PARTICAL SWARM OPTIMIZATION
METHOD

A project
Submitted to Department of Computer Science, College of Science, and University of Baghdad in partial fulfillment of the Requirements for the degree of B.SC. In Computer Science.

By
Riyam Muhanad Al.Anie
Hadeel Ayad Al.Quraishy

Supervised by
Assis.
Lecturer: Khulood E. Dagher
Date 2010-2011
الأهـمـداء

الحمد والشكر لله تعالى والصلاة والسلام على المصطفى محمد وآل بيته الطيبين الطاهرين وصحابه المنتجبين ...

إلى من علمني لغة الحنان .................................. أمي العزيزة
إلى من ذلل لي مصاعب الحياة ................................ والطيب الطيب
إلى مصدر سعادتي ........................................................ عائلتي الكريمة
إلى جميع استاذتي وتدريسيوا قسم علوم الحاسبات بجامعة بغداد
إلى استاذتي السيدة خلود سكندر داغر المحترمة التي مدتي لي يد العـبود واعطنتي كل النصائح المهمة بكل إخلاص وتفاني ...

اهди هذه الكلمات المتواضعة والبسيطة والقليلة لترفعني شرفًا وحكمة على أن يكون قادرا على تقديم ولو بالشيء البسيط إلى بلدي العظيم العراق.

الله ي람عكم جميعًا.
بسم الله الرحمن الرحيم

الله لا إله إلا هو الحكيم العزيز لا تأخذنه سنة ولا نوم له ما في السماوات وما في الأرض من حاذي يشفع عنده إلا بأذنه يعلم ما بين أيديهم وما خلفهم ولا يحيطون بشيء من علمه إلا بما شاء وسع خبريه السماوات والأرض ولا يؤذه حفظهما و هو العلي العظيم

صدق الله العلي العظيم
Abstract

Chapter One “Introduction”
1.1 Introduction ........................................................................... (1)
1.2 Aim of project .................................................................. (2)
1.3 Project layout .................................................................. (3)

Chapter Two “Theoretical Background”
2.1 Introduction ..................................................................... (4)
2.2 Optimization ................................................................... (4)
2.3 pseudo code .................................................................. (7)
2.4 Advantage and Disadvantage .............................................. (8)
2.5 PSO applications .............................................................. (8)

Chapter Three “PSO”
3.1 Introduction ..................................................................... (9)
3.2 Algorithm ....................................................................... (9)
3.3 Flow chart ..................................................................... (16)

Chapter four “PSO”
4.1 Introduction ..................................................................... (17)
4.2 Mathematical example ...................................................... (17)
4.3 Banana function ............................................................... (22)

Chapter five “conclusions and future work”
5.1 Introduction ..................................................................... (27)
5.2 Conclusions .................................................................... (27)
5.3 future work ................................................................. (27)

Reference
Abstract

This project study Particle Swarm Optimization method and gives the MATLAB code for it. Finally it gives the advantage and the disadvantage of PSO with its practical application.
Chapter One

Introduction

1.1 Introduction

Particle Swarm Optimization (PSO) is a population based stochastic optimization technique developed by Dr. Eberhart and Dr. Kennedy in 1995.

PSO shares many similarities with evolutionary computation techniques such as Genetic Algorithms (GA). The system is initialized with a population of random solutions and searches in PSO, the potential solutions, called particles (same as the GA chromosome). They contain the variable value and are not binary encoded. Each particle moves about the cost surface with velocity.
The thought process behind the algorithm was inspired by the social behavior of animals, such as bird flocking or fish schooling.

PSO has been successfully applied in many areas: function optimization, artificial neural network training, fuzzy system control, and other areas where GA can be applied.

1.2 The aim of the project

This project study PSO and then use it for minimizing the banana function.

\[ F(x) = 100*(x(2) - x(1))^2 + (1-x(1))^2 \]
1.3 Project layout

This project consists of

**Chapter One:** Introduction.

**Chapter Two:** Theoretical background

**Chapter Three:** Particle Swarm Optimization.

**Chapter Four:** Results.

**Chapter Five:** Conclusion and future work.
Chapter Two

Theoretical Background

2.1 Introduction
This chapter introduces the optimization then study advantage, disadvantages and applications of it.

2.2 Optimization
Is the mechanism by which one finds the maximum or minimum value of a function or process economics, and engineering where the goal is to maximize efficiency, production, or some other measure. Optimization can refer to either minimization or maximization; this mechanism is used in fields such as physics, chemistry, maximization of a function f is equivalent to minimization of the opposite of this function, −f [1].

