率低下の原因は並列処理できない部分の存在と、通信によるオーバーヘッドである。PCG法のアルゴリズムにおける浮動小数点計算の量により単純に並列化効率を推定すると、以下のようになる。

$$E = \frac{2n^2 + 18n}{2n^2 + 18ns + l} \quad (s=5)$$

ただし、 n は部分ブロックの自由度、 s は分割数、 l は通信量を表している。通信量を適当に仮定したこの式を、実際の計算結果における並列化効率と同時に Fig. 6に示す。この式は非常に粗い近似であり、実際の効率との一致はあまりよくないが、粒度が大きくなっていくに従って並列化効率が上がってきて、各部分の自由度が数百のオーダーになれば、十分な並列化効率を得られることがわかる。

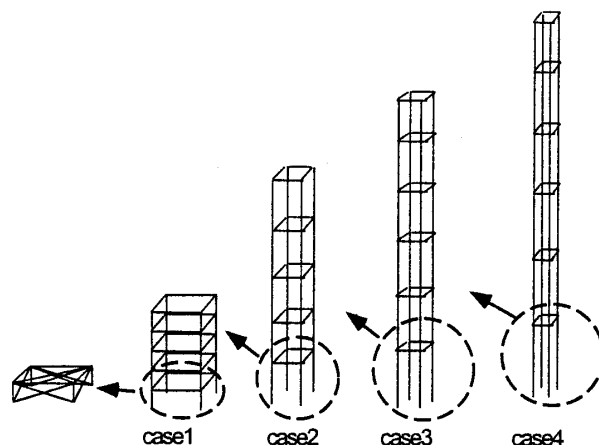


Fig. 5 Example Problem of the Parallel Analysis

Table 1 Results of Parallel Computation of Analysis

Case	Degree of Freedom		Time (sec)		S	E
	Total	Sub-system	1 PE	5 PE		
1	72	24	0.02	0.02	1.00	0.20
2	312	72	0.29	0.17	1.71	0.33
3	1512	312	26.12	7.40	3.53	0.70
4	3012	612	172.37	44.78	3.85	0.78

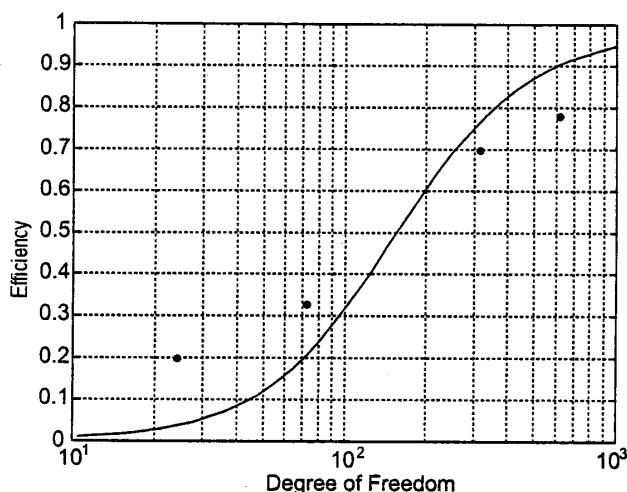


Fig. 6 Efficiency of the Parallel Computation of Analysis

4.3 最適化の並列計算結果

最適化の並列処理の例として Fig. 7に示す450部材(450設計変数)、104節点からなる3次元不静定トラスを25分割し、25プロセッサを用いて分散並列最適設計を行った結果を、分割しないで全体を一度に最適化した場合と比較した。2種類の荷重ケースによる結果を Table 2に示す。ただし、Iterationとは全体を一度に最適化した場合は可能方向法における反復回数、分散最適化した場合は各部分システムの最適化を行った回数を示す。

