The No Free Lunch Theorem

Does Not Apply to Continuous Optimization

George I. Evers

george@georgeevers.org
Independent Consultant

Abstract

After deriving the particle swarm equations from basic physics, this paper shows by contradiction that NFL Theorem 1, and consequently Theorems 2 and 3, are irrelevant to continuous optimization. As the discrete nature of matter at its most fundamental level is generally irrelevant from the broader classical perspective, so to is the discrete nature of an optimization problem at its most fundamental level generally irrelevant from the broader continuous perspective.

Key words

No free lunch, NFL, disproof

1 Introduction

Particle swarm optimization (PSO) is derived from the kinematic equation for translational particle motion to shed light on the conceptual underpinnings of the algorithm. The derivation supports the counterexample of the second section by clarifying that if the static value of $1/2$ from the position update equation of physics is utilized in lieu of stochastic numbers, the counterexample maintains its integrity, which shows that the counterargument applies equally well to the static case.

1.1 Derivation of the PSO Equations

University students of the mathematics and physics are familiar with the kinematic equation by which a particle's final position vector can be calculated from its initial position and velocity if acceleration is constant over the time period:

$$\vec{x} = \vec{x}_0 + \vec{v}_0 t + \frac{1}{2}\vec{a}t^2 . \tag{1}$$

For PSO, the "time" between position updates is 1 iteration; hence, t and the corresponding dimensional analysis can be dropped to simplify iterative computations. Instead, authors generally use k to denote values of the current iteration and $k+1$ for values of the ensuing iteration. Using this notation, the basic position update equation of physics can be rewritten for iterative computation as

$$\vec{x}(k+1) = \vec{x}(k) + \vec{v}(k) + \frac{1}{2}\vec{a}(k) . \tag{2}$$

Since PSO was jointly created by a social psychologist and an electrical engineer, it incorporates very simple concepts from both disciplines. Via shared computer memory, particles model social communication by iteratively accelerating toward the *global best*, which is the best solution found by the swarm through the current iteration, k. Modeling cognition, each particle also iteratively accelerates toward its *personal best*, which is the best solution that it personally has found.

The cognitive and social acceleration constants, c_1 and c_2 respectively, determine how aggressively particles accelerate based on the cognitive and social information available to them: these

can be set identically, or a preference can be given to either acceleration type [1]. The subtraction in Expression (3) ensures that any particle which is distant from the social best will accelerate toward it more strongly than a particle nearby. Similarly, the subtraction in Expression (4) ensures that any particle that is distant from its cognitive best will accelerate toward it more strongly than were it nearby. As a conceptual example, one could imagine children playing both indoors and out when their mother announces that dinner is ready: those farther away might be expected to run more quickly toward the dinner table.

$$\begin{aligned}&\text{Social acceleration}: c_2\left(\vec{g}(k)-\vec{x}_i(k)\right)\\&\text{where:}\\&c_2 \text{ is the social acceleration constant,}\\&\vec{x}_i(k) \text{ is the position vector of particle "}i\text{"}\\&\quad \text{at iteration "}k\text{",}\\&\vec{g}(k) \text{ is the global best of all particles at}\\&\quad \text{iteration "}k\text{"}\end{aligned} \tag{3}$$

$$\begin{aligned}&\text{Cognitive acceleration}: c_1\left(\vec{p}_i(k)-\vec{x}_i(k)\right)\\&\text{where:}\\&c_1 \text{ is the cognitive acceleration constant,}\\&\vec{x}_i(k) \text{ is the position vector of particle "}i\text{"}\\&\quad \text{at iteration "}k\text{",}\\&\vec{p}_i(k) \text{ is the personal best of particle "}i\text{"}\\&\quad \text{at iteration "}k\text{"}\end{aligned} \tag{4}$$

Expression (3) defines the social acceleration of Global Best (Gbest) PSO. Local Best (Lbest) PSO limits each particle's social sphere to knowledge of the best solution found by its neighbors instead of immediately granting each particle knowledge of the best solution found so far by the entire search team.

Substituting the sum of these two acceleration terms for $\vec{a}(k)$ in Equation (2), while applying the subscript i adopted in (3) and (4), produces Equation (5).

$$\vec{x}_i(k+1)=\vec{x}_i(k)+\vec{v}_i(k)+\frac{1}{2}c_1\left(\vec{p}_i(k)-\vec{x}_i(k)\right)+\frac{1}{2}c_2\left(\vec{g}(k)-\vec{x}_i(k)\right) \tag{5}$$

Having replaced physical acceleration in the position update equation of physics with social and cognitive modeling, the next step toward producing a stochastic search algorithm is the replacement of $1/2$ with a pseudo-random number sampled per dimension from the uniform distribution between 0 and 1, $U(0,1)$. Note that the expected or mean value of the distribution is still $1/2$. Designating the first vector of pseudo-random numbers as $\vec{r}_{1i}$ and the second as $\vec{r}_{2i}$ produces Equation (6).

$$\begin{aligned}&\vec{x}_i(k+1)=\vec{x}_i(k)+\vec{v}_i(k)+c_1\vec{r}_{1,i}\circ\left(\vec{p}_i(k)-\vec{x}_i(k)\right)+c_2\vec{r}_{2,i}\circ\left(\vec{g}(k)-\vec{x}_i(k)\right)\\&\text{where "}\circ\text{" denotes the Hadamard or "element-wise" product}\end{aligned} \tag{6}$$

