Alternative Analysis Networking
A Multi-characterization Algorithm

Kevin Albarado and Roy Hartfield
Department of Aerospace Engineering, Auburn University, Auburn, AL, U.S.A.

Keywords: Particle Swarm, Neighborhooding, Unsupervised Training, Kohonen.

Abstract: A neural network technique known as unsupervised training was coupled with conventional optimization schemes to develop an optimization scheme which could characterize multiple "optimal" solutions. The tool discussed in this study was developed specifically for the purposes of providing a designer with a method for designing multiple answers to a problem for the purposes of alternative analysis. Discussion of the algorithm is provided along with three example problems: unconstrained 2-dimensional mathematical problem, a tension-compression spring optimization problem, and a solid rocket motor design problem. This algorithm appears to be the first capable of performing the task of finding multiple optimal solutions as efficiently as typical stochastic based optimizers.

1 INTRODUCTION

Numerous methods for optimizing constrained, complex problems have been proposed and proven to be successful in implementation (Holland, 1992; Eberhart and Kennedy, 1995). In 1975, John Holland first suggested modeling optimization schemes after phenomena observed in nature such as evolution, crossover, mutation, etc. which eventually developed into what is now called a genetic algorithm. Eberhart and Kennedy followed suit by developing the particle swarm optimizer, which mimics behavior of crowding and flocking observed in birds, insects, and mammals alike. Both of these concepts are population based stochastic methods. On their own, population based methods are very efficient at finding designs near the global optimum, but can never be gauranteed to find the absolute global minimum point. For that reason, Jenkins and Hartfield (Jenkins and Hartfield, 2010) added a pattern search optimization routine to the particle swarm routine to form a hybrid optimization routine that can effectively move off of local optima and still climb the hills to find the global optima. This same setup has been implemented into a strategy for finding multiple optimal solutions by using a concept known as neighborhooding. Neighborhooding traditionally is the limiting of information for a which particle can access in the particle swarm optimization technique. Typically, this information limitation is random with no real structure or organization (Eberhart and Kennedy, 1995). In this work it is shown that by simply restricting the communication topology amongst the particles, self-organized communication networks can be formed, resulting in an environment suitable for multi-characterization of local optima. The restricting topology was formed using unsupervised training to classify the initial population into individual clusters of particles (henceforth called neighborhoods).

2 UNSUPERVISED TRAINING

Unsupervised Training was developed by the neural network community as a method for classifying clusters of data when no metric for classification was available (Masters, 1993). The concept of competitive learning and self organization was developed by Tuevo Kohonen (Kohonen, 2001). Typical neural network training updates all neuron weights through each training iteration. For unsupervised training, however, neurons compete for learning. The Kohonen network follows this same logic, however input normalization occurs before training thereby creating a three layer network. Kohonen training is a six step exercise that is repeated iteratively until the network converges to a steady state. These steps are:

1. Normalize input patterns such that inputs fall onto an N-dimensional hypersphere where N is the

Albarado K. and Hartfield R..
Alternative Analysis Networking - A Multi-characterization Algorithm.
DOI: 10.5220/0004147901830188
In *Proceedings of the 4th International Joint Conference on Computational Intelligence* (ECTA-2012), pages 183-188
ISBN: 978-989-8565-33-4

number of inputs.

$$z_n = \frac{x_n}{\sqrt{\sum_{i=1}^{N} x_i^2}} \quad (1)$$

where x is the initial set of inputs, and z are the normalized inputs.

