

NetSolveを用いたトラス構造物の最適化計算

Truss Structural Optimization using NetSolve System

正 廣安 知之 (同志社大工) 正 三木 光範 (同志社大工)
○非 下坂 久司 (同志社大院) 非 佐野 正樹 (同志社大院)
非 谷村 勇輔 (同志社大院)
正 三村 泰成 (東京大院) 正 吉村 忍 (東京大院)
非 Jack Dongarra (Department of Computer Science, University of Tennessee)

Tomoyuki HIROYASU, Doshisha University, tomo@is.doshisha.ac.jp
Mitsunori MIKI, Doshisha University, Tatara Miyakodani 1-3, Kyo-Tanabe, Kyoto
Hisashi SHIMOSAKA, Graduate School of Engineering, Doshisha University
Masaki SANO, Graduate School of Engineering, Doshisha University
Yusuke TANIMURA, Graduate School of Engineering, Doshisha University
Yasunari MIMURA, Graduate School of Frontier Sciences, The University of Tokyo
Shinobu YOSHIMURA, Graduate School of Frontier Sciences, The University of Tokyo
Jack Dongarra, Department of Computer Science, University of Tennessee

In this paper, the APIs for general optimization system in the Grid is proposed. The system is supposed to be constructed with the Grid RPC. The proposed APIs consists of three modules. One of them is the API for searching for the data from the optimizer and analyzer. The others are the APIs for the optimizer and analyzer. In this paper, the optimization system is implemented with the proposed APIs and the NetSolve that is one of the Grid RPC systems. To discuss the effectiveness the proposed APIs and the implemented system, the structural optimization problem of truss structure is solved by the implemented system. Through the optimization, it is found that the proposed APIs are very useful to construct the optimization system in the Grid. At the same time, since there is an overhead time of NetSolve, the calculation time of analyzing the problem should be longer to shorten the total execution time.

Key word: Grid Computing, Grid RPC, NetSolve System, Optimization Computation

1 はじめに

近年、コンピュータとネットワークの発達により Grid が注目されている。Grid は広域にまたがって存在する種々の資源を結びつけて、様々なサービスの構築を容易にする¹⁾。例えば、大規模な計算資源を統合し、さらに大規模な計算システムの構築も可能である。Grid を利用した遠隔テレビ会議やリモートセンシングなどに応用されることも期待される。大規模計算と大規模データベースのシームレスな統合も Grid の目指すところであろう。これらの Grid に関連する研究および規定は主に Global Grid Forum にて検討されている²⁾。

本研究の目的は、Grid 環境下における最適化システムの構築にある。最適化問題の多くは、一般に非常に計算コストが高く、かつ、実際の問題は複合的な問題が多いため、Grid 環境下においてその処理を行うことで効率的なシステムの構築が可能であることが期待される。「Grid」とは元々 Power Grid (電力網) にその語源がある。すなわち、エンドユーザは電気製品をコンセントに接続するだけで、電力の供給を受けることが可能である。ユーザは供給された電力が実際にどこで製造されたのか、どのように配送されたのか、また、それらの品質がどの程度であるのかを知る必要はない。Grid 環境下における最適化

システムにおいても、エンドユーザは物理的にどこで最適化や解析が行われているのかを知ることがなくとも最適化の結果が得られることが重要である。また、最適化の手法の種類や解析対象は、モジュール化され簡単に拡張可能であることが必要となる。

本研究ではシステムの構築を Grid RPC と呼ばれるミドルウェアの利用を考える。特に、実際の実装においては、Netsolve³⁾ と呼ばれる Grid のミドルウェアを用いてシステムを構築する。提案する最適化システムでは、まず、最適化における次探索点を決定する Optimizer とある設計変数に対応する目的関数値や制約条件値を決定する Analyzer (Function Call) 部分に分離して考える。Optimizer と Analyzer を統合したシステムは特定の問題に対しては効率的な場合もあるが、最適化手法や対象問題の拡張性に問題があり、上記の条件を満たさない。次に提案システムは、個々のサーバに用意されている Optimizer を管理し、システム利用者から要求を受けつける Optimizer Agent と、個々の Optimizer から要求を受けつけ、個々の Analyzer に計算を依頼する Analyzer Agent の2つの Agent からなる。最適化システム開発者たちは、個別に Optimizer と Analyzer を開発する。システム利用者は、それが Grid 上のどこに存在し、どう利用できるのかを詳しく知らずとも、簡単にシステムを利用すること

