Biogeography-Based Optimization

Dan Simon, *Senior Member, IEEE*

***Abstract*—Biogeography is the study of the geographical distribution of biological organisms. Mathematical equations that govern the distribution of organisms were first discovered and developed during the 1960s. The mindset of the engineer is that we can learn from nature. This motivates the application of biogeography to optimization problems. Just as the mathematics of biological genetics inspired the development of genetic algorithms (GAs), and the mathematics of biological neurons inspired the development of artificial neural networks, this paper considers the mathematics of biogeography as the basis for the development of a new field: biogeography-based optimization (BBO). We discuss natural biogeography and its mathematics, and then discuss how it can be used to solve optimization problems. We see that BBO has features in common with other biology-based optimization methods, such as GAs and particle swarm optimization (PSO). This makes BBO applicable to many of the same types of problems that GAs and PSO are used for, namely, high-dimension problems with multiple local optima. However, BBO also has some features that are unique among biology-based optimization methods. We demonstrate the performance of BBO on a set of 14 standard benchmarks and compare it with seven other biology-based optimization algorithms. We also demonstrate BBO on a real-world sensor selection problem for aircraft engine health estimation.**

***Index Terms*—Biogeography, evolutionary algorithms, Kalman filter, optimization, sensor selection.**

List of Acronyms

ACO	Ant colony optimization.
BBO	Biogeography-based optimization.
CPU	Central processing unit.
DARE	Discrete algebraic Riccati equation.
DE	Differential evolution.
ES	Evolutionary strategy.
GA	Genetic algorithm.
HSI	Habitat suitability index.
MAPSS	Modular aero propulsion system simulation.
PBIL	Probability-based incremental learning.
PSO	Particle swarm optimization.
SGA	Stud genetic algorithm.
SIV	Suitability index variable.
SVD	Singular value decomposition.

Manuscript received March 28, 2007; revised September 14, 2007. First published March 18, 2008; current version published December 2, 2008.

The author is with the Department of Electrical Engineering, Cleveland State University, Cleveland, OH 44115 USA (e-mail: d.j.simon@csuohio.edu).

Digital Object Identifier 10.1109/TEVC.2008.919004

I. Introduction

THE SCIENCE OF biogeography can be traced to the work of nineteenth century naturalists such as Alfred Wallace [1] and Charles Darwin [2]. Until the 1960s, biogeography was mainly descriptive and historical. In the early 1960s, Robert MacArthur and Edward Wilson began working together on mathematical models of biogeography, their work culminating with the classic 1967 publication *The Theory of Island Biogeography* [3]. Their interest was primarily focused on the distribution of species among neighboring islands. They were interested in mathematical models for the extinction and migration of species. Since MacArthur and Wilson's work, biogeography has become a major area of research [4]. A recent search of Biological Abstracts (a biology research index) reveals that 25,452 papers were written in the year 2005 that were related to the subject of biogeography. However, a search of INSPEC, an engineering research index, reveals that no biogeography papers have ever been written. In view of this, part of the motivation of this paper is to merge the burgeoning field of biogeography with engineering in order to see how the two disciplines can be of mutual benefit. The application of biogeography to engineering is similar to what has occurred in the past few decades with genetic algorithms (GAs), neural networks, fuzzy logic, particle swarm optimization (PSO), and other areas of computer intelligence.

Mathematical models of biogeography describe how species migrate from one island to another, how new species arise, and how species become extinct. The term "island" here is used descriptively rather than literally. That is, an island is any habitat that is geographically isolated from other habitats. We therefore use the more generic term "habitat" in this paper (rather than "island") [4]. Geographical areas that are well suited as residences for biological species are said to have a high habitat suitability index (HSI) [5]. Features that correlate with HSI include such factors as rainfall, diversity of vegetation, diversity of topographic features, land area, and temperature. The variables that characterize habitability are called suitability index variables (SIVs). SIVs can be considered the independent variables of the habitat, and HSI can be considered the dependent variable.

Habitats with a high HSI tend to have a large number of species, while those with a low HSI have a small number of species. Habitats with a high HSI have many species that emigrate to nearby habitats, simply by virtue of the large number of species that they host. Habitats with a high HSI have a low species immigration rate because they are already nearly saturated with species. Therefore, high HSI habitats are more static in their species distribution than low HSI habitats. By the same token, high HSI habitats have a high emigration rate; the large number of species on high HSI islands have many opportunities

to emigrate to neighboring habitats. (This does not mean that an emigrating species completely disappears from its home habitat; only a few representatives emigrate, so an emigrating species remains extant in its home habitat, while at the same time migrating to a neighboring habitat.) Habitats with a low HSI have a high species immigration rate because of their sparse populations. This immigration of new species to low HSI habitats may raise the HSI of the habitat, because the suitability of a habitat is proportional to its biological diversity. However if a habitat's HSI remains low, then the species that reside there will tend to go extinct, which will further open the way for additional immigration. Due to this, low HSI habitats are more dynamic in their species distribution than high HSI habitats.

Biogeography is nature's way of distributing species, and is analogous to general problem solutions. Suppose that we are presented with a problem and some candidate solutions. The problem can be in any area of life (engineering, economics, medicine, business, urban planning, sports, etc.), as long as we have a quantifiable measure of the suitability of a given solution. A good solution is analogous to an island with a high HSI, and a poor solution represents an island with a low HSI. High HSI solutions resist change more than low HSI solutions. By the same token, high HSI solutions tend to share their features with low HSI solutions. (This does not mean that the features disappear from the high HSI solution; the shared features remain in the high HSI solutions, while at the same time appearing as new features in the low HSI solutions. This is similar to representatives of a species migrating to a habitat, while other representatives remain in their original habitat.) Poor solutions accept a lot of new features from good solutions. This addition of new features to low HSI solutions may raise the quality of those solutions. We call this new approach to problem solving biogeography-based optimization (BBO).

BBO has certain features in common with other biology-based algorithms. Like GAs and PSO, BBO has a way of sharing information between solutions. GA solutions "die" at the end of each generation, while PSO and BBO solutions survive forever (although their characteristics change as the optimization process progresses). PSO solutions are more likely to clump together in similar groups, while GA and BBO solutions do not necessarily have any built-in tendency to cluster.

The goals of this paper are threefold. First, we want to give a general presentation of the new optimization method called BBO. We do this by first studying natural biogeography, and then generalizing it to obtain a general-purpose optimization algorithm. Second, we want to compare and contrast BBO with other population-based optimization methods. We do this by looking at the commonalities and differences from an algorithmic point-of-view, and also by comparing their performances on a set of benchmark functions. Third we want to apply BBO to the real-world problem of sensor selection for aircraft engine health estimation. This will demonstrate the applicability of BBO to real-world problems.

Section II reviews the ideas and mathematics of biogeography, and Section III discusses how biogeography can be used to formulate a general optimization algorithm. Section IV reviews aircraft engine health estimation and how Kalman filtering can be used to estimate engine health. Section V provides some simulation results comparing BBO with other optimization methods, both for general benchmark functions and for a sensor selection problem. Section VI presents some concluding remarks and suggestions for further work.

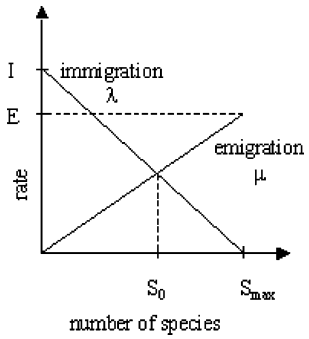

Fig. 1. Species model of a single habitat based on [3].

II. Biogeography

Fig. 1 illustrates a model of species abundance in a single habitat [3]. The immigration rate λ and the emigration rate μ are functions of the number of species in the habitat.