2. Randomly initialize weights, w, for the desired number of neurons and normalize weights according to Equation 2.

$$w_n = \frac{w_n}{\sqrt{\sum_{i=1}^{N} w_i^2}} \quad (2)$$

3. One input pattern at a time, apply the inputs to each neuron.

$$net = \sum_{i=1}^{N} z_i w_i = \mathbf{ZW}^T \quad (3)$$

4. Update neuron with the highest net value

$$W_k = W_k + \alpha Z \quad (4)$$

where α is a learning constant

5. Renormalize new weights using Equation 2.
6. Repeat 3, 4, and 5 for each input pattern.

Two important properties of note for this classification method are that during the training, some neurons will never take part in learning, which will lead to neighborhoods being unoccupied, and there is no restriction on how large or small a neighborhood can be. In a 2-dimensional sense, the Kohonen technique divides the domain into unequally sized "slices of pie", and whichever slice of pie a particle falls within dictates which neighborhood that particle belongs to. This is demonstrated in Figure 1. From the Figure, it is shown how seemingly unclustered data can be clustered into distinct groups. These individual groups will form neighborhoods of particles that will be applied to the optimization schemes to be discussed. Having the ability to define the number of neighborhoods desired effectively gives the user the freedom to estimate the number of local optima to search for.

3 OPTIMIZATION

After setting up the neighborhoods, three distinct optimization phases take place: base level optimization within the individual neighborhoods, a second level optimization amongs the top performers from each neighborhood, and a gradient based routine for solution refinement. This last point is directly inspired by the successes demonstrated by (Jenkins and Hartfield, 2010).

3.1 Base Level Optimization

The particle swarm technique was used for base level optimization. Particles were limited in communication to only the particles within a given neighborhood. The equations of motion for a particle were unchanged from those presented by Mishra (Mishra, 2006) in the repulsive particle swarm optimizer (RPSO).

$$v_{i+1} = R_1 \alpha\omega (\hat{x}_m - x_i) + R_2 \beta (\hat{x}_i - x_i) + \omega v_i + R_3 \gamma \quad (5)$$

$$x_{i+1} = x_i + v_{i+1} \quad (6)$$

The Mishra RPSO algorithm was used due to its robustness as compared to the basic particle swarm optimizer. In particular, the Mishra algorithm is more effective in especially complex design spaces, and has been tested on a wide array of functions.

3.2 Second Level Optimization

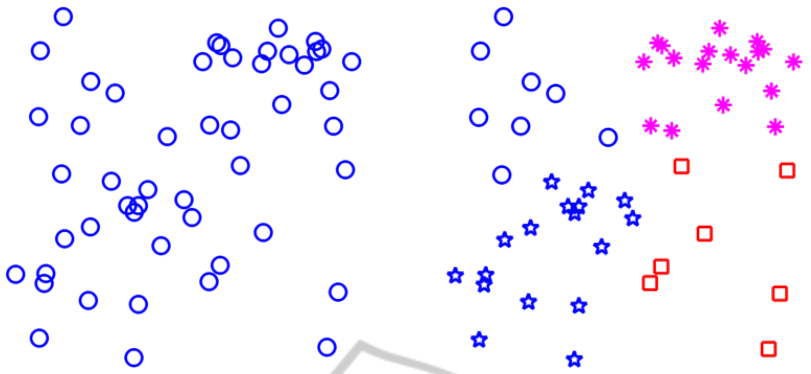
The second level optimization was implemented so that the neighborhoods could move throughout the design space. Without a second level optimization in place, each neighborhood would locate an optimum within the local domain, but all collectively could fail to find the optimal "region" in the entire design space. The second level optimization provides a way for the neighborhoods to move as whole. The method implemented in this study was an atomic attraction/repulsion model. This model was chosen somewhat arbitrarily but proved to be successful. Future work would involve investigating other models for this phase of optimization. The attraction/repulsion model is represented as

$$x_{i+1} = -(\hat{x}_m - x_i)\left(\frac{\gamma}{(\hat{x}_m - x_i)^2} - \frac{\omega}{(\hat{x}_m - x_i)^4}\right) \quad (7)$$

where $\hat{x}_m$ represents the global best member, and x represents the positions of the neighborhood best particles. γ and ω are tuning parameters.

3.3 Gradient based Optimization

A gradient based minimization routine was added to refine the best members of each neighborhood. This improves the probability that true local optima are found. The minimization scheme was the pattern search method developed by Hooke and Jeeves (Hooke and Jeeves, 1961). This method is robust and does not require exact calculation of the gradient. Rather, the pattern search method evaluates the gradient in each direction and performs a collective pattern



(a) Initial patterns. (b) Assigned patterns.

