Hybridization of Particle Swam Optimization and Pattern Search Algorithms with Application

Eric Koessler, Dr. Ahmad Almomani

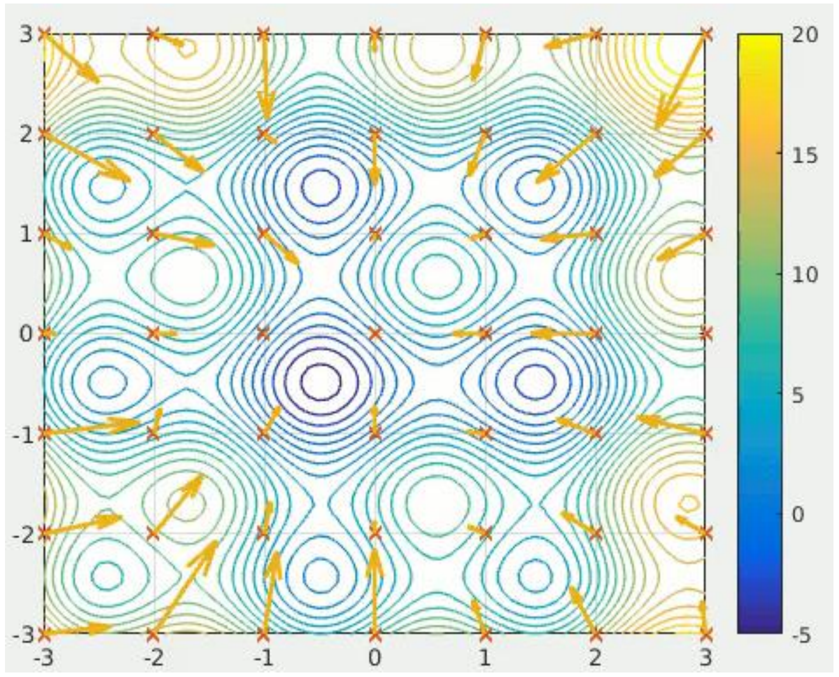
What is Derivative-Free Optimization

- Design algorithm to take input function and return optimal location and optimal function value (usually the minimum)
- Derivative free: some functions are non-differentiable, noisy, and/or costly to differentiate (meaning you can’t just find the slope at a point in the function and slide down the slope to the bottom of the function) so we need an algorithm that does not try to calculate derivatives/find slopes
- Goal is an accurate, quick, and robust algorithm

Particle Swarm Optimization

- Global algorithm: explores entire search space to find global instead of local-only minimum
- “Particles”: points where the function is being evaluated
- Particle velocity in each iteration determined by both a personal and swarm intelligence

Example of PSO on a 2D function



Click play! (only works on the Powerpoint file)

