Appears in "Computer Science Today: Recent Trends and Developments", Ed: J. van Leeuwen, Springer-Verlag LNCS 1000, pp275-291, 1995.

Fundamental Limitations on Search Algorithms: Evolutionary Computing in Perspective

Nicholas J. Radcliffe[a,b] & Patrick D. Surry[a,b]
{njr,pds}@quadstone.co.uk

[a]Quadstone Ltd, 16 Chester Street, Edinburgh, EH3 7RA, UK
[b]Department of Mathematics, University of Edinburgh, The Kings Buildings, EH9 3JZ, UK

Abstract. The past twenty years has seen a rapid growth of interest in stochastic search algorithms, particularly those inspired by natural processes in physics and biology. Impressive results have been demonstrated on complex practical optimisation problems and related search applications taken from a variety of fields, but the theoretical understanding of these algorithms remains weak. This results partly from the insufficient attention that has been paid to results showing certain fundamental limitations on universal search algorithms, including the so-called "No Free Lunch" Theorem. This paper extends these results and draws out some of their implications for the design of search algorithms, and for the construction of useful representations. The resulting insights focus attention on tailoring algorithms and representations to particular problem classes by exploiting domain knowledge. This highlights the fundamental importance of gaining a better theoretical grasp of the ways in which such knowledge may be systematically exploited as a major research agenda for the future.

1 Overview

The last two decades has seen increasing interest in the application of stochastic search techniques to difficult problem domains. Strong biological metaphors led to the development of several schools of evolutionary computation, including work on genetic algorithms and classifier systems (Holland, 1975), evolution strategies (Rechenberg, 1973, 1984; Baeck & Schwefel, 1993), evolutionary programming (Fogel *et al.,* 1966) and genetic programming (Koza, 1991). Physical analogues provided both the motivation and theoretical framework for the method of simulated annealing (Kirkpatrick *et al.,* 1983). Other randomised methods, such as tabu search (Glover, 1986) and a variety of stochastic hill climbers and related search techniques have also attracted much interest.

Although some theoretical progress has been made in the study of stochastic search techniques, most attention has focused on the artificial situation of arbitrary ("black-box") search problems. Despite occasional warnings from various researchers (Vose & Liepins, 1991; Radcliffe, 1992, 1994; Wolpert & Macready, 1995) a great deal of research seems oblivious to the fact that in such a situation there is no scope for distinguishing any of these biased sampling methods from enumeration—random or fixed—as we demonstrate below. There are exceptions to this, for example the convergence results for simulated annealing which exploit knowledge of the problem domain (e.g. Laarhoven & Aarts, 1989, Shapiro *et al.,* 1994) but much effort continues to be devoted

to the "general" case, (Vose, 1992; Goldberg, 1989c, 1989a, 1989b). While such studies lead to elegant mathematics, and provide occasional insights into stochastic search, their practical utility in guiding—or even making contact with—practitioners tackling real search problems remains to be demonstrated.

This paper explicitly demonstrates these fundamental limitations on search algorithms, and highlights the necessity of theory incorporating knowledge of the problem domain of interest. Much interest in this topic has been aroused lately as awareness has grown of the so-called "No Free Lunch Theorem" (Wolpert & Macready, 1995), which—broadly—states that if a sufficiently inclusive class of problems is considered, no search method can outperform an enumeration. This paper discusses the consequences of this and other fundamental limitations on search for the practical design of stochastic search algorithms, particularly evolutionary algorithms, and emphasizes that there are many possible improvements that are *not* strongly limited by the theorems.

A more accessible and general form of the No Free Lunch Theorem is first presented, based on arguments put forward previously (Radcliffe, 1992, 1994). Strictly, this result shows that over the ensemble of all representations of one space with another, all algorithms (in a rather broad class) perform identically on any reasonable measure of performance. This has as an immediate corollary the No Free Lunch Theorem, and provides an interesting context for the "minimax" results given in Wolpert & Macready (1995).