Mathematically, a minimization task is defined as:
Given f: R (n) → R
Find ∀x ∈ R (n) such that f (x̂) ≤ f(x), ∀x ∈ R (n).

Similarly, a maximization task is defined as:
Given f: R (n) → R
Find x̂ ∈ R (n) such that f (x̂) ≤ f(x), ∀x ∈ R (n).

The domain R (n) of f is referred to as the search space (or parameter space [2]). Each element of R (n) is called a candidate solution in the search space, with x̂ being the optimal solution. The value n denotes the number of dimensions of the search space, and thus the number of parameters involved in the optimization problem. The function f is called the objective function, which maps the search space to the function space. Since a function has only one output, this function space is usually one-dimensional.
The function space is then mapped to the one-dimensional fitness space, providing a single fitness value for each set of parameters. This single fitness value determines the optimality of the set of parameters for the desired task. In most cases, including all the cases discussed in this paper, the function space can be directly mapped to the fitness space. However, the distinction between function space and fitness space is important in cases such as multi objective optimization tasks, which include several objective functions drawing input from the same parameter space [2, 3].

For a known (differentiable) function $f$, calculus can fairly easily provide us with the minima and maxima of $f$. However, in real-life optimization tasks, this objective function $f$ is often not directly known. Instead, the objective function is a “black box” to which we apply parameters (the candidate solution) and receive an output value. The result of this evaluation of a candidate solution becomes the solution’s fitness.

The final goal of an optimization task is to find the parameters in the search space that maximize or minimize this fitness [2]. In some optimization tasks, called constrained optimization tasks, the elements in a candidate solution can be subject to certain constraints (such as being greater than or less than zero) [1]. For the purposes of this paper, we will focus on unconstrained optimization tasks. A simple example of function optimization can be seen in Figure 1.

This figure shows a selected region the function $f$, demonstrated as the curve seen in the diagram. This function maps from a one-dimensional parameter space—the set of real numbers $\mathbb{R}$ on the horizontal $x$-axis—to a one-dimensional function space—the set of real numbers $\mathbb{R}$ on the vertical $y$-axis. The $x$-axis represents the candidate solutions, and the $y$-axis represents the results of the objective function when applied to these candidate solutions.

This type of diagram demonstrates what is called the fitness landscape of an optimization problem [2]. The fitness landscape plots the n-dimensional parameter space against the one-dimensional fitness for each of these parameter.
Figure 1 also shows the presence of a local maximum in addition to the marked global maximum. A local maximum is a candidate solution that has a higher value from the objective function than any candidate solution in a particular region of the search space. For example: - if we choose the interval [0, 2.5] in Figure 1, the objective function has a local maximum located at the approximate value $x = 1.05$. Many optimization algorithms are only designed to find the local maximum, ignoring other local maxima and the global maximum. However, the PSO algorithm as described intended to find the global maximum.
2.3 Pseudo Code

For each particle
    Initialize particle
END

Do
    For each particle
        Calculate fitness value
        If the fitness value is better than the best fitness value (pBest) in history
            set current value as the new pBest
        End
    End

Choose the particle with the best fitness value of all the particles as the gBest

For each particle
    Calculate particle velocity according equation (a)
    Update particle position according equation (b)
End

While maximum iterations or minimum error criteria is not attained

Particles' velocities on each dimension are clamped to a maximum velocity Vmax. If the sum of accelerations would cause the velocity on that dimension to exceed Vmax, which is a parameter specified by the user. Then the velocity on that dimension is limited to Vmax.
2.4 The Advantage and the Disadvantage of PSO

The Advantages of PSO

- Insensitive to scaling of design variables.
- Simple implementation.
- Easily parallelized for concurrent processing.
- Derivative free.
- Very few algorithm parameters.
- Very efficient global search algorithm.

The Disadvantages of PSO

- Slow convergence in refined search stage (weak local search ability).

