Grid RPCを利用したディーゼルエンジン燃料噴射スケジュール最適化

廣安 知之[†], 三木 光範[†], 澤田 淳二[††], Jack Dongarra[‡]

[†] 同志社大学工学部 [††] 同志社大学大学院 [‡] University of Tennessee/Oak Ridge National Laborator

環境問題に対する関心の高まりから，ガソリンエンジンに比べ二酸化炭素排出量の少ないディーゼルエンジンが注目されている．しかし，ディーゼルエンジンの燃料噴射スケジュールを最適化するためには高い計算負荷を必要とし，最適化に非常に長い時間が必要という問題点がある．本研究では，グリッドで動作するアプリケーションを作成するためのミドルウェアとしてGrid RPCシステムの一つであるNetSolveを用いて遺伝的アルゴリズムにおける候補解評価の並列化を行った．実験の結果，高い並列化効率を示し，その有効性が確認できた．

Optimization of injection schedule of diesel engine using Grid RPC

Tomoyuki HIROYASU[†], Mitsunori MIKI[†], Junji SAWADA[††], and Jack Dongarra [‡]

[†] Knowledge Engineering Dept., Doshisha University
[††] Graduate Student, Doshisha University

[‡] University of Tennessee/Oak Ridge National Laborator

In order to meet increasing environmental concerns and more stringent emission regulations, diesel engines that have high fuel efficiency with small amounts of NOx and soot are desired to design. Recently, parameter survey of diesel engines by computer simulation has been focused. However, for parameter survey by computer simulation, high calculation cost is necessary. Especially, when parameters are optimized by varying fuel scheduling and other parameters, it takes a long time to derive the optimum solution. In this paper, the optimization system for diesel engine combustion is performed on the Grid system using NetSolve that is one of Grid PRC systems. In the proposed system, the genetic algorithm is performed to derive the optimum solution. During the genetic algorithm is performed, the evaluation part is executed in parallel and NetSolve is utilized. Through the experiments, it is expressed that the proposed system is very effective to use on the Grid envrionment.

1 はじめに

近年，環境問題に対する関心の高まりから，ガソリンエンジンに比べ二酸化炭素排出量の少ないディーゼルエンジンが注目されている．しかし，ディーゼル燃焼のシミュレートには高い計算負荷を必要とし，ディーゼルエンジンの燃料噴射スケジュールを最適化するためには非常に長い時間が必要という問題点がある．

その問題を解決する一手法としてグリッド計算環境の利用が挙げられる．グリッド計算とは，ネットワークで接続された複数の資源を統合して1台の大規模計算機として利用する技術である[1]．

そこで，グリッド上の計算資源を利用し，複数のディーゼルエンジンの解析計算を並列に実行することにより，最適化に要する時間の短縮を試みる．本研究では，グリッドで動作するアプリケーションを作成するためのミドルウェアとしてGrid RPC[2]に注目する．Grid RPCを用いることでグリッド上に存在するライブラリやハードウェアを簡単に利用することが可能となる．Grid RPCシステムの1つであるNetSolve[3]を用いて遺伝的アルゴリズム(Genetic Algorithm : GA)[4]における候補解評価の並列化を行い，ディーゼルエンジン燃料噴射スケジュール最適化問題に適用することにより，その有効性を示す．

2 グリッドとそのミドルウェア

2.1 グリッド

大規模な計算を行うための環境として，近年，グリッドが注目されている．ネットワーク技術がめざましい進歩を遂げることにより，高い通信性能を持つネットワークが利用できるようになった．これ

に伴い，広い地域に配置された高性能な計算サーバや大規模なストレージを持つファイルサーバといった計算資源や，計算を行うのに必要なデータやプログラムのような情報資源を結びつけて，広域的に分散／並列計算を行うための基盤となるグリッドと呼ばれる計算環境が研究されるようになった[1]．

2.2 GridRPC

グリッドで動作するアプリケーションを作成するための方法の 1 つとして，GridRPC[2] の利用がある．GridRPC は，遠隔地に存在する計算機上のライブラリ呼び出しを提供する規格であり，Global Grid Forum にて API が規定されている．GridRPC は以下のようなことを考慮した設計となっている．