Figure 1: This Kohonen training example demonstrates 50 randomly distributed, unclustered particles separated into 4 distinct groups.

move in the direction of steepest descent. The pattern search method implemented into AAN based on the results observed by Jenkins and Hartfield (Jenkins and Hartfield, 2010). At the end of each base and second level maneuver, a pattern search is performed on the best performing members of each neighborhood.

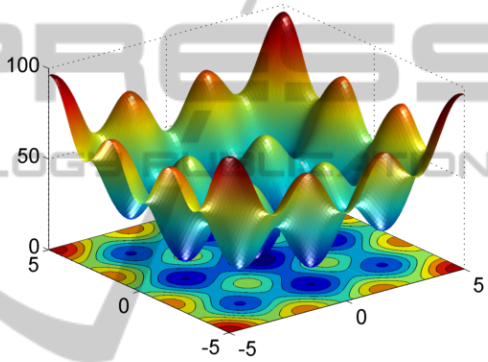


Figure 2: Surface/Contour for unconstrained problem $z = 25(\sin^2 x + \sin^2 y) + x^2 + y^2$.

4 RESULTS

Three test cases were performed with increasing complexity. The first test case is a simple 2-dimensional mathematical optimization problem that is unconstrained but fraught with numerous local optima. The second test case is a constrained tension/compression spring design problem with four input variables. The final case is a rocket motor design problem which presents numerous constraints, and is fraught with numerous local and global optima.

4.1 Case 1: Unconstrained Math Problem

The unconstrained math problem provides a validation that the optimization scheme works correctly. The answer is known beforehand, and can be verified quickly. In the case of classifying multiple optimal solutions, the unconstrained math problem still needs to be somewhat complex. Equation 8 results in the surface shown in Figure 2 which is fraught with numerous local minima and maxima.

$$z = 25\left(\sin^2 x + \sin^2 y\right) + x^2 + y^2 \qquad (8)$$

where x and y lie in the domain of $[-5,5]$. The surface has 8 local minima, 1 global minimum, 12 local maxima, and 4 global maxima. The global minimum lies at the origin with the global maxima at the corners of the domain. Table 1 gives a listing of the global and local minima and their associated output. These locations will serve as comparison markers for the optimization performance. The difference between the major local optima and the minor local optima is the associated function cost. The minor local have a higher function cost than that of the major local minima, making finding them less of a priority. The optimizer was tested with 150 members in the population, with the desired number of optima found to be 9. For every iteration of the optimizer, the pattern search was performed twice on each of the top members of the neigbhorhoods, with an inspection range of 5% of the domain in each direction. The

Table 1: Global and local minima, locations and output values.

Minima Number	X	Y	Output
Global Minima	0	0	0
Major Local Minima	± 3	0	9.5
Major Local Minima	0	± 3	9.5
Minor Local Minima	± 3	± 3	19

Table 2: Final results for the unconstrained math problem.

Minima Found	%	Total Found	%
2 Major	97%	5	93%
3 Major	77%	6	62%
2 Minor	62%	7	27%
3 Minor	19%		

tuning parameters for the particle swarm were all set to 0.3. The optimizer iteration limit was set to 25. This run was repeated 100 times to gauge how well the optimizer performs at identifying all local minima near the global minimum. Table 2 gives the statistics from the 100 runs. The global minimum was found in every run. From Table 2, the minor local minima were found less often than the major local minima. This was expected because as the solution develops, the major local minima draw particles at minor local minima away due to their lower function cost. It was expected that finding 3 of the 4 minor local minima should not happen nearly as often as finding 3 out of the 4 major local minima. The optimizer will always favor local minima with a better function cost over others because the second level optimization scheme will draw the neighborhoods toward the global minimum. Also from Table 2, five minima were found 93% of the time. Although not listed, at least four minima were found in every run, including the global minima in every run. This example demonstrates the optimizer's ability to simultaneuosly seek out local optima but also jump from minor local optima to major local optima and global optima. Demonstrating this ability is a major step forward from a conventional optimizer, which is capable of either only finding global optima or only finding local optima, toward an optimizer that can do both, and in some cases simultaneously.