For convenience, the rather long Equation (6) is separated into a velocity update equation (7) and a position update equation (8). This primarily helps with record keeping since each value can be stored separately for post-simulation analysis. Substituting Equation (7) into (8) shows equivalency to (6).

$$\vec{v}_i(k+1)=\vec{v}_i(k)+c_1\vec{r}_{1,i}\circ\left(\vec{p}_i(k)-\vec{x}_i(k)\right)+c_2\vec{r}_{2,i}\circ\left(\vec{g}(k)-\vec{x}_i(k)\right) \tag{7}$$

$$\vec{x}_i(k+1)=\vec{x}_i(k)+\vec{v}_i(k+1) \tag{8}$$

Since its conception, Equation (7) has developed two mechanisms by which to improve search behavior. The inertia weight, ω, roughly simulates friction in a computationally inexpensive manner

by carrying over to the next iteration only a user-specified percentage of the current iteration's velocity. This is done by multiplying the velocity of the current iteration by $\omega \in (-1,1)$ [1] as shown in the first term of Equation (9) [2]. The constriction models use a constriction coefficient instead [3], but the popular Type 1" parameters can be converted to Clerc's Equivalents for use in Equation (9) [4].

$$\vec{v}_i(k+1) = \omega \vec{v}_i(k) + c_1 \vec{r}_{1,i} \circ (\vec{p}_i(k) - \vec{x}_i(k)) + c_2 \vec{r}_{2,i} \circ (\vec{g}(k) - \vec{x}_i(k)) \quad (9)$$

The other restriction imposed on velocity is essentially a speed limit [5]. Rather than limiting the vector magnitude itself, the computationally simpler approach of limiting each component is implemented as shown in Equation (10), which limits the magnitude indirectly.

$$v_{i,j}(k+1) = \text{sign}(v_{i,j}(k+1)) \times \max(|v_{i,j}(k+1)|, v_j^{\max})$$
$$\text{where } j \in \{1, 2, ..., n-1, n\}, \text{and} \quad (10)$$
$$n \text{ denotes the problem dimensionality}$$

This limits the maximum step size on dimension j by clamping: (i) values greater than $v_j^{\max}$ to a maximum value of $v_j^{\max}$, and (ii) values less than $-v_j^{\max}$ to a minimum of $-v_j^{\max}$. From a physical perspective, particles with clamped velocities are analogous to birds with limited flying speeds. Considering the psychological aspects of the algorithm, clamped velocities could also be considered analogous to self-limited emotive responses.

Concerning the calculation of $v_j^{\max}$, suppose the feasible candidates for an application problem are [12, 20] on some dimension to be optimized. Clamping velocities to 50%, for example, of $x_{\max}$ would allow particles to take excessively large steps relative to the range of the search space on that dimension; in this case, the maximum step size would be $0.5 \times 20 = 10$. But stepping 10 units in any direction when the search space is only 8 units wide would be nonsensical. Since real-world applications are not necessarily centered at the origin of Euclidean space, it is preferable to clamp velocities based on the range of the search space per dimension in order to remove dependence on the frame of reference [10]; hence, subscript j is included in Equation (10) for sake of generality; but it can be dropped for applications with the same range of values per dimension.

1.2 Swarm Initialization

The optimization process begins by randomly initializing positions between a minimum and maximum per dimension as per Relation (11). The most common benchmarks use the same minimum and maximum per dimension. For application problems, however, these might differ depending on the characteristics being optimized; hence, the general formula is provided, which uses subscript j to indicate the dimension.

$$x_{i,j}(k=0) \in U(x_j^{\min}, x_j^{\max}) \quad (11)$$

Velocities are similarly initialized according to Relation (12). For application problems with a different range of feasible values on one dimension than on another, different step sizes per dimension would make sense; hence, the general form is presented, which avoids unnecessarily imposing the same range of feasible values on all characteristics to be optimized.

$$v_{i,j}(k=0) \in U(-v_j^{\max}, v_j^{\max}) \quad (12)$$

Each particle's personal best is initialized to its starting position as shown in Equation (13).

$$\vec{p}_i(k=0) = \vec{x}_i(k=0) \quad (13)$$

1 ω is often selected to lie within (0, 1), but a small negative value models trust and distrust quite effectively [4].

The global best is always the best of all personal bests as shown in Equation (14).

$$\vec{g}(k) = \arg\min_{\forall p_i(k)} f\left(p_i(k)\right) \tag{14}$$

1.3 Iterative Optimization Routine

Once the swarm has been initialized, particles iteratively: (i) accelerate (i.e. adjust their velocity vectors) toward the global best and their own personal bests, (ii) update and clamp their velocities, (iii) update their positions, and (iv) update their personal bests and the global best. This routine is repeated until reaching a user-specified termination criterion.