ができる。

本研究では、提案する最適化システムを実現するために必要な Grid RPC のための API と Netsolve を用いた実装方法について述べる。続いて、提案システムの一部を遺伝的アルゴリズム、および有限要素法を用いたトラス構造解析により実装して数値実験を行った結果を示し、考察を行う。

2 Grid

2.1 Grid の概要と特徴

Grid とは広域ネットワークに接続されているあらゆる計算資源及び情報資源を統合して、ユーザが手軽に利用できることを目指し、これを実現する基盤技術(ソフトウェア、ハードウェア、ネットワーク)と応用プログラムを意味する⁴⁾。

Grid では種々のサービスが広域に接続された状況で、安全かつ効率的に計算資源を利用するには、ユーザ認証やスケジューリング、通信、セキュリティ、フォールトトレランス、どこにそんな情報が存在しているかといった情報サービスなど、多岐にわたる点を考慮しなければならない。

このような問題を考慮しながら、アプリケーションのプログラミングを行うことは非常に困難である。そのため、Grid に関する研究は初期の頃より、上記の問題を解決する Grid のミドルウェアの開発が数多くなされている。代表例として、Globus⁵⁾ や Legion⁶⁾、NetSolve³⁾、Ninf⁷⁾ といったシステムが挙げられる。これら Grid のミドルウェアを利用することで、簡単に Grid 上の資源を利用するシステムを構築することができる。

2.2 Grid RPC

現在 Grid のミドルウェアはさまざまなものが開発されており、いくつかのアプリケーション・プログラミングモデルを提供する。Grid RPC はその代表的なものの一つである。

RPC(RPC:Remote Procedure Call) とはリモートシステム上に存在するサブルーチンをネットワーク経由で呼び出す機能であり、Grid 上で使える RPC は Grid RPC と呼ばれている。RPC に基づく計算システムはユーザビリティに優れ、広域計算環境である Grid 上のミドルウェアの一形態として有望である。Grid RPC はすでに様々な分野で実用的に使用されつつあり、標準化への取り組み⁸⁾ も行われている。このため、Grid RPC は今後ますます発展すると考えられる。

2.3 NetSolve System

NetSolve System³⁾ は、Jack Dongarra 等によって開発された Grid RPC に基づくシステムである。

NetSolve は次の 3 つの主要コンポーネントから構成されている。

1. The NetSolve Client
2. The NetSolve Agent
3. The NetSolve computational resources (or servers)

図 1 のように、NetSolve システムを使用し Grid 上のライブラリやハードウェアを使用したいユーザやアプリケーションは、NetSolve Client となる。NetSolve Client は NetSolve Client Interface(これは API として提供される)を使用し、NetSolve Agent に対して、ライブラリの使用を要求する。

NetSolve Client からの要求を受けつけた NetSolve Agent は、あらかじめ登録されている各 NetSolve Server のロードバランスや使用可能なライブラリを考慮し、適切な Server へと、NetSolve Client の要求を転送する。

NetSolve Agent によって、選択された NetSolve Server は、NetSolve Client からの要求を受け、ライブラリを実行し、NetSolve Client の要求を処理し、結果を返す。

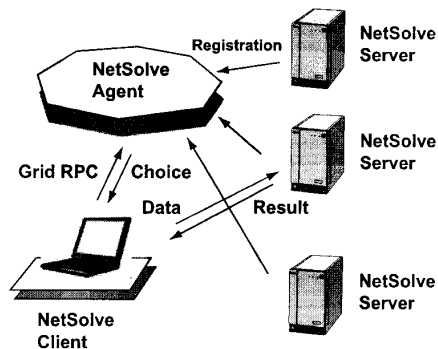


図 1: Use of The NetSolve System

3 Grid RPC を用いた最適化計算のための提案システム

本研究では Grid RPC を利用した際の最適化システムのフレームワークを提案する。また、実際に、NetSolve を利用して実装し、簡単な評価実験を行う。

3.1 最適化システム

本研究では、最適化計算における次探索点を決定する Optimizer と解析を行い目的関数や制約条件の値を決定する Analyzer を分離して考える。先に述べたとおり、Optimizer と Analyzer を統合したシステムは、特定の問題に対しては非常に優れたシステムとなる。しかしながら、実際には、Optimizer と Analyzer の開発者、研究者は異なる。そのため、Optimizer と Analyzer と分離することで、それぞれの開発を独立に行うことができ、高性能なシステムを構築することが可能となる。また Optimizer は

