**Proceedings of the 2007 IEEE Symposium on
Foundations of Computational Intelligence (FOCI 2007)**

Simulated Annealing with Opposite Neighbors

Mario Ventresca and Hamid R. Tizhoosh

***Abstract*— This paper presents an improvement to the vanilla version of the simulated annealing algorithm by using opposite neighbors. This new technique, is based on the recently proposed idea of opposition based learning, as such our proposed algorithm is termed opposition-based simulated annealing (OSA). In this paper we provide a theoretical basis for the algorithm as well as its practical implementation. In order to examine the efficacy of the approach we compare the new algorithm to SA on six common real optimization problems. Our findings confirm the theoretical predictions as well as show a significant improvement in accuracy and convergence rate over traditional SA. We also provide experimental evidence for the use of opposite neighbors over purely random ones.**

***Index Terms*— Opposition based learning, simulated annealing, optimization.**

I. Introduction

IMPROVING properties of accuracy, convergence and reliability of a global search algorithm is an important area of research. There are direct practical effects, in that higher quality solutions to real world problems can be discovered in shorter periods of time. This can lead to improved organization as well as saving money and time for a myriad of applications from a variety of different areas.

Simulated annealing is a popular global search technique that has been widely utilized over the past quarter century [1], [2]. This paper proposes a method based on the concept of opposition-based learning to improve simulated annealing. We will provide theoretical evidence to support a claim that using this concept we are guaranteed a more desirable evaluation function. By using opposition-based learning we will show an improvement in the accuracy, convergence rate and reliability of simulated annealing.

The remainder of this paper is organized as follows: Section II describes the motivation and required theoretical foundations of opposition-based learning that are needed for this paper. Section III will describe the OSA algorithm followed by a description of the benchmark functions in Section IV. The experimental setup will be outlined in Section V and our results will be presented in Section VI. Conclusions and directions for future research will be given in Section VII.

II. Opposition Based Learning

The concept of opposition-based learning (OBL) is a recently proposed computational intelligence idea [3]. OBL is a simple concept that can be used to accelerate the convergence rate and/or accuracy and reliability of a learning algorithm. In this section we will provide the motivating principles and theoretical foundations of OBL.

This work has been supported in part by Natural Sciences and Engineering Council of Canada (NSERC).

M. Ventresca is a student member of the Pattern Analysis and Machine Intelligence (PAMI) laboratory in the Systems Design Engineering Department, University of Waterloo, Waterloo, ONT, N2L 3G1, CANADA (email: mventres@pami.uwaterloo.ca)

H. R. Tizhoosh is a faculty member of the Pattern Analysis and Machine Intelligence (PAMI) laboratory in the Systems Design Engineering Department, University of Waterloo, Waterloo, ONT, N2L 3G1, CANADA (email: tizhoosh@uwaterloo.ca)

A. Motivating Principles

It seems that nearly everything in nature has an opposite, for example we can observe male/female sexes, positive/negative electrical charges, hot/cold temperatures, and so on. We can also observe opposites in a social context where revolutions represent a rapid change from one doctrine to another, typically opposite one. For computational intelligence, this idea manifests itself as a solution and opposite solution and can be represented as a pair $< s, \breve{s} >$, respectively. The manner in which these solutions correspond can be defined according to some mapping function.

Consider the problem of discovering the minima of some unknown real function $f: \Re \rightarrow \Re$. In general a search/optimization algorithm will begin by making some random guess/solution s over the interval on which f is defined. If we are not satisfied with $f(s)$ then we can choose some other solution (either dependant or independent on s) in the hopes of discovering some s such that $|f(s^*) - f(s)| < \epsilon$, where $\epsilon \in \Re$ is a desired accuracy and s^* is the optimal solution. Alternatively, we may terminate after a maximum number of guesses have been evaluated. In any event this stochastic process has an associated computational time. Thus, it is important to develop algorithms and techniques that can determine high quality solutions quickly and reliably, which is where OBL can be used.

The premise of OBL is that in lieu of just selecting some random guess, it is beneficial to also consider its opposite guess. If we take two random guesses then it is possible that they are relatively close to each other in solution space, which can hinder the performance of the algorithm. On the other hand, if the two guesses are sufficiently distant from each other, say on opposite sides, then we cover the search space more adequately. Effectively, we are maintaining a higher degree of diversity. With reference to the above example, an OBL-based search algorithm will simultaneously consider s and its opposite $\breve{s}$ and continue with the more desirable of the two. For a minimization problem this can be accomplished by taking the minimum of the two guesses (with respect to some evaluation function). In order to avoid convergence to a local optima we can probabilistically accept this value.