- 簡単なプログラミングインターフェースの提供
- 並列処理を実装するためのノンブロッキング呼び出しの提供
- セキュリティ
- フォールトトレランス
- 動的な計算資源の発見とスケジューリング
- サーバサイドのみでの IDL 管理

GridRPC の実装には，次節で述べる NetSolve が存在する．

2.3 NetSolve

NetSolve[3] は，Tennessee 大学 Innovative Computing Laboratory の Jack Dongarra 等によって開発されている Grid RPC システムの 1 つである．

2.3.1 NetSolve システムの構成

NetSolve の構成を図 1 に示す．NetSolve では，Client，Agent，Server の三者が存在する．グリッドのライブラリやハードウェアを使用したいアプリケーションは Client となる．Client は，まず，Agent に対して利用したい問題の問い合わせを行う．Agent では，適切な Server を選択し，Client に紹介する．その後，Client は紹介された Server に計算対象のデータを送信する．Server で計算が実行され，Client は計算結果を受信する．

2.3.2 NetSolve における RPC の実行形式

NetSolve の API を用いて RPC を実行する方法を図 2 に示す．この例では，x, dim, f のアドレスを引数にして rastrigin という問題の処理要求を行っている．status には，RPC が成功したかどうかの情報が格納される．

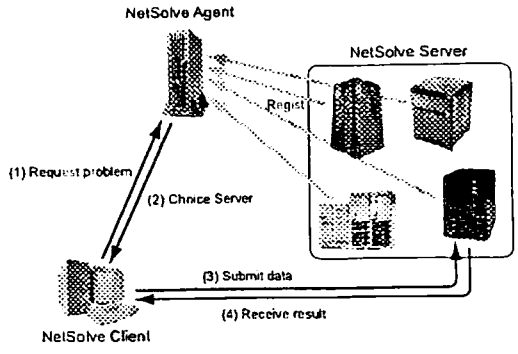

図 1: Architecture of NetSolve

```
status = netsl("rastrigin()", x, dim, &f);
```

図 2: Example of RPC

NetSolve には，RPC の実行要求の結果が得られるまで処理が停止されるブロッキング RPC と，処理が停止されないノンブロッキング RPC が提供されている．ノンブロッキング RPC を用いる場合，プログラマは RPC を実行する処理とは別に，結果を受け取るための処理を記述する必要がある．この様子を図 3 に示す．

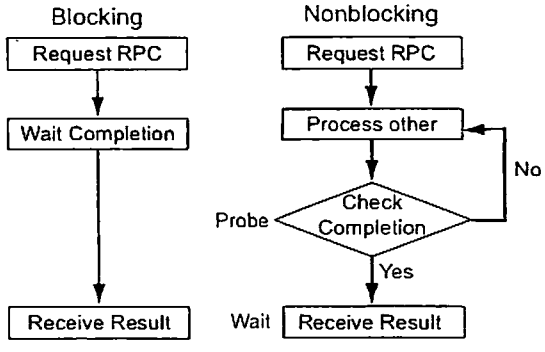

図 3: Flow of blocking RPC and non-blocking RPC

RPC の並列化を行う場合，ノンブロッキング RPC を用いることになる．しかし，実行する RPC の数が多い場合には，RPC を管理するのは困難な作業となる．そこで，NetSolve では，Farming と呼ばれる同じ問題に対する複数の処理要求を同時に発行する API が用意されている．これは，RPC の引数を配列に格納し，Farming API を呼び出すことで API 内でノンブロッキング RPC を用いて並列処理が実行されるというものである．Farming を用いれば，プログラマはノンブロッキングに送った RPC を管理する必要がなくなる．Farming の実行の模式図を図 4 に示す．Farming では，一定

の間隔を空けてRPCを投入される．その後，全てのRPCが完了するまで待機する．Serverの中に処理能力の遅いものが存在する場合，そのServerに投入されたRPCが完了するまで待機することになる．