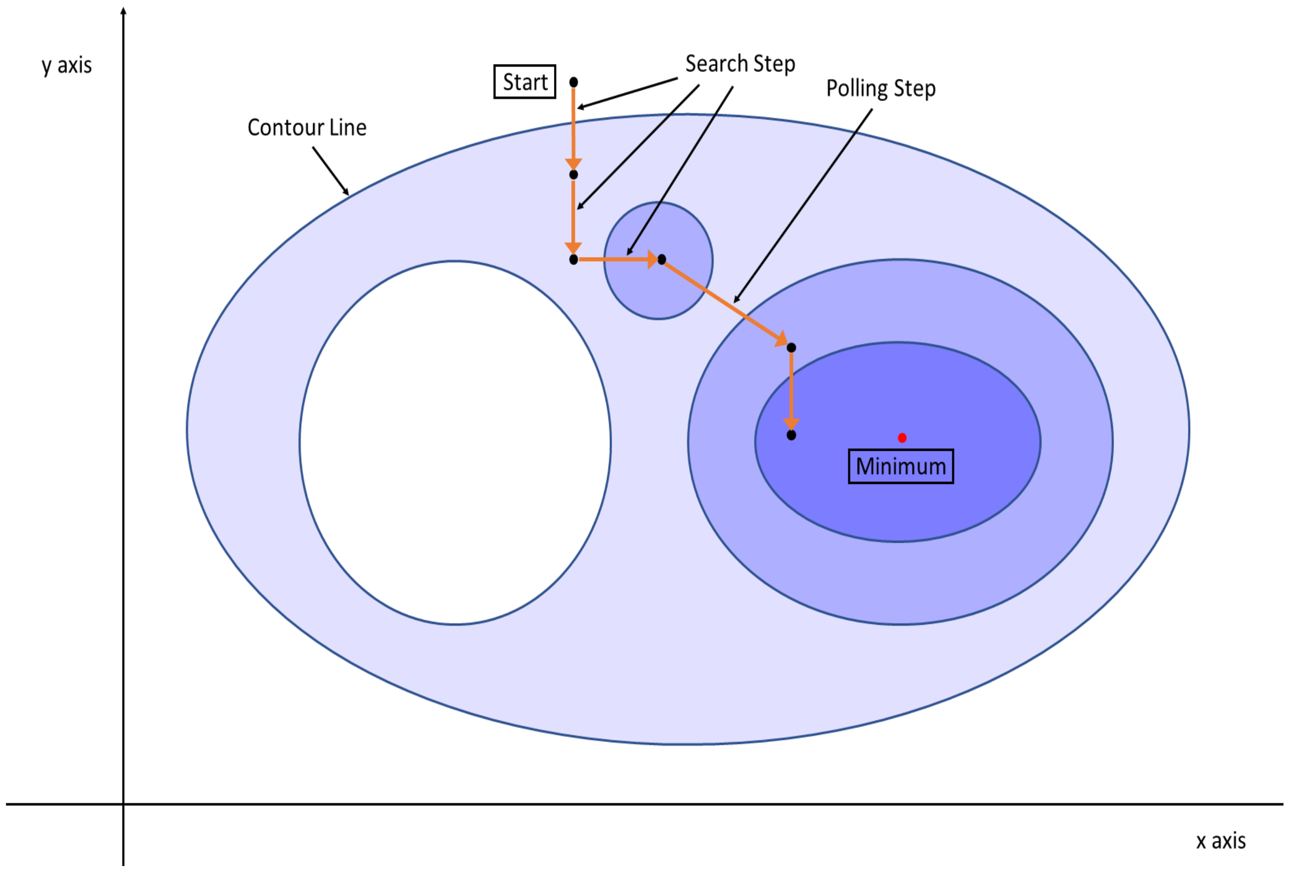
PSO Algorithm

$$\boldsymbol{V}(\boldsymbol{i}+\mathbf{1}) = \boldsymbol{w} \cdot \boldsymbol{V}(\boldsymbol{i}) + \boldsymbol{C}_{\boldsymbol{p}} \cdot \boldsymbol{r}_{\boldsymbol{p}} \cdot [\boldsymbol{p}(\boldsymbol{i}) - \boldsymbol{x}(\boldsymbol{i})] + \boldsymbol{C}_{\boldsymbol{g}} \cdot \boldsymbol{r}_{\boldsymbol{g}} \cdot [\boldsymbol{g}(\boldsymbol{i}) - \boldsymbol{x}(\boldsymbol{i})]$$

- $\boldsymbol{V}(\boldsymbol{i}+\mathbf{1})$ is the particle velocity at the next iteration
- $\boldsymbol{x}(\boldsymbol{i})$ is the particle’s current location
- $\boldsymbol{p}(\boldsymbol{i})$ is the particle’s best location so far
- $\boldsymbol{g}(\boldsymbol{i})$ is the global best location so far
- $\boldsymbol{w}$ is the inertial coefficient (helps maintain current direction)
- $\boldsymbol{C}_{\boldsymbol{p}}$ and $\boldsymbol{C}_{\boldsymbol{g}}$ are the personal and global coefficients
- $\boldsymbol{r}_{\boldsymbol{p}}$ and $\boldsymbol{r}_{\boldsymbol{g}}$ are random numbers from (0,1)

Pattern Search

- Local algorithm: doesn’t attempt to explore entire search space, quickly finds the local minimum
- Checks function values near current location using a certain “pattern” (list of directions to search)



PS Algorithm

1. Search step: Starting at current point, evaluate points in each direction at a certain step size (e.g. evaluate points up, down, left, and right of the current point)
2. If a better point is found than the current point, replace with better point and repeat step 1
3. Polling step: evaluate points around current point using a designated pattern (can include diagonals)
4. If a better point is found, replace with better point and repeat step 1; if no better point is found, reduce step size and repeat step 1
5. Stop algorithm when a minimum tolerance is reached

Why Hybridize?