Consider the immigration curve. The maximum possible immigration rate to the habitat is I, which occurs when there are zero species in the habitat. As the number of species increases, the habitat becomes more crowded, fewer species are able to successfully survive immigration to the habitat, and the immigration rate decreases. The largest possible number of species that the habitat can support is $S_{\max}$, at which point the immigration rate becomes zero.

Now consider the emigration curve. If there are no species in the habitat then the emigration rate must be zero. As the number of species increases, the habitat becomes more crowded, more species are able to leave the habitat to explore other possible residences, and the emigration rate increases. The maximum emigration rate is E, which occurs when the habitat contains the largest number of species that it can support.

The equilibrium number of species is S_0, at which point the immigration and emigration rates are equal. However, there may be occasional excursions from S_0 due to temporal effects. Positive excursions could be due to a sudden spurt of immigration (caused, perhaps, by an unusually large piece of flotsam arriving from a neighboring habitat), or a sudden burst of speciation (like a miniature Cambrian explosion). Negative excursions from S_0 could be due to disease, the introduction of an especially ravenous predator, or some other natural catastrophe. It can take a long time in nature for species counts to reach equilibrium after a major perturbation [4], [6].

We have shown the immigration and emigration curves in Fig. 1 as straight lines but, in general, they might be more complicated curves. Nevertheless, this simple model gives us a general description of the process of immigration and emigration. The details can be adjusted if needed.

Now, consider the probability P_s that the habitat contains exactly S species. P_s changes from time t to time $(t+\Delta t)$ as follows:

$$P_s(t+\Delta t) = P_s(t)(1-\lambda_s\Delta t - \mu_s\Delta t) + P_{s-1}\lambda_{s-1}\Delta t + P_{s+1}\mu_{s+1}\Delta t \quad (1)$$

where λ_s and μ_s are the immigration and emigration rates when there are S species in the habitat. This equation holds because in order to have S species at time $(t+\Delta t)$, one of the following conditions must hold:

1) there were S species at time t, and no immigration or emigration occurred between t and $(t+\Delta t)$;
2) there were $(S-1)$ species at time t, and one species immigrated;
3) there were $(S+1)$ species at time t, and one species emigrated.

We assume that Δt is small enough so that the probability of more than one immigration or emigration can be ignored. Taking the limit of (1) as $\Delta t \to 0$ gives equation (2) shown at the bottom of the page. We define $n = S_{\max}$, and $P = [P_0 \quad \dots \quad P_n]^T$, for notational simplicity. Now, we can arrange the $\dot{P}_s$ equations (for $S = 0, \dots, n$) into the single matrix equation

$$\dot{P} = AP \quad (3)$$

where the matrix A is given as (4) shown at the bottom of the page. For the straight line curves shown in Fig. 1, we have

$$\mu_k = \frac{Ek}{n}$$
$$\lambda_k = I\left(1 - \frac{k}{n}\right). \quad (5)$$

Now, consider the special case $E = I$. In this case, we have

$$\lambda_k + \mu_k = E \quad (6)$$

and the A matrix becomes

$$A = E\begin{bmatrix} -1 & \frac{1}{n} & 0 & \dots & 0 \\ \frac{n}{n} & -1 & \frac{2}{n} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \frac{2}{n} & -1 & \frac{n}{n} \\ 0 & \dots & 0 & \frac{1}{n} & -1 \end{bmatrix} = EA' \quad (7)$$

where A' is defined by the above equation.

1) Observation 1: Zero is an eigenvalue of A', with the corresponding eigenvector

$$v = [\, v_1 \quad \dots \quad v_{n+1} \,]^T$$
$$v_i = \begin{cases} \frac{n!}{(n-1-i)!(i-1)!} & (i = 1, \dots, i') \\ v_{n+2-i} & (i = i'+1, \dots, n+1) \end{cases} \quad (8)$$

where i' is the smallest integer that is greater than or equal to $(n+1)/2$; that is, $i' = \text{ceil}((n+1)/2)$.

This observation can be verified by a straightforward but somewhat tedious solution of the eigenvalue equation $A'v = kv$ for the unknown scalar k and the unknown vector v. As an example, with $n = 4$, we obtain

$$v = [\, 1 \quad 4 \quad 6 \quad 4 \quad 1 \,]^T. \quad (9)$$

With $n = 5$, we obtain

$$v = [\, 1 \quad 5 \quad 10 \quad 10 \quad 5 \quad 1 \,]^T. \quad (10)$$

2) Conjecture 1: The eigenvalues of A' are given as

$$k = \{\, 0, \quad \tfrac{-2}{n}, \quad \tfrac{-4}{n}, \quad \dots \quad -2 \,\}. \quad (11)$$

This conjecture has not yet been proven, but it has been observed to be true for all values of n that have been investigated up to this point in time.

$$\dot{P}_s = \begin{cases} -(\lambda_s + \mu_s)P_s + \mu_{s+1}P_{s+1}, & S = 0 \\ -(\lambda_s + \mu_s)P_s + \lambda_{s-1}P_{s-1} + \mu_{s+1}P_{s+1}, & 1 \le S \le S_{\max} - 1 \\ -(\lambda_s + \mu_s)P_s + \lambda_{s-1}P_{s-1} & S = S_{\max} \end{cases} \quad (2)$$

$$A = \begin{bmatrix} -(\lambda_0 + \mu_0) & \mu_1 & 0 & \dots & 0 \\ \lambda_0 & -(\lambda_1 + \mu_1) & \mu_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \lambda_{n-2} & -(\lambda_{n-1} + \mu_{n-1}) & \mu_n \\ 0 & \dots & 0 & \lambda_{n-1} & -(\lambda_n + \mu_n) \end{bmatrix} \quad (4)$$

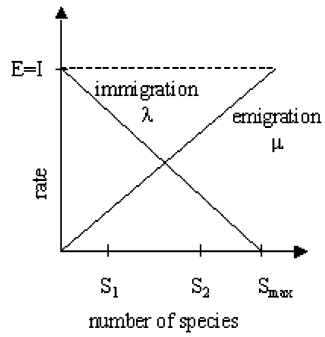

Fig. 2. Illustration of two candidate solutions to some problem. S_1 is a relatively poor solution, while S_2 is a relatively good solution.

Theorem 1: The steady-state value for the probability of the number of each species is given by

$$P(\infty) = \frac{v}{\sum_{i=1}^{n+1} v_i} \tag{12}$$

where v and v_i are given in (8).

Proof: See the appendix.

III. Biogeography-Based Optimization (BBO)

In this section, we discuss how the biogeography theory of the previous section can be applied to optimization problems with a discrete domain.

A. Migration

Suppose that we have a problem and a population of candidate solutions that can be represented as vectors of integers. Each integer in the solution vector is considered to be an SIV. Further suppose that we have some way of assessing the goodness of the solutions. Those solutions that are good are considered to be habitats with a high HSI, and those that are poor are considered to be habitats with a low HSI. HSI is analogous to "fitness" in other population-based optimization algorithms (GAs, for example). High HSI solutions represent habitats with many species, and low HSI solutions represent habitats with few species. We assume that each solution (habitat) has an identical species curve (with $E = I$ for simplicity), but the S value represented by the solution depends on its HSI. S_1 in Fig. 2 represents a low HSI solution, while S_2 represents a high HSI solution. S_1 in Fig. 2 represents a habitat with only a few species, while S_2 represents a habitat with many species. The immigration rate λ_1 for S_1 will, therefore, be higher than the immigration rate λ_2 for S_2. The emigration rate μ_1 for S_1 will be lower than the emigration rate μ_2 for S_2.

