研究報告

粒子群最適化法のための Particle 言語の開発

Development of Particle Language for Particle Swarm Optimization Method

森山 賀文　Yoshifumi MORIYAMA
有明工業高等専門学校 電気工学科
Department of Electrical Engineering, Ariake National College of Technology

飯村 伊智郎　Ichiro IIMURA
熊本県立大学 総合管理学部 総合管理学科
Department of Administration, Faculty of Administration, Prefectural University of Kumamoto

中山　茂　Shigeru NAKAYAMA
鹿児島大学 工学部 情報工学科
Department of Information and Computer Science, Faculty of Engineering, Kagoshima University

要　旨

最適化問題の近似解法の一つである粒子群最適化（Particle Swarm Optimization: PSO）は，単峰性の関数最適化問題において遺伝的アルゴリズムよりも効率的に解を探索できることが知られている。しかしながら，PSO を実装する場合，粒子の群知能に関する知識が必要であり，またそれらをシミュレートするために煩雑なコーディング作業をプログラム開発者が行う必要がある。そこで本論文では，そのコーディングの煩雑さを軽減し，一般ユーザでも PSO を容易に適用できる環境構築を目的として，PSO のための Particle 言語を提案する。提案する Particle 言語を関数最適化問題に適用した結果，プログラムのステップ数とファイルサイズの大幅な削減ができ，コーディングの煩雑さを軽減することができた。

Abstract

Particle swarm optimization (PSO) that is used to calculate quasi-optimal solution of optimization problems can obtain the solution from unimodal function optimization problems more effectively than genetic algorithm. However, knowledge concerning swarm intelligence of particles are indispensable for programmers, therefore complex coding work is demanded to programmers. In this paper, we propose "Particle language" in order to reduce the complex coding work and to construct an environment in which general users, who are not specialists, can easily apply the PSO algorithm to function optimization problems. The result of the experiment in which the proposed Particle language is applied to function optimization problems clarifies that the use of the Particle language reduces complex coding work and greatly decreases program steps and file size.

1. はじめに

ある制約条件のもとで目的関数を最大化または最小化にする最適解を求める最適化問題（Optimization Problems）の近似解法として，生物の進化を模倣した遺伝的アルゴリズム（Genetic Algorithm: GA）[1]や生体の獲得免疫の仕組みを模倣した免疫アルゴリズム（Immune Algorithm: IA）[2]，アリの採餌行動を模倣したアントコロニー最適化（Ant Colony Optimization: ACO）[3]などが提案されている。これらの手法には対象とする問題によって得意・不得意があることが知られている。GA は大局的な一つの最適解を探索する能力に優れており，IA は大局的な一つ以上の最適解を探索する能力を有している。また，ACO は，巡回セールスマン問題（Traveling Salesman Problem: TSP）において，GA と比較して，より優れた探索能力を有していることが分かる。

筆者らはこれまでに，コーディングの煩雑さを軽減し一般ユーザでも容易に利用できる環境構築を目的として，GA と IA，ACO にそれぞれ特化した，Gene 言語[4]と Immune 言語[5]，Ant 言語[6]の開発を行ってきたが，今回は，粒子群最適化（Particle Swarm Optimization: PSO）[7]を取り上げる。GA，IA，ACO，そして今回対象とした PSO は，すべて進化論的計算手法に分類されるものの，アルゴリズムはまったく異なり，また一般ユーザが使用できることを考えているために，それぞれ個別の対応が必要となる。

PSO は，鳥や魚など（粒子）が群を成して移動する集団行動をコンピュータ上でシミュレートしたものである。各粒子の動きは単純であるが，それらの粒子が群を成すと非常に複雑な動きをする。この PSO は，単峰性の関数最適化問題において，単峰性の最適化を得意とする GA よりも優れた解探索能力を有していることが示されている[8]。しかしながら，この PSO のプログラムを実装する場合，粒子群の集団行動に基づいた群知能（Swarm Intelligence）に関する知識や煩雑なコーディング作業が必要となる。コーディング作業の負担を軽減する策としては，PSO 用のクラスや PSO 専用の DLL（Dynamic