To date, OBL has been utilized to improve the convergence rate and/or accuracy of differential evolution (by chromosomes/anti-chromosomes) [4], [5], [6], reinforcement learning (by opposite actions/states) [7], [8], [9] and backpropagation learning in feed forwards neural networks (by opposite

transfer functions) [10]. This paper will aim to improve yet another popular global search algorithm, namely simulated annealing [11], [12].

B. Theoretical Foundations

The key to understanding opposition-based learning, is in the definition of an opposite, which is accomplished by the opposition map and opposite evaluation functions. As we will show below, this map creates a symmetric evaluation function which is guaranteed to have a more desirable expected value than otherwise. However, we must first define what it means to be opposite, as it is used in this paper.

Definition 1 (Opposite): Given a representation space S, two solutions $s_1, s_2 \in S$ are considered opposite if $d(s_1, r^*) = d(s_2, r^*)$, where $d(\cdot)$ is a distance function and r^* is some reference point in S. We will denote the opposite of s as $\breve{s}$. Furthermore, s_1 and s_2 must lie on opposite sides of r^*.

This definition allows for multiple opposites. However, to be computationally efficient it is recommended that each element have only 1 unique opposite. Furthermore, knowledge of $d(\cdot)$ is not required in practice. Now, we can define the opposite map as

Definition 2 (Opposite map): Any function $\Psi: S \rightarrow S$ that given some solution s, returns the set ϑ containing s and the set of its opposites $\breve{s}$. That is $\vartheta = s \cup \breve{s}$, where $|\breve{s}| \geq 1$.

An important consequence of this definition, is that elements of ϑ_j can only be opposites of each other. That is, given some $s_2 \in S$, $s_2 \cap \vartheta_j = \phi$ if and only if s_2 is not an opposite of some element in ϑ_j. Additionally it is possible, although not practically useful, to define $\breve{s} = \{s\}$ (i.e. s is its own opposite).

If we let Θ be the set of all resulting sets of the representation space R by some opposite map Ψ, then $|\Theta| \leq |R|$. If each element $\vartheta_j \in \Theta$ contains 2 elements (each value has a single unique opposite), then $|\Theta| = |R|/2$, which effectively halves the search space. In the event $|R| = \infty$, there will still be an improvement in expected result as a consequence of the opposite evaluation function, which results from the opposite map.

Definition 3 (Opposite evaluation function): Any function $\Phi: S \rightarrow \Re$ that takes an element $s \in S$ and returns the most desirable (w.r.t. the problem) of $\Psi(s)$ can be utilized as an opposite evaluation function.

As an example, if we consider a minimization problem, evaluation of any $s_i \in \vartheta_j$ returns $min(s_i)\ \forall\ s_i \in \vartheta_j$. The dotted line in Figure 1 represents the original evaluation function f to some hypothetical problem. Assuming an algorithm not using OBL has selected a solution of 1, then $f(1) = 16$. However, if we define an opposite map as $\Psi(s) = s, -s$, which is symmetric about $r^* = 0$, then using the concept of opposition $f(1) = \min(f(1), f(-1)) = -5.1378$. Clearly, -5.1378 is a much more desirable value than 16. The solid line in Figure 1 represents the opposite evaluation function for this problem.

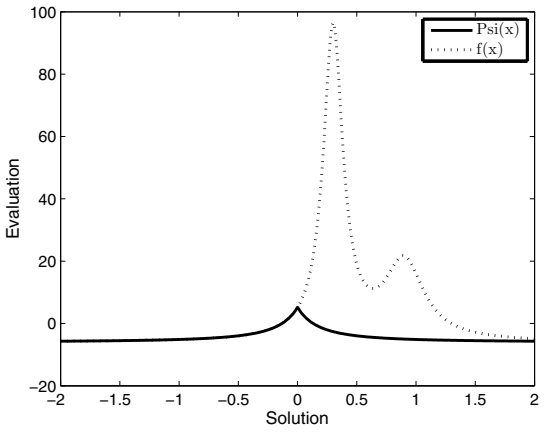

Fig. 1: Example of an opposite evaluation function w.r.t. the original evaluation function.