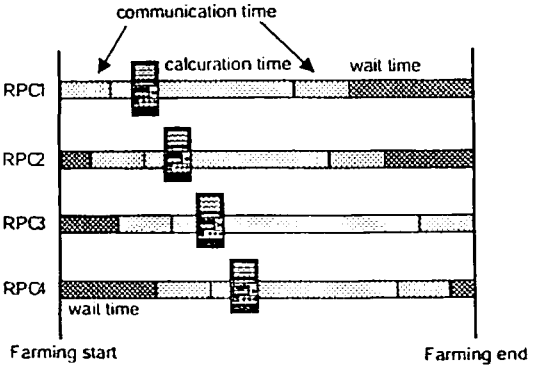

図 4: Flow of Farming

Farmingは，図5のような書式で呼び出す．この例では，rastriginという問題について，最初のRPCは，x_arr[0], dim_arr[0], f_arr[0]の値が使われ，次のRPCは，x_arr[1], dim_arr[1], f_arr[1]の値が使われる．同様に，合計100個のRPCが実行される．いずれかのRPCでエラーが起こった場合は，配列statusにエラー番号が格納される．

```
status = netsl_farm(
  "i=0,99",
  "rastrigin()",
  ns_ptr_array(x_arr,"$i"),
  ns_int_array(dim_arr,"$i"),
  ns_ptr_array(f_arr,"$i")
);
```

図 5: Example of Farming

3 最適化計算

3.1 最適化計算の構造

最適化は，OptimizerとAnalyzerという2つのソフトウェアから構成される(図 6).

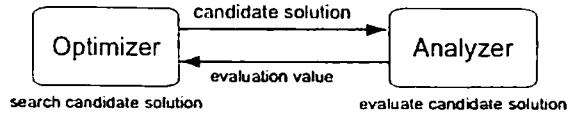

図 6: Structure of optimization

Optimizerには，遺伝的アルゴリズムやシミュレーテッドアニーリング，傾斜法などが存在し，候補解の探索が行われる．Analyzerには，ディーゼル燃焼のシミュレートやタンパク質のエネルギー計算などが存在し，Optimizerが探索した候補解に対して解析計算が行われ，評価値をOptimizerに返す．

3.2 遺伝的アルゴリズム

遺伝的アルゴリズム(Genetic Algorithm : GA)は，生物の進化の過程を模倣した確率的な最適化手法である[4]．GAでは，対象問題の候補解を生物個体とみなし，個体の集まりを母集団と呼ぶ．母集団内の各個体は，設計変数値を符号化した染色体を持つ．通常，染色体は0, 1のビット列で構成される．母集団内では，ある世代を構成している個体群のうち環境への適合度の高い個体が高い確率で生き残るように選択される．さらに，個体間の交叉や突然変異によって次世代の個体が生成される．このような世代の更新が繰り返されることによってよりよい個体が増加し，やがて最適解が得られるというのがGAの基本的な概念である(図7).

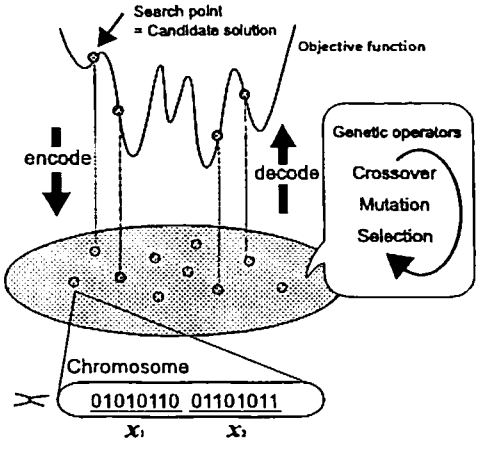

図 7: Schematic of GA

GAでは1世代につき，母集団内の個体数だけ適合度関数の評価が必要になる．そのため，一点探索の最適化手法と比較して，計算負荷が高くなる．これを解決するための方法として，GAにはいくつかの並列モデルが存在する[5]．今回は選択と交叉，突然変異は1つのプロセッサで行い，個体の評価のみを複数のプロセッサで分散処理させるというマスタースレーブモデル(図 8)を利用する．

