Comparison of BAT with PSO for Path Planning Problems

Yogita Gigras, Oshina Vasishth
1 Asst.professor, 2 Student
1 Department of Computer Science and Engineering,
2 ITM University, Gurgaon, India

Abstract - Path planning is a fundamental but an essential task in the field of robotics. The completion and correctness of a problem in planning a path depends upon the accuracy of the algorithms used. One category of algorithms used in Path planning has a great influence from nature and are known as bio-inspired algorithms. These algorithms imitate the nature for discovering the path in an artificial environment. Bio inspired algorithms in path planning has evoked large attention from research community in past two decades which can we seen by the way large number of algorithms are being developed today to improve and increase the efficiency of finding the path. This paper shows the growth in this field by comparing two algorithms of initial and recent phases in which one is PSO which was developed in 1995 while the other is BAT algorithm developed in 2010. These two algorithms are implemented in dynamic environment and the path is evaluated using matlab, while the performance of algorithms are compared or calculated by changing the number of population.

Index Terms - Bat algorithm, Particle Swarm Optimization, Path Planning, Bio-inspired algorithms.

I. INTRODUCTION

Path planning is a mechanism for finding efficient path between two distinct points by overcoming the disturbances caused by obstacles. Path planning has shown an immense growth in past years. There are large number of algorithms which are developed and can be broadly classified and pictorially represented as:

![Division of path planning algorithms](image)

Although all these categories of the algorithms are being found to find a path but the most evolving and attention gainer is the category of bio-inspired algorithms which is further divided in categories of algorithms based on Swarm intelligence(S.I) and non swarm(non S.I) Bio-inspired algorithms. Eg: of S.I based algorithms are: Bee colony optimization[12], Bat algorithm, Cuckoo search[11], Pso[9] etc and non SI based are genetic algorithm[14], Atmosphere cloud model[7], Brain Storm Optimization[13] etc.

Particle Swarm Optimization

Dr. Eberhart and Dr. Kennedy in 1995[9] first time implemented Particle swarm optimization which was based on the swarm behaviour of insects such as ants, a flock of birds etc. this swarm behaviour of insect in nature can be seen at the time of gathering food or searching a new colony(home) for themselves. During this action each member of the swarm contributes by sharing of information and takes a global step together to move. The number of individuals is known as particles and the whole population is come to be known as swarm. So the bigger is the number of particles the bigger is the swarm and represented as:

Swarm= {par1, par2, par3…..parN} where par1, par2, par3,…..parN are the participant in forming a swarm.

In a search space these particles have their own position and velocity represented as follow

Particle position = {pos1, pos2, pos3, ……posN}

Particle velocity = {vel1, vel2, vel3, ……….velN}
Each particle in swarm have its own best position value \( (X_{pbest}) \). The particle which is having the best position which is calculated by using objective function or fitness function is considered to be the global best position \( (X_{gbest}) \).

After getting the two best values, the particle updates its velocity and positions using equation (a) and (b)

\[
\begin{align*}
vel &= vel + c1 \times (Rand1 \times (pbest - pos)) + c2 \times (Rand2 \times (gbest - pos)) \\
pos &= pos + vel
\end{align*}
\]

(a) \hspace{2cm} (b)

\( Rand1 \) and \( Rand2 \) are random numbers between \((0,1)\) and \( c1, c2 \) are learning factors. Usually the values of \( c1 \) and \( c2 \) is taken as 2.

**Bat Algorithm**

Xin-She Yang in 2010[10] developed an algorithm which was emulating the behaviour of bats, so this algorithm came to be known as bat algorithm. Bat algorithm uses one of the most important behavioural characteristic of bat known as echolocation to explore its path. Echolocation is a mechanism in which sound pulses are emitted by bat during his search for food and listen to the echo bouncing back from the obstacles which helps bat in finding obstacle free path. The sound pulses emitted by bats are usually short and loud and are emitted at a rate of 10 to 20 pulses per sec. Bats uses these reflected pulses as a source of information for finding their food.

The bat sound pulses are made from three mathematical factors, they are wavelength, loudness and frequency. As wavelength and frequency are inversely proportional, hence only one factor is used during implementation of the algorithm.

The bat algorithm uses the following steps:

1. **Initialization**:
   - Objective function \( ob(x), x=(x1,x2,…..xd) \).
   - In this step the bat population location(loc), initial velocity(vel), maximum frequency\[F_{max}\] and minimum frequency\[F_{min}\], loudness\[A\] and pulse rate\[P\] are initialized.

2. **Generation of new solution**:
   - a) by adjusting frequency –
     \[
     F(i) = F_{min} + (F_{max} - F_{min}) \times \beta,
     \]
   - b) by updating velocity and location –
     \[
     vel(i) = vel(i) + (x(i) - bestx) \times F(i)
     \]
     \[
     loc (i) = loc(i) + vel(i).
     \]

3. **Local Search**: it is done by choosing a result among the best result
   - And the condition applied is \( (Rand>P) \)

4. **Generating new solution by flying randomly**.
   - \( (Rand<A \ and \ ob(xi)<ob(x*)\) )

**PROBLEM DEFINITION AREA**

Considering a space of 120X120 dimension in which robot will move. The space consist of starting point \((X1,Y1)\) in left bottom corner represented using red square while green diamond represent destination point \((X2,Y2)\) on right upper corner .20 randomly generated obstacles are plotted in the space which will create hindrance for robot while finding a path. The aim is to find a path between starting and destination point by avoiding the obstacles using bat and pso algorithms.