Another possible interpretation is to dynamically alter Θ after each iteration, using the previous solution as the reference r^*. For example, some OBL-based algorithm has solution $s^t = 1$ at time t, then after the algorithm applies some perturbation method to generate $s^{t+1} = 1.5$. Then, we can modify our opposite map such that $\breve{s}^{t+1} = 0.5$, resulting in values of $f(1.5) = -2.8103$ and $f(0.5) = 19.0$. So, without using opposition we have a 50% chance of selecting $f(1.5)$, whereas using OBL it is a guarantee. In either case, it is required that the opposite map definition is strictly adhered to. This is a necessary condition to ensure a lower expected value of the evaluation function than otherwise.

Theorem 1: For any evaluation function $f: S \rightarrow \Re$ over solution space S and opposite evaluation function $\Phi: S \rightarrow \Re$, $\Phi(s)$ yields at worst case an equal expected value to $f(s)$. That is, for a minimization problem $\mathbb{E}[\Phi(s)] \leq \mathbb{E}[f(s)]$, and for a maximization problem, $\mathbb{E}[\Phi(s)] \geq \mathbb{E}[f(s)]$.

Proof: The proof is presented without loss of generality for a minimization problem where each solution has 1 opposite and follows directly from the definition of expected value of continuous variables.

$$\begin{aligned} \mathbb{E}[f(s)] &= \int_S f(s) Pr(s) ds \\ \mathbb{E}[\Phi(s)] &= \int_S \Phi(s) Pr(s) ds \\ &= \int_S \min(f(s), f(\breve{s})) Pr(s) ds \end{aligned}$$

Since $\min(f(s), f(\breve{s})) \leq f(s)\ \forall s \in S$ then it must follow that $\mathbb{E}[\Phi(s)] \leq \mathbb{E}[f(s)]$. ■

Note that taking the minimum of two purely random solutions r_1, r_2 does not lead to much improvement over traditional random sampling since the $\mathbb{E}[r_1] = \mathbb{E}[r_2]$. While, the extra guess does increase the probability of discovering a better solution it is not guaranteed. More importantly, r_1, r_2 are not dependant on each other to exist. That is for a given r_1, r_2 can take a different value each time it is evaluated.

Also, the situation where $\mathbb{E}[\Phi(s)] = \mathbb{E}[f(s)]$ occurs is when the difference between all pairs of elements for each $\vartheta_j \in \Theta$ is zero, i.e. $\forall\ \Theta$, $|s_i - s_j| = 0\ \forall s_i, s_j\ \in\ \vartheta_j$. For this to occur, the opposite map must be a perfectly symmetric function, which is very highly improbable for real world problems.

Of course for a given problem many possible opposite maps (which directly influence the opposite evaluation function) may exist, the best of which will provide the lowest expected evaluation. However, defining the optimal mapping is not required to observe beneficial results.

III. Opposition based Simulated Annealing

Simulated annealing (SA) is a stochastic global search technique developed during the early 1980's [11], [12]. This section will present a modified version of the SA algorithm based on opposition-based learning.

Algorithm 1 presents this new opposition-based simulated annealing (OSA) approach. Much of the algorithm is identical to traditional SA, with the exception of lines 14-21, which implement opposition. The function $randomGuess()$ generates and returns a random solution; $evaluate(Guess)$ evaluates the quality of the given $Guess$; $neighbor(Guess)$ returns a neighbor of the passed $Guess$; $\mathcal{U}(0,1)$ generates a uniform random value between 0 and 1, and $update(Temp)$ calculates the cooling schedule, returning the new temperature.

The function $opposite(N, Guess)$ generates a guess that is opposite to neighbor N with respect to $Guess$, effectively creating an *opposite neighbor*. Thus, we are utilizing a temporal version of OBL, as described in Section II. The reasoning for a time-dependant frame of reference over a constant one is due to an assumption of the search space and behavior of SA. Specifically, a search algorithm tends to discover regions of the space where high quality solutions reside. Additionally we assume that if we are currently exploring one of these regions, then the probability of a higher quality solution existing on the exact opposite side of the search space is small.

Lines 14-21 define the use of opposition in the OSA algorithm (the lines are italicized). In line 14 we probabilistically consider to take into account the opposite neighbor. This probability decreases over time, proportional to the number of iterations completed and the constant K. This accounts for the fact that opposition tends to be more beneficial during early stages of learning than later on [1]. Lines 15 and 16 generate and evaluate the opposite guess with respect to the current guess, respectively. In lines 17-20 we decide to choose the best of the two neighbors.

IV. Test Functions

In order to test the quality of the OSA approach versus traditional SA we have utilized five common real optimization functions. Each of these is a minimization problem.