4 GridRPCを利用したディーゼルエンジン燃料噴射スケジュール最適化システム

本研究で提案するシステムの構成を図 9に示す．

本システムでは，NetSolve Clientにて，選択，交叉，突然変異が行われる．個体評価の際には，まず，NetSolve Serverに送る個体の情報を送信バッファ

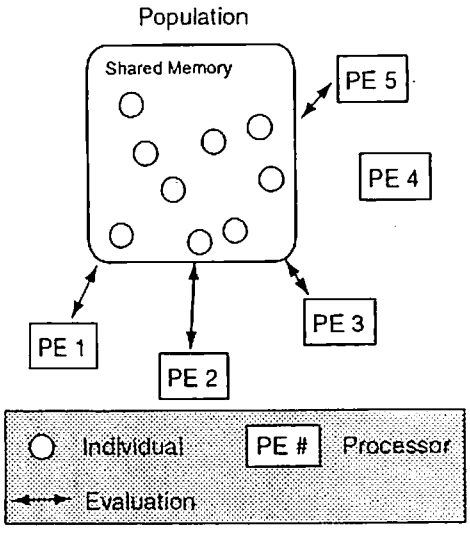

図 8: Master-Slave model

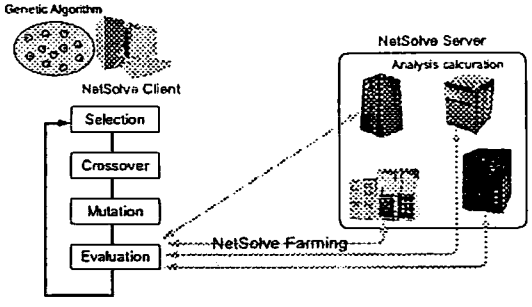

図 9: Structure of system

に登録する．全ての個体を登録後，前述の Farming を利用し，NetSolve Server への評価計算要求を並列に処理する．

本研究では，ディーゼルエンジン解析に HIDECS [6] を使用する．HIDECS は，廣安らによって開発されたディーゼル燃焼の現象論的モデルの 1 つで，噴射率 (Injection Rate)，大気状態 (Atmosphere Condition) などをもとに，ディーゼル燃焼をシミュレートし，燃費や炭素化合物の生成量を求めるソフトウェアである．

NetSolve Server では，HIDECS を用いてディーゼルエンジンの解析計算が行われる．全ての評価計算が終了したら，NetSolve Client は評価値を利用して次の遺伝的操作を行う．

NetSolve Server には，図 10 のような IDL を用意する．Server にこの問題を登録することにより，HIDECS を用いたディーゼルエンジンの解析計算を行うことが可能となる．Client から Server には，設計変数とその個数が送信される．Server は，HIDECS を実行し，結果として得られる SFC，NOx，Soot の値を返す．設計変数の数は 36 個であり，解析結果の数は 3 個である．そのため，1 回の評価計算での通信量はそれほど多くない．

```
@PROBLEM hidecs
@FUNCTION execute_hidecs
@LIB $(HIDECS_PATH)/lib/libhidecs.a
@DASHI $(HIDECS_PATH)/include
@INCLUDE "hidecs.h"
@LANGUAGE C
@MAJOR ROW
@PATH /engine/
@DESCRIPTION
HIDECS diesel engine simulator
@INPUT 1
@OBJECT VECTOR D x
Design variables
@OUTPUT 1
@OBJECT VECTOR D eval
Analysis results(SFC, NOx, Soot)
@COMPLEXITY 1,1

@CALLINGSEQUENCE
@ARG IO
@ARG mIO
@ARG OO

@CODE
@OO@ = (double *)calloc(3, sizeof(double));
*@mOO@ = 3;
execute_hidecs(@IO@, *@mIO@, @OO@);
@END_CODE
```

図 10: NetSolve IDL of HIDECS

5 数値実験

GA の個体数を 100 とし，1000 回の評価計算 (100 個の RPC 要求を 10 回) を実行するのに要した時間の計測を行う．なお，今回利用した NetSolve のバージョンは，1.5 である．

利用した実験環境を表 1 に示す．表 1 に示すように，利用した計算資源は性能の異なるノードから構成されている．各ノードは 100Mbps の FastEthernet でフラットに接続されている．

表 1: Specification of PC cluster

Type A	CPU	Pentium III 600MHz × 32
	Memory	128MB × 32
Type B	CPU	Pentium III 850Hz × 8
	Memory	256MB × 8
Type C	CPU	Pentium III 500MHz × 12
	Memory	256MB × 6

表 1 のクラスタを利用し，次のパターンについ

て計測を行う.