2.5 PSO applications

- Training of neural network.
- Optimization of electric power distribution network.
- Structural optimization.
- Process biochemistry.
- System identification in biomechanics.
- Images processing.
Chapter Three

PSO

3.1 Introduction

This chapter introduces the PSO algorithm, flow chart, Topologies and then studies Variants of it.

3.2 Algorithms

The PSO algorithm works by simultaneously maintaining several candidate solutions in the search space. During each iteration of the algorithm, each candidate solution is evaluated by the objective function being optimized, determining the fitness of that solution. Each candidate solution can be thought of as a particle “flying” through the fitness landscape finding the maximum or minimum of the objective function.

Initially, the PSO algorithm chooses candidate solutions randomly within the search space. Figure 2 shows the initial state of a four-particle PSO algorithm seeking the global maximum in a one-dimensional search space. The search space is composed of all the possible solutions along the x-axis; the curve denotes the objective function. It should be noted that the PSO algorithm has no knowledge of the underlying objective function, and thus has no way of knowing if any of the candidate solutions are near to or far away from a local or global maximum.

The PSO algorithm simply uses the objective function to evaluate its candidate solutions, and operates upon the resultant fitness values.
Each particle maintains its position, composed of the candidate solution and its evaluated fitness, and its velocity. Additionally, it remembers the best fitness value it has achieved thus far during the operation of the algorithm, referred to as the individual best fitness, and the candidate solution that achieved this fitness, referred to as the individual best position or individual best candidate solution.

Finally, the PSO algorithm maintains the best fitness value achieved among all particles in the swarm, called the global best fitness, and the candidate solution that achieved this fitness, called the global best position or global best candidate solution [1].

1. Create a ‘population’ of agents (called particles) uniformly distributed over X.
2. Evaluate each particle’s position according to the objective function.

3. If a particle’s current position is better than its previous best position, update it.

4. Determine the best particle (according to the particle’s previous best positions).
5. Update particles’ velocities according to \( \mathbf{v}_i(t+1) = \mathbf{v}_i(t) + c_1 r_1 (\mathbf{p}_i(t) - \mathbf{x}_i(t)) + c_2 r_2 (\mathbf{p}_g(t) - \mathbf{x}_i(t)). \)

6. Move particles to their new positions according to \( \mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1). \)

7. Go to step 2 until stopping criteria are satisfied.
2. Evaluate each particle’s position according to the objective function.

3. If a particle’s current position is better than its previous best position, update it.
4. Determine the best particle (according to the particle’s previous best positions).

5. Update particles’ velocities according to \( v_i(t+1) = v_i(t) + c_1 r_1(p_i(t) - x_i(t)) + c_2 r_2(p_g(t) - x_i(t)) \).

6. Move particles to their new positions according to \( x_i(t+1) = x_i(t) + v_i(t+1) \).
7. Go to step 2 until stopping criteria are satisfied.
3.3 Flow chart

1. Initialize $x_i(k)$, $y_i(k)$.
2. Compute $f(x_i(k))$.
3. Reorder the particles.
4. Generate neighborhoods.
5. Determine best particle in neighborhood of $i$.
6. If $i \leq N_x$ then go to step 7, else go to step 8.
7. Compute $x_i(k+1)$, $f(x_i(k+1))$.
8. Compute $x_i(k+1)$, $f(x_i(k+1))$.
9. Update previous best if necessary.
10. If $k \leq K$ then go to step 11, else stop.
11. $i = i + 1$.
12. $k = k + 1$.
13. Stop.
4.1 Introduction
This chapter introduces mathematical example (Hand Work) and Banana function.

4.2 Mathematical Example
Maximize the function \( f(X) = 2X - \frac{X^2}{16} \) which defines within \([0, 31]\) interval:-

Let \text{pop size}=4 \hspace{1em} \text{(population size)}

Maxit=2 \hspace{1em} \text{(maximum iteration)}

C1=C2=2

C=1

<table>
<thead>
<tr>
<th>Particles</th>
<th>Velocity</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>18.66</td>
<td>12.69</td>
<td>15.556</td>
</tr>
<tr>
<td>28.574</td>
<td>0.0106</td>
<td>9.01</td>
</tr>
<tr>
<td>30.639</td>
<td>16.7689</td>
<td>2.605</td>
</tr>
<tr>
<td>28.816</td>
<td>6.444</td>
<td>5.732</td>
</tr>
</tbody>
</table>

Hint: - Particles and velocity created randomly within (0, 31).