The sphere problem [13] is defined as

[1]Recall that opposition is embedded into an existing algorithm. Initially an algorithm is unbiased as to what solutions it guesses and so the influence of opposition is potentially large. However, as the number of iterations increases focus of the algorithm changes, resulting in a smaller range of values under consideration. So, the influence of opposition is likely small. If the algorithm did behave in this manner, it would be purely random.

Algorithm 1 Opposition-based Simulated Annealing (OSA)

Require: $K := constant$
1: {Initial Conditions}
2: $CurGuess \Leftarrow randomGuess()$
3: $BestGuess \Leftarrow Guess$
4: $CurScore \Leftarrow evaluate(CurGuess)$
5: $BestScore \Leftarrow CurScore$
6: $Temp \Leftarrow BestScore$
7: $iteration \Leftarrow 0$
8:
9: **while** termination criteria not satisfied **do**
10: $N \Leftarrow neighbor(CurGuess)$
11: $Ne \Leftarrow evaluate(N)$
12:
13: {*Check opposite neighbor*}
14: ***if*** $\mathcal{U}(0,1) \leq e^{-iteration/K}$ ***then***
15: $O \Leftarrow opposite(N, CurGuess)$
16: $Oe \Leftarrow evaluate(O)$
17: ***if*** $Oe \leq Ne$ ***then***
18: $N \Leftarrow 0$
19: $Ne \Leftarrow Oe$
20: ***end if***
21: ***end if***
22:
23: {Found new best?}
24: **if** $Ne \leq BestScore$ **then**
25: $BestGuess \Leftarrow N$
26: $BestScore \Leftarrow Ne$
27: **end if**
28:
29: {Do we accept N?}
30: **if** $\mathcal{U}(0,1) \leq e^{(CurScore-Ne)/Temp)}$ **then**
31: $CurGuess \Leftarrow N$
32: $CurScore \Leftarrow Ne$
33: **end if**
34:
35: $iteration \Leftarrow iteration + 1$
36: $Temp \Leftarrow update(Temp)$
37: **end while**

$$sphere(\mathbf{x}) = \sum_{i=1}^{n} {x_i}^2 \tag{1}$$

where $-5.12 \leq x_i \leq 5.12$. The minimum of this function is 0 and occurs at $\mathbf{x} = 0, ..., 0$.

Rosenbrock's Valley function [13] is evaluated according to

$$rosenbrock(\mathbf{x}) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2] \tag{2}$$

where $-2 \leq x_i \leq 2$. This function has a minimum at $\mathbf{x} = 1, ...1$, with a corresponding value of 0.

A solution to Rastrigin's function [14] shown in equation (3) is defined over $-5.12 \leq x_i \leq 5.12$. The optimal value of

this function is $rastrigin(0, ..., 0) = 0$.

$$rastrigin(\mathbf{x}) = 10n + \sum_{i=1}^{n}(x_i^2 - 10\cos(2\pi x_i)) \quad (3)$$

We also have utilized a Schwefel function [15] (equation 4) which is bounded according to $-10 \leq x_i \leq 10$. The function has a minimum at $schwefel(0, ..., 0) == 0$.

$$schwefel(\mathbf{x}) = \sum_{i=1}^{n}|x_i| + \prod_{i=1}^{n}|x_i| \quad (4)$$

The Alpine function is calculated according to

$$alpine(\mathbf{x}) = \sum_{i=1}^{n}|x_i \sin(x_i) + 0.1x_i| \quad (5)$$

where $-10 \leq x_i \leq 10$. The minimum of this function occurs at $alpine(0, ..., 0) = 0$.

The final function we have utilized is De Jong's noiseless function #4 [13], which is defined as

$$dejong(\mathbf{x}) = \sum_{i=1}^{n} i{x_i}^4 \quad (6)$$

where $-1.28 \leq x_i \leq 1.28$. The corresponding minimum value is found at $dejong(0, ..., 0) = 0$.

V. Experimental Setup

This section will describe the behavior of the functions needed for OSA to operate, as described in Section III. Specifically, we will outline the $randomGuess()$, $neighbor(Guess)$, $update(Temp)$ and $opposite(N, Guess)$ functions.

A. Generating Guesses

Both the SA and OSA algorithms commence by generating a random starting solution via the $randomGuess()$ function. This is accomplished by randomly selecting a uniform value for each variable of an n-dimensional solution. That is, $x_i = \mathcal{U}(L, H)$, where L and H are the bounds of each variable as described in Section IV.

B. Neighborhood Function

A neighbor of some current guess $CurGuess$ is generated using the $neighbor(Guess)$ function. Specifically, we select m of the n variables in $CurGuess$ and add a uniform random value $\pm\Delta$ where $\Delta = (L + H)/15$, where L and H represent the lower and upper bound of the current problem, respectively. This operator is not optimal, but performs well on each test function.