For convenience, the relevant equations are restated below as needed in order of implementation.

$$\vec{v}_i(k+1) = \omega\vec{v}_i(k) + c_1\vec{r}_{1,i} \circ \left(\vec{p}_i(k) - \vec{x}_i(k)\right) + c_2\vec{r}_{2,i} \circ \left(\vec{g}(k) - \vec{x}_i(k)\right) \tag{9}$$

$$v_{i,j}(k+1) = \text{sign}\left(v_{i,j}(k+1)\right) \times \max\left(\left|v_{i,j}(k+1)\right|, v_j^{\max}\right) \tag{10}$$

$$\vec{x}_i(k+1) = \vec{x}_i(k) + \vec{v}_i(k+1) \tag{8}$$

A particle's personal best is only updated when the new position offers a better function value:

$$\vec{p}_i(k+1) = \begin{cases} \vec{x}_i(k+1) \text{if } f\left(\vec{x}_i(k+1)\right) < f\left(\vec{p}_i(k)\right) \\ \vec{p}_i(k) \text{otherwise} \end{cases}. \tag{15}$$

The global best is always the best of all personal bests:

$$\vec{g}(k+1) = \arg\min_{\forall p_i(k)} f\left(p_i(k+1)\right). \tag{14}$$

Rather than accelerating due to external physical forces, particles adjust toward solutions of relative quality. Each position encountered as particles swarm is evaluated and compared to existing bests. Though the behavior of each individual is simple, the collective result is an optimization algorithm capable of maximizing or minimizing problems that would be difficult to tackle with straightforward mathematical analyses, either because the problem is not well understood in advance or simply because the problem is quite complicated.

1.4 The No Free Lunch (NFL) Theorems

The NFL Theorems [6, 7, 8] essentially state that any pair of optimizers must perform equivalently across the set of all problems, which implies that the attempt to design a general purpose optimizer is necessarily futile. The theorems are well established and have even become the basis for a book that attempts to draw biological inferences from them [9]; consequently, it becomes useful for the validity of the theorems to be closely scrutinized.

One could hardly question the productivity of designing for a particular purpose and clearly communicating the intended problem class, yet the assertion that no algorithm can outperform any other on the whole is disconcerting since it implies that no true general purpose optimizer can exist. If physical tools are analogous to algorithmic tools, then for Algorithm A to be no more effective across the set of all problems than Algorithm B would imply that no rope, wheel, or pulley could be more effective across the set of all problems than a shoe horn; yet pulleys clearly have more applications than shoe horns, which draws the No Free Lunch concept into question. Even comparing shoe horns to shoe horns, to design one of such width as to be practically useless would be a trivial matter; this too is incongruous with the No Free Lunch concept since the mean performance of a shoe horn that is 20 cm wide could hardly match the mean performance of a shoe horn that is 5 cm wide.

Disproof is often simpler than proof since a counterexample suffices. Subsection 2.2 demonstrates via counterexample that the first NFL Theorem is too strong to be true, which also invalidates the corollary that only via problem-specific knowledge can an algorithm outperform randomness [8].

Were algorithms only written for specific problem types, each problem would need to be classifiable in advance unless multiple algorithms were to be randomly applied in series until

producing a quality solution, which would be quite an inefficient approach. Since modern optimization algorithms can operate on systems capable of mapping inputs to output(s) "black box" style, a quality general purpose optimizer, were such a thing possible, would offer an increased probability of solving problems that are not classifiable in advance.

Furthermore, accidental misclassification of a problem would likely produce a relatively low-quality solution. A quality general purpose optimizer would therefore also be useful for checking answers produced by more problem-specific tools. So the question is, "Can one optimizer produce better mean performance than another in the long run?"

2 The NFL Theorems Do not Pertain to Continuous Optimization

The first NFL Theorem is arguably the most important to address since its notion of "a-independence" is the seed from which the tree of NFL Theorems has sprung. It is quoted below for convenience.

"Theorem 1: For any pair of algorithms a_1 and a_2

$$\sum_f P\left(d_m^y \mid f, m, a_1\right) = \sum_f P\left(d_m^y \mid f, m, a_2\right)."$$

where:

$f : X \to Y$ is an element of F, which denotes the space of all possible problems,

a_1 and a_2 are the algorithms being compared,

m is the number of iterations allotted to the comparison, assuming no revisitation [7] (i.e. the number of unique cost evaluations divided by the population size) (16)

d_m^y denotes the "ordered set of cost values"

$P\left(d_m^y \mid f, m, a\right)$ measures "the performance of an algorithm, a, iterated m times on a cost function, f"

Theorem 1 only asserts to be true for deterministic algorithms. While PSO is stochastic in the sense that it uses pseudo-random numbers, the counterexample of Subsection 2.2 works equally well with the static $1/2$ of the traditional physics formula replacing the stochastic random number vectors, $\vec{r}_{1,i}$ and $\vec{r}_{2,i}$. Following this substitution, only the randomized starting locations would be stochastic; however, initial positions can be set identically for both algorithms either via any non-stochastic initialization scheme or via identical seeding of the randomizer. Consequently, the pseudo-stochastic nature of particle swarm does not exempt it from the NFL Theorems. Since the pair of algorithms presented as a counterexample to NFL Theorem 1 in Subsection 2.2 presumes the seed of the randomizer to be set identically as a practical matter, all starting positions are identical; hence, removal of the stochasm is unnecessary, though it would be straightforward to accomplish.