ここで特徴的なのが、それぞれの荷重ケースに関して、目的関数である重量の最適値が大きく異なることである。

すなわち、分割しないで一度に最適化を行った場合はローカルな最適解に落ち込んでしまったと考えられる。一方において、分散最適化を行った場合は、よりよい解に到達している。この現象を数学的にきちんと説明することは難しいが、概念的には以下のように説明できる。すなわち、個々のサブシステムの最適化を十分に収束させる前に全体のシステムの最適化をある程度行うことによって、Fig. 8(a)に示すように多峰性の解の空間に対してある種のスムージングを行っていると考えることができる。また、同じ意味であるが、グローバルな最適化において個々の細かい制約条件を考慮しない事により、Fig. 8(b)のように可能空間も単純な形になり、ローカルな最適解への落ち込みがおこりにくいと考える。

この例以外にも、いくつかのケースに対して、一度に最適化を行った場合と分散的に最適化を行った場合を比較したが、分散的に最適化を行った方がよりよい解に到達する傾向が一般的に見られた。最適値が大きく異なるので全体を一度に行った場合と分散的に行った場合の計算時間の

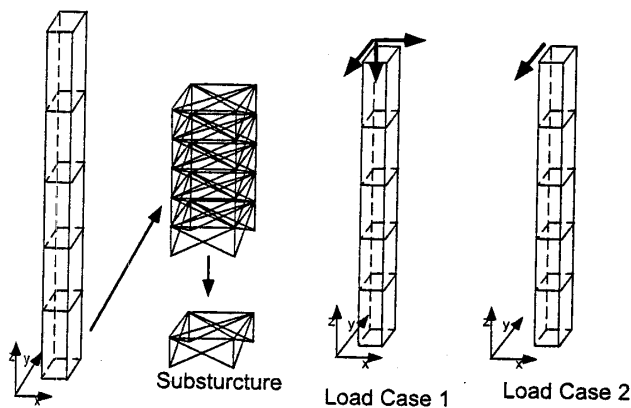


Fig. 7 Example Problem of the Parallel Optimization

比較は意味を持たないが、一般に分散的に行うことによって互いに関連の薄い制約条件、設計変数同士の感度計算が不用になるため、計算量が少なくてすむ。

また、それぞれの荷重ケースに対して並列化効率率は44%, 64%となった。並列化効率率がこれ以上に上がらないのは、それぞれの部分構造で最適化に必要な時間が異なるので、プロセッサ間の不均一な負荷による待ち時間が生じるためであると考えられる。Fig. 9にそれぞれのプロセッサの使用率を示す。明らかに、荷重ケース2のほうが様な負荷率により近く、高い並列化率を裏づけている。一般に分割数を増やすとこの負荷の不均一が大きくなり、並列化効率率が下がることが観測された。

4.4 解析および最適化の並列計算

最後に、Fig. 5 Case 3のトラスの例題(自由度1512)に対し、部材をグループ化することによって100個の設計変数

Table 2 Results of Parallel Computation of Optimization

		Weight	Time (min)	Iteration
Case 1	No Division	21.50	248	130
	25 Division	18.52	39	67
Case 2	No Division	18.44	251	98
	25 Division	18.52	40	60