各 Analyzer 用の Wrapper を用意することで, Analyzer 間の相違点を吸収することができる。このことは, ある一つの Analyzer に依らない汎用的な Optimizer の開発を容易にする。

3.2 Grid RPC を用いた最適化システムの論理モデル

本研究では, Grid RPC を用いた最適化システムを提案する。提案する Grid RPC を用いた最適化システムを図 2 に示す。図 2 では, 一例として, Optimizer として Gradient Method, GA, SA の各 Server があり, Analyzer として FEM, CFD, MDA の各 Server があることを仮定している。

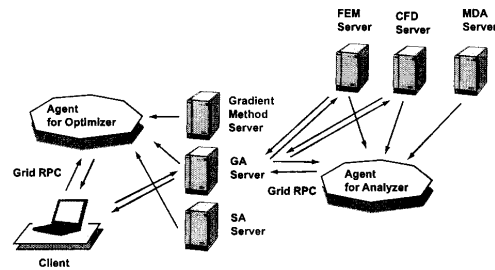


図 2: Optimization System using Grid RPC

このシステムの特徴は次の通りである。

- Optimizer 用の Agent (Agent for Optimizer) と Analyzer 用の Agent (Agent for Analyzer) が存在する。
- 各 Optimizer は Agent for Optimizer, 各 Analyzer は Agent for Analyzer に登録する。
- 各 Optimizer の Server は Agent for Analyzer の所在を知っており, 解析する問題によって Agent を通じて Grid RPC による実行要求により適正な解析用の Server を選択できる。

本システムでは, 前節で述べたように最適化計算を行う Optimizer と解析を行う Analyzer を分離して考え, Grid 環境下でのシステムを構築することで, 次のような利点を持つ。

- エンドユーザは Agent for Optimizer が動作するサーバのみを知っていればよく, どこに Optimizer や Analyzer, Agent for Analyzer がどこにあるのかを知る必要はない。
- ユーザは Optimizer と Analyzer の実行ファイルを用意する必要がない。
- Optimizer, Analyzer は Grid として提供される計算資源で動作するため, ユーザは強力なマシンを用意しなくても, 高速に計算が可能である。

表 1: Arguments of API for Optimizer

Name_of.Optimizer	Specification of Optimizer
Tag	Specification of Tag
Name_of.Analyzer	Specification of Analyzer
Inputfile	Path to Input File
Outputfile	Path to output File

3.3 Grid RPC による最適化システムのための API

Grid RPC により最適化システムを構築するために, 本研究ではいくつかの API を用意した。以下にそれらについて説明する。システム開発者はこの API に沿った Optimizer および Analyzer を構築しなければならない。

3.3.1 サーバの検索

提案システムでは, 以下の Calling Sequence を用意し, 利用可能な Optimizer あるいは Analyzer の情報を得ることができる仕組みを提供する。これは NetSolve での, サーバの検索を行う Calling Ssequence を基にしている。

[opt.analyzers]

現在使用可能な Analyzer の一覧を表示する

[opt.analyzers show Analyzer_Name]

指定した Analyzer (Analyzer_Name) の説明を表示する

[opt.analyzers subanalyze Analyzer_Name]

指定した Analyzer が, 内部で使用するサブ Analyzer を表示する

[opt.optimizers]

現在使用可能な Optimizer の一覧を表示する

[opt.optimizers show Optimizer_Name]

指定した Optimizer (Optimizer_Name) の説明を表示する

3.3.2 Optimizer の API

Optimizer の API はシステム利用者のクライアントからのみ呼び出される。Optimizer の API は次に示すもので, 表 1 に示す 5 つの引数をとる。

```
optimize(Name_of.Optimizer, Tag,
Name_of.Analyzer, Inputfile, Outputfile);
```