C. Temperature Update

Fundamental to the operation of simulated annealing is the temperature update function, $update(Temp)$. For these experiments we have empirically decided to update the temperature according to

$$Temp^i = \alpha \cdot Temp^{i-1} \quad (7)$$

for iteration i. Initially, we set the decay constant $\alpha = 0.95$.

D. Determining Opposites

As mentioned in Section III, we are utilizing a dynamic reference point in determining opposites, which is accomplished by the $opposite(N, Guess)$ function. Here, N is a neighbor of $Guess$ as generated by the $neighbor(Guess)$ function. An opposite of N with respect to $Guess$ is a point $\breve{N}$ directly opposite to N. We can summarize this as

$$\breve{N} = Guess_i \pm \breve{\Delta} \quad (8)$$

where $\breve{\Delta}$ represents the opposite value added to the i^{th} variable of $Guess$ to arrive at N by the $neighbor$ function. For example, if the neighbor function adds a value of 0.1 to $Guess$ to arrive at N, then $\breve{N} = Guess - 0.1$.

VI. Experimental Results

This section describes the experimental results using the test functions described in Section IV. The results that we present have been taken from runs of length 5000 and averaged over 250 runs. For experiments using the OSA algorithm, the constant $K = 500$ was empirically decided, but not optimal. This resulted in an additional 500 calls to the $evaluate(Guess)$ function.

Our aim in these experiments is to examine the influence of dimensionality and neighborhood function on the performance of the OSA and SA algorithms in order to determine any significant differences. Additionally, we confirm the influence of opposition by considering a second random neighbor instead of an opposite one.

A. Dimensionality

Here, we vary the dimensionality of each benchmark function in order to determine the sensitivity of OSA to various sized problems. We will compare the results to those found with traditional simulated annealing. The neighbor function will alter only 1 variable as is described in Section V for these experiments. The results for varying the problem dimensionality size of $\{10, 25, 50, 100\}$ are presented in Table I.

The results found using OSA show a much lower expected value (μ) and standard deviation (σ) than with SA. In each case the results of the OSA approach are more desirable than those found with SA. For the 24 experiments 19 were better using OSA, and 5 were equal (for ¿ 3 decimal places OSA outperforms SA on all the experiments).

We tested the null hypothesis that the two means are equal using a t-test with a 0.95 confidence. From the results we find that all results except for rosenbrock with 10 dimensions are statistically significant. Additionally, the results found by OSA exhibited a lower standard deviation, meaning that the results are more reliable.

Figure 2 shows the influence of dimensionality on convergence of the OSA algorithm for the Schwefel function. Increasing the dimensionality has the typical effect of requiring more computation time for the algorithm to converge. The behavior of the convergence curve is also very similar as the dimensionality increases.

TABLE I: Results for experiments on dimensionality with fixed variable change=1. The bolded values represent a statistically significant results at 0.95 confidence.

		SA		OSA	
	dim	μ	σ	μ	σ
sphere	10	0.000	0.000	0.000	0.000
	25	0.001	0.000	0.001	0.000
	50	0.007	0.002	**0.005**	0.002
	100	0.269	0.449	**0.151**	0.487
rosenbrock	10	7.678	6.898	**7.674**	6.890
	25	38.218	25.779	**35.465**	23.910
	50	89.842	42.118	**87.762**	41.673
	100	263.658	66.382	**240.794**	60.877
rastrigin	10	82.371	24.925	**63.258**	22.654
	25	215.611	38.620	**182.938**	36.001
	50	422.267	56.710	**390.629**	54.155
	100	849.204	77.428	**806.422**	76.165
schwefel	10	0.028	0.009	**0.025**	0.009
	25	0.178	0.036	**0.159**	0.033
	50	0.793	0.114	**0.692**	0.093
	100	4.633	1.144	**3.550**	0.572
alpine	10	0.128	0.056	**0.099**	0.047
	25	0.798	0.225	**0.682**	0.225
	50	3.080	0.614	**2.809**	0.512
	100	12.789	1.935	**11.139**	1.674
dejong	10	0.000	0.000	0.000	0.000
	25	0.000	0.000	0.000	0.000
	50	0.000	0.000	0.000	0.000
	100	0.138	0.991	**0.019**	0.143