(A) Type A のノードのみを利用

(B) Type A と Type B のノードを利用

(C) 全てのノードを利用

まず，対象問題として，Rastrigin 関数を最適化することを考える．Rastrigin 関数は，式 (1) で定義される．この問題は非常に簡単に計算が可能なため，その計算時間はほとんど無視できる．また，今回は対象問題の次元数を 10 とした．よって，Client - Server 間での通信量もほとんど無視できる程度である．

$$f(\boldsymbol{x}) = -\left(10n + \sum_{i=1}^{n}\left(x_i^2 - 10\cos(2\pi x_i)\right)\right) \quad (1)$$
$$-5.12 < x_i < 5.12$$
$$\max(f) = f(0,\ldots,0) = 0$$

20 回試行での実行時間の平均値を図 11 に示す.

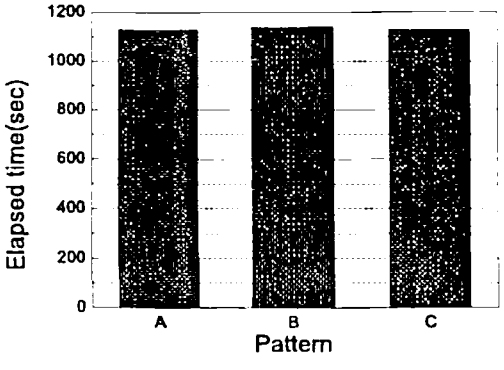

図 11: Elapsed time(Rastrigin)

図 11 より，利用ノード数を増加させても実行時間に変化がないことがわかる．このことから，計算負荷の低い問題については，NetSolve による評価計算処理の並列化が機能しないことがわかる．また，1 世代の計算に約 110 秒を要していることがわかる．Server での計算時間や Client - Server 間の通信時間は無視できる程度であるので，100 個の RPC を投入する際の Farming のオーバーヘッドは 110 秒程度であることがわかる.

次に，提案システムをディーゼルエンジン燃料噴射スケジュール最適化問題に適用する．この問題は高い計算負荷を必要とし，1 回の評価計算に Type A のノードでは，約 29 秒，Type B のノードでは，約 20 秒，Type C のノードでは，約 34 秒を要する．20 回試行での実行時間の平均値を図 12 に示す.

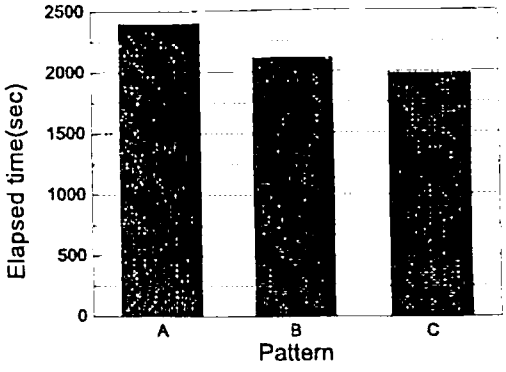

図 12: Elapsed time(Diesel engine)

図 12 より，利用するノード数を増加させるに伴い，実行に要した時間が短縮されることがわかる．このことから，NetSolve による評価計算処理の並列化が有効であることがわかる.

1 世代の計算にパターン A では約 240 秒，パターン B では約 210 秒，パターン C では約 200 秒を要している．100 個の RPC を処理する際の Farming のオーバーヘッドは 110 秒程度であるので，残りの時間は送られた RPC が全て完了するまでの待機時間だと考えられる．Farming で最後の処理要求が送られてから，結果が全て返ってくるまでの待機時間について検討した．各パターンにおける待機時間の最大値，最小値，平均値を表 2 に示す.