We use the emigration and immigration rates of each solution to probabilistically share information between habitats. With probability P_{mod}, we modify each solution based on other solutions. If a given solution is selected to be modified, then we use its immigration rate λ to probabilistically decide whether or not to modify each suitability index variable (SIV) in that solution. If a given SIV in a given solution S_i is selected to be modified, then we use the emigration rates μ of the other solutions to probabilistically decide which of the solutions should migrate a randomly selected SIV to solution S_i.

The BBO migration strategy is similar to the global recombination approach of the breeder GA [7] and evolutionary strategies [8] in which many parents can contribute to a single offspring, but it differs in at least one important aspect. In evolutionary strategies, global recombination is used to create new solutions, while BBO migration is used to change existing solutions. Global recombination in evolutionary strategy is a reproductive process, while migration in BBO is an adaptive process; it is used to modify existing islands.

As with other population-based optimization algorithms, we typically incorporate some sort of elitism in order to retain the best solutions in the population. This prevents the best solutions from being corrupted by immigration.

B. Mutation

Cataclysmic events can drastically change the HSI of a natural habitat. They can also cause a species count to differ from its equilibrium value (unusually large flotsam arriving from a neighboring habitat, disease, natural catastrophes, etc.). A habitat's HSI can, therefore, change suddenly due to apparently random events. We model this in BBO as SIV mutation, and we use species count probabilities to determine mutation rates.

The probabilities of each species count will be governed by the differential equation given in (2). By looking at the equilibrium point on the species curve of Fig. 2, we see that low species counts and high species counts both have relatively low probabilities. This can also be inferred from Theorem 1. Medium species counts have high probabilities because they are near the equilibrium point.

As an example, consider the case where $S_{\max} = 10$. Then, the steady-state solution of (2) is independent of the initial condition $P(0)$ and can be computed either numerically or from Theorem 1 as shown in (13) at the bottom of the page. The elements of $P(\infty)$ sum to one (within rounding error), and a plot of the $P(\infty)$ elements is an even function with respect to its midpoint.

Each population member has an associated probability, which indicates the likelihood that it was expected *a priori* to exist as a solution to the given problem. Very high HSI solutions and very low HSI solutions are equally improbable. Medium HSI solutions are relatively probable. If a given solution S has a low probability P_s, then it is surprising that it exists as a solution. It is, therefore, likely to mutate to some other solution. Conversely, a solution with a high probability is less likely to mutate to a

$$P(\infty) \approx [0.001 \quad 0.001 \quad 0.044 \quad 0.117 \quad 0.205 \quad 0.246 \quad 0.205 \quad 0.117 \quad 0.044 \quad 0.001 \quad 0.001]^T \tag{13}$$

different solution. This can be implemented as a mutation rate m that is inversely proportional to the solution probability

$$m(S) = m_{\max}\left(\frac{1-P_s}{P_{\max}}\right) \tag{14}$$

where $m_{\max}$ is a user-defined parameter. This mutation scheme tends to increase diversity among the population. Without this modification, the highly probable solutions will tend to be more dominant in the population. This mutation approach makes low HSI solutions likely to mutate, which gives them a chance of improving. It also makes high HSI solutions likely to mutate, which gives them a chance of improving even more than they already have. Note that we use an elitism approach to save the features of the habitat that has the best solution in the BBO process, so even if mutation ruins its HSI, we have saved it and can revert back to it if needed. So, we use mutation (a high risk process) on both poor solutions and good solutions. Those solutions that are average are hopefully improving already, and so we avoid mutating them (although there is still some mutation probability, except for the most probable solution).

The implemented mutation mechanism is problem dependent, just as it is for GAs. In our sensor selection problem (discussed in Section IV), if a solution is selected for mutation, then we simply replace a randomly chosen sensor in the solution with a new, randomly generated sensor. We have not explored alternative mutation schemes in this paper, but presumably all of the mutation schemes that have been implemented for GAs could also be implemented for BBO.

C. BBO Definitions and Algorithm

In this section, we provide some definitions as a first step towards formalizing the BBO algorithm. We also provide an outline of the algorithm. We use R to refer to the set of real numbers, Z to refer to the set of integers, and $\emptyset$ to refer to the empty set.

Definition 1: A habitat $H \in \mathrm{SIV}^m$ is a vector of m integers that represents a feasible solution to some problem.

Definition 2: A suitability index variable $SIV \in C$ is an integer that is allowed in a habitat. $C \subset Z^q$ is the set of all integers that are allowed in a habitat.

The requirement that $SIV \in C$ is called a constraint. At a higher level, the requirement that $H \in \mathrm{SIV}^m$ is also called a constraint.

Definition 3: A habitat suitability index HSI: $H \rightarrow R$ is a measure of the goodness of the solution that is represented by the habitat.

Note: In most population-based optimization algorithms, HSI is called fitness.

Definition 4: An ecosystem H^n is a group of n habitats.

The size n of an ecosystem is constant. Future work could allow variable-sized ecosystems, just as some flavors of GAs allow for variable population sizes.

Definition 5: Immigration rate $\lambda(\mathrm{HSI}) : R \rightarrow R$ is a monotonically nonincreasing function of HSI. λ_i is proportional to the likelihood that SIVs from neighboring habitats will migrate into habitat H_i.

Definition 6: Emigration rate $\mu(\mathrm{HSI}) : R \rightarrow R$ is a monotonically nondecreasing function of HSI. μ_i is proportional to the likelihood that SIVs from habitat H_i will migrate into neighboring habitats.

In practice, we assume that λ and μ are linear with the same maximum values. However, these assumptions are made only for mathematical convenience, and better performance might be attainable if these assumptions are relaxed.

Definition 7: Habitat modification $\Omega(\lambda, \mu) : H^n \rightarrow H$ is a probabilistic operator that adjusts habitat H based on the ecosystem H^n. The probability that H is modified is proportional to its immigration rate λ, and the probability that the source of the modification comes from H_j is proportional to the emigration rate μ_j.

Habitat modification can loosely be described as follows.

```
Select H_i with probability ∝ λ_i
If H_i is selected
    For j = 1 to n
        Select H_j with probability ∝ μ_i
        If H_j is selected
            Randomly select an SIV σ from H_j
            Replace a random SIV in H_i with σ
        end
    end
end
```

From this algorithm, we note that elitism can be implemented by setting $\lambda = 0$ for the p best habitats, where p is a user-selected elitism parameter. Also note that the definition of Ω ensures that the modified habitat H satisfies the SIV constraints.

Definition 8: Mutation $M(\lambda, \mu) : H \rightarrow H$ is a probabilistic operator that randomly modifies habitat SIVs based on the habitat's *a priori* probability of existence.

A habitat's probability of existence is computed from λ and μ as discussed in Section II. Mutation can be described as follows.

```
For j = 1 to m
    Use λ_i and μ_i to compute the probability P_i
    Select SIV H_i(j) with probability ∝ P_i
    If H_i(j) is selected
        Replace H_i(j) with a randomly generated SIV
    end
end
```

As with habitat modification, elitism can be implemented by setting the probability of mutation selection P_i to zero for the p best habitats. From the above definition, we see that mutation must be constrained to result in an HSI that satisfies the SIV constraints.

Definition 9: An ecosystem transition function $\Psi = (m, n, \lambda, \mu, \Omega, M) : H^n \rightarrow H^n$ is a 6-tuple that modifies the ecosystem from one optimization iteration to the next.

An ecosystem transition function can be written as follows:

$$\Psi = \lambda^n \circ \mu^n \circ \Omega^n \circ \mathrm{HSI}^n \circ M^n \circ \mathrm{HSI}^n. \qquad (15)$$

In other words, the ecosystem transition function begins by computing the immigration and emigration rates of each habitat. Then, habitat modification is performed on each habitat, followed by an HSI recalculation. Finally, mutation is performed, followed again by an HSI recalculation for each habitat.