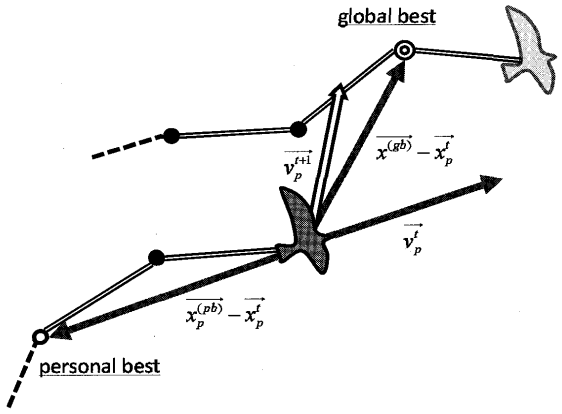


図 1　探索点の移動
Fig. 1　Updating position of a particle.

Link Library）などの開発が考えられる。しかし，本研究では，PSO に対する一般ユーザのハードルを下げ利便性を向上するという観点から PSO 用のクラスや DLL などでの実装は避け，PSO 専用の言語として開発を行った。

そこで本論文では，Java 言語に関する基本的な知識さえ持っていれば，一般ユーザでも PSO を容易に利用できる環境構築を目的として，PSO のための Particle 言語を提案する。また，Particle 言語には，PSO による解探索だけでなく，Graphical User Interface（GUI）を用いた解探索過程の描画機能や，PSO のアルゴリズムが繰返される度に探索結果を保存する機能が組込まれているため，PSO による探索過程や結果を容易に把握することができる。

情報文化においては，言語開発の問題は，情報文化空間を構成する理念系に位置付けられる。これまで筆者らは，この問題を，前述した Gene 言語や Immune 言語，Ant 言語の開発の立場から行ってきたが，本研究は群知能の一つである PSO のための言語開発の立場から行うものである。また，PSO は鳥や魚などが群を成して行動するパターンをモデル化した群知能の一つであり，情報文化社会における社会知能とも深く関係すると思われる。例えば，社会知能として，人は真似をして行動することがあり，これも単純な群知能の一つである。

以下，まず第 2 節では PSO について概説し，次に第 3 節では，本論文で提案する Particle 言語の仕様と，コーディング事例について述べる。その後，第 4 節では，実験内容について述べ，その実験の結果について考察する。

2.　粒子群最適化法の概要

2.1　概要

PSO は，J.Kennedy[7] らによって提案された，鳥や魚の群が移動する行動パターンを模倣した最適化問題の近似解法である。

PSO では，例えば鳥の群の場合，1 羽の鳥のことを粒子（particle）と呼び，複数羽の鳥の集合を群（swarm）と呼ぶ。各粒子は，ある時刻 t における自身の位置情報と速度（進行方向と速さ）を把握している。また，自身がこれまでに移動した探索点の中で最良の評価が得られた解（personal best）の位置を記憶しており，さらに群を形成する粒子全体の中での過去最良の評価が得られた解（global best）の位置情報を群全体で共有している。次の時刻（t +1）における各粒子の進行方向は，時刻 t での速度，personal best の位置，global best の位置によって決定される。

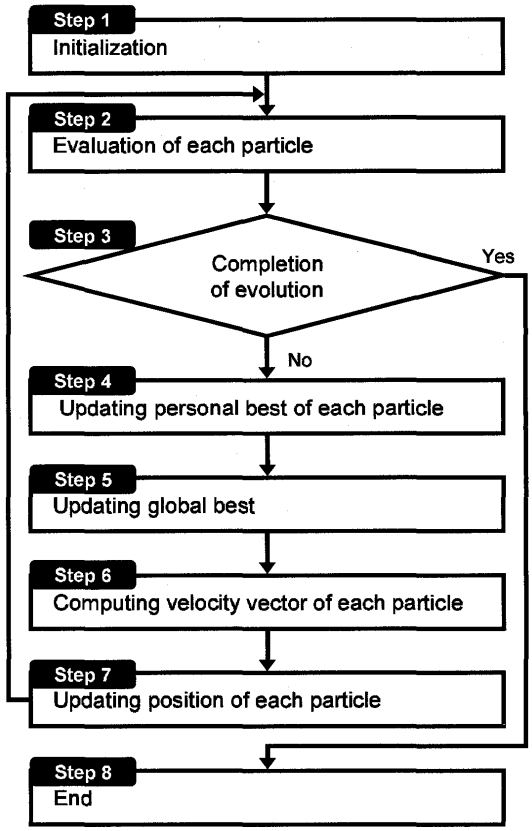


図 2　PSO の処理の流れ
Fig. 2　Processing procedure of PSO.