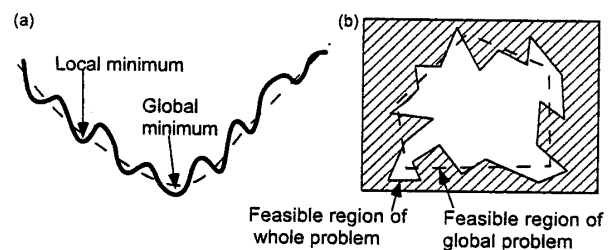


Fig. 8 The Local and Global Optimal Points

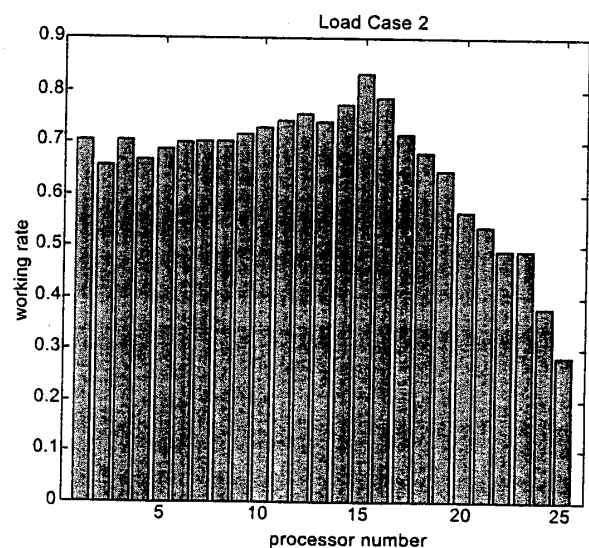
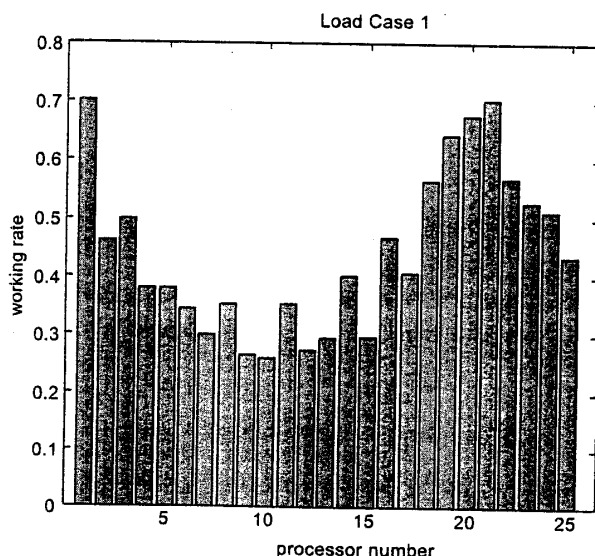


Fig. 9 Working Rate of Each Processing Element

にし、解析および最適化の両方の並列計算を3.3節に述べた計算モデルにしたがって行った。全体を5つの部分構造に分割し、それぞれを縦横の25のプロセッサに割り当てることによって、12.1倍のスピードアップ率を達成することができた。また、その際の並列化効率率は48%であるが、これは解析の並列化効率と最適化の並列化効率を掛けたものであると解釈することができる。これにより、問題規模が大きい場合には解析、最適化の両方を並列化することにより非常に大きなスピードアップ率を得ることができ、ここで提案する並列計算のモデルによって大幅な計算時間の短縮が可能であることがわかる。

5. 結 論

本論文では、構造最適設計を分散的に行うことを試みた。解析と最適化をそれぞれ並列化することによって、超並列計算機を用いた計算に適した計算モデルを提案した。超並列計算機により3次元トラスの問題に対して最適設計を行った結果、以下の結論を得た。

- ・構造解析の並列処理を行い、十分な並列化効率を得た。
- また、粒度の小さい問題では十分な並列化効率を得られ

ないことを確認した。

- ・最適化の並列処理を行い、超並列計算機によって高い計算効率を得られるのみならず、多くの場合よりグローバルな最適解に近い良い解が得られやすいことがわかった。しかし、プロセッサ間の負荷の不均一により並列化効率をある程度以上上げることは難しい。
- ・最適化と解析の両方を同時に並列処理を行う事を試み、超並列計算機で多くのプロセッサを効率的に使うことができることを確認した。

参 考 文 献

- 1) 鈴木克幸, 大坪英臣「多段階最適化手法による船体構造最適設計」日本造船学会論文集第178号, pp. 405-411 (1995)
- 2) 鈴木克幸, 大坪英臣「分散構造最適設計におけるシステム間の制約の導入」計算工学講演会論文集 Vol. 2 No. 2 pp. 613-616 (1997)
- 3) Haftka, R. T. and Gurdal, Z. "Element of Structural Optimization, Third Revised and Expanded Edition", Kluwer Academic Publishers (1992)
- 4) DOT User's Manual, VMA Engineering (1993)