Definition 10: A BBO algorithm $\mathrm{BBO} = (\mathcal{I}, \Psi, T)$ is a 3-tuple that proposes a solution to an optimization problem. $\mathcal{I} : \emptyset \rightarrow \{H^n, \mathrm{HSI}^n\}$ is a function that creates an initial ecosystem of habitats and computes each corresponding HSI. Ψ is the ecosystem transition function defined earlier, and $T : H^n \rightarrow \{\mathrm{true}, \mathrm{false}\}$ is a termination criterion.

$\mathcal{I}$ could be implemented with random number generators, heuristic solutions to the optimization problem, or some other problem-dependent procedure. T could depend on the number of Ψ iterations, or the HSI of the best habitat, or some other problem-dependent criterion. A BBO algorithm can be described as follows.

$\mathcal{I}$

while not T

Ψ

end

The BBO algorithm can be informally described with the following algorithm.

1) Initialize the BBO parameters. This means deriving a method of mapping problem solutions to SIVs and habitats (see Definitions 1 and 2), which is problem dependent. We also initialize the maximum species count $S_{\max}$ and the maximum migration rates E and I (see Fig. 2), the maximum mutation rate $m_{\max}$ [see (14)], and an elitism parameter (see the last paragraph of Section III-A). Note that the maximum species count and the maximum migration rates are relative quantities. That is, if they all change by the same percentage, then the behavior of BBO will not change. This is because if E, I, and $S_{\max}$ change, then the migration rates μ, λ, and the species count S will change by the same relative amount for each solution.
2) Initialize a random set of habitats, each habitat corresponding to a potential solution to the given problem. This is the implementation of the $\mathcal{I}$ operator described in Definition 10.
3) For each habitat, map the HSI to the number of species S, the immigration rate λ, and the emigration rate μ (see Fig. 2 and Definitions 5 and 6).
4) Probabilistically use immigration and emigration to modify each non-elite habitat as discussed in Section III-A, then recompute each HSI (see Definition 7).
5) For each habitat, update the probability of its species count using (2). Then, mutate each non-elite habitat based on its probability as discussed in Section III-B, and recompute each HSI (see Definition 8).
6) Go to step (3) for the next iteration. This loop can be terminated after a predefined number of generations, or after an acceptable problem solution has been found. This is the implementation of the T operator described in Definition 10.

Note that after each habitat is modified (steps 2, 4, and 5), its feasibility as a problem solution should be verified. If it does not represent a feasible solution, then some method needs to be implemented in order to map it to the set of feasible solutions.

D. Differences Between BBO and Other Population-Based Optimization Algorithms

In this section, we point out some of the distinctives of BBO. First, we note that although BBO is a population-based optimization algorithm it does not involve reproduction or the generation of "children." This clearly distinguishes it from reproductive strategies such as GAs and evolutionary strategies.

BBO also clearly differs from ACO, because ACO generates a new set of solutions with each iteration. BBO, on the other hand, maintains its set of solutions from one iteration to the next, relying on migration to probabilistically adapt those solutions.

BBO has the most in common with strategies such as PSO and DE. In those approaches, solutions are maintained from one iteration to the next, but each solution is able to learn from its neighbors and adapt itself as the algorithm progresses. PSO represents each solution as a point in space, and represents the change over time of each solution as a velocity vector. However, PSO solutions do not change directly; it is rather their *velocities* that chane, and this indirectly results in position (solution) changes. DE changes its solutions directly, but changes in a particular DE solution are based on differences between other DE solutions. Also, DE is not biologically motivated. BBO can be contrasted with PSO and DE in that BBO solutions are changed directly via migration from other solutions (islands). That is, BBO solutions directly share their attributes (SIVs) with other solutions.

It is these differences between BBO and other population-based optimization methods that may prove to be its strength. Some open research questions are: How do these differences make the performance of BBO differ from other population-based optimization methods? What do these differences say about the types of problems that are most appropriate for BBO? This paper presents the initial explorations into BBO but leaves these questions for later work.

IV. Aircraft Engine Health Estimation

In this section, we review the sensor selection problem for aircraft engine health estimation, which we will later use as a test problem for the BBO theory.

Fig. 3 shows a schematic of an aircraft turbofan engine [9]. An inlet supplies air to the fan. The air that leaves the fan separates into two streams, one through the engine core, and the other through the bypass duct. The fan is driven by a low-pressure turbine. The air that passes through the engine core goes through a compressor, which is driven by a high-pressure turbine. Fuel is injected and ignited in the combustor to produce hot gas that drives the turbines. The two air streams recombine in the augmentor duct, where additional fuel may be added to increase the temperature. The air leaves the augmentor at a high

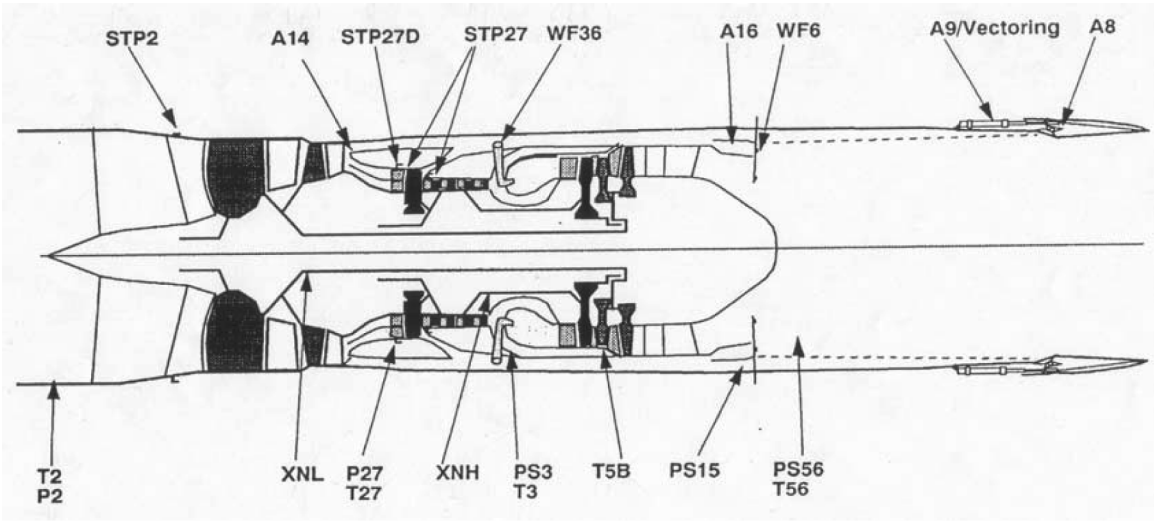

Fig. 3. Schematic of an aircraft turbofan engine.

velocity through the nozzle (which has an adjustable cross section area) and thereby produces thrust.

The engine simulation used in this paper is called Modular Aero Propulsion System Simulation (MAPSS) [9], and was written using Matlab Simulink. The controller update rate is 50 Hz. The three state variables used in MAPSS are low-pressure rotor speed, high-pressure rotor speed, and average hot section metal temperature.

The discretized time invariant equations that model the turbofan engine can be summarized as

$$\begin{aligned} x(k+1) &= f[x(k), u(k), p(k)] + w_x(k) \\ p(k+1) &= p(k) + w_p(k) \\ y(k) &= g[x(k), u(k), p(k)] + e(k) \end{aligned} \tag{16}$$

where k is the time index, x is the three-element state vector, u is the three-element control vector, p is the ten-element health parameter vector, and y is the measurement vector. The measurement consists of the outputs of the sensors with which we instrument the engine. The health parameters change slowly over time. Between measurement times their deviations can be approximated by the zero mean noise $w_p(k)$. The noise term $w_x(k)$ represents inaccuracies in the system model, and $e(k)$ represents measurement noise. The states, controls, health parameters, and measurements are summarized in [10], along with their values.