図 1 はこれらの処理をベクトルで表現したものである。ただし，●は各粒子の過去の探索点，○は各粒子の personal best の位置，◎は群の中での global best の位置である。また $\overrightarrow{x_p^t}$ と $\overrightarrow{v_p^t}$ はそれぞれ粒子 p の時刻 t における位置ベクトルと速度ベクトルを表し，$\overrightarrow{x_p^{(pb)}}$ は粒子 p の personal best の位置ベクトル，$\overrightarrow{x^{(gb)}}$ は群の中での global best の位置ベクトルを表す。

2.2　処理手順

PSO の処理手順を図 2 に示し，各処理の詳細を以下に説明する。

Step 1〔初期化〕　初期（$t = 0$）の位置ベクトル $\overrightarrow{x_p^0}$ と速度ベクトル $\overrightarrow{v_p^0}$ をランダムに決定した粒子 p を N_p 個生成し，群を形成する。

Step 2〔評価〕　各粒子 p の評価を行う。PSO を関数最適化問題に適用する場合，各粒子 p の位置ベクトル $\overrightarrow{x_p^t}$ が指し示す座標 x_p^t を用いて評価する。最大値探索の場合はこの評価値が大きいほど評価の高い粒子であるものとし，最小値探索の場合はこの評価値が小さいほど評価の高い粒子であるものとする。

Step 3〔終了判定〕 予め指定された進化の終了条件を満たしていればStep 8〔終了〕へ，そうでなければStep 4〔personal best の更新〕へ進む。

Step 4〔personal best の更新〕 群を成すそれぞれの粒子の personal best を更新する。粒子 p がこれまでの探索で得た最良解である personal best の位置座標 $x_p^{(pb)}$ よりも，現在の位置座標 x_p^t の方が評価が高かった場合，personal best の位置座標 $x_p^{(pb)}$ を位置座標 x_p^t で更新する。

Step 5〔global best の更新〕 群を成す粒子全体での global best を更新する。これまでの探索で得られた最良解である global best よりも，さらに評価の高い粒子 p' が得られた場合，global best の位置座標 $x_p^{(gb)}$ をその粒子 p' の位置座標 $x_{p'}^t$ で更新する。

Step 6〔速度ベクトルの計算〕 各粒子の速度ベクトルを計算する。粒子 p の次の時刻（t+1）の速度ベクトル $\overrightarrow{v_p^{t+1}}$ は，以下の式で求めることができる。

$$\overrightarrow{v_p^{t+1}} = \chi(\omega \cdot \overrightarrow{v_p^t} + \phi_1(\overrightarrow{x_p^{pb}} - \overrightarrow{x_p^t}) + \phi_2(\overrightarrow{x^{gb}} - \overrightarrow{x_p^t})) \quad (1)$$

ただし，収束係数 χ は 0.9 以上 1.0 以下の乱数，減衰係数 ω は任意の定数，係数 ϕ_1，ϕ_2 は 0.0 以上 2.0 以下の乱数である。また，粒子 p の速さ $\left|\overrightarrow{v_p^{t+1}}\right|$ が最大速 v_{max} を超えた場合は，$\left|\overrightarrow{v_p^{t+1}}\right| = v_{max}$ となるよう正規化を行う。

Step 7〔探索点更新〕 Step 6〔速度ベクトルの計算〕で求めた速度ベクトルを用いて，各粒子の位置座標を更新する。粒子 p の位置ベクトルは，以下の式で更新される。

$$\overrightarrow{x_p^{t+1}} = \overrightarrow{x_p^t} + \overrightarrow{v_p^{t+1}} \quad (2)$$