**Figure 2: Problem Configuration**

**II. PATH PLANNING ALGORITHM FOR DETECTING COLLISION FREE PATH**

This algorithm is designed to find an effective path between starting and destination point in an artificial environment after avoiding the obstacles.

**Input**: starting point \((X1,Y1)\), destination point \((X2,Y2)\), artificial environment having randomly generated obstacles.

**Output**: reaching to the destination point without colliding with obstacles
Algorithm:
Robot starts from the initial point and calculate the shortest distance between (X1, Y1) and (X2, Y2) using distance formula:
\[
\text{dist} = \sqrt{(x2-x1)^2 + (y2-y1)^2};
\]  
(1)

Then the direction or the new coordinates are calculated using the following equations:
\[
x1 = x1 + \text{step} \times \cos(\theta);
\]  
(2)
\[
y1 = y1 + \text{step} \times \sin(\theta);
\]  
(3)
the \( \theta \) is obtained as:
\[
\theta = \arctan\left(\frac{y2-y1}{x2-x1}\right).
\]  
(4)

**Condition 1:** if no obstacle is encountered
If the robot encounters no obstacle during its search for the optimal path, it continues to apply the above formulas to calculate distance and directions (coordinates) and reaches to the final destination point.

![Figure 3: Reaching to destination without encountering obstacle](image)

**Condition 2:** if obstacles are encountered
If the robot encountered the obstacle in path while moving toward the destination path it will stop and take the following steps:
Robot moves three steps back by using following equations:
\[
x1 = x1 - 3 \times \cos(\theta);
\]  
(5)
\[
y1 = y1 - 3 \times \sin(\theta);
\]  
(6)
\[
\theta = \arctan\left(\frac{y2-y1}{x2-x1}\right).
\]  
(7)

![Figure 4: Obstacle encountered](image)
Now we can apply the path planning algorithms BAT and PSO to get escape from the obstacles and afterwards compares the performance of the algorithms.

**Applying PSO**

Initial random population is generated by assuming the size of the population as 10 particles. The position of these 10 particles is generated using the equation:

\[
st_{1x}(i) = \cos(par(i)); \\
st_{1y}(i) = \sin(par(i));
\]

now the distance formula is applied which is acting as the objective function so that the particle having the minimum objective function could be selected and PSO equation can be applied.

\[
vx(i) = v(i) + c1*(R1*(pbestx(i) - st1x(i)))+c2*(R2*(gbestx - st1x(i))); \\
vy(i) = v(i) + c1*(R1*(pbesty(i) - st1y(i)))+c2*(R2*(gbesty - st1y(i)));
\]

the updated velocities are then used to update the positions as:

\[
posx(i) = posx(i) + vx(i); \\
posy(i) = posy(i) + vy(i);
\]

hence whenever the obstacle is encountered the PSO is applied and the new coordinates are generated which helps the robot in avoiding the obstacles and finding the obstacle-free path.

**Applying BAT algorithm**

Bat algorithm is recently developed algorithm which is applied as follows:

Generate the initial population of 10 bats. There positions are randomly generated as:

\[
st_{1x}(i) = \cos(bat(i)); \\
st_{1y}(i) = \sin(bat(i));
\]

using the minimum objective function the best bat is selected and then the bat algorithm equations are applied separately on X-coordinate and Y-coordinate.

\[
F(i) = F_{\text{min}} + (F_{\text{min}} - F_{\text{max}}) \times \text{rand};
\]

\[
vx(i) = v(i) + (st_{1x}(i) - \text{bestx}) \times Q(i); \\
vy(i) = v(i) + (st_{1y}(i) - \text{besty}) \times Q(i);
\]

\[
st_{1x}(i) = st_{1x}(i) + vx(i); \\
st_{1y}(i) = st_{1y}(i) + vy(i);
\]

keep on applying bat algorithm whenever the robot encounters the obstacles.

![Figure 5: Our aim is obtained as robot finds the path between final and destination point after avoiding obstacle](image)

© 2015 IJEDR | Volume 3, Issue 2 | ISSN: 2321-9939
III. RESULT AND DISCUSSION

The pso and bat algorithms for path planning are being compared by increasing the number of population against number of iteration being executed and we found that BAT algorithm is much more efficient than PSO. As iteration taken by BAT are lesser than PSO which also shows that time taken by BAT would be lesser than PSO in finding a obstacle free path. A comparison tables between bat and pso

<table>
<thead>
<tr>
<th>Population size</th>
<th>PSO(no.of iteration)</th>
<th>BAT(no.of iteration)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td>11</td>
<td>2</td>
</tr>
<tr>
<td>15</td>
<td>16</td>
<td>2</td>
</tr>
<tr>
<td>20</td>
<td>21</td>
<td>2</td>
</tr>
<tr>
<td>25</td>
<td>26</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 2. simulation results applied on whole path planning using PSO and bat algorithm

<table>
<thead>
<tr>
<th>Population size</th>
<th>PSO(no.of iteration)</th>
<th>Bat(no of ieration)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>859</td>
<td>213</td>
</tr>
<tr>
<td>10</td>
<td>729</td>
<td>442</td>
</tr>
<tr>
<td>15</td>
<td>435</td>
<td>338</td>
</tr>
<tr>
<td>20</td>
<td>630</td>
<td>242</td>
</tr>
<tr>
<td>25</td>
<td>759</td>
<td>204</td>
</tr>
</tbody>
</table>

IV. CONCLUSIONS

Both the techniques used in this paper are applied for path planning though both of them have wide scope of usage like Pso can be used to solve problems related to engineering field because of its exceptional features like proximity, quality, stability and adaptability while Bat algorithm can be applied in the field of optimization, image processing, feature selection, scheduling, and others.

REFERENCES