A Kalman filter can be used with (16) to estimate the state vector x and the health parameter vector p. One of the nice features of the Kalman filter is that it not only provides an estimate of x and p, but it also provides a measure of the uncertainty of the estimate. The uncertainty of the estimate is provided by the error covariance Σ, which is computed as part of the Kalman filter recursion [11].

Since we have three states and ten health parameters, the covariance Σ is a 13×13 matrix. The diagonal elements give the variance of the estimation errors of the states and health parameters. The first three diagonal elements give the variance of the state estimation errors, and the last ten diagonal elements give the variance of the health parameter estimation errors. In the problem we consider in this paper, we are interested only in the health parameter estimation errors, so we are concerned about the diagonal elements $\Sigma(4,4), \Sigma(5,5), \ldots, \Sigma(13,13)$.

We can choose which sensors to use for the health estimation process. We can also duplicate sensors if we want. We have 11 unique sensors as described in [10], but we can use multiple sensors at a single location if desired. For example, we could use two or three identical sensors to measure the fan exit pressure, thereby effectively reducing our signal-to-noise ratio for that measurement, or we could completely eliminate one of the sensors to achieve a financial savings. The use of more sensors results in smaller elements Σ, which means that our health estimate will be better. However, there is a point of diminishing returns. The use of more sensors costs more money, and it may not be worth the extra cost to obtain a marginally improved health estimate. The optimality criterion for the health estimation problem can, therefore, be written

$$J = \sum_{i=4}^{13} \sqrt{\frac{\Sigma(i,i)}{\Sigma_0(i,i)}} + \frac{\alpha C}{C_0}. \tag{17}$$

Σ_0 and C_0 are reference values used for normalization. Σ_0 is the covariance that results if we use all 11 sensors with no duplicates, and C_0 is the financial cost of fitting the aircraft engine with all 11 sensors. α is a scale factor that weights the importance of financial cost relative to estimation accuracy. J is the objective function for the health estimation problem. This approach to sensor selection was first proposed using GAs [12]. When BBO is used to solve the problem, J is referred to as the HSI.

The choice of what sensors to use to minimize J is an optimization problem. Recall that we have 11 sensors available. We typically have some constraints on the problem, such as the

constraint that we are to use a total of N sensors, with each individual sensor used no more than M times. If $N = 12$ and $M = 3$, then we have the following examples:

$1,2,3,4,4,5,6,7,8,8,8,11$
– legal set (no sensor is used more than 3 times)
$1,2,3,4,4,4,4,5,6,7,8,9$
– illegal set (sensor 4 is used more than 3 times).

In general, we want to use a total of N sensors out of K unique sensors (in our example, $K = 11$) with each sensor being used no more than M times. (The numerical values of N, K, and M will be problem dependent.) The total number of possible sensor sets is found by the following procedure. First, we generate a polynomial $q(x)$ as

$$\begin{aligned} q(x) &= (1 + x + x^2 + \cdots + x^M)^K \\ &= 1 + q_1 x + q_2 x^2 + \cdots + x^{MK}. \end{aligned} \tag{18}$$

The total number of sets containing exactly N sensors is equal to q_N. This is known as the multinomial theorem [13].

As a simple example, suppose that we want to use a total of four sensors out of three unique sensors (sensor numbers 1, 2, and 3) with each sensor being used no more than two times. The possible sensor sets are shown in (19) at the bottom of the page. We see that there are six possible sensor sets. The polynomial associated with this problem is

$$\begin{aligned} q(x) &= (1 + x + x^2)^3 \\ &= 1 + 3x + 6x^2 + 7x^3 + 6x^4 + 3x^5 + x^6. \end{aligned} \tag{20}$$

The coefficient of x^4 in $q(x)$ is equal to 6; that is, there are six unique sensor sets that use a total of four sensors.

V. Simulation Results

In this section, we look at the performance of BBO as compared with other population-based optimization methods. First, we compare performances for a set of commonly used benchmark functions, and then we compare performances for the turbofan sensor selection problem. The code that was used to generate the results is available at `http://academic.csuohio.edu/simond/bbo`.

A. Benchmark Results

In order to explore the benefits of BBO, we compared its performance on various benchmark functions with seven other population-based optimization methods. ACO [14]–[17] is an algorithm that is based on the pheromone deposition of ants. DE [17]–[19] is a simple method that uses the difference between two solutions to probabilistically adapt a third solution. An ES [8], [20]–[22] is an algorithm that generally gives about equal importance to recombination and mutation, and that allows more than two parents to contribute to an offspring. A GA [8], [20], [23] is a method that is based on natural selection in the theory of biological evolution. PBIL [24], [25] is a type of GA that maintains statistics about the population rather than maintaining the population directly. PSO [17], [26]–[28] is based on the swarming behavior of birds, fish, and other creatures. A stud genetic algorithm (SGA) [29] is a GA that uses the best individual at each generation for crossover.

The benchmarks that we minimized are functions that are representative of those used in the literature for comparison of optimization methods. Some are multimodal, which means that they have multiple local minima. Some are nonseparable, which means that they cannot be written as a sum of functions of individual variables. Some are regular, which means they are analytical (differentiable) at each point of their domain. Each of the functions in this study has 20 independent variables. The functions are summarized in Table I. More information about these functions, including their domains, can be found in [8], [30], and [31].

The benchmarks were compared by implementing integer versions of all the optimization algorithms in Matlab. The granularity or precision of each benchmark function was 0.1, except for the quartic function. Since the domain of each dimension of the quartic function was only ± 1.28, it was implemented with a granularity of 0.01.

We did some rough tuning on each of the optimization algorithms to get reasonable performance, but we did not make any special efforts to fine-tune the algorithms. For ACO, we used the following parameters: initial pheromone value $\tau_0 = 1E-6$, pheromone update constant $Q = 20$, exploration constant $q_0 = 1$, global pheromone decay rate $\rho_g = 0.9$, local pheromone decay rate $\rho_l = 0.5$, pheromone sensitivity $\alpha = 1$, and visibility sensitivity $\beta = 5$. For BBO, we used the following parameters: habitat modification probability $= 1$, immigration probability bounds per gene = $[0, 1]$, step size for numerical integration of probabilities $= 1$, maximum immigration and migration rates for each island $= 1$, and mutation probability $= 0$. (For BBO mutation is beneficial primarily for small population sizes.) For DE, we used a weighting factor $F = 0.5$ and a crossover constant $CR = 0.5$. For the ES, we produced $\lambda = 10$ offspring each generation, and standard deviation $\sigma = 1$ for changing solutions. For the GA, we used roulette wheel selection, single point crossover with a crossover probability of 1, and a mutation probability of 0.01. For PBIL, we used a learning rate of 0.05, 1 good population member and 0 bad population members to use to update the probability vector each generation, an elitism parameter of 1, and a 0 probability vector mutation rate. For PSO, we used only global learning (no local neighborhoods), an inertial constant $= 0.3$, a cognitive constant $= 1$, and a social constant for swarm interaction $= 1$. For the SGA, we used

$$\{2,2,3,3\}, \{1,2,3,3\}, \{1,2,2,3\}, \{1,1,3,3\}, \{1,1,2,3\}, \{1,1,2,2\} \tag{19}$$

TABLE I
BENCHMARK FUNCTIONS. THE GRANULARITY OF EACH DOMAIN WAS 0.1 EXCEPT FOR THE QUARTIC FUNCTION, WHICH HAD A GRANULARITY OF 0.01