Step 8〔終了〕 予め指定された条件を満たしたため，探索を終了する。

3. 提案する Particle 言語の仕様およびそのコーディング事例

Particle 言語は，Java 言語をベースに，PSO 特有の処理に関するキーワードを追加したものである。以下，提案する Particle 言語の仕様について述べ，その後 Particle 言語による開発の流れについて述べる。

3.1 文法

ユーザが Particle 言語を用いて記述すべきプログラムは「PSO 処理開始メソッドの呼び出し」，「評価関数定義メソッドの定義」，「**particle** 構文による必要情報の指定」のみである。それぞれの文法を，本論文で実験対象とした関数最適化問題を例に取り上げて説明する。

PSO 処理開始メソッドの呼び出し　PSO の処理を開始するためのメソッドを呼び出す。ここではメソッドの呼び出しのみを行い，メソッド自体の定義は Particle 言語コンパイラが行う。

開始メソッドでは，引数として，探索を行う粒子数，1 回の実行で PSO の処理が適用される繰返し回数，そして探索結果を保存するためのファイル名を指定する。アルゴリズムの繰返し回数とファイル名は省略することができ，繰返し回数が省略された場合は 1 回の実行でアルゴリズムを 1 度だけ実行することなり，ファイル名が省略された場合は探索結果は保存されない。ただし，粒子数とアルゴリズムの繰返し回数は，プログラム実行時に表示される GUI により後で変更可能である。

粒子数を **nParticles**，1 回の実行におけるアルゴリズムの繰返し回数を **nSteps**，探索結果を保存するためのファイル名を **fName** とし，開始メソッド名を **start** とした場合のオーバーロード記述例を以下に示す。

```
// 記述例 1
start(nParticles, nSteps, fName);
// 記述例 2
start(nParticles, nSteps);
// 記述例 3
start(nParticles, fName);
// 記述例 4
start(nParticles);
```

評価関数定義メソッドの定義　1 変数の関数を評価するメソッドを定義する。最適化を行いたい関数はユーザによって異なるため，ユーザが問題に応じて定義する必要がある。ただし，この関数評価メソッドの引数には探索点の座標を用い，戻り値にはその探索点での関数の評価結果を用いる。評価すべき関数のメソッド名を **function**，引数となる変数を **x** とし，座標 **x** での評価値を戻り値として返すメソッドの記述例を以下に示す。

```
// 記述例　関数 F1
//f(x) = sin(x)
public double function(double x) {
  return Math.sin(x);
}
// 記述例　関数 F2
//f(x) = e^{-4·log2((x-0.1)/0.8)} · sin^6(5πx)
public double function(double x) {
  return Math.exp((-4 * Math.log((x
      - 0.1) / 0.8)) / Math.log(2))
      * Math.pow(Math.sin(5
      * Math.PI * x), 6);
}
```

```
// 記述例　関数 F3
//f(x) = e^{-4·log2((x-0.1)/0.8)} · sin^6(5πx^{3/4})
public double function(double x) {
  return Math.exp((-4
      * Math.log((x - 0.1) / 0.8))
      / Math.log(2)) * Math.pow(
      Math.sin(5 * Math.PI
      * Math.pow(x, 3.0 / 4)), 6);
}
```

particle 構文による必要情報の指定　particle 構文では，これまで定義してきたメソッドや変数に関する情報を指定する。Particle 言語コンパイラは，この情報をもとに PSO 処理に必要なメソッド等を補完して Java 言語プログラムを自動生成する。

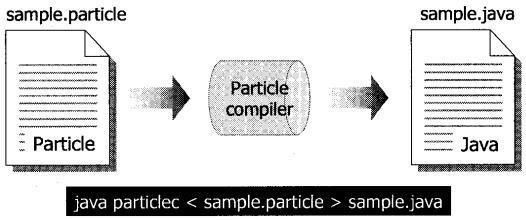
「PSO 処理開始メソッドの呼び出し」で述べた PSO の処理を開始するために呼び出すメソッド名を **start**，「評価関数定義メソッドの定義」で述べた 1 探索点の評価を行うメソッド名を **function** としたの場合の記述例を，以下に示す。

```
// 記述例
particle start(function);
```