4.2 Case 2: Tension/Compression Spring Design

The tension/compression spring design problem has been optimized previously by Coello (Coello Coello, 2000) and Hu et al. (Hu et al., 2003). This problem represents a highly constrained engineering design problem for which the near optimal solution has been reported already. There have been no reported local optima to the problem. Mathematically, the spring design problem is formalized as minimization of the weight of a tension compression spring via Equation 9

$$f(\mathbf{X}) = (N+2)\,Dd^2 \tag{9}$$

where D is the mean coil diameter, d is the wire diameter, and N is the number of active coils. The spring design is subject to the following constraints.

$$G(1) = 1 - \frac{D^3 N}{71785d^4} \leq 0$$

$$G(2) = \frac{4D^2 - dD}{12566\,(Dd^3 - d^4)} + \frac{1}{5108d^2} \leq 0$$

$$G(3) = 1 - \frac{140.45d}{D^2 d} \leq 0$$

$$G(4) = \frac{D+d}{1.5} - 1 \leq 0$$

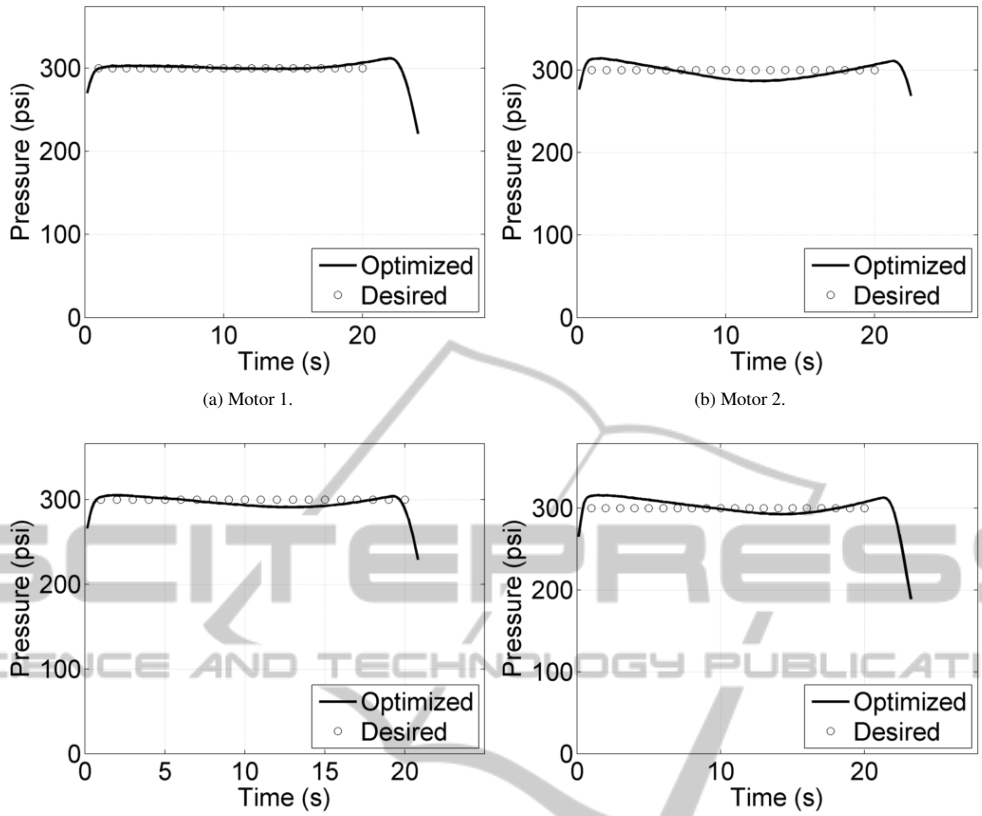
Table 3: Constrained tension/compression spring results.