Name	Multimodal?	Separable?	Regular?	Range of each Dimension
Ackley	yes	no	yes	± 30
Fletcher-Powell	yes	no	no	$\pm \pi$
Griewank	yes	no	yes	± 600
Penalty #1	yes	no	yes	± 50
Penalty #2	yes	no	yes	± 50
Quartic	no	yes	yes	± 1.28
Rastrigin	yes	yes	yes	± 5.12
Rosenbrock	no	no	yes	± 2.048
Schwefel 1.2	no	no	yes	± 65.536
Schwefel 2.21	no	no	no	± 100
Schwefel 2.22	yes	no	no	± 10
Schwefel 2.26	yes	yes	no	± 512
Sphere	no	yes	yes	± 5.12
Step	no	yes	no	± 200

TABLE II
MEAN NORMALIZED OPTIMIZATION RESULTS AND CPU TIMES ON BENCHMARK FUNCTIONS. THE NUMBERS SHOWN ARE THE MINIMUM FUNCTION VALUES FOUND BY THE ALGORITHMS, AVERAGED OVER 100 MONTE CARLO SIMULATIONS, AND NORMALIZED SO THAT THE SMALLEST NUMBER IN EACH ROW IS 100. NOTE THAT THESE ARE NOT THE ABSOLUTE MINIMA FOUND BY EACH ALGORITHM, BUT THE *AVERAGE* MINIMA FOUND BY EACH ALGORITHM

	ACO	BBO	DE	ES	GA	PBIL	PSO	SGA
Ackley	182	**100**	146	197	197	232	192	103
Fletcher	1013	**100**	385	494	415	917	799	114
Griewank	162	117	272	696	516	2831	1023	**100**
Penalty #1	2.22E7	1.16E4	9.70E4	1.26E6	2.46E5	2.82E7	2.09E6	**100**
Penalty #2	5.02E5	715	5862	4.23E4	1.06E4	5.37E5	6.35E4	**100**
Quartic	3213	262	1176	7008	2850	4.81E4	8570	**100**
Rastrigin	454	**100**	397	536	421	634	470	134
Rosenbrock	1711	102	253	716	428	1861	516	**100**
Schwefel 1.2	202	**100**	391	425	166	606	592	110
Schwefel 2.21	161	**100**	227	162	184	265	179	146
Schwefel 2.22	688	**100**	290	1094	500	861	665	142
Schwefel 2.26	108	118	137	140	142	177	142	**100**
Sphere	1347	**100**	250	910	906	2785	1000	109
Step	248	112	302	813	551	3271	1161	**100**
CPU Time	3.2	2.4	3.3	2.3	2.1	**1.0**	2.9	2.1

single point crossover with a crossover probability of 1, and a mutation probability of 0.01.

Each algorithm had a population size of 50, an elitism parameter of 2 (unless noted otherwise in the previous paragraph), and ran for 50 generations. We ran 100 Monte Carlo simulations of each algorithm on each benchmark to get representative performances. Tables II and III shows the results of the simulations. Table II shows the *average* minima found by each algorithm, averaged over 100 Monte Carlo runs. Table III shows the absolute best minima found by each algorithm over 100 Monte Carlo runs. In other words, Table II shows the *average* performance of each algorithm, while Table III shows the *best* performance of each algorithm. Note that the normalizations in the tables are based on different scales, so numbers cannot be compared between the two tables.

From Table II, we see that BBO and SGA both performed the best (on average) on seven of the 14 benchmarks. Table III shows that SGA was the most effective at finding function minima when multiple runs are made, performing the best on seven of the 14 benchmarks. BBO was the second most effective, performing the best on four of the benchmarks, while ACO performed the best on three of the benchmarks.

Benchmark results must always be taken with a grain of salt. First, we did not make any special effort to tune the optimization algorithms in this section. Different tuning parameter values in the optimization algorithms might result in significant changes in their performance. Second, real-world optimization problems may not have much of a relationship to benchmark functions. Third, benchmark tests might result in different conclusions if the grading criteria or problem setup change. In this section, we examined the mean and best results attained with a certain population size and after a certain number of generations. However, we might arrive at different conclusions if (for example) we change the generation limit, or look at how many generations it takes to reach a certain function value, or if we change the population size. In spite of these caveats, the benchmark results shown here are promising for BBO, and indicate that this new paradigm might be able to find a niche among the plethora of population-based optimization algorithms.

The computational requirements of the eight optimization methods were similar. We collected the average computational time of the optimization methods as applied to the 14 benchmarks discussed in this section. The results are shown in Table II. PBIL was the quickest optimization method. BBO was

TABLE III
BEST NORMALIZED OPTIMIZATION RESULTS ON BENCHMARK FUNCTIONS. THE NUMBERS SHOWN ARE THE BEST RESULTS FOUND AFTER 100 MONTE CARLO SIMULATIONS OF EACH ALGORITHM, AND NORMALIZED SO THAT THE SMALLEST NUMBER IN EACH ROW IS 100. NOTE THAT THESE ARE THE ABSOLUTE BEST MINIMA FOUND BY EACH ALGORITHM

	ACO	BBO	DE	ES	GA	PBIL	PSO	SGA
Ackley	205	**100**	178	220	224	325	262	114
Fletcher	1711	109	527	544	632	1947	1451	**100**
Griewank	240	181	576	1081	404	4665	2241	**100**
Penalty #1	**100**	3660	2.67E5	5.47E7	6198	1.65E10	4.05E7	1090
Penalty #2	**100**	4651	3.42E7	4.69E8	8.79E5	2.60E10	1.13E9	4878
Quartic	1.64E4	432	4847	2.50E4	4378	1.57E5	3.51E4	**100**
Rastrigin	541	**100**	502	564	466	798	544	123
Rosenbrock	2012	**100**	418	615	443	2696	558	103
Schwefel 1.2	391	174	1344	1209	186	2091	1742	**100**
Schwefel 2.21	259	109	571	381	249	597	307	**100**
Schwefel 2.22	779	**100**	374	560	468	1297	670	142
Schwefel 2.26	**100**	119	215	174	161	231	188	104
Sphere	1721	115	278	111	751	5196	1445	**100**
Step	279	106	585	1155	530	5595	1580	**100**

the fifth fastest of the eight algorithms. However, it should be noted that in the vast majority of real-world applications, it is the fitness function evaluation that is by far the most expensive part of a population-based optimization algorithm.

B. Sensor Selection Results

The sensor selection problem can be solved with population-based optimization methods. A population member consists of a vector of integers, with each element in the vector representing a sensor number. The fitness or HSI of a population member is given by (17) with $\alpha = 1$. If an invalid sensor set arises during the optimization process due to too many of a certain sensor type, then we replace some of the duplicated sensor types with a randomly chosen sensor to enforce feasibility.

We assumed here that we could use a total of 20 sensors (out of our unique 11 sensors) with each sensor being used no more than four times. The total number of sensor sets to choose from is the coefficient of x^{20} in the polynomial

$$q(x) = (1 + x + x^2 + x^3 + x^4)^{11}. \tag{21}$$

The coefficient of x^{20} in this polynomial is equal to 3 755 070. That is the total number of sensor sets that must be searched in order to find the minimum value of J in (17). In order to compute J for a single sensor set, we need to solve for Σ for that sensor set. In order to solve for Σ, we need to solve a discrete algebraic Riccati equation (DARE) [11]. This can be done with the DARE function in Matlab's Control System Toolbox. A DARE solution with 13 states (the three original states plus the ten health parameters) and 20 measurements takes 0.02 s on an admittedly outdated 1.2 GHz personal computer. So in order to search all 3 755 070 sensor sets, we require about 21 h of CPU time. Note that the minimum cost sensor set and its cost will be computer-dependent because of numerical issues in Matlab's DARE computation. Twenty-one hours of CPU time is not unreasonable if it only needs to be done once. However, if it needs to be done many times (once for 20 sensors, once for 19 sensors, once for 21 sensors, etc.), or if it needs to be done repeatedly as different aspects of the problem change (signal-to-noise ratios, system operating point, etc.), then the CPU time quickly becomes impractical.