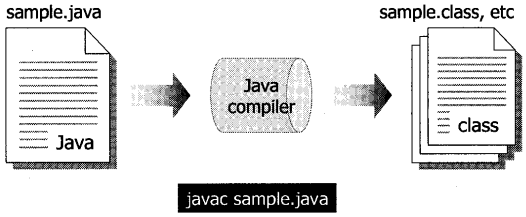
3.2 Particle 言語コンパイラと Particle 言語によるプログラム開発の流れ

本研究では，UCLA の Computer Science Department による “The Cooperative Database Project Web Page” で公開されていた “java1_4c.jj” をベースとして，Particle 言語用コンパイラ “particlec” を開発した。この Particle 言語用コンパイラを用いることで，Particle 言語で記述されたプログラムを Java 言語のプログラムに変換することができる。また，Particle 言語用コンパイラは Particle 言語で必要な字句解析や構文解析だけでなく，Java 言語のすべての文法（JDK1.4 仕様）に則した構文解析も可能である。

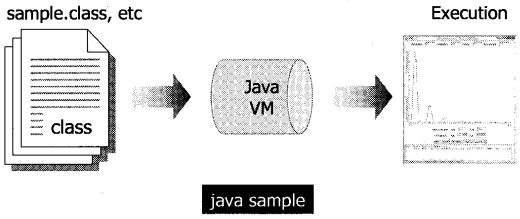
図 3 に Particle 言語による開発の流れを示す。第 3.1 項で述べた Particle 言語の文法に基づいて記述されたプログラムは，Particle 言語用コンパイラを通すことで構文解析が行われ，文法的にミスがなければ PSO 処理に必要なメソッド等が補完された Java 言語プログラムに変換される。その際，**particle** 構文で指定された情報を用い，PSO 特有の処理に関するメソッドと，GUI による視覚的な結果表示を行うためのメソッドが追加される。その後，変換された Java 言語プログラムは Java 言語コンパイラを通すことで Java バイトコードに変換され，Java バイトコードが Java 仮想マシン上で実行されることになる。



(a) Converting Particle file to Java file by using Particle compiler “particlec”.



(b) Converting Java file to class files by using Java compiler “javac”.



(c) Executing class files on Java VM.

図 3　Particle 言語による開発と実行の流れ
Fig. 3　Procedure of developement and execution in Particle language.

4. 実験

4.1 実験内容

本節では，Particle 言語で記述されたプログラムと，そのプログラムを Particle 言語コンパイラで Java 言語に変換したプログラムとを比較することにより，Particle 言語による開発効率を考察する。また，プログラムのステップ数は，コーディングする際の書式によって多少増減することが考えられるため，併せてプログラムのファイルサイズによる比較も行うものとする。

さらに，被験者 17 名を対象として，Particle 言語を用いて関数最適化問題を解くプログラムを実際に作成してもらう実験を行い，その後，Java 言語の理解度や PSO に関する知識，Particle 言語の利便性に関する 5 段階評価と自由記述によるアンケート調査を行った。このアンケート結果をもとに，PSO に関する知識の有無と Particle 言語を用いた PSO の取扱いの関係について考察する。

参考までに，第 3.1 項の「評価関数定義メソッドの定義」で述べた関数 F1，F2，F3 に対する Particle 言語プログラムの実行画面例をそれぞれ図 4，図 5，図 6 に示し，関数 F3 を対象とした Particle 言語で記述されたプログラムを付録の A.1 節に示す。プログラムの実行画面上では，上部のテキストフィールドを用いて粒子数（Population），探索世代数（Generation），減衰係数（omega），最大速（Max velocity）の変更が可能である。また，ユーザが定義した探索すべき評価関数を簡略化したグラフとその探索結果が中央部に描画され，グラフの横軸と縦軸の表示範囲の変更も可能である。このとき，表示範囲の初期値は自動で計算され，またユーザが定義した関数は下部に表示されるため実行後に確認することができる。