Optimizer の実行するジョブの指定をする Tag は次の 3 つの指定ができる。

[Analyze.Init]

指定された Analyzer のデフォルトの入力ファイルを Analyzer に送る。

表 2: Arguments of API for Analyzer

Name_of_Analyzer	Specification of Analyzer
Tag	Specification of Tag
Inputfile	Path to Input File
Outputfile	Path to output File

[Run]

指定された Optimizer の入力ファイルから Optimizer を実行し、出力ファイルに結果を返す。

[Analyze_Config]

指定された Analyzer に Input Config ファイルと Output Config ファイルを送る。Input Config ファイルとは analyzer の入力ファイルと設計変数との対応表のことで、これを基に Analyzer の wrapper は Analyzer の入力ファイルを変更する。Output Config ファイルとは analyzer の出力ファイルと目的関数および制約条件の対応表のことで、これを基に Analyzer の出力ファイルから wrapper は目的関数値および制約条件値を抽出する。

3.3.3 Analyzer の API

Analyzer の API は主に Optimizer から呼び出される。つまり、Optimizer は前節で示した Optimizer の機能のいくつかを Analyzer の API を用いて実装する。Analyzer の API は次に示すもので、表 2 に示す 4 つの引数をとる。

```
analyze(Name_of_Analyzer, Tag,
Inputfile, Outputfile);
```

Optimizer の実行するジョブの指定をする Tag は次の 4 つの指定ができる。

[Initialize]

指定された Analyzer の設定ファイルを Analyzer に送る

[Config]

指定された Analyzer に Input Config ファイルと Output Config ファイルを送る

[Solve]

指定された Analyzer の入力ファイルから Analyzer を実行し、出力ファイルに結果を返す

[Result]

指定された Analyzer の全ての解析結果を出力ファイルに返す

3.4 提案システムの実行

ユーザは次のような手順で提案システムでジョブを実行する。

1. opt.analyzers などを利用し、Analyzer を選択する
2. opt.optimizers などを使用し、Optimizer を選択する
3. opt.analyzers subanalyze Analyzer_Name などを使用し、選択した Analyzer のサブ Analyzer をチェックする。
4. 全ての Analyzer の設定ファイルを用意する。
5. 全ての Analyzer の Input Config ファイルと Output Config ファイルを用意する。
6. 全ての Analyzer に対して、設定ファイルを送信する
optimize(Name_of_Optimizer, Analyze_Init, Name_of_Analyzer, Inputfile, Dummy);
7. Analyzer に対して、Input Config ファイルと Output Config ファイルを送信する
optimize(Name_of_Optimizer, Analyze_Config, Name_of_Analyzer, Input.config_file, Output.Config_file);
8. 選択した Optimizer の入力ファイルを用意し、Optimizer を実行する。
optimize(Name_of_Optimizer, Run, Name_of_Analyzer, Inputfile, Outputfile);
この時結果は Outputfile に返される。

3.5 NetSolve を用いた最適化システムの実装

上記の Optimizer および Analyzer の API を基に NetSolve を用いて実装を行った。実装は極めて容易で、NetSolve で用意されている関数を用いて API に沿ったライブラリを構築すればよい。

例えば、Optimize() を実装する場合には以下のようになる。

```
netst(Name_of_Optimizer, Tag,
Name_of_Analyzer, Inputfile, Outputfile);
```

Analyzer() を実装する場合には以下のようになる。

```
netst(Name_of_Analyzer, Tag,
Inputfile, Outputfile);
```

これからもわかるように、実装には他の通信関数を利用することも可能である。

図 2 の提案システムにおいて、Optimizer に GA, Analyzer に CFD を選択した場合の、各 Client, Sever における Optimizer 及び Analyzer, NetSolve の API が呼び出される手順について図 3 に示す。

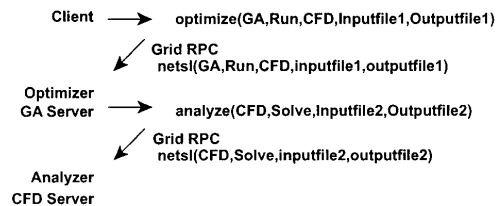


図 3: Flow Chart of API Call

4 数値実験

4.1 遺伝的アルゴリズム及びトラス構造解析を用いた提案システムの実装

NetSolve を用いた実装した提案最適化システムの性能を評価するために、Optimizer に遺伝的アルゴリズム (GA:Genetic Algorithm)⁹⁾、Analyzer にトラス構造解析を用いて実装する。対象問題はトラス構造物の最適設計問題である。