TABLE IV
OPTIMIZATION RESULTS FOR THE SENSOR SELECTION PROBLEM. THE NUMBERS SHOWN ARE THE MINIMUM FUNCTION VALUES FOUND BY EACH ALGORITHM AVERAGED OVER 100 MONTE CARLO SIMULATIONS, AND THE BEST SOLUTIONS FOUND DURING THOSE 100 SIMULATIONS

	ACO	BBO	DE	ES	GA	PBIL	PSO	SGA
Mean Minimum	8.22	**8.01**	8.06	8.15	8.04	8.18	8.14	8.02
Best Minimum	8.12	**7.19**	7.60	8.05	8.02	8.08	8.06	8.02

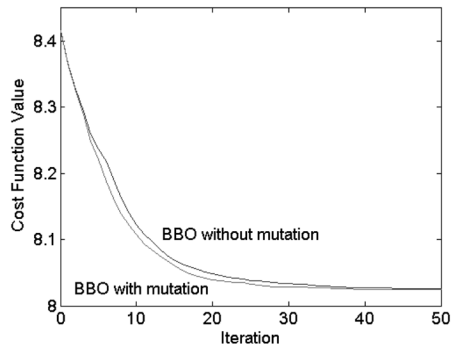

Fig. 4. Average sensor selection results of BBO without mutation, and BBO with probability-based mutation.

Instead of a brute-force 21-h search, we can use computer intelligence to find a near-optimal sensor set. We implemented population-based optimization algorithms to search for the best sensor set. The algorithms we used were the same as those used for the benchmark tests in Section V-A. For BBO, we used the algorithm given in Section III-C. For each algorithm, we used a population size of 50, a generation count of 100, and an elitism count of 2. One run of each optimization algorithm therefore required 4802 DARE evaluations, a computational savings (relative to an exhaustive search) of approximately 99.87%.

Table IV shows the results of the optimization methods on the sensor selection problem. We see that BBO performs the best in terms of both average performance and best performance.

Fig. 4 shows the results of the BBO search with and without probability-based mutation (see Section III-B) when the population size is 20. The figure shows the results of each method averaged over 100 Monte Carlo simulations. We see that the performances of the methods are comparable, but BBO with

probability-based mutation is clearly better than BBO without mutation. Note that we used a small population size for Fig. 4. Mutation can be detrimental for large population sizes, but with small population sizes mutation helps increase diversity and increases the changes for a good solution.

These simulation results should not be taken to mean that BBO is "better" than other population-based optimization algorithms. Such a general statement would be an oversimplification, especially in view of the no free lunch theorem [32]. However, the results presented here show that BBO provides better performance than most of the other algorithms we tested for the particular benchmarks that we examined. The results shown here indicate that BBO is at least competitive with other population-based optimization algorithms, and could provide a valuable tool for practical problems.

VI. Conclusion

We have shown how biogeography, the study of the geographical distribution of biological species, can be used to derive algorithms for optimization. This new family of algorithms is called BBO. We have applied BBO to benchmark functions and to a sensor selection problem, and shown that it provides performance on a par with other population-based methods. We cannot conclude that BBO is universally better than other methods, or *vice versa*, in view of the no free lunch theorem. However, it may be possible in future work to quantify the performance of BBO relative to other algorithms for problems with specific features. The good performance of BBO on the benchmarks and the sensor selection problem provides some evidence that BBO theory can be successfully applied to practical problems. This paper is preliminary in nature and, therefore, opens up a wide range of possibilities for further research.

It would be interesting to prove the conjecture in Section II about the eigenvalues of A'. The matrix A' has a very special structure that has apparently not yet appeared in the literature. The properties of A' could have important implications for the behavior of BBO with respect to stability, convergence, equilibria, and other issues.

Another important extension of this work would be to apply BBO to the optimization of problems with dynamic fitness landscapes. This could be done by using optimal filters to estimate solution fitnesses, similar to what has been suggested for GAs [33].

It might be fruitful to explore the idea of species sharing only between similar solutions (neighboring habitats). Species are more likely to migrate to habitats that are close to their place of origin. This is similar to niching in GAs [23] (where subspecies do not compete with each other), and is also reminiscent of the speciating island model [34].

The details of the species model in Fig. 1 could be adjusted to improve optimization performance. We used linear and symmetric immigration and emigration curves, but perhaps other shapes could give better performance under certain conditions. In addition, it could be supposed that a habitat must have a minimum nonzero HSI in order to support any species, which would give a species count lower bound that is greater than zero [4].

We formulated BBO to optimize functions of discrete variables. It would be valuable to modify the BBO algorithm so that it could be used to directly optimize functions of continuous variables.

We have seen that BBO has features in common with other population-based methods. These connections should be explored further. Under what conditions might BBO be equivalent to these other methods?

An issue that has not been explored in this paper is that the reproductive value of an individual as a function of its age looks like a triangular function. Reproductive value is low at young ages (due to infant mortality), high at child-bearing ages, and low again at old ages (due to loss of fertility). The same could be said of species. A young species has a chance of being poorly adapted to its environment and so has only a small chance of speciating, a middle-aged species is both mature enough and dynamic enough to speciate, and an old species is too stagnant to speciate. This could lead to the introduction of an age criterion in BBO, similar to that which has been used in GAs [35].

Other approaches and aspects of biogeography could inspire variants to the BBO suggested in this paper. The biogeography literature is so rich that there are many possibilities along these lines. For example, how can population sizes be incorporated into BBO? How can predator/prey relationships be incorporated into BBO? How can variations in species mobilities be incorporated into BBO? How can the evolution of migration rate for a particular species be incorporated into BBO? How can population models be incorporated into BBO [36], [37]?

We note that CPU time is a bottleneck to the implementation of many population-based optimization algorithms. If an algorithm does not converge rapidly, it will be impractical, since it would take too long to find a near-optimal solution. BBO does not seem to require an unreasonable amount of computational effort; of the eight optimization algorithms compared in this paper, BBO was the fifth fastest. Nevertheless, finding mechanisms to speed up BBO could be an important area for further research. For example, perhaps knowledge could be incorporated to replace selected SIVs in a way such that the modified solution is always better than the original solution.

Another bottleneck to population based optimization algorithms, and one that is related to computational effort, is the problem of creating infeasible solutions. In BBO as presented here, it is not possible to check for feasibility while a new solution is being completed. The feasibility check has to wait until after the new solution is already complete. This procedure may result in creating too many infeasible solutions and may slow down the algorithm considerably. We conclude that finding mechanisms to ensure feasibility during solution generation could be an important area for further research. For example, perhaps knowledge could be incorporated to replace selected SIVs in a way such that the modified solution is always feasible. Note that this suggestion (in general) also applies to other population based optimization algorithms. This paper has introduced a new optimization tool that can hopefully be applied to many different types of problems. Almost every problem in engineering (and in life) can be interpreted as an optimization problem [38]. The new optimization algorithm introduced here opens up promising avenues of productive research. The software that was used to generate the results shown in this paper is available at http://academic.csuohio.edu/simond/bbo.