表 2: Wait time

Pattern	max(sec)	min(sec)	average(sec)
A	241	58	121.3
B	204	48	94.37
C	202	40	81.43

表 2 より，ノード数を増加させるにつれて待機時間が短くなっていることがわかる．これは，利用するノード数を増やすことにより，1 ノードに割り当てられる個体数が減少し，各ノードでの処理時間が短くなるからであると考えられる．また，試行によって待機時間の長さにばらつきがあることがわかる．NetSolve では，利用可能な Server の中からランダムに選択された Server に処理が投入されるため，1 台のノードに処理が集中することが考えられる．複数の処理が同時に投入された場合，1 つの処理に割り当てられる処理時間が短くなるため，全体の処理時間が長くなるのだと考えられる.

次に解析例の一例を示す．対象としているエンジンは，シングルシリンダータイプの Caterpillar 3400 シリーズトラックエンジンである．エンジンの仕様を表 3 に示す.

初期設計の設定は，文献 [7] と同一である．これ

表 3: Specification of Caterpillar 3400 Series

Bore (m)	0.1372
Stroke (m)	0.08255
Connecting Rod (m)	0.24
Cavity (m)	0.06
EPS Compress Ratio	15.6
Nozzle Number	6
Nozzle Diameter (m)	0.000214
Displacement (l)	2.44

らの設定を表 4 にまとめる.

表 4: Operation conditions of baseline case

Engine Speed (rpm)	1737
Load (% of Maximum)	57
Start of Injection (ATDC)	3.5
Injection Duration (CA)	20.5
Fuel Rate (kg/hr)	6.97
Intake Temperature (Co)	32
Intake Pressure (kPa)	184
Exhaust Pressure (kPa)	181
Exhaust Pressure (kPa)	181
EGR rate	0%

実験では文献[7]と同様の設計変数の設定により最適化を行う. 文献[7]では, 3 目的を各目的に重みをつけることにより, 1 目的の問題に変換している. また, 解法は応答曲面法と詳細論モデルの実装の一つである KIVA を利用している. KIVA は計算コストが非常に高いため, 多くの繰り返し計算を行うことができない. そのため, 設計変数値を同時に変化させるのではなく, ステップごとに変動させる設計変数を決定し, 最適化を行っている. 我々の手法では, 同時に多目的最適化を行っている. 使用している遺伝的アルゴリズムは, Neighborhood Cultivation Genetic Algorithm (NCGA) である[8, 9].

本実験では, 2 段階の多段噴射形状, ブースト圧 (boost pressure) および EGR が設計変数である. 2 段階の多段噴射形状を表現するためには, 全体の噴射期間 (duration angle), 前後の噴射間隔 (dwell between injections), 前半部分での噴射量割合 (percent of fuel in the first part) が必要となる. 前後の噴射の際の噴射間隔は同一としている. このうち, dwell between injections と percent of fuel in the first part が設計変数となる

本実験では 4 種類の設計変数が存在するが, それらの値の上下限値をまとめて表 5 に示す.

表 5: Range of design variables

Item	Min	Max	bit for GA
Dwell between injections (angle)	0	12	7
Percentage of first part (%)	50	84	7
Boost Pressure (kg/cm2)	1.62	1.83	5
EGR rate	0.0	0.50	5

NCGA では 1 個体あたり 24 ビットを利用している. 各設計変数に対して利用しているビット数は表 5 に合わせて示している.

また, NCGA におけるパラメータは表 6 に示す.

表 6: GA parameters

Population Size	200
Crossover Rate	1.0
Mutation Rate	1/bit length
Terminal Generation	100
Trials	2

図 13 に得られたパレート最適解集合を示す.

この図では, 解集合を 3 目的空間に表示したものと合わせて 2 次元平面に射影した図を示している.

この図において, 解 B は初期設計解を示す. また, 解 A は文献[7]で最適解として得られたものを HIDECS で計算した結果である. この解 A は応答曲面法にて単一目的最適化によって得られた結果である.