GA は生物の進化を模倣した優れた最適化手法の一つである。GA は複数の解候補 (個体) による多点探索を行う。

トラス構造物の最適設計問題とは、ある節点に力を加えて、複数の制約条件を与えたとき、それらを満たす最小体積のトラス構造物を設計することである。制約条件には、各節点の変位、各部材の応力や座屈が考えられる。

本研究で用いるトラス構造解析は、各部材の断面積を設計変数とする。Optimizer である GA は、各個体の設計変数をトラス構造物の部材の断面積として Analyzer に解析を依頼する。Analyzer は、トラス構造解析を行い、トラス構造物の体積、全ての節点の変位、全ての部材の軸応力と座屈応力のいずれかを返す。

GA は一般的に制約条件を陽に扱わないため、本研究では、トラス構造物最適設計問題を解く上で制約条件を扱うために、ペナルティ関数法を適用した。

ペナルティ関数法は満たされない制約条件を罰金化した項を目的関数と組み合わせた関数を最適化するという方法である。制約条件として、全ての節点の変位量の上限、全ての部材の軸応力の上限を設け、また全ての部材に座屈がないこととした。このため、GA がトラス解析によって目的関数値を得るには、トラス構造物の体積と節点の変位量、部材の軸応力、座屈応力が必要になる。

本節で、対象としたトラス構造物は、図 4 に示すような、22 節点 50 部材のトラス構造物である。このトラス構造物は、5 部材の基本構造のトラスを 10 段重ねた構造となっている。またこのトラス構造物は最下層の二つの節点を固定し、最上層の一つの節点に水平加重 1000.0N を付加する。使用材料は簡単のため縦弾性係数 1.0GPa の線形弾性材とする。制約条件は、変動する節点の変位が 0.003m 以下、全部材の応力が 2.0MPa 以下、全部材にお

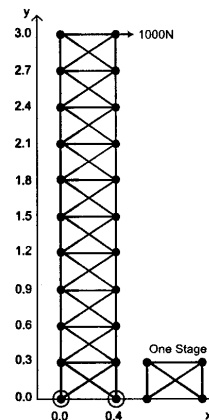


図 4: 10 Stage truss

表 3: Machine Spec.

CPU	PentiumIII 800MHz
Memory	256MB
Network	100Mbps(Fast Ethernet)

いて座屈がおこらないことの 3 つである。

よって、22 節点 50 部材 144 制約のトラス構造物の最適設計問題である。

4.2 実験環境

実験に用いたマシンスペックを表 3 に示す。全ての Client, Agent, Server に表 3 のノードを割り当てた。

4.3 複数台の同一アナライザによる並列処理

Grid 環境においては、計算資源が膨大に存在することが考えられ、同一の解析を行う Analyzer も複数台存在することが考えられる。その場合には、それらを並列処理にて使用することにより、最適化の高速化が望まれる。

図 5 に示す最適化システムの場合、Optimizer の GA Server は複数の個体の解析を独立に扱えるため、並列に処理することができる。図 5 ではトラス解析を行う同じ種類の Analyzer が 2 つあり、2 個体のトラス解析を同時に処理できる。

図 5 に示す、NetSolve を用いた 2 つのトラス解析を並列に実行する最適化システムと、NetSolve を用いず、GA によって逐次にトラス解析を行いトラス構造物最適設計問題を解く最適化システムの実行時間の計測を行い、比較を行う。今回の対象問題は規模が小さいために、解析に必要な時間は非常に小さい。そのため、解析時間によって、結果がどのように変化するかを検討するため、一回のトラス解析に 0ms, 1500ms, 3000ms の時間をそれぞれ