Global cost \hspace{1em} 15.556

Global particle \hspace{1em} 18.66
NEW SWARM  IT=1

<table>
<thead>
<tr>
<th>R1</th>
<th>R2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2193</td>
<td>0.7485</td>
</tr>
<tr>
<td>0.325</td>
<td>0.5433</td>
</tr>
<tr>
<td>0.095</td>
<td>0.3381</td>
</tr>
<tr>
<td>0.745</td>
<td>0.8323</td>
</tr>
</tbody>
</table>

Hint: - R1 and R2 created randomly.

****W = maxit /it.

****Vel = (W*vel+C1*R1*(local par-par) +C2*R2 (global par-par))

****Par = par + vel

<table>
<thead>
<tr>
<th>particle</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>25.011</td>
<td></td>
</tr>
<tr>
<td>17.983</td>
<td></td>
</tr>
<tr>
<td>30.924</td>
<td></td>
</tr>
<tr>
<td>15.1346</td>
<td></td>
</tr>
</tbody>
</table>

Check par>0 and par<31

<table>
<thead>
<tr>
<th>New cost</th>
<th>Local cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.924</td>
<td>15.55</td>
</tr>
<tr>
<td>15.7541</td>
<td>9.0107</td>
</tr>
<tr>
<td>2.07</td>
<td>2.6050</td>
</tr>
<tr>
<td>15.953</td>
<td>5.7329</td>
</tr>
</tbody>
</table>

Compare it with local cost Then chose better cost.
<table>
<thead>
<tr>
<th>Better cost</th>
<th>Better particle</th>
</tr>
</thead>
<tbody>
<tr>
<td>15.5567</td>
<td>18.66</td>
</tr>
<tr>
<td>15.7541</td>
<td>17.983</td>
</tr>
<tr>
<td>2.6050</td>
<td>30.63</td>
</tr>
<tr>
<td>15.9532</td>
<td>15.134</td>
</tr>
</tbody>
</table>

**NOW**

Global cost 15.953

Global particle 15.1346

Local cost=better cost

Local particle=better particle

<table>
<thead>
<tr>
<th>It</th>
<th>x</th>
<th>f(x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>15.1346</td>
<td>15.953</td>
</tr>
</tbody>
</table>

**NEW SWARM  IT=2**

<table>
<thead>
<tr>
<th>R1</th>
<th>R2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5529</td>
<td>0.5464</td>
</tr>
<tr>
<td>0.9575</td>
<td>0.3967</td>
</tr>
<tr>
<td>0.892</td>
<td>0.6228</td>
</tr>
<tr>
<td>0.356</td>
<td>0.7960</td>
</tr>
</tbody>
</table>
Hint: - R1 and R2 created randomly.

*****W= maxit /it.

*****Vel= (W*vel+C1*R1*(local par-par) +C2*R2 (global par-par))

*****Par=par +vel

<table>
<thead>
<tr>
<th>vel</th>
<th>particle</th>
</tr>
</thead>
<tbody>
<tr>
<td>17.8</td>
<td>7.2024</td>
</tr>
<tr>
<td>1.9</td>
<td>16.00</td>
</tr>
<tr>
<td>20.177</td>
<td>10.7471</td>
</tr>
<tr>
<td>0</td>
<td>15.1346</td>
</tr>
</tbody>
</table>

Check par>0 and par<31

<table>
<thead>
<tr>
<th>New cost</th>
<th>Local cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.16</td>
<td>15.556</td>
</tr>
<tr>
<td>16.00</td>
<td>5.7541</td>
</tr>
<tr>
<td>14.27</td>
<td>2.6</td>
</tr>
<tr>
<td>15.9</td>
<td>15.953</td>
</tr>
</tbody>
</table>

Compare it with local cost
Then chose better cost
<table>
<thead>
<tr>
<th>Better cost</th>
<th>Better particle</th>
</tr>
</thead>
<tbody>
<tr>
<td>15.556</td>
<td>18.663</td>
</tr>
<tr>
<td>16.00</td>
<td>16.00</td>
</tr>
<tr>
<td>14.27</td>
<td>10.7471</td>
</tr>
<tr>
<td>15.953</td>
<td>15.1346</td>
</tr>
</tbody>
</table>