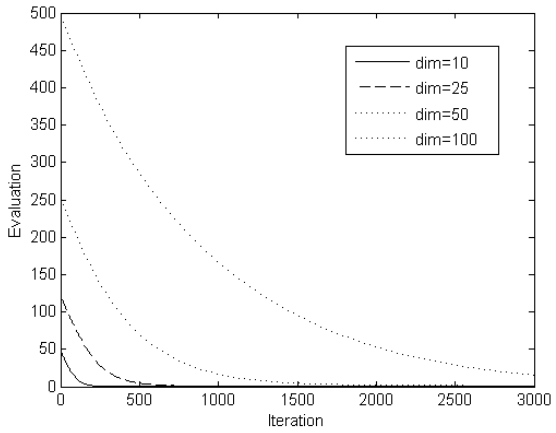

Fig. 2: Influence of dimensionality on convergence of OSA on the Schwefel function.

A comparison of the convergence of the OSA and SA algorithms on the 25-dimensional De Jong function is presented in Figure 3. Although both approaches eventually achieve a similar final result, the OSA approach exhibits a much more rapid convergence. This characteristic curve is can also be seen above in Figure 2 for the Schwefel function.

B. Neighborhood

This experiment is aimed to discover the degree to which the neighborhood function influences the outcome. We ran the experiments for a fixed dimensionality of 100 and varied the number variables that are altered by the $neighbor$ function from $\{1, 3, 5\}$.

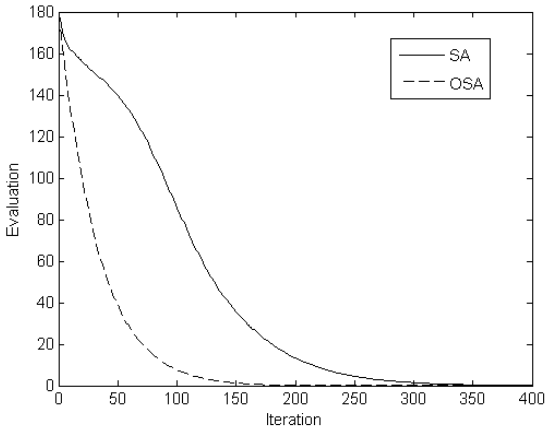

Fig. 3: Convergence of OSA versus SA on the De Jong function of dimensionality 25.

TABLE II: Results for experiments on neighborhood with fixed dimensionality = 100. The bolded values represent a statistically significant results at 0.95 confidence.

		SA		OSA	
	dim	μ	σ	μ	σ
sphere	1	0.269	0.449	**0.151**	0.487
	3	0.909	0.111	**0.818**	0.094
	5	1.846	0.187	**1.727**	0.177
rosenbrock	1	263.658	66.382	**240.794**	63.877
	3	190.613	62.386	**181.011**	54.081
	5	202.753	56.991	**196.079**	59.565
rastrigin	1	849.204	77.428	**806.422**	76.165
	3	882.186	76.033	**832.605**	79.160
	5	957.445	72.301	**899.624**	73.026
schwefel	1	4.633	1.144	**3.550**	0.572
	3	14.018	1.091	**13.095**	0.991
	5	20.406	1.334	**19.428**	1.318
alpine	1	12.789	1.935	**11.139**	1.674
	3	35.514	4.500	**31.862**	4.192
	5	55.293	6.538	**49.644**	5.989
dejong	1	0.138	0.991	**0.019**	0.143
	3	0.004	0.001	**0.003**	0.001
	5	0.017	0.003	**0.015**	0.003

As with the dimensionality experiments, the average OSA outcomes are relatively low compared to those achieved with SA. In fact all results obtained using the OSA algorithm were more desirable than those reached using SA. A t-test with a 0.95 confidence level confirmed that all the results are indeed statistically significant. Furthermore, the standard deviations of results found via the OSA algorithm are also relatively low, and so the results are more reliable.

The influence of the number of variables altered at each call of the $neighbor(Guess)$ function can be seen in Figure 4 for the Alpine function. When the $neighbor$ function only perturbs a single variable the algorithm converges more rapidly than when more variables are considered. This is reasonable since the likelihood of escaping a local optima is lower if the operator makes only a small change to the solution. Nevertheless, the behavior of the convergence curve is relatively similar.