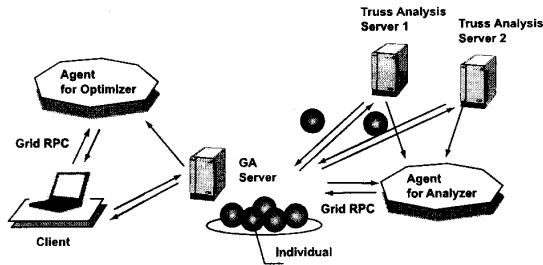


図 5: Optimization System used for the experiment

表 4: Parameter of Experiment

Number of Individuals	20
End Condition	100 Generations
Number of Truss Analysis	2020
Number of NetSolve RPC	2023

れ付加した。実験に用いたパラメータを表 4、トラス解析に各時間を付加した実行時間の比較結果を図 6 に示す。

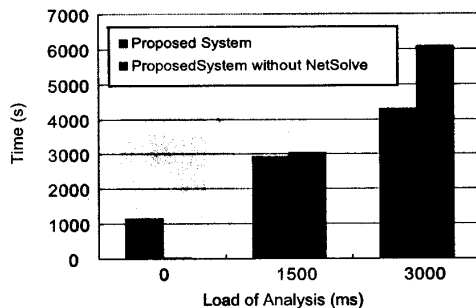


図 6: Comparison of Optimization System and Optimization System without NetSolve

結果より、トラス解析に時間を付加しなかった場合、NetSolve を使用しない最適化システムでは約 10 秒、NetSolve を使用した最適化システムでは約 1150 秒の実行時間となる。つまり、NetSolve を使用することによるオーバーヘッドは全体で約 1140 秒、一回の RPC では 0.56 秒程度となる。しかしながら、トラス解析に時間を付加し、解析時間が長くなるにつれ、NetSolve を使用した最適化システムでは、このオーバーヘッドが隠れ、並列化による効果ができることがわかる。すなわち、NetSolve のオーバーヘッドを隠蔽するだけの解析時間が必要である。今回の場合には約 1500ms を必要とした。

5 まとめ

本研究では、Grid RPC を用いた最適化システムを構築するための API を提案した。また、実際のシステムにおいて、Grid RPC の一つである NetSolve を用いて実装した。

提案システムでは、Optimizer と Analyzer を分離して考える。このことは、それぞれの開発を容易にし、システムの高性能化につながる。また Grid RPC を使用することで、システム利用者にとっては、優れた Optimizer と Analyzer を簡単に使用できる仕組みを提供し、計算資源も利用することができる。Optimizer や Analyzer の開発者にとっても、Grid の利用は互いの協調作業を促進させる。

さらに本研究では NetSolve を用いた最適化システムに、Optimizer に遺伝的アルゴリズム、Analyzer に有限要素法を用いたトラス解析を実装し、実験を行った。

その結果、NetSolve を使用することで、ある程度のオーバーヘッドが発生することが明らかとなった。しかしながら、Optimizer において Analyzer への複数の RPC を並列に処理することにより、解析にある程度の時間を要する処理においては、高速化が期待できることを明らかにした。

謝辞 本研究は文部科学省科学研究費補助金、および文部科学省学術フロンティア推進事業により支援されている。

参考文献

- 1) I. Foster, C. Kesselman, and S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International J. Supercomputer Applications* 15(3), 2001.
- 2) Global Grid Forum. <http://www.gridforum.org/>.
- 3) H. Casanova and J. Dongarra. NetSolve: A Network Server for Solving Computational Science Problems. *Proc. of Supercomputing '96 Conference*, 1996.
- 4) I. Foster and C. Kesselman. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 1998.
- 5) I. Foster et.al. Globus: A metacomputing infrastructure toolkit. *International J. Supercomputer Applications* 11(2), 1997.
- 6) A.S. Grimshaw, W.A. Wulf, and the Legion team. The Legion Vision of a Worldwide Virtual Computer. *Communications of the ACM*, 1997.
- 7) H. Nakada, M. Sato, and S. Sekiguchi. Design and implementations of ninf: towards a global computing infrastructure. *Future Generation Computing Systems, Meta-computing Issue*, 1999.
- 8) S. Matsuoka et.al. GridRPC: A Remote Procedure Call API for Grid Computing. <http://www.eece.unm.edu/apm/>.
- 9) D.E. Goldberg. *Genetic Algorithms in Search Optimization and Machine Learning*. Addison-Wesley, 1989.