This perspective is consistent with Wolpert's statement, "A search algorithm is deterministic; every sample maps to a unique new point. Of course, essentially, all algorithms implemented on computers are deterministic, and in this our definition is not restrictive" [7]. In a footnote, Wolpert further clarifies, "In particular, note that pseudorandom number generators are deterministic given a seed" [7]. For purposes of the counterexample, the pseudo randomness employed is deterministic as evidenced by: (i) the full repeatability of any particular computer simulation, and (ii) the ability to give multiple algorithms the same starting positions via identical seeding of the randomizer.

Theorem 1 essentially states that for any performance measure selected (e.g. the best function value returned over "m" iterations), the sum of an algorithm's performance over all possible functions

is identical to the sum of any other algorithm's performance. This is equivalent to stating that the mean performances of all algorithms are equal across the set of all problems. Quoting Wolpert and Macready, "This theorem explicitly demonstrates that what an algorithm gains in performance on one class of problems is necessarily offset by its performance on the remaining problems…."

Perhaps it is common sense that an algorithm specially designed for one problem type should be expected to perform worse on another problem type than an algorithm specially designed for the latter type; however, asserting that all algorithms should be expected to perform equivalently across the set of all problems is tantamount to stating that a shoe horn should be expected to perform as well across the set of all problems as a rope, wheel, pulley, or Swiss army knife. Since some tools are more widely applicable than others, the notion that all tools must produce precisely the same mean performance across the same large set of problems seems counterintuitive.

2.1 One Optimizer Can Outperform Another if Re-visitations Count

This example shows that one algorithm can outperform another across the set of all static, continuous problems, S_1, if any possible re-visitations producing repeated evaluation of the same locations are counted for comparison purposes. While the NFL Theorems do not claim to apply to the case considered in Subsection 2.1, the demonstration provides an intuitive basis for the counterexample of Subsection 2.2, which counts evaluations of *approximately* the same locations for comparison: a case to which the NFL Theorems are thought to apply. In practice, continuous optimizers generally do not attempt to prevent re-visitation since modern optimization software can generate 10^{16n} distinct points to model continuous search spaces such that the probability of any location being revisited is essentially zero if the number of iterations is consistent with values used in practice, as discussed further in Subsection 2.2.

Let Gbest PSO as described in Section 1 be algorithm a_1. To fully define the algorithm, let the velocities of a_1 be clamped to 15% of the range of the search space per dimension using Formula (10). Let the inertia weight be set to 0.72984379, and let both acceleration constants be set to 1.49617977: this parameter selection produces the same search behavior as the Type 1" constriction model of PSO [3] when $K = 1$, $\Phi = 4.1$, and c_1 and c_2 of Equations (2.14) and (2.16) of [4] are both 2.05. These parameters define Gbest PSO throughout this paper and were used to generate the swarm trajectory snapshots of Figure 1 on the following page. More important than defining a particular swarm size is using the same population size for both algorithms to produce a straightforward comparison; nevertheless, Figure 1 was generated using a swarm size of 10 particles.

Define a_2 as a simple algorithm that uses the same population size as a_1 but never updates positions: its initial positions are simply preserved. Let the initial positions be set identically for both algorithms, which can be accomplished either by seeding the randomizer identically or by utilizing any non-stochastic initialization scheme, such as the maximum possible equidistant spacing of particles per dimension.

Except for m and F, whose definitions would not necessarily apply to Subsection 2.1, the symbology of [7] is utilized to avoid symbolic clutter. Let $k_{\max}$ denote the number of iterations allotted to the comparison. Let the performance measure be the lowest Y value in sample $d^y_{k\max}$. Then $\Phi\left(d^y_{k\max}\right) = \min_i \left\{ d^y_{k\max}(i) : i = 1...k_{\max} \right\}$; for Gbest PSO, this equates to the function value of the global best when the search concludes at iteration $k_{\max}$.

Since algorithms a_1 and a_2 are initialized identically for sake of a controlled experiment, and since the global best of Gbest PSO updates only when a better position is discovered (i.e. only when the lowest Y value improves), Gbest PSO can never perform worse than a_2 on any static problem.

Consequently, proving that a_1 produces better mean performance than a_2 only requires that Gbest PSO be demonstrated to improve upon its initial positions for any one problem.

Figure 1 shows the swarm migrating from a poorly initialized state to premature convergence at a local minimizer offering the best function value of all positions occupied during the course of the search, which is what attracts all particles to it via the social component [4]. Even in the worst-case scenario that for all other problems in S_1, Gbest PSO should somehow fail to improve upon the best position of the initialization phase in all successive iterations, a_1 would still outperform a_2 on set S_1 since a_1 can perform no worse than a_2 on any static problem for which positions are identically initialized by definition of the global best in Equation (14). In other words, since a_1 improves performance over a_2 on at least one problem, and since a_1 performs at least as well as a_2 on all other problems, algorithm a_1 achieves better mean performance than algorithm a_2 across the set of all problems; hence, what algorithm a_1 gains in performance on one problem is never offset by its performance on the remaining problems.