- PSO problem: good at finding deepest well, slow at finding very bottom of the well
- PS problem: doesn't explore most of search space → find local-only minimum
- Hybrid algorithm: let PSO find the deepest well, then let PS find bottom of the well

3 Methods of Hybridization

- Method 1: Use PSO with a lot of particles for a small number of iterations, then run PS
- Method 2: Use PSO with standard number of particles until a minimum tolerance is reached, then run PS
- Method 3: Use PSO with standard number of particles until average particle distance from global best location is under threshold, then run PS

Overview of Benchmark Results

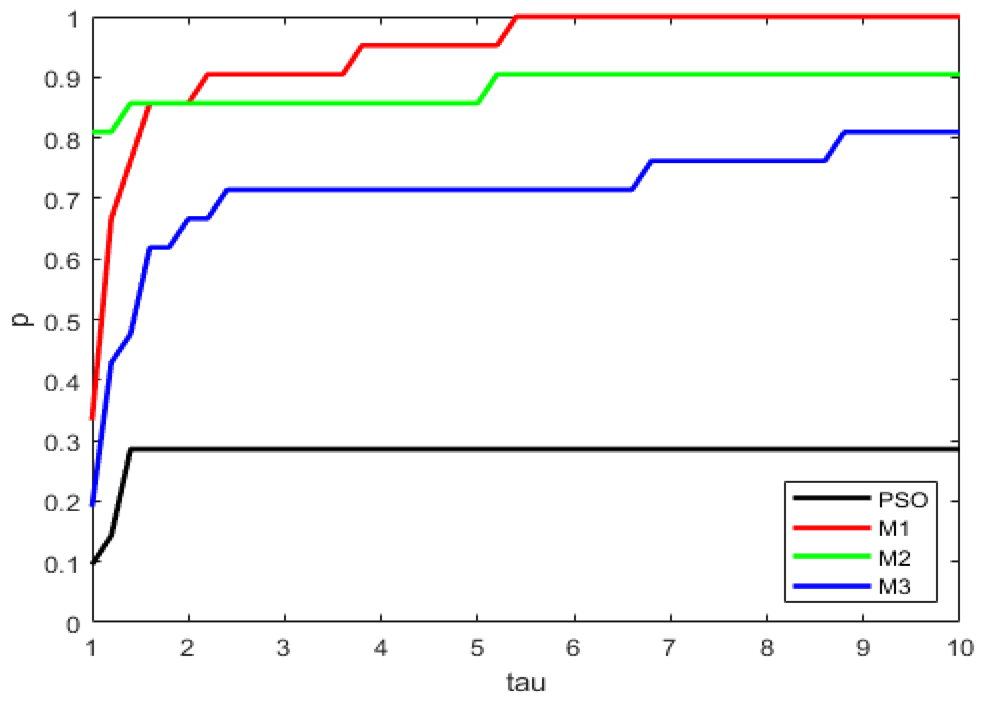
- Hybrid methods compared to PSO against 21 benchmark functions
- All hybrid methods found better (lower) y-values than PSO
- Only method 3 used fewer function evaluations than PSO
- All hybrid methods were more robust (less variance) than PSO
- Performance profile used to evaluate performance (higher is better)

Performance of Mean and Median Best Y-values

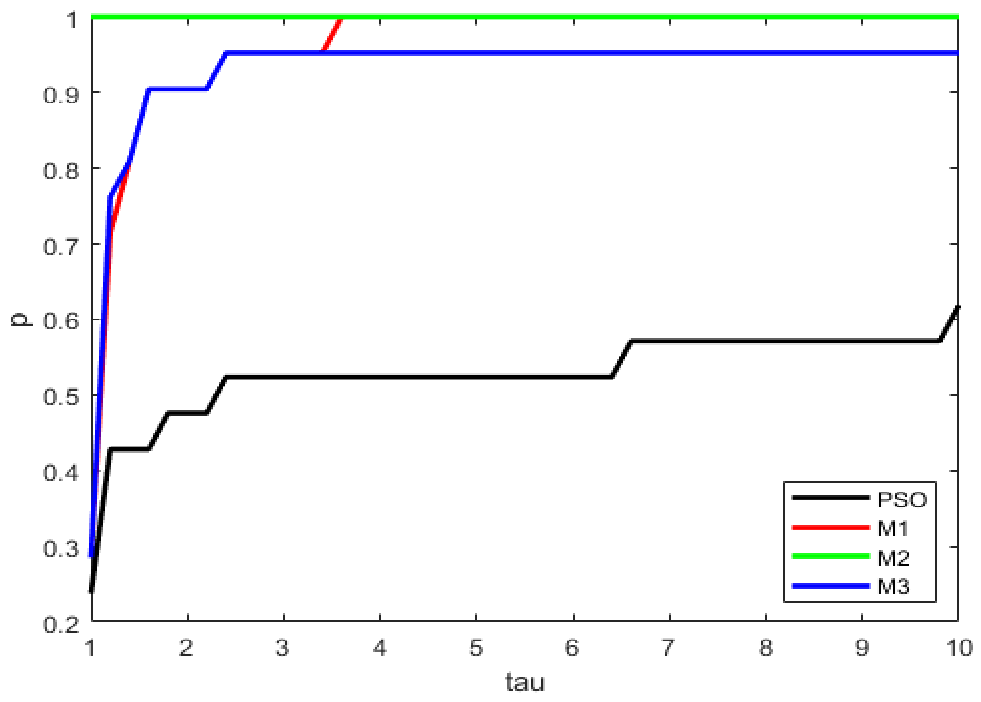
(how good are the best values that the algorithm usually finds)