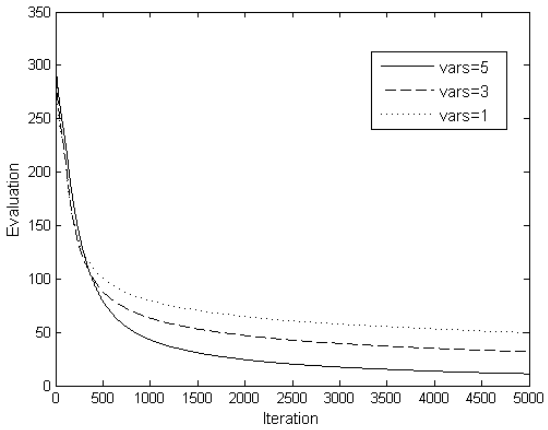

Fig. 4: Influence of neighbor generation on convergence of 100-dimensional OSA for the Alpine function.

C. Randomness

As described by Yao [16] increasing the neighborhood size corresponds to a higher probability of arriving at a global minimum. So, to examine whether the observed improvements are a result of opposition or just increased neighborhood size we compare them against the results obtained by a randomized version of OSA. If the results obtained by OSA are of higher quality, then the results are not simply due to a larger neighborhood. This is simply accomplished by replacing line 15 in Algorithm 1 with $neighbor(CurGuess)$, which effectively generates two random, independent neighbors. The randomized version of the OSA algorithm is referred to as RSA below.

The results for this experiment are presented in Table III where we fixed the dimensionality at 100. We also vary the number of variables altered by each call to the $neighbor$ function as above.

TABLE III: Comparing OSA vs RSA, fixed dimension=100

		RSA		OSA	
	dim	μ	σ	μ	σ
sphere	1	0.220	0.530	**0.151**	0.487
	3	0.826	0.096	**0.818**	0.094
	5	1.787	0.185	**1.727**	0.177
rosenbrock	1	**230.556**	63.818	240.794	63.877
	3	182.201	54.387	**181.011**	54.081
	5	**192.794**	49.155	196.079	59.565
rastrigin	1	843.953	78.029	**806.422**	76.165
	3	864.800	71.285	**832.605**	79.160
	5	934.826	77.407	**899.624**	73.026
schwefel	1	3.890	0.858	**3.550**	0.572
	3	13.249	0.983	**13.095**	0.991
	5	19.522	1.365	**19.428**	1.318
alpine	1	11.508	1.447	**11.139**	1.674
	3	33.430	4.440	**31.862**	4.192
	5	52.503	6.299	**49.644**	5.989
dejong	1	0.024	0.224	**0.019**	0.143
	3	0.003	0.001	0.003	0.001
	5	0.015	0.003	0.015	0.003

Most of the final results obtained with OSA are relatively lower than those obtained with RSA. In total 14/18=78% results favored OSA, 2/18=11% for RSA and 2/18=11% were equal. The $rosenbrock$ experiments yielded a slightly more favorable result for the RSA algorithm. However, according to a t-test at a 0.95 confidence level all but the $dejong$-5 and rosenbrock-3 experiments exhibit a statistically significant difference in μ. Also, as with the previous experiments, the σ values are lower, indicating a more reliable result. The average number of function calls to the $evaluate(Guess)$ function is the same for both techniques.

D. Summary

The above experiments provide evidence to support the theoretical expectation that opposition-based algorithms tend to achieve a more desirable accuracy than those without. The experiments considered both problem dimensionality and neighbor generating functions to arrive at this conclusion. As was also briefly discussed in Section II, OBL algorithms should also converge at a more rapid rate. This behavior was also observed in our experimentation.

The efficacy of opposite neighbors was also tested against random neighbor generation. We find that in nearly all cases, OSA outperformed the RSA algorithm in terms of final accuracy. Although, the RSA approach was able to also improve on the vanilla version of SA. Convergence of OSA and RSA was not significantly different, perhaps as a result of the respective benchmark problem difficulties.

Figure 5 shows a plot of the results of the 100 dimensional Rastrigin function with 5-variable neighbor generation. In this case, the convergence trajectories are similar for each approach since the dynamics of the underlying search process have not been drastically altered. However, this is another example of the improved accuracy that can be achieved with OBL.

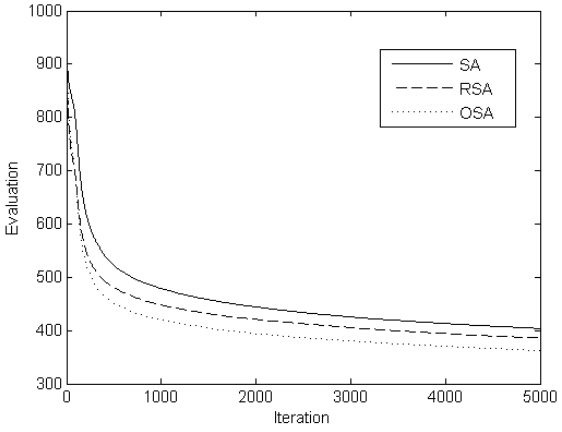

Fig. 5: Comparison of convergence for OSA, RSA and SA on the Rastrigin function with dimension=100 and 5-variable neighbor generation.