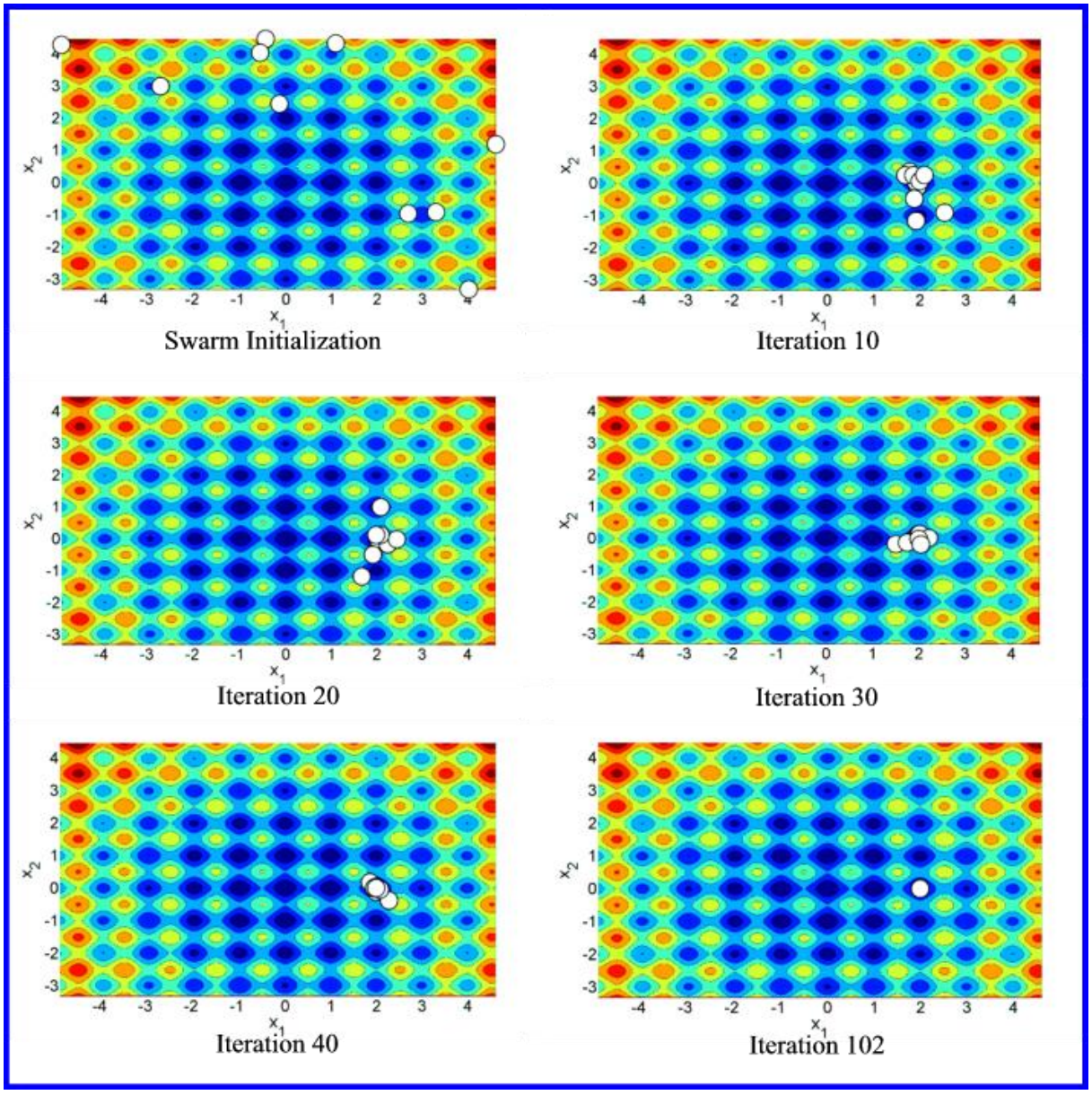

Figure 1

The snapshots of Figure 1 were generated from data produced by an actual simulation on the Rastrigin benchmark [10]. They illustrate that Gbest PSO improves upon its initial positions on at least one problem; since algorithm a_2 cannot outperform algorithm a_1 on any static problem by definition of the global best of a_1, algorithm a_1 outperforms algorithm a_2 on the set of all static problems when precise re-visitations are counted for comparison purposes. Section 2.2 extends this concept by proving that one algorithm can outperform another across the set of all problems if *approximate* re-visitations are counted for comparison purposes, which in turn disproves the applicability of the NFL Theorems to continuous optimization.

2.2 One Optimizer Can Outperform Another if Approximate Re-visitations Count

To demonstrate that one continuous optimizer can outperform another across the set of all possible problems, F, even if the outperformed algorithm never revisits any precise location, let Gbest PSO as defined in Subsection 2.1 be algorithm a_1. Let P_1 denote the percentage of trials written as a decimal for which a_1 successfully approximates a global minimizer across set F within a user-specified tolerance for error (e.g. $P_1 = 0.5$ if algorithm a_1 successfully approximates the global minimizer on 50 attempts out of 100).