*the higher the line, the better the algorithm did relative to the title statistic (mean/median/ect.)

Mean



Median

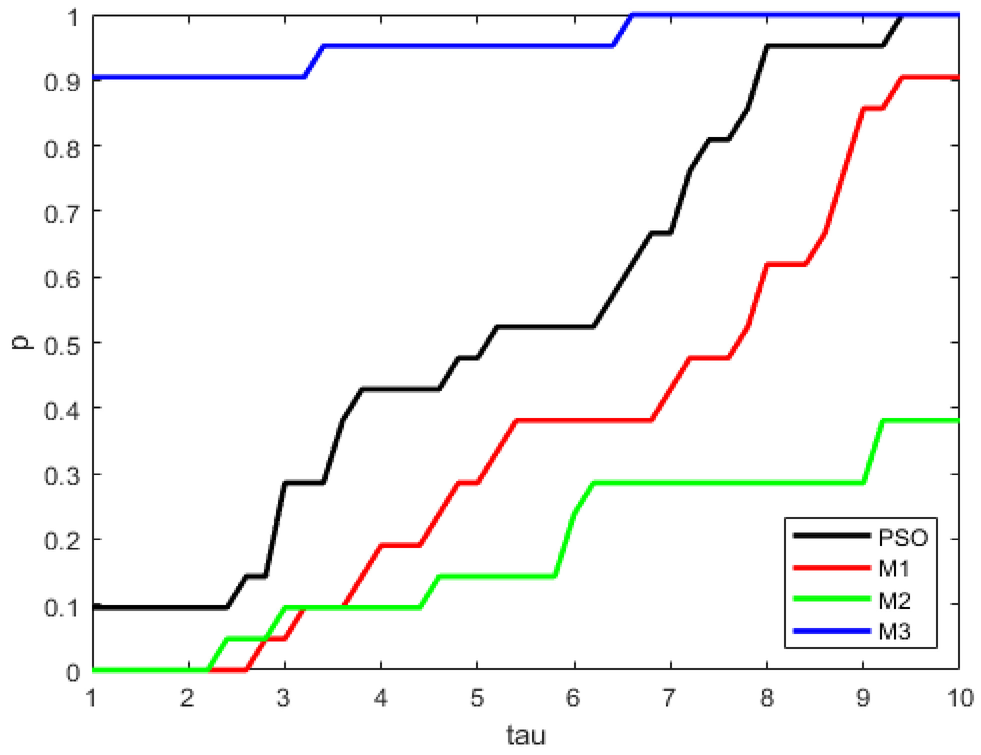


Performance of Median Function Evaluations