この図より SFC と NOx および Soot と SFC 間にはトレードオフの関係が存在することが明らかである. また, 同時に SFC と Soot 間にはトレードオフの関係はなく線形の関係が存在することもわかる. さらに, 初期設計解はパレート最適解集合からは大きく離れている.

解 A とパレート最適解集合の比較を行うと次の点が明らかとなる. 解 A はパレート最適解集合の一つであり, その NOx 値は非常に小さい. しかしながら, SFC と Soot の値はそれほど良いものではない. これは, 解 A 付近の最適解集合が弱パレート解ともみなせる解集合であり, SFC の値がその付近ではほとんど同じであることが理由の一つで

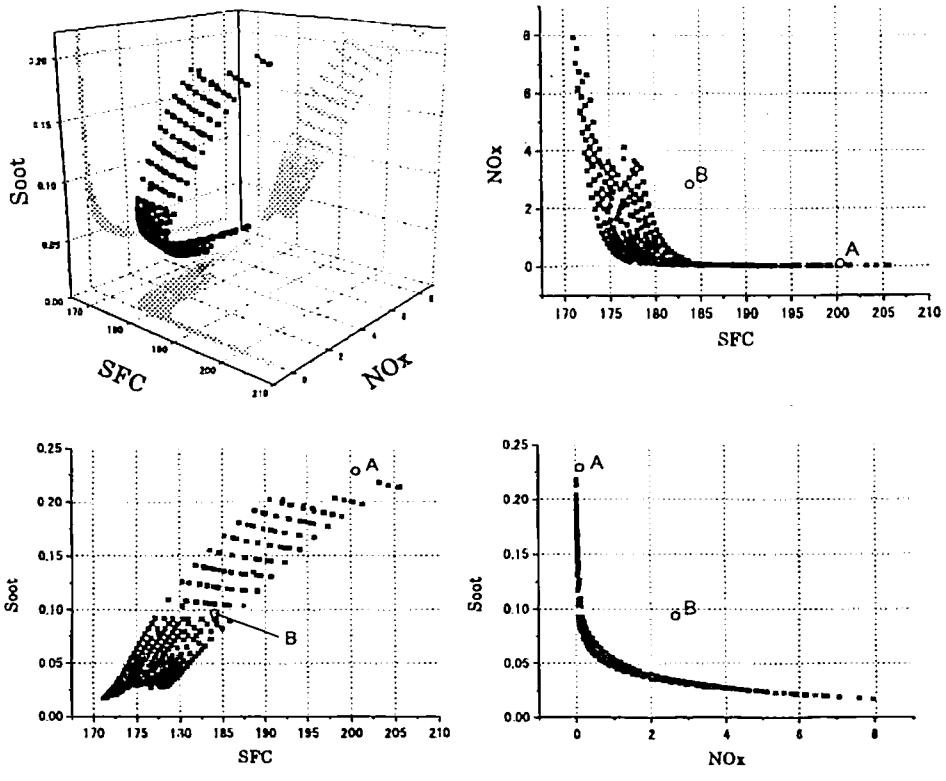

図 13: Pareto Optimum Solutions

ある．このような場合，複数の目的関数に対して重みを設定し，単一目的の問題に変更して，解を求めることは非常に困難であることがこれらの結果からも明らかである．

また，文献[7]では解 A は複数の最適化のステップから解が得られている．それに対して，本手法では 1 度のステップでパレート最適解集合のすべてが得られる．これは計算負荷の低い現象論モデルを利用しているため遺伝的アルゴリズムにおいて十分，評価計算が行えるからである．これらの結果からも現象論モデルが遺伝的アルゴリズムに適していることがわかる．

図 14, 15 には，得られた最適解の燃料噴射形状のうち，SFC および NOx の値をそれぞれ最小化するものを表している．

図 14 から SFC を最小とする形状は，噴射の最初の段階で多くの燃料を費やす形状であることがわかる．一方，NOx 値を最小化する形状は，噴射期間中一定に燃料を噴射している．