Let each dimension be linearly normalized to lie between 0 and 1 prior to the optimization process, which both simplifies the ensuing calculations and ensures that each significant figure used by the software assumes all 10 integer values from 0 to 9, thereby generating the maximum number of discrete points within the search space. If MATLAB's "format" is set appropriately, 16 significant figures are reliably stored per variable; hence, 10^{16} feasible values exist per dimension, constituting an n-dimensional search space of precisely 10^{16n} locations. Though the search space is literally discrete from a microscopic perspective, with 10^{16n} points, it is certainly continuous from the macroscopic perspective. The following counterexample demonstrates the NFL Theorems to be irrelevant to optimization that is continuous from a macroscopic perspective, regardless of whether it becomes discrete at a microscopic level or not.

Let $k_{\max}$ be a reasonable number of iterations over which to optimize a system. "Reasonable" in this context means that the number is consistent with values used in practice. The first benefit of this selection is that velocity vectors will have nonzero magnitudes throughout the simulation such that no precise n-dimensional location will be revisited due to absolute immobility: in practice, continuing the optimization process once the swarm has literally stopped moving from even the most microscopic of perspectives would expend computational time and energy without any chance of improving solution quality, which is why the situation is avoided. Let S denote both the swarm size of Gbest PSO and the population size of the comparison algorithm, which are set identically for sake of a controlled experiment: the second benefit of using a reasonable value for $k_{\max}$ is that step size $\Delta \equiv \frac{P_1}{S \times k_{\max}}$ will be large enough for the software to distinguish it from 0; for example, step sizes larger than 10^{-323} are reliably distinguished from 0 by MATLAB.

Since NFL Theorem 1 claims to be true for any pair of algorithms, define a_2 as a population-based optimizer that, following any given initialization, iteratively updates each particle's position vector by stepping to the right (i.e. in the positive direction) per dimension via increments of user-specified width $\Delta \equiv \frac{P_1}{S \times k_{\max}}$. If a particle ever reaches the rightmost edge of the search space on any dimension, its subsequent position update will move it to the leftmost point on that dimension, from which it will continue stepping to the right. Since a_2 searches uni-directionally and step

size $\Delta \equiv \frac{P_1}{S \times k_{\max}}$, each particle will step across precisely $\frac{P_1}{S} \times 100\%$ of each dimension in $k_{\max}$ iterations. Since $\frac{P_1}{S} < 1$, neither any initial location nor any subsequently visited location will be revisited; since no re-visitation occurs, $k_{\max}$ is equivalent to the "m" of [7] from the perspective of algorithm a_2. Whether algorithm a_1 revisits any location or not is irrelevant since it will be shown to outperform algorithm a_2 regardless.

Linearly normalizing each dimension's range of feasible values to lie between 0 and 1 produces 10^{16} locations per dimension since each of the 16 significant figures maintained by the software is allowed to assume the full range of 10 potential values (i.e. 0 through 9). The distance between locations is then $1/\left(10^{16}-1\right) \approx 10^{-16}$ per dimension. Provided that step size $\Delta \gg 10^{-16}$, the discrete nature of the search space at a microscopic level does not become a limiting factor; rather, it becomes irrelevant to the large-scale continuous nature of the problem as proved in the following paragraph.

Even if the first algorithm, a_1, only successfully approximates a global minimizer once per one million simulations with a population size of 100 (i.e. $P_1 = 10^{-6}$, $S = 10^2$), the resulting value of 10^8 iterations for $k_{\max} = \frac{P_1}{S \times \Delta}$ would still be larger than the number of iterations used in practice by a factor of thousands. Since the number of iterations can reasonably be selected such that $k_{\max} \ll \frac{P_1}{S} \times 10^{16}$, it follows that the step size can reasonably be selected such that $\Delta \gg 10^{-16}$; hence, the computational requirements for this counterexample are satisfied. Since the number of iterations is not set nearly as large as 10^8 in practice, the microscopic level at which the search space produces discrete limitations is not reached so that the optimization process is continuous rather than discrete from the perspective of the optimization algorithm.

For problems that are continuous from the large-scale perspective, an optimizer can be designed to search such a small fraction of the search space that it will necessarily perform worse than a comparison algorithm. The following counterexample considers two mutually exclusive and mutually exhaustive cases, both of which lead to the same conclusion: one algorithm can produce better overall performance than another across the set of all problems.

Case I: Suppose that over the set of all relevant problems, F, global minimizers are uniformly distributed relative to the search spaces containing them. In this case, dividing each dimension of each problem's search space into 100 equidistant regions and counting the frequency with which the global minimizers of F occur within each region would produce a histogram showing precisely a 1% probability of occurrence per region. In other words, suppose that all relative regions of the search space are equally likely to contain a global minimizer for any problem chosen at random.

Continuous optimization algorithms sample locations within the search space and compare their qualities to propose the best location visited as a solution. Such an algorithm can only locate a global minimizer within the portion of the search space from which it successfully samples locations; consequently, if a_2 searches within a region comprising less than $P_1 \times 100\%$ of the entire search space, its success rate, P_2, will necessarily be less than P_1 across set F due to the supposed uniform distribution of global minimizers. Since step size $\Delta \equiv \frac{P_1}{S \times k_{\max}}$, each particle of algorithm a_2 will search precisely $\frac{P_1}{S} \times 100\%$ of each dimension of the search space in $k_{\max}$ iterations. The entire swarm will then search $P_1 \times 100\%$ of each dimension; therefore, a_2 will search precisely $P_1^n \times 100\%$ of the n-

dimensional search space in $k_{\max}$ iterations. Since $P_1 < 1$, it must be true that $P_1^n < P_1$; hence, due to the supposed uniform distribution of global minimizers across set F, algorithm a_1 with its actual success rate of P_1 will necessarily outperform algorithm a_2 with its maximum success rate of P_1^n. Hence, Gbest PSO is a better general purpose optimizer than the comparison algorithm if global minimizers are distributed uniformly within their containing search spaces.

