Particle Swarm Optimization for Engineering Applications using Python and PYSWARM

Nathan Brakband, Mechanical Engineering

Mentor: Dr. Todd D. Coburn

Kellogg Honors College Capstone Project

Abstract: Particle Swarm Optimization (PSO) has been developed as a method for solving optimization problems where mathematical methods are difficult and direct numerical methods are computationally intensive. PSO mimics the behavior of animal swarms in nature by searching a function randomly and utilizing comparison and movement of "particles" to find an optimum or minimum point. This allows a solution to be found in fewer computations while accommodating various problems and constraints. In particular, this method and its benefits are of interest to the field of engineering for optimization of design. The difficulty lies in making it accessible to those of varying levels of programming competence and those who may not have access to mathematical software packages. Python has been chosen as a language which is simple in syntax and freely available. Using PYSWARM [2] as a foundation, several modifications were made to create a PSO function that can receive entry of any problem. These modifications are highlighted and the results of several test cases are shown.

What is PSO?

- Method of solving minimization/optimization problems by mimicking the behavior of swarms in nature [3]
- Randomly initializes swarm of "particles" with "velocities" that are test solutions to the problem
- Evaluates which particles are best and modifies velocity towards them [1] (See Fig. 3)
- Evaluates constraints to ensure solution is feasible

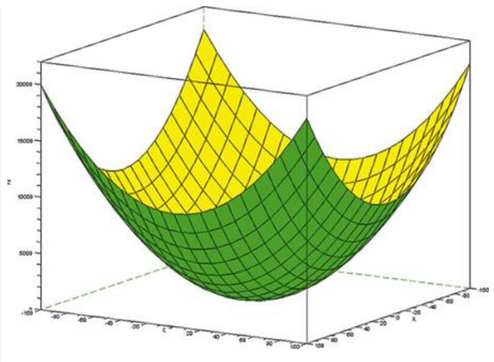

Figure 1. Problem 1 parabola [4]

Python and PYSWARM Modifications

- Python is a language with relatively simple syntax.
- PYSWARM[2] is a free PSO module that included basic functionality, but failed to handle narrow solution spaces.
- Feasibility function was created to evaluate the magnitude of constraint violation.
- New logic was applied (Figure 8.)

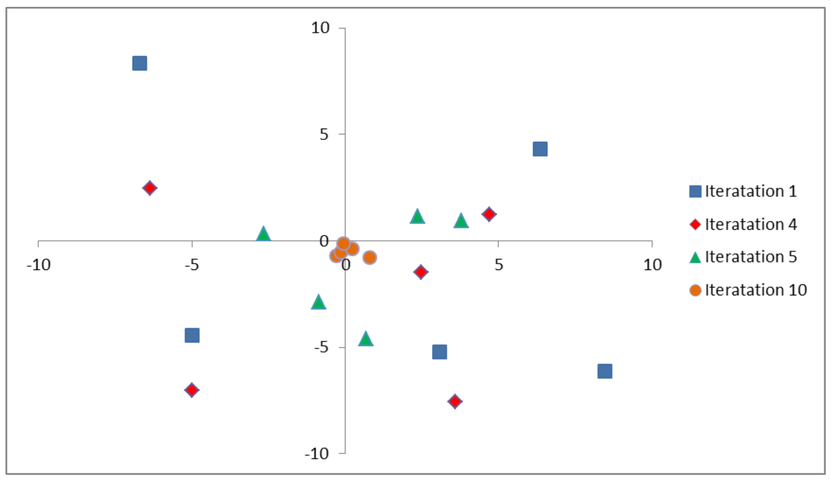

Figure 2. Problem 1 particle positions

```
# Import the PYSWARM module
import pso

# Function to be minimized
    def f(x):
        # Extract variables from variable array
        x1 = x[0]
        x2 = x[1]
        x3 = x[2]
        # etc...
        return x1+x2+x3... # Insert function here

# Side constraint functions
    def g1(x):
        # Extract necessary variables from variable array
        x1 = x[0]
        x3 = x[2]
        return x1/x3 # Insert variable portion of constraint

    def g2(x):
        # Variables
        return # Function
    # etc...

# Combine side functions into single constraint
    def con(x):
        # Extract side constraint function values
        cg1 = g1(x)
        cg2 = g2(x)
        cg3 = g3(x)
        # etc...
        # Subtract constant (usually normalized to 1)
        return [cg1-1.0, cg2-1.0, cg3-1.0, ... ]
            # Output array

# Variable ranges
    lb = [a, c, e…] # Lower Bound
    ub = [b, d, f…] # Upper Bound

# Use PSO function
# xopt is the best position
# fopt is value at xopt
xopt, fopt = pso.pso(f, lb, ub, f_ieqcons=con)
```