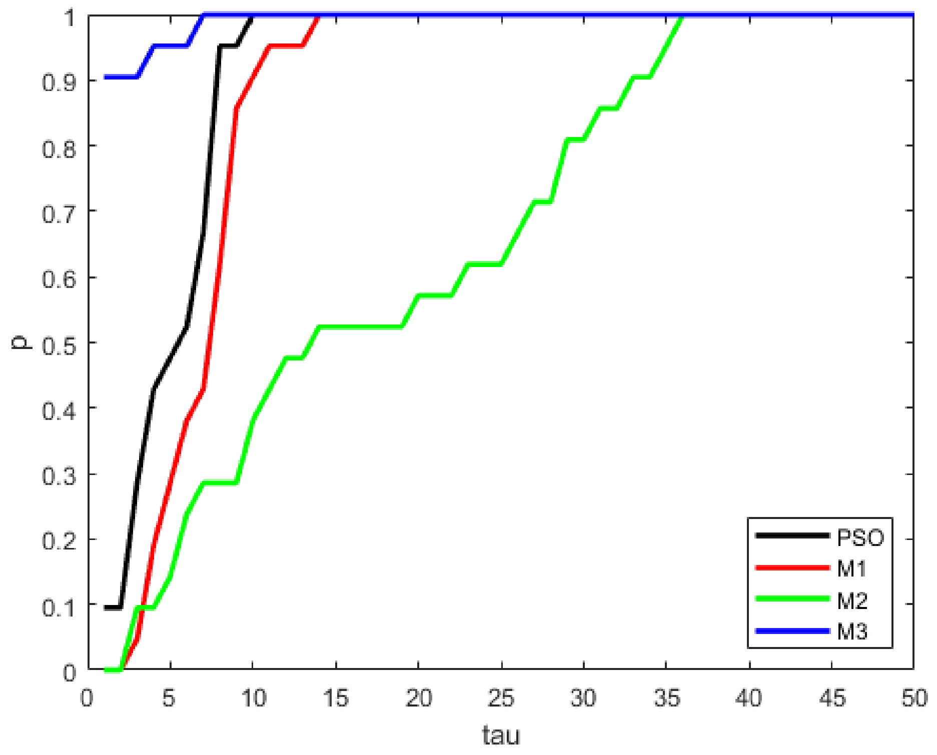
(how fast does the algorithm usually work)

*low lines at larger values of tau indicates much worse performance

Median (tau: 1-10)



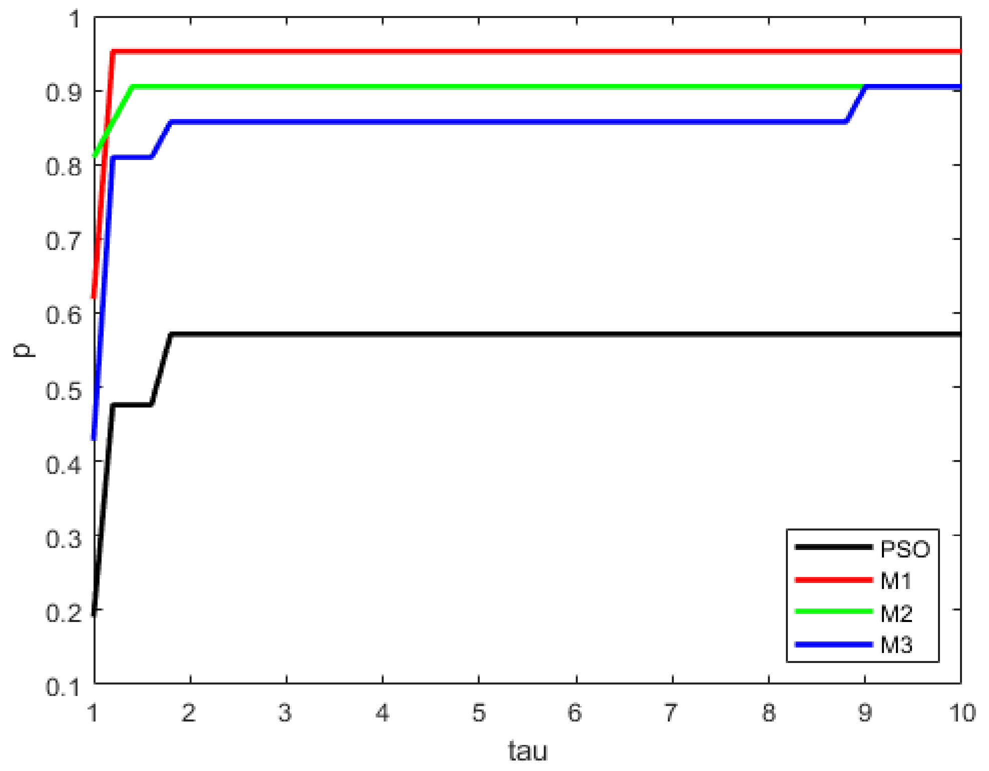
Median (tau: 1-50)