Another important result that was discovered concerns the reliability of the final results. That is, the OSA algorithms was able to not only achieve a more desirable result, but also

do so with less standard deviation. This means that the OSA algorithm tends to yield more reliable results than either the RSA or SA approaches.

VII. Conclusions and Future Work

We have proposed a modification to the simulated annealing algorithm based on the concept of opposition-based learning. We first provided theoretical reasoning behind OBL, and proved that it's associated opposite evaluation function is more desirable with respect to expected value than originally. From these theoretical properties of OBL the new algorithm, termed OSA, should be able to achieve more accurate results than traditional SA. We conducted experiments varying the dimensionality and neighbor generation to confirm this. Our experimentation also revealed that the convergence of OSA was at a higher rate than SA, and that the final results exhibited less variance.

Additionally, we conducted experiments aimed at showing the benefit of OSA over a random version, RSA. From these results it was determined that opposition indeed did yield more desirable results, which was also predicted theoretically.

Since the concept of OBL is in its infancy, there is a need to further study its theoretical properties. In this direction we may be able to provide recommendations regarding opposite maps, as well as how to best incorporate OBL into existing algorithms. It is possible that this could also lead to the development of a new search algorithm. Other future work involves examination of a wider array of benchmark and practical applications. Also, exploration of possible methods to extend existing algorithms using the concept of opposition is an important research direction.

References

[1] J. D. Pinter, *Global Optimization: Scientific and Engineering Case Studies*. Springer, 2006.

[2] P. J. M. Van Laarhoven and E. H. L. Aarts, *Simulated Annealing: Theory and Applications*. Springer, 1987.

[3] H. R. Tizhoosh, "Opposition-based Learning: A New Scheme for Machine Intelligence," in *International Conference on Computational Intelligence for Modelling, Control and Automation*, 2005.

[4] S. Rahnamayn, H. R. Tizhoosh, and S. Salama, "A Novel Population Initialization Method for Accelerating Evolutionary Algorithms," *(to appear) Computers and Mathematics with Applications*, 2006.

[5] S. Rahnamayn, H. R. Tizhoosh, and S. Salama, "Opposition-based Differential Evolution Algorithms," in *IEEE Congress on Evolutionary Computation*, pp. 7363–7370, 2006.

[6] S. Rahnamayn, H. R. Tizhoosh, and S. Salama, "Opposition-based Differential Evolution Algorithms for Optimization of Noisy Problems," in *IEEE Congress on Evolutionary Computation*, pp. 6756–6763, 2006.

[7] M. Shokri, H. R. Tizhoosh, and M. Kamel, "Opposition-based Q(λ) Algorithm," in *IEEE International Joint Conference on Neural Networks*, pp. 646–653, 2006.

[8] H. R. Tizhoosh, "Opposition-based Reinforcement Learning," *Journal of Advanced Computational Intelligence and Intelligent Informatics*, vol. 10, no. 4, pp. 578–585, 2006.

[9] H. R. Tizhoosh, "Reinforcement Learning Based on Actions and Opposite Actions," in *International Conference on Artificial Intelligence and Machine Learning*, 2005.

[10] M. Ventresca and H. R. Tizhoosh, "Improving the Convergence of Backpropagation by Opposite Transfer Functions," in *IEEE International Joint Conference on Neural Networks*, pp. 9527–9534, 2006.

[11] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by Simulated Annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[12] V. Cerney, "A Thermodynamical Approach to the Travelling Salesman Problem: an Efficient Simulation Algorithm," *Journal of Optimization Theory and Applications*, vol. 45, pp. 41–51, 1985.

[13] K. A. De Jong, *An Analysis of the Behavior of a class of Genetic Adaptive Systems*. PhD thesis, University of Michigan, 1975.

[14] D. Mhlenbein, H. Schomisch, and J. Born, "The Parallel Genetic Algorithm as Function Optimizer," *Parallel Computing*, pp. 619–632, 1991.

[15] J. P. Schwefel, *Evolution and Optimum Seeking*. John Wiley & Sons, 1995.

[16] X. Yao, "Simulated Annealing with Extended Neighbourhood," *International Journal of Computer Mathematics*, vol. 40, pp. 169–189, 1991.