**NOW**

Global cost 16  
Global particle 16  
Local cost=better cost  
Local particle=better particle

<table>
<thead>
<tr>
<th>It</th>
<th>x</th>
<th>f(x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>16</td>
<td>16</td>
</tr>
</tbody>
</table>
4.3 Banana function

% particle swarm optimization pso

Clear
clc
Close all
Pop size=20;   %size of swarm
Npar=2;   %dimension of the problem
Maxit=15;   %maximum number of iteration
c1=1;   %cognitive parameter
c2=1;   %social parameter

%initialization swarm
% par=rand (popsize,npar)
% vel=rand (popsize,npar)

Par=rand (popsize,npar).*((2/9999));   %%%% randn to get +ve and -ve
Vel=rand (popsize,npar).*((2/9999));
Cost=fev (par,npar);
Minc (1) =min (cost);
Mean (1) =mean (cost);
Global min=minc (1);
Local par=par;
Local cost=cost;
[globalcost,indx]=min (cost);
Global par=par (index, :);

[x,y]=mesh grid([-2:0.05:2]);
Zz=100*(y-x.^2).^2+(1-x).^2;
Mesh(x,y,zz)
Hold on

%hsv2=hsv;
%hsv3=[hsv2(11:64,:);hsv2(1:10,:)];
View(10,55);
%color map (hsv3);
Hold on

Iter=0;
While (iter<maxit)
Iter=iter+1;
w=(maxit-iter)/maxit
r1=rand(popsize,npar)
r2=rand(popsize,npar)
vel=(w*vel+c1*r1.*(local par-par)+c2*r2.*(ones(popsize,1)*global par-par));
'New vel';
vel;
'New par';
Par=par+vel;
Par;
Overlimit=par<=2  \%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%rang
Underlimit=par>=-2  \%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%rang
%
%
Par=par.*over limit;
Par=par.*under limit;
'Par under over limit new swarm'
Par;
Cost=fev(par)
Better cost=cost<local cost
Local cost=local cost.*not (better cost) +cost.*better cost
Local par (find (better cost), :) =par (find (better cost), :)
[Temp, t]=min (local cost)
If temp<global cost
   Global par=par (t, :)
   Index=t
   Global cost=temp
End%

'iter global par global cost'
datt=[iter global par global cost]  \%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%for the current iteration

Minc (iter+1) =min (cost);
Global min (iter+1) =global cost;
meanc(iter+1)=mean(cost);

[parfit,parind]=mini(cost);
    Parsol=par(parind,:);
    plot3(parsol(1),parsol(2),parfit,'r+');
    Hold on
    Axis([-2,2,-2,2])
    xlabel(parsol(1))
    ylabel(parsol(2))
    %zlabel(parfit)
    Title([iter,parfit])
    Pause(0.001)
End

Figure (2)

Iters=0: maxit;

Plot(iter,minc,_iters,meanc,'-',iters,globalmin,'*');
Xlabel('generation'); ylabel('cost');
%


Figure 5: Banana function
Chapter Five
Conclusions and Future work

5.1 Introduction

This chapter introduces the conclusions and presents the future work.

5.2 conclusions

1) Particle Swarm Optimization is an extremely simple algorithm that seems to be effective for optimizing a wide range of functions.

2) Social optimization occurs in the time frame of ordinary experience - in fact, it is ordinary experience.

3) Particle swarm optimization seems to lie somewhere between genetic algorithms and evolutionary programming.

4) It is highly dependent on stochastic processes, like evolutionary programming.

5) The adjustment toward pbest and gbest by the particle swarm optimizer is conceptually similar to the crossover operation utilized by genetic algorithms.

6) PSO uses the concept of fitness, as do all evolutionary computation paradigms.

7) Much of the success of particle swarms seems to lie in the agents' tendency to hurtle past their target.

8) The Particle Swarm Optimizer serves both of these fields equally well.

5.3 Future work

Study Genetics algorithm and compare it with Particle Swarm Optimization (PSO).