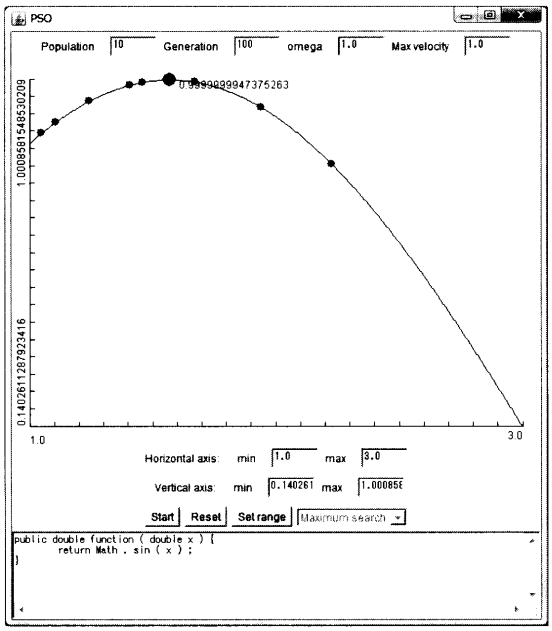


図 4　Particle 言語プログラムの実行画面例（関数 F1）
Fig. 4　An executing example of an Particle language program (function F1).

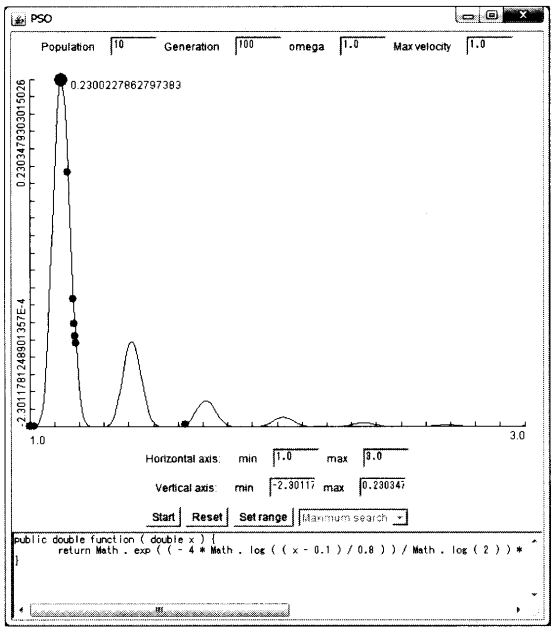


図 6　Particle 言語プログラムの実行画面例（関数 F3）
Fig. 6　An executing example of an Particle language program (function F3).

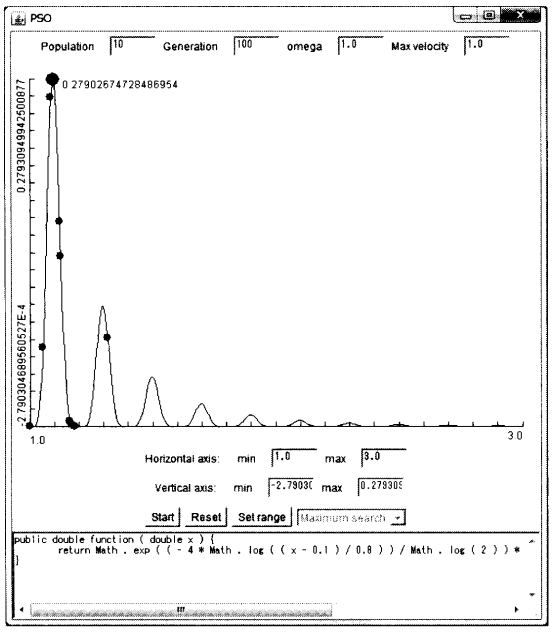


図 5　Particle 言語プログラムの実行画面例（関数 F2）
Fig. 5　An executing example of an Particle language program (function F2).