	Coello	Hu et. al	This paper
d	.05148	.05146	.05000
D	.35166	.35138	.34749
N	11.63	11.60	11.13
G(1)	-.00333	6.84e-9	-.04163
G(2)	-.07397	-.07390	-.00354
G(3)	-4.026	-4.0431	-4.2216
G(4)	-.7312	-.73143	-.73500
$f(\mathbf{X})$	.01270	.01266	.01141

Because no local optima exist for this problem, the end goal for the optimizer should be to find the single global optima which meets the constraints. The optimizer was executed using 150 members with 150 generations with 4 neighorhoods. The final results given by Coello and Hu et al. are presented along side the results for this study in Table 3. The optimizer developed in this study was able to retain the ability to find the global optimum (and as it turns out, a better answer for this problem than previously reported) when no local optima exist despite the highly constrained design space, demonstrating the communication ability available between the neighborhoods.

4.3 Solid Rocket Motor Design Problem

The solid rocket motor design problem is associated with a highly complex design space typical of what would be observed in a real world engineering design scenario. The solid rocket motor program uses 9 geometric variables to describe a solid rocket motor. The specific geometric inputs used for this study are described in full detail in (Ricciardi, 1992; Hartfield et al., 2004). The goal for the optimizer in this case study was to find as many motor geometries that produces 300 psi pressure for 20 seconds exactly as possible. The performance metric for this problem is the RMS error between the output pressure-time profile and the neutral 300 psi-20 second curve desired. This is a common solid rocket motor problem but presents a number of challenges. There are a total of nine independent variables, making the design space complex



(a) Motor 1.

(b) Motor 2.

(c) Motor 3.

(d) Motor 4.

Figure 3: Pressure-time profile for four distinct solutions from solid rocket motor optimization.

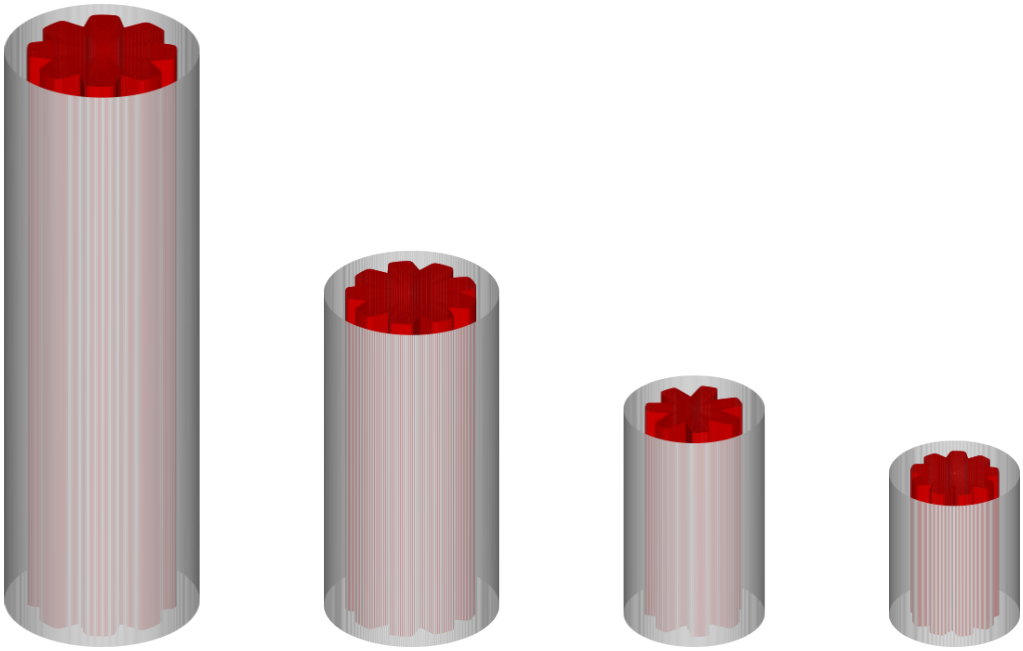


Figure 4: Motor geometries 1-4 based on 9 independent variables.