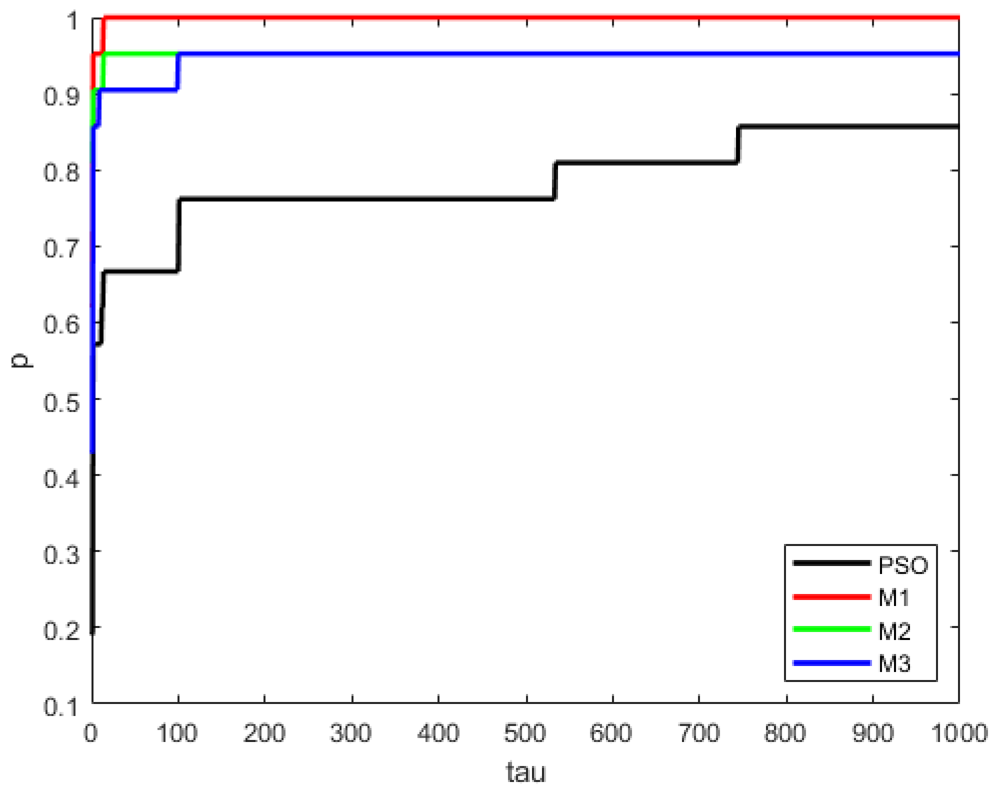
Performance of Variance of Best Y-values

(how reliable/consistent is the algorithm usually)

Variance (tau: 1-10)



Variance (tau: 1-1000)



Application to Water Basin Network Problem

- To test our hybrid algorithm on a real-world problem, a water basin network problem that was previously optimized with other algorithms was chosen to compare against
- To collect rainwater that flows down into rivers, a network of water basins is needed to capture the water. The problem is: where to place these basins and how big should each one be
- Need to minimize the cost of building/maintaining these basins as well as make sure they never overflow or dry up

Network Equation and Results

- This is the function we had to minimize (wow!)

$$f(\mathbf{x}) = \sum_{i=1}^{n} \left(\frac{L_i}{y_i} x_i + C_i x_i^{D_i} \right) + \alpha \left(Q_0 - \sum_{i=1}^{n} x_i \right)^2 + P(\mathbf{x}, \rho_0, \alpha_0),$$

$$P(\mathbf{x}, \rho_0, \alpha_0) = \sum_{i \in E} \alpha_0 \ln \left(2 + 2 \cosh \left(\frac{\rho_0 c_i(\mathbf{x})}{\alpha_0} \right) \right) + \sum_{i \in I} \alpha_0 \ln \left(1 + \exp \left(\frac{\rho_0 c_i(\mathbf{x})}{\alpha_0} \right) \right)$$

- Our hybrid algorithm (using method 3) found a lower cost (\$757,391) than the best result using a Genetic Algorithm from the original paper (\$760, 572)
- This indicates that our hybrid algorithm can successfully compete against other currently in-use optimization algorithms

Conclusions

- Letting PS run after PSO improved performance with the minimum y-values and robustness (reliability) compared to PSO
- Cutting off PSO after reaching a minimum average particle distance (method 3) reduced function evaluations
- The hybrid algorithm compares favorably to other current global algorithms for real-world problems