Case II: Suppose that over the set of all possible problems, F, global minimizers are *not* uniformly distributed relative to the search spaces containing them. In this case, at least one dimension of each problem's search space could be divided into some number of equidistant regions for which a histogram plotting the frequency with which the global minimizers of F occur per region would show at least one region to have a greater probability of producing a global minimizer than at least one other region. In other words, suppose that *not* all relative regions of the search space are equally likely to contain a global minimizer for a problem chosen at random.

Different algorithms can favor different regions across the search spaces of F as a result of their characteristic search behaviors. To exemplify this, consider the following two options by which the PSO Research Toolbox confines particles to the search space [10]. The position clamping option works analogously to velocity clamping by instituting a minimum and maximum position per dimension, which places a particle flush against an edge of the feasible search space every time a particle attempts to exit. The velocity reset option sets to zero each dimension of a particle's velocity vector that would cause it to leave the search space, thus pulling it back into the interior of the search space in the same iteration [11, 12].

Whereas velocity reset pulls each exiting particle back into the search space, position clamping puts each straying particle at an edge location, which necessarily increases the number of edge locations sampled across set F. Since position clamping increases the amount of edge searching relative to velocity reset, the two algorithms favor different regions of the search space. Since each algorithm favors different regions, any notion that all algorithms search identically across set F is invalidated; hence, some algorithm should be expected to favor a region or regions more likely than others to produce a global minimizer, thereby outperforming some other algorithm on set F. In other words, some algorithm would, by its unique search behavior, surely favor a region or regions of the search space less likely than others to produce a global minimizer, thereby causing it to be outperformed by some other algorithm on set F (e.g. an algorithm using position clamping if the edges of the search space are less likely to contain a global minimizer than the interior).

As a second counterexample, the uni-directional search algorithm of Subsection 2.2 could incorporate an initialization scheme that would initialize particles to the left of the least promising regions of width $\Delta \times k_{\max}$ per dimension. Exactly which scheme would accomplish this is irrelevant: only the existence of such a scheme is important from this perspective, and there is certainly nothing to prevent any particular starting positions from being used by some particular initialization scheme. The algorithm would then have a lower probability of finding a global minimizer for any problem selected from F at random than would Gbest PSO using the same initialization scheme if initial velocities were set large enough to ensure that particles would be perturbed from their starting locations in case they should happen to be initialized near each other. For that matter, searching the regions that are the least likely across set F to produce global minimizers would probabilistically cause the uni-directional search algorithm to be outperformed by other optimization algorithms in general.

As a thought experiment, if search spaces that are defined at least partly by a priori knowledge, rather than wholly by physical constraints, are cautiously defined so as to avoid excluding quality solutions as a matter of intuition, this expert judgment could easily produce a somewhat normal distribution of global minimizers relative to the search space per dimension since at the very least, such expert judgment would be expected to put quality solutions well within the search space. In this

case, random search sampling from a normal distribution would be expected to outperform random search sampling from a uniform distribution since each region would be searched in proportion to its probability of producing a global minimizer. The same logic applies equally well for any other non-uniform distribution, which alone suffices to demonstrate that if global minimizers are not distributed uniformly within their containing search spaces, there must exist at least one algorithm that would outperform at least one other algorithm across the set of all relevant problems, F.

Any algorithm that produces better mean performance than another does so not by exploiting knowledge of any particular problem, but essentially by exploiting statistics concerning the set of all problems. This statistical knowledge is not programmed into the algorithm since the programmer possesses no such knowledge in advance; rather, the seeming knowledge merely results from the fact that different algorithms can favor different regions of the search space characteristically.

Since an algorithm can favor particular regions of the search space as a result of its characteristic search behavior, it follows that if global minimizers are not uniformly distributed across the set of all search spaces, some algorithm will spend valuable computation time favoring less promising regions, thus producing worse mean performance across the set of all possible problems. Regardless of which case is considered, what an algorithm gains or loses in performance on one class of problems is *not necessarily* offset by its performance on the remaining problems in contradiction to NFL Theorem 1.

3 Conclusions

This paper demonstrates that when either precise or approximate re-visitations count toward performance comparisons, NFL Theorem 1 is irrelevant. While the NFL Theorems do not claim to apply when precise re-visitations are counted for comparison purposes, they are thought to apply when approximate re-visitations from the large-scale continuous perspective are counted. This stems from the fact that the derivation of the NFL Theorems assumed a discrete search space in which all positions were simply regarded as distinct [7] rather than accounting for the fact that so many discrete points exist via modern software that approximate re-visitation has a similar effect as precise re-visitation for simulations of reasonable length, as evidenced specifically by the counterexample of Subsection 2.2. This renders the NFL Theorems inapplicable to continuous optimization, for which positions that are approximately the same must be compared in order to refine solution quality; otherwise, proposed solutions would unnecessarily be of lower quality than the algorithm is capable of producing. In addition to refining the quality of the final solution, approximate re-visitation is necessary to reliably ascertain the relative qualities of various potential solutions; otherwise, continuous optimizers such as RegPSO [4] would not be able to reliably distinguish local minimizers from global minimizers in order to select and propose the best solution.

