Outline:

1. What is PSO
2. Formulation
3. Examples
4. Further Reading / Other types of PSOs

**Introduction**

The need to solve for the global minimum of a multidimensional function frequently arises in mathematics and engineering. There are many different ways to solve for a minimum of a function including gradient-based solvers that take small steps in the direction of the negative gradient and symbolic solvers that solve for extrema to find when the gradient is zero. For some applications the following ways are adequate to solve for the global minimum of a function, however, both of the following methods have shortcomings. If a function is complicated enough it may have many local minimums and would cause the gradient solvers to solve for a local minimum rather than a global minimum. Furthermore, if a function is complicated, it may be impossible to set its derivative equal to zero and solve for the variable and therefore impossible to find the global minimum.

**Formulation**

This is where particle swarm optimization (PSO) can be partially useful. Particle swarm optimization uses intelligent swarms of particles, also known as agents, that uses an iterative algorithm that finds the global minimum of any function. Every particle has a position [xi], velocity [vi], and a personal best [pi]. The swarm has a single position known as the global best [pg] that has the “best” or lowest value out of all the particles and is computed with every iteration. The velocity update from one iteration to the next is based on three factors, its current velocity, its personal best, and the global best. This is best shown by the velocity update equation below where [k] is the current time step.

[eq 1]

The [w] term in the above equation, also known as the inertia term, is a constant that determines how much of an effect the current velocity has on the next velocity. The [c1] term, also known as the cognitive term, is the effect its personal best has on the next velocity. Finally, the [c2] term, also known as the social term, is the effect the swarms best have on the next velocity. The r1 and r2 terms are random variables from zero to one and change for every particle and iteration. Below is a visual understanding of the position and velocity update.

[eq 2]

From the velocity, the position of the next particle can be known from the following equation.

The next step would be to find some values for [w] [c1] and [c2]. One way would be to give them some arbitrary values, however, some values may lead to the particle swarm optimizer being unstable. This is the problem Clerc and Kennedy solved in their paper titled “The Particle Swarm—Explosion, Stability, and Convergence in a Multidimensional Complex Space”[1]. In this paper, they came up with a way to generate values for [w] [c1] and [c2] that guarantees stability. To do this let,

[eq 3]

Where \phi > 4 and let,

[eq 4]

Where 0 > \kappa > 1

Then,

[eq 5-7]

As long as \phi > 4 and 0 > \kappa > 1 the PSO is guaranteed to be stable. The variable kappa can be thought of as how much the PSO searches a space. The higher kappa the more it searches but the slower it converges. If a search space is highly nonlinear then a kappa closer to 1 is recommended. Phi 1 is similar to c1 in that it is how much the social component affects the particles. Phi 2 is similar to c2 in that it is how much the cognitive component affects the particles. Once all the constants are initialized the algorithm follows the flow chart shown below

[photo]

One term that is not explained in the flow chart is the [w] update. As a reminder [w] is the inertia term and is how much the current velocity affects the next velocity. The reason to update [w] can be shown experimentally below in the two charts shown below.

[graph 1-2]

(Comparison of the addition of the w damp term with 10-dimensional normal function with 100 particles and 200 iterations. The best cost of the no w-damp is 3.35\*10^-6 and the best cost with a w-damp set to 0.985 is 2.81\*10^-32)

As shown above the PSO with w-damp is approximately 26 orders of magnitude better than with no w-damp in this example. One intuitive way of thinking of the reduction of [w] is that it removes some of the “kinetic energy” of the particle. As the algorithm iterates, it becomes less relent on its own velocity and more on its personal and global best. This causes the average velocity of the swarm to decrease and converge to a solution.

**Examples**

Example 1:

As an example, let the function we want to minimize be the normal function defined below.

[norm(x)]

This is a good function to test the PSO due to its simplicity and the minimum of this function is trivial; x = 0. Furthermore, the dimension of the function can be changed to see how the number of dimensions affects convergence. The results of the PSO show an exponential decrease in error as shown in the figure below.

10-dimensional minimization of normal function

[ex 1 photos]

Example 2:

Another benchmark of the PSO is Shaffer’s f6 function defined below.

[Shaffer’s f6]

The global minimum of this function is located at (0,0) however it may be difficult to know that numerically. The PSO, with 100 particles and 100 iterations, finds the solution so close to the exact solution that it reaches MATLAB floating point precision and returns an error of 0 after 60 iterations.

[ex 2 photos]

Example 3:

In this final example, a more applied problem is conducted. Imagine

**Conclusion**

This technical memo has only scratched the surface of particle swarm optimization. There are many variations of this algorithm that extend its capability. For example, there are gradient-based PSOs that use the gradient as extra information and PSOs that add an acceleration term to the particles. These methods can be added to existing optimization algorithms to create a hybrid optimizer. Furthermore, additional efforts can be made to make the algorithm more computationally efficient by writing it in c or parallelizing it on GPUs.

**References**

<https://www.analyticsvidhya.com/blog/2021/10/an-introduction-to-particle-swarm-optimization-algorithm/>

<https://youtu.be/ICBYrKsFPqA>

//

The reason why particle swarms are so effective is that all the particles can communicate with each other.

When the minimum of a multidimensional can not be solved using traditionally then

For example one of the calculus

Many times in mathematics and engineering the need to solve for the minimum function

\begin{equation}

v\_{k+1}^i = wv\_{k}^i + c\_1r\_1(p\_k^i - x\_k^i) + c\_2r\_2(p\_k^g - x\_k^i)

\end{equation}

\begin{equation}

x\_{k+1}^i = x\_k^i + v\_{k+1}^i

\end{equation}

\begin{equation}

\phi = \phi\_1 + \phi\_2 \quad \phi\geq4

\end{equation}

\begin{equation}

\chi = \frac{2\kappa}{|2-\phi-\sqrt{\phi^2-4\phi}|} \quad 0\geq\kappa\geq1

\end{equation}

\begin{equation}

w = \chi

\end{equation}

\begin{equation}

c\_1 = \chi\phi\_1

\end{equation}

\begin{equation}

c\_1 = \chi\phi\_2

\end{equation}

\begin{equation}

f(x\_k) = \sqrt{\sum\limits\_{i=1}^k x\_i^2}

\end{equation}

\begin{equation}

f(x,y) = 0.5 + \frac{sin^2(\sqrt{x^2+y^2})-0.5}{(1 + 0.001 \cdot (x^2 + y^2))^2}

\end{equation}