and difficult to visualize. Two of the independent parameters are integers. The first integer parameter, the number of star points, has a large effect on the profile of the pressure versus time curve (progressive, regressive, neutral, etc.). The second integer parameter, propellant selection is a series of choices and has no meaningful gradient, as some propellant choices are similar and some are very different. There are numerous geometric constraints, some of which can be handled implicitly by nondimensionalizing some of the independent variables in an intelligent manner. The optimizer was executed with 100 particles for 50 generations with 6 neighborhoods. The results of the optimization were 4 unique solutions, shown in Figures 3 and 4. While the pressure-time curves were not perfect, it is important to remember that the optimizer is looking for global and local optima. The results in Figure 3 should be considered local optima, as they are close (10% RMS) but not exact. From Figure 4, clearly the optimizer was searching vastly different areas of the design space. While the fuel type selection was the same for all four, and the number of star points for each motor was between 7 and 9, the lengths of each motor are drastically different. This example effectively demonstrates the full capabilities of the optimizer at locating unique local optima in a complex and constrained design space.

5 CONCLUSIONS

The development of population based optimization routines brought about the capability to locate solutions in a complex and constrained design space. These stochastic schemes typically only develop a single solution, usually the global optimum. In most instances, however, it is desirable to find multiple optimal solutions. The algorithm described in this paper is capable of accomplishing this feat. It was shown in this study that the algorithm developed has the following advantages:

- From the unconstrained mathematical problem, it was shown that the algorithm is capable of jumping from minor local optima toward major local optima and the global optima. This fact is important in verifying that the optimizer will not simply find a local optima within a local neighborhood domain, but instead will make some attempt to improve.
- The constrained tension/compression spring example demonstrated the algorithm's ability to navigate a complex design space and constraints as well as the algorithm's ability to find a global optima when no local optima are known to exist.
- The solid rocket motor example proved that the algorithm can be effective in practical real world engineering design problems.
- The Kohonen unsupervised training technique provides the user the ability to define the number of desired optimal solutions to search for.

While the development thus far shows great promise, some improvements can still be made to make the algorithm more efficient. A study of optimization techniques should be performed for the base level and second level optimization phases to determine which combination of optimization schemes are most efficient for a wide range of problems. The gradient based scheme can also be improved by switching to a more robust and efficient hill climbing routine.

REFERENCES

Coello Coello, C. (2000). Use of self-adaptive penalty approach for engineering optimization problems. *Computers in Industry*, 41(2):113–127.

Eberhart, R. and Kennedy, J. (1995). A new optimizer using particle swarm theory. *Proceedings Sixth Symposium on Micro Machine Human Science*.

Hartfield, R., Jenkins, R., Burkhalter, J., and Foster, W. (2004). A review of analytical methods for predicting grain regression in tactical solid rocket motors. *AIAA Journal of Spacecraft and Rockets*, 41(4).

Holland, J. (1992). *Adaptation in Natural and Artificial Systems*. MIT Press.

Hooke, R. and Jeeves, T. (1961). Direct search solution of numerical and statistical problems. *Journal of Association of Computing Machinery*, 8(2).

Hu, X., Eberhart, R., and Shi, Y. (2003). Engineering optimization with particle swarm. Technical report, IEEE Swarm Intelligence Symposium.

Jenkins, R. and Hartfield, R. (2010). Hybrid particle swarm-pattern search optimizer for aerospace applications. Technical report, AIAA Paper 2010-7078.

Kohonen, T. (2001). *Self-Organizing Maps*. Springer.

Masters, T. (1993). *Practical Neural Network Recipes in C++*. Academic Press.

Mishra, S. (2006). Repulsive particle swarm method on some difficult test problems of global optimization. Technical report, Munich Persona RePEc Archive Paper No. 1742.

Ricciardi, A. (1992). Generalized geometric analysis of right circular cylindrical star perforated and tapered grains. *AIAA Journal of Propulsion and Power*, 8(1).