このように，多目的最適化を行うことで，複数の解が一度に得ることができ，ディーゼルエンジンの設計者にとって有意義な情報を得ることが可能となる．

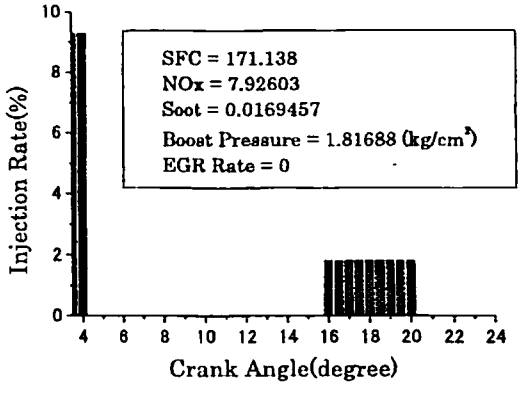

図 14: Injection Shape that gives minimum SFC

6 おわりに

本研究では，大規模計算環境であるグリッドに注目し，Grid RPC システムの 1 つである NetSolve を用いて遺伝的アルゴリズムにおける候補解評価の並列化を行った．計算負荷の低い問題として Rastrigin 関数に適用した結果，計算負荷の低い問題の場合は Farming によるオーバヘッドの割合が大きくなってしまい，並列化が有効に機能しないことがわかった．次に，高い計算負荷を必要とするディーゼルエンジン燃料噴射スケジュール最適化問題に適用

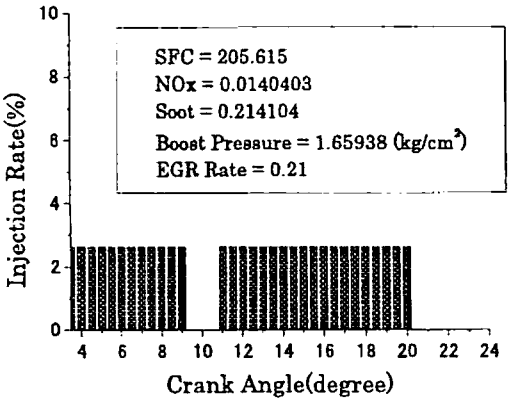

図 15: Injection Shape that gives minimum NOx

した結果，ノード数の増加に伴って実行時間の短縮が確認された．このことより，計算負荷の高い問題については NetSolve による並列化が有効であるといえる．

参考文献

1) Ian Foster and Carl Kesselman, editors. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 1999.

2) Hidemoto Nakada, Satoshi Matsuoka, Keith Seymour, Jack Dongarra, Craig Lee, and Henri Casanova. GridRPC: A Remote Procedure Call API for Grid Computing. In *Global Grid Forum 5*, Edinburgh, Scotland, July 2002.

3) Henri Casanova and Jack Dongarra. NetSolve: A Network Enabled Server for Solving Computational Science Problems. *The International Journal of Supercomputer Applications and High Performance Computing*, Vol. 11, No. 3, pp. 212–223, Fall 1997.

4) David E. Goldberg. *Genetic Algorithms in Search Optimization and Machine Learning*. Addison-Wesley, 1989.

5) Erick Cantú-Paz. A survey of parallel genetic algorithms. *Calculateurs Paralleles*, Vol. 10, No. 2, 1998.

6) Hiroyuki Hiroyasu and Toshikazu Kadota. Models for Combustion and Formation of Nitric Oxide and Soot in DI Diesel Engines. *SAE No.760129*, Vol. 85, pp. 513–526, 1976.

7) D. T. Montgomery and Rolf D. Reitz. Optimization of Heavy-Duty Diesel Engine Operating Parameters Using A Response Surface Method. SAE Paper 2000-01-1962, 2000.

8) Tomoyuki Hiroyasu, Mitsunori Miki, Jiro Kamiura, Sinya Watanabe and Hiroyuki Hiroyasu. Multi-Objective Optimization of Diesel Engine Emissions and FuelEconomy Using Genetic Algorithms and Phenomenological Model. SAE paper 2002-01-2778, 2002.

9) Sinya Watanabe, Tomoyuki Hiroyasu and Mitsunori Miki. Multi-Objective Rectangular Packing Problem and Its Applications. Proceedings of Second International Conference on Evolutionary Multi-Criterion Optimization(EMO' 03), pp. 565–577, 2003.