The counterexample of Subsection 2.2 considers two mutually exclusive and mutually exhaustive cases that both produce the same conclusion: regardless of whether global minimizers are distributed uniformly or non-uniformly within their search spaces, NFL Theorem 1 is incorrect in asserting that all pairs of algorithms produce identical performance across the set of all problems. The counterexample demonstrates that the theorem does not apply to problems that are continuous from the macroscopic perspective, regardless of whether they become discrete at a microscopic level or not.

The counterexample of Subsection 2.2 casts doubt on the other NFL Theorems as well. Since it does not depend on any position to repeatedly produce the same function value or otherwise make any assumption about function values with respect to time, the counterexample is not restricted to the static case. NFL Theorem 2 for the time-varying case was justified similarly to Theorem 1: specifically, "the proof for the time-dependent case proceeds by establishing the a-independence of the sum $\sum_T P(c\,f, T, m, a)$ …" [7]; yet the counterexample of Section 2.2 shows a-independence to be a false conjecture with respect to continuous optimization. Furthermore, Theorem 2 is based on the assumption of bijectivity: "We impose bijectivity because if it did not hold, the evolution of cost

functions could narrow in on a region of f's for which *some algorithms may perform better than others*. This would constitute an a priori bias in favor of those algorithms, a bias whose analysis we wish to defer to future work" [7]. Yet the deferred analysis has not been published in more than a decade since the promise was made, which alone renders the theorem unfounded.

Theorem 3 also built on the irrelevant notion of a-independence: "At first glance this seems to be an intractable question, but the NFL theorem provides a way to answer it. This is because – according to the NFL Theorem – the answer must be independent of the algorithm used to generate $\vec{c}$" [7].

In summary, at least in the continuous case, an algorithm can be designed to perform worse than a comparison algorithm across the set of all relevant problems; consequently, the notion that "what an algorithm gains in performance on one class of problems is necessarily offset by its performance on the remaining problems" [7] does not apply to continuous optimization. Hence, the development of quality general-purpose optimizers is a valid pursuit. The irrelevance of NFL Theorem 1 and its foundational concept of a-independence furthermore render Theorems 2 and 3 irrelevant to continuous optimization, which might also cast doubt on the relevance of the remaining theorems.

The goal of this paper is not to clarify to which cases the NFL Theorems apply since the author is not fully convinced of the validity of the concept; rather, the idea is simply to show by counterexample that the No Free Lunch concept definitely does not apply to continuous optimization.

References

[1] Kennedy, J., *The Particle Swarm: Social Adaptation of Knowledge*, in Proceedings of IEEE International Conference on Evolutionary Computation, Indianapolis, IN, 1997, pp. 303-308.

[2] Shi, Y. and Eberhart, R. *A Modified Particle Swarm Optimizer*, in Proceedings of IEEE International Conference on Evolutionary Computation, (IEEE World Congress on Computational Intelligence), Anchorage, AK, pp. 69-73, 1998.

[3] Clerc, M. and Kennedy, J., *The Particle Swarm - Explosion, Stability, and Convergence in Multidimensional Complex Space*, IEEE Transactions on Evolutionary Computation, vol. 6, pp. 58-73, 2002.

[4] Evers, G., *An Automatic Regrouping Mechanism to Deal with Stagnation in Particle Swarm Optimization*, M.S. thesis, The University of Texas – Pan American, Edinburg, TX, 2009 <http://www.georgeevers.org/thesis.pdf>

[5] Eberhart, R. C. and Kennedy, J., *A New Optimizer Using Particle Swarm Theory*, in Proceedings of the Sixth International Symposium on Micro Machine and Human Science, Nagoya, Japan, pp. 39-43, 1995.

[6] Macready, W.G., and Wolpert, D.H., *What Makes an Optimization Problem Hard?*, Complexity, 5, 1996.

[7] Wolpert, D.H., and Macready, W.G., *No Free Lunch Theorems for Optimization*, IEEE Transactions on Evolutionary Computation, 1, 1997.

[8] Wolpert, D.H., and Macready, W.G., *Coevolutionary Free Lunches, IEEE Transactions on Evolutionary Computation*, 9, 721-735, 2005.

[9] Dembski, W., *No Free Lunch: Why Specified Complexity Cannot Be Purchased without Intelligence*. (Lanham, Md.: Rowman & Littlefield, 2002).

[10] Evers, G., *PSO Research Toolbox Documentation,* M.S. thesis code with documentation, 2011, < http://www.georgeevers.org/pso_research_toolbox.htm>

[11] Venter, G. and Sobieski, J., *Particle Swarm Optimization*, Proceedings of the 43rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, Denver, CO, 2002.

[12] Ben Ghalia, M. , *Particle Swarm Optimization with an Improved Exploration-exploitation Balance*, Circuits and Systems, MWSCAS. 51st Midwest Symposium, pp. 759-762 2008.