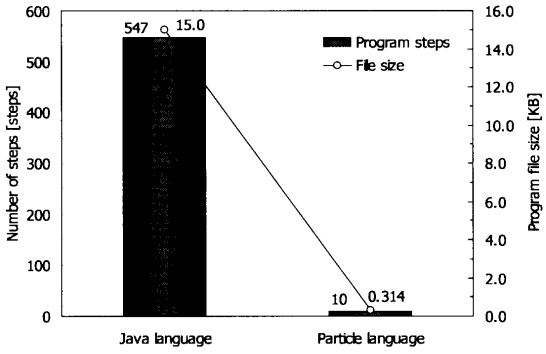


図 7　Java 言語と Particle 言語におけるプログラムステップ数とファイルサイズの比較
Fig. 7　Comparison of both the program step numbers and file sizes in "Java language" and "Particle language".

4.2　実験結果と考察

第 3.1 項の「評価関数定義メソッドの定義」で示した関数 F1, F2, F3 の中で最も複雑な関数 F3 について，Particle 言語で記述されたプログラムと Java 言語で記述されたプログラムとを，ステップ数とファイルサイズで比較した結果を，図 7 に示す。

まずステップ数で比較すると，Particle 言語での記述に要した 10 ステップは，Java 言語での記述に要した 547 ステップの約 1/55 倍であることが分かる。次にプログラムのファイルサイズについて比較すると，Particle 言語での記述に要した 0.314KB は，Java 言語での記述に要した 15.0KB の約 1/48 倍であることが分かる。これらのステップ数とファイルサイズはユーザが対象とする問題によって異なるが，さらに複雑な問題を解く場合であっても，評価関数定義メソッドのみを変更するだけで，PSO の処理と探索結果の描画に関するソースが自動的に付加されるため，PSO プログラムのコーディングに必要な労力の削減が可能であると考えられる。

以上の結果から，Particle 言語を用いることでプログラム開発者によるコーディングのステップ数およびファイルサイズを大幅に削減でき，PSO プログラムの開発効率を向上できると考えられる。また，"particlec" で変換されたプログラムは Java 言語で記述されているため，Java 言語の知識さえあればプログラムの変更も可能であり，容易に PSO を取り扱うことができる。Particle 言語を用いることで，PSO による組合せ最適化問題の解法に対するハードルを下げ，より多くの一般ユーザに対して PSO 活用の機会を拡大させる効果も期待できる。

4.3　アンケートによる主観評価

Java 言語の基本文法の学習経験がある社会科学系学部生 17 名を被験者として，アンケートによる主観評価を行った。本実

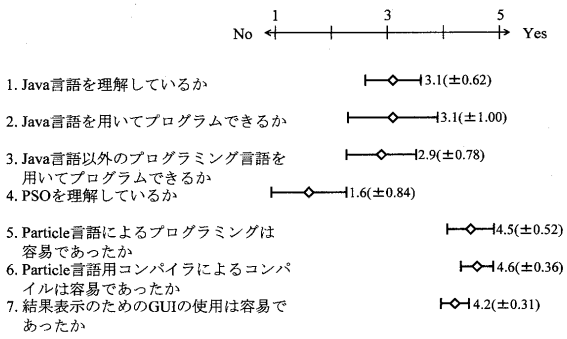


図 8 アンケート結果
Fig. 8 Results of questionnaries.

験では，まず被験者に対して筆者らが作成したマニュアルを用いて PSO の概要および Particle 言語の使用方法について説明し，その後，Particle 言語を用いたコーディングからプログラムの実行までの作業を被験者にしてもらうものとした。また，「Particle 言語用コンパイラの作成」，「Particle ファイルから Java ファイルへのコンパイル」，「PSO のプログラムファイルの実行」の処理を一括して実行するバッチファイルを用意し，バッチファイルを使用する場合と，バッチファイルを使用せず各処理を一つずつ実行する場合の実験を行った。アンケート調査では，Java 言語の理解度や PSO に関する知識，Particle 言語の利便性について 5 段階で評価を行った。そのアンケート結果を図 8 に示す。図中のエラーバーは 95% の信頼区間を表す。