APPENDIX

This appendix provides a proof of Theorem 1. If the species count probabilities are in steady-state, then from (3), we have $AP(\infty) = 0$. Taking the singular value decomposition (SVD) [39] of A in this equation gives $U\Sigma V^H P(\infty) = 0$. (We use the H superscript to indicate the Hermitian transpose of a matrix.) Since U in an SVD is always nonsingular, this implies that

$$\Sigma V^H P(\infty) = 0. \tag{22}$$

Combining (7) with Observation 1 shows us that A has rank n. Therefore, $A^H A$ also has rank n, which means that the singular value matrix Σ has n nonzero diagonal elements and one zero diagonal element (the lower right element in Σ is zero). Combining this information with (22) shows that

$$V^H P(\infty) = \begin{bmatrix} 0 & \dots & 0 & 1 \end{bmatrix}^T. \tag{23}$$

Since V in an SVD is always a unitary matrix, this equation implies that $P(\infty)$ is equal to the last column of V multiplied by some scalar. However, from SVD theory, we know that the last column of V is equal to the eigenvector that corresponds to the zero eigenvalue of $A^H A$. We know that v in (8) is the eigenvector that corresponds to the zero eigenvalue of A. That means that $Av = 0$, which means that $A^H Av = 0$, which means that v in (8) is the eigenvector that corresponds to the zero eigenvalue of $A^H A$. Therefore $P(\infty)$ is equal to v multiplied by some scalar. The elements of $P(\infty)$ must add up to one, and so we obtain (12). QED

ACKNOWLEDGMENT

The comments of the reviewers were instrumental in improving this paper from its original version.

REFERENCES

[1] A. Wallace, *The Geographical Distribution of Animals (Two Volumes)*. Boston, MA: Adamant Media Corporation, 2005.
[2] C. Darwin, *The Origin of Species*. New York: Gramercy, 1995.
[3] R. MacArthur and E. Wilson, *The Theory of Biogeography*. Princeton, NJ: Princeton Univ. Press, 1967.
[4] I. Hanski and M. Gilpin, *Metapopulation Biology*. New York: Academic, 1997.
[5] T. Wesche, G. Goertler, and W. Hubert, "Modified habitat suitability index model for brown trout in southeastern Wyoming," *North Amer. J. Fisheries Manage.*, vol. 7, pp. 232–237, 1987.
[6] A. Hastings and K. Higgins, "Persistence of transients in spatially structured models," *Science*, vol. 263, pp. 1133–1136, 1994.
[7] H. Muhlenbein and D. Schlierkamp-Voosen, "Predictive models for the breeder genetic algorithm: I. Continuous parameter optimization," *Evol. Comput.*, vol. 1, pp. 25–49, 1993.
[8] T. Back, *Evolutionary Algorithms in Theory and Practice*. Oxford, U.K.: Oxford Univ. Press, 1996.
[9] K. Parker and K. Melcher, "The modular aero-propulsion systems simulation (MAPSS) users' guide," NASA, Tech. Memo. 2004-212968, 2004.
[10] D. Simon and D. L. Simon, "Kalman filter constraint switching for turbofan engine health estimation," *Eur. J.Control*, vol. 12, pp. 331–343, May 2006.
[11] D. Simon, *Optimal State Estimation*. New York: Wiley, 2006.
[12] R. Mushini and D. Simon, "On optimization of sensor selection for aircraft gas turbine engines," in *Proc. Int. Conf. Syst. Eng.*, Las Vegas, NV, Aug. 2005, pp. 9–14.
[13] C. Chuan-Chong and K. Khee-Meng, *Principles and Techniques in Combinatorics*. Singapore: World Scientific, 1992.
[14] M. Dorigo and T. Stutzle, *Ant Colony Optimization*. Cambridge, MA: MIT Press, 2004.
[15] M. Dorigo, L. Gambardella, M. Middendorf, and T. Stutzle, Eds., "Special section on 'ant colony optimization'," *IEEE Trans. Evol. Comput.*, vol. 6, no. 4, pp. 317–365, Aug. 2002.
[16] C. Blum, "Ant colony optimization: Introduction and recent trends," *Phys. Life Reviews*, vol. 2, pp. 353–373, 2005.
[17] G. Onwubolu and B. Babu, *New Optimization Techniques in Engineering*. Berlin, Germany: Springer-Verlag, 2004.
[18] K. Price and R. Storn, "Differential evolution," *Dr. Dobb's Journal*, vol. 22, pp. 18–20, 22, 24, 78, Apr. 1997.
[19] R. Storn, "System design by constraint adaptation and differential evolution," *IEEE Trans. Evol. Comput.*, vol. 3, pp. 22–34, Apr. 1999.
[20] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*. New York: Springer, 1992.
[21] H. Beyer, *The Theory of Evolution Strategies*. New York: Springer, 2001.
[22] E. Mezura-Montes and C. Coello, "A simple multimembered evolution strategy to solve constrained optimization problems," *IEEE Trans. Evol. Comput.*, vol. 9, pp. 1–17, Feb. 2005.
[23] D. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA: Addison-Wesley, 1989.
[24] I. Parmee, *Evolutionary and Adaptive Computing in Engineering Design*. New York: Springer, 2001.
[25] , D. Dasgupta and Z. Michalewicz, Eds., *Evolutionary Algorithms in Engineering Applications*. New York: Springer, 2001.
[26] R. Eberhart, Y. Shi, and J. Kennedy, *Swarm Intelligence*. San Mateo, CA: Morgan Kaufmann, 2001.
[27] R. Eberhart and Y. Shi, "Special issue on particle swarm optimization," *IEEE Trans. Evol. Comput.*, vol. 8, no. 3, pp. 201–228, Jun. 2004.
[28] M. Clerc, *Particle Swarm Optimization*. Amsterdam, The Netherlands: ISTE Publishing, 2006.
[29] W. Khatib and P. Fleming, "The stud GA: A mini revolution?," in *Parallel Problem Solving from Nature*, A. Eiben, T. Back, M. Schoenauer, and H. Schwefel, Eds. New York: Springer, 1998.
[30] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," *IEEE Trans. Evol. Comput.*, vol. 3, pp. 82–102, Jul. 1999.
[31] Z. Cai and Y. Wang, "A multiobjective optimization-based evolutionary algorithm for constrained optimization," *IEEE Trans. Evol. Comput.*, vol. 10, pp. 658–675, Dec. 2006.
[32] Y. Ho and D. Pepyne, "Simple explanation of the no-free-lunch theorem and its implications," *J. Opt. Theory Appl.*, vol. 155, pp. 549–570, 2002.
[33] P. Stroud, "Kalman-extended genetic algorithm for search in nonstationary environments with noisy fitness evaluations," *IEEE Trans. Evol. Comput.*, vol. 5, pp. 66–77, 2001.
[34] S. Gustafson and E. Burke, "Speciating island model: An alternative parallel evolutionary algorithm," *Parallel and Distributed Computing*, vol. 66, pp. 1025–1036, 2006.
[35] Y. Zhu, Z. Yang, and J. Song, "A genetic algorithm with age and sexual features," in *Proc. Int. Conf. Intell. Comput.*, 2006, pp. 634–640.
[36] H. Caswell, *Matrix Population Models*. Sunderland, MA: Sinauer Associates, 1989.
[37] C. Li and S. Schreiber, "On dispersal and population growth for multistate matrix models," *Linear Algebra and Its Applications*, vol. 418, pp. 900–912, 2006.
[38] D. Bernstein, "Optimization r us," *IEEE Control Systems Mag.*, vol. 26, pp. 6–7, 2006.
[39] B. Noble and J. Daniel, *Applied Linear Algebra*. Englewood Cliffs, NJ: Prentice-Hall, 1987.

Dan Simon (S'89–M'90–SM'01) received the B.S. degree from Arizona State University, Tempe, the M.S. degree from the University of Washington, Seattle, and the Ph.D. degree from Syracuse University, Syracuse, NY, all in electrical engineering.

He worked in industry for 14 years at Boeing, TRW, and several smaller companies. His industrial experience includes work in the aerospace, automotive, agricultural, GPS, biomedical, process control, and software fields. In 1999, he moved from industry to academia, where he is now an Associate Professor in the Electrical and Computer Engineering Department, Cleveland State University. His teaching and research involves embedded systems, control systems, and computer intelligence. He has published over 60 refereed conference and journal papers, and is the author of the text *Optimal State Estimation* (Wiley, 2006).