Figure 4. Problem entry template

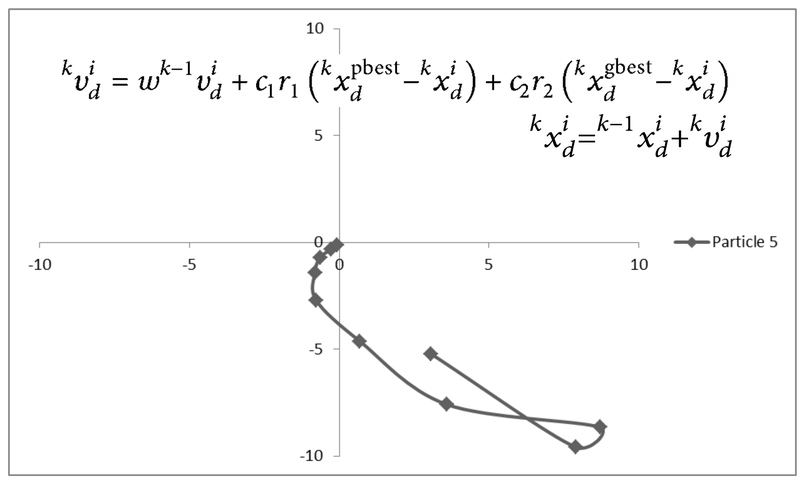

Figure 3. Problem 1 particle trajectory and PSO equations

Test Problems

- Problem 1: Trivial Solution, minimize parabola
  - Answer: [0, 0]
- Problem 2: Maximize Volume, obtain maximum volume of a cylinder given a limit on surface area.
  - Best Answer: r = 5.00027955 h = 9.99888179
- Problem 3: Speed Reducer Design, minimize weight, E03[1]
  - Best Answer: x = [3.5, 0.7, 17, 7.3, 7.8, 3.35021467]
- Problem 4: Spring Design, minimize weight, E04[1]
  - Best Answer: x = [0.05, 0.28202298, 2.0]

Problem Entry

- Define function f(x) to be minimized (cost, weight, etc.)
- Define side constraints, g(x) ≥ c
- Define variable ranges, a < x < b

Conclusion

- For problems 1-3, the function performed as expected, although with less accuracy.
- For problem 4, a better solution was found than published. Discrepancies in side constraints could have occurred.
- Accuracy increased with more particles.

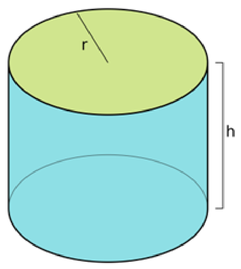

Figure 5. P2 Cylinder [Wikipedia]

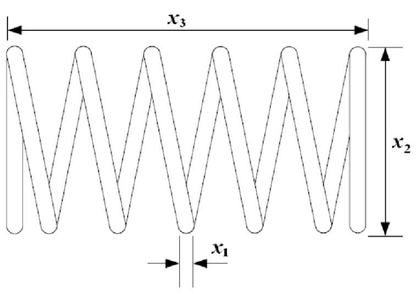

Figure 6. P4 Spring [5]

Table 1: Summary of Results

		# of Particles/ Max Iterations	Best	Worst	Mean	St. Dev
Problem 2	**Modified PYSWARM**	20 / 1000	-785.398128	-783.5826	-785.1333	0.3535
		40 / 500	-785.398156	-784.9837	-785.3376	0.0864
Problem 3	**MCEPSO[1]**	10 / 3000	2996.348166	2996.3560	2996.3495	0.0027
		20 / 1500	2996.348165	2996.3482	2996.3482	0.0000
	SiCPSO[1]	10 / 3000	2996.348165	2996.3481	2996.3481	0.0000
		20 / 1500	2996.348165	2996.3481	2996.3481	0.0000
	Modified PYSWARM	20 / 1000	2996.348173	3222.2627	3064.6861	65.3745
		40 / 500	2996.348170	3165.9279	3025.9751	29.0723
Problem 4	**MCEPSO[1]**	10 / 3000	0.012665	0.0156	0.0134	0.0007
		20 / 1500	0.012666	0.0146	0.0132	0.0005
	SiCPSO[1]	10 / 3000	0.012665	0.0170	0.0136	0.0010
		20 / 1500	0.012665	0.0146	0.0133	0.0005
	Modified PYSWARM	20 / 1000	0.002820	0.0030	0.0029	0.0001
		40 / 500	0.002820	0.0030	0.0028	0.0001

*PYSWARM runs were performed with w=0.5 and c_1=c_2=1.8

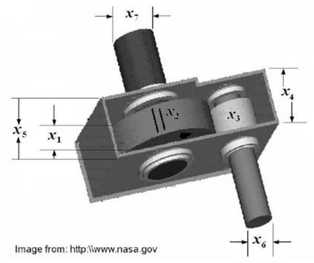

Figure 7. P3 Speed Reducer [5]

If $f(x^i) < f(x^{pbest})$ and $dist(x^i) \leq dist(x^{pbest})$:
  $x^{pbest} = x^i$
  If $f(x^i) < f(x^{gbest})$ and $dist(x^i) \leq dist(x^{gbest})$:
    $x^{gbest} = x^i$
Else If $dist(x^i) < dist(x^{pbest})$:
  $x^{pbest} = x^i$
  If $f(x^i) < f(x^{gbest})$ and $dist(x^i) \leq dist(x^{gbest})$:
    $x^{gbest} = x^i$

Figure 8. Logic for selecting best particles

References

1. Giordano Tomassetti and Leticia Cagnina, "Particle Swarm Algorithms to Solve Engineering Problems: A Comparison of Performance," Journal of Engineering, vol. 2013, Article ID 435104, 13 pages, 2013. doi:10.1155/2013/435104
2. Lee, Abraham. "Particle Swarm Optimization (PSO) with Constraint Support." PYSWARM. 27 Feb. 2014. Web. 2 Mar. 2015. <http://pythonhosted.org/pyswarm/>.
3. Brownlee, Jason. "Particle Swarm Optimization." Clever Algorithms: Nature-Inspired Programming Recipes. 1 Jan. 2014. Web. 2 Mar. 2015. <http://www.cleveralgorithms.com/nature-inspired/swarm/pso.html>.
4. McCaffrey, James. "Particle Swarm Optimization." MSDN Magazine. Microsoft, 1 Aug. 2011. Web. 2 Mar. 2015. <https://msdn. microsoft.com/en-us/magazine/hh335067.aspx>.
5. L.C. Cagnina, S.C. Esquivel, "Solving engineering optimization problems with the simple constrained particle swarm optimizer", Informatica 32 (3) (2008)319–326.