図 8 より，Java 言語をある程度理解していれば PSO に関する知識がなくとも，Particle 言語を用いることで PSO を容易に扱えることが分かる。

また，自由記述を集計した結果，「PSO について分からない私でもたった数行で実行できた」，繰返しごとの探索結果を GUI 上で表示することで「global best と現在の粒子の位置が視覚的な変化として分かりやすかった」，「煩雑なコーディングやコンパイルを簡単に行え，自由度も損なっていないので便利だと思った。PSO を扱いたい（研究したい）がプログラムは良く分からないという人への入り口として特に便利と感じた」，バッチファイルを用いることにより「コンパイルもとても容易にできて良かった」などの回答が得られた。一方で，「PSO に極端に特化した類の言語ではないかと思った」，「GUI が少し操作しづらい」，「2 次元関数だけでなく，3 次元関数にも対応できると，PSO が視覚的に捕らえることができると思う」，また Java 言語に不慣れな学生からは「今後 Particle 言語を使うことになったら最初はちょっと分からないかもしれない」という回答も得られた。

以上の結果より，Particle 言語を用いることで PSO に関する知識がなくとも PSO を容易に適用できると考えられる。しかしながら，現在の Particle 言語では，1 変数関数しか最適化できないため 2 変数関数も最適化できるよう拡張し，よりユーザが探索結果を理解しやすい GUI を開発することが必要であると考えられる。また，Java 言語に不慣れなユーザであっても Particle 言語を利用できるよう，Java 言語の基本的な操作法をまとめたマニュアル等を準備する必要があると考えられる。

5. おわりに

本論文では，Java 言語に **particle** キーワードを追加定義した Particle 言語を提案した。Particle 言語を用いることでコーディングの煩雑さを軽減でき，また Java 言語の知識さえあれば PSO を容易に取り扱えることが分かった。

今回開発した Particle 言語は関数最適化問題を解くことに焦点を絞り PSO を実装したため，TSP などの離散値を扱う問題には対応できない。そのため，より多くのユーザのニーズに応じられるよう，離散値を取り扱うことができる Discrete PSO[9] へと拡張することが必要であると考えられる。

参考文献

[1]D.E. Goldberg: Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, 1989.
[2] 森一之，築山誠，福田豊生：免疫アルゴリズムによる多峰性関数最適化，電気学会論文誌 C, Vol. 117, No. 5, pp. 593–598, 1997.
[3]E. Bonabeau, M. Dorigo, and G. Teraulaz: Swarm Intelligence — From Aatural to Aritficial Systems, Oxford University Press, New York, 1999.
[4] 吉永孝二，飯村伊智郎，中山茂：遺伝的アルゴリズムのための Gene 言語の開発，情報文化学会論文誌，Vol. 12, No. 1, pp. 18–25, 2005.
[5] 森山賀文，飯村伊智郎，中山茂：免疫アルゴリズムのための Immune 言語の開発，情報文化学会論文誌，Vol. 13, No. 2, pp. 3–11, 2006.
[6] 森山賀文，飯村伊智郎，中山茂：アントコロニー最適化法のための Ant 言語の開発，情報文化学会論文誌，Vol. 14, No. 2, pp. 6–10, 2007.
[7]J. Kennedy, R. Eberhart: Particle swarm optimization, Proc. of IEEE International Conference on Neural Networks, Vol. 4, pp. 1942–1948, 1995.
[8] 伊庭斉志：進化論的計算手法，オーム社，2005.
[9]J. Kennedy, R. Eberhart: A discrete binary version of the particle swarm optimization algorithm, Proc. of IEEE International Conference on Systems, Man, and Cybernetics, Vol. 5, pp. 4104–4108, 1997.

付録

A.1 Particle 言語プログラム記述例

以下に，第 3.1 項の「評価関数定義メソッドの定義」で述べた関数 F3 を最適化するためのプログラムを，Particle 言語で記述した例を示す。

```
public class PSO {
  public static void main(String[] args) {
    PSO psol = new PSO();
    //PSO 処理開始メソッドの呼び出し
    psol.start(10, 100, "test");
  }
  // 評価関数定義メソッドの定義
  public double function(double x) {
    return Math.exp((-4 * Math.log((x - 0.1) / 0.8))
        / Math.log(2)) * Math.pow(Math.sin(5 * Math.PI
        * Math.pow(x, 3.0 / 4)), 6);
  }
  particle start(function);
}
```