Having proved the main results, attention turns to a discussion of their implications. First, issues concerning searches that re-sample points previously visited are discussed. After this, representational issues are discussed and restricted problem classes are considered in two stages. The first focuses on the case in which some unspecified representation is used which is known—in an appropriately defined sense—to fit the problem class well, and it is shown that different algorithms can be expected to perform differently in these cases. The second focuses on how knowledge concerning the structure of some class of functions under consideration can be used to choose an effective algorithm (consisting of a representation, a set of move operators and a sampling strategy). In particular, the critical rôle of representation in determining the potential efficacy of search algorithms is once again highlighted.

Finally, an agenda for future research is set forward, focusing on integrating the restrictions stated herein with the wide scope for theory incorporating domain-specific knowledge. The authors draw attention to what they feel are important unanswered questions concerning performance measurement, comparison of families of stochastic algorithms and difficulty of real-world problem domains, and suggest a new methodology for comparing search algorithms.

2 Terminology

2.1 Search Space Concepts

Before formulating theorems about search, it will be useful to introduce some terminology, to reduce the scope for misinterpretation. A search problem will be taken to be characterised by a *search space,* $\mathcal{S}$—the set of objects over which the search is to be conducted—and an *objective function,* f, which is a mapping from $\mathcal{S}$ to the space of

objective function values, $\mathcal{R}$, i.e.

$$f : \mathcal{S} \longrightarrow \mathcal{R}. \tag{1}$$

Although the most familiar situation arises when $\mathcal{R} = \mathbb{R}$ (the set of real numbers) and the goal is minimisation or maximisation of f, it is not necessary to require this for present purposes. For example, $\mathcal{R}$ could instead be a vector of objective function values, giving rise to a multicriterion optimisation.

Note that the set of all mappings from $\mathcal{S}$ to $\mathcal{R}$ is denoted $\mathcal{R}^{\mathcal{S}}$, so $f \in \mathcal{R}^{\mathcal{S}}$.

2.2 Representation

The issue of selecting an appropriate representation is central to search, as arguments in this paper will once again confirm. The present section gives a formal definition of a representation space, but further motivation for concentrating on representation as an issue is given in section 5.

A *representation of* $\mathcal{S}$ will be taken to be a set $\mathcal{C}$ (the *representation space*, or the space of *chromosomes*) of size at least equal to $\mathcal{S}$, together with a *surjective* function g (the *growth function*) that maps $\mathcal{C}$ onto $\mathcal{S}$:

$$g : \mathcal{C} \longrightarrow \mathcal{S}. \tag{2}$$

(Surjectivity is simply the requirement that g maps at least one point in $\mathcal{C}$ to each point in $\mathcal{S}$.) The set of surjective functions from $\mathcal{C}$ to $\mathcal{S}$ will be denoted $\mathcal{S}^{\mathcal{C}}_{>}$, so $g \in \mathcal{S}^{\mathcal{C}}_{>} \subset \mathcal{S}^{\mathcal{C}}$.

If $\mathcal{C}$ and $\mathcal{S}$ have the same size, then g is clearly invertible, and will be said to be a *faithful representation.*

Objective function values will also be associated with points in the representation space $\mathcal{C}$ through composition of f with g,

$$f \circ g : \mathcal{C} \longrightarrow \mathcal{R} \tag{3}$$

being given by

$$f \circ g(x) \stackrel{\triangle}{=} f(g(x)). \tag{4}$$

The relationship between $\mathcal{C}$, $\mathcal{S}$ and $\mathcal{R}$ is shown in figure 1.

2.3 Sequences

Given any set $\mathcal{A}$, $\mathbb{S}(\mathcal{A})$ will be used to denote the set of all finite sequences over $\mathcal{A}$, including the empty sequence $\langle\rangle$, i.e.:

$$\mathbb{S}(\mathcal{A}) \stackrel{\triangle}{=} \left\{ \langle a_1, a_2, \ldots, a_n \rangle \;\middle|\; a_i \in \mathcal{A}, 1 \leq i \leq n \right\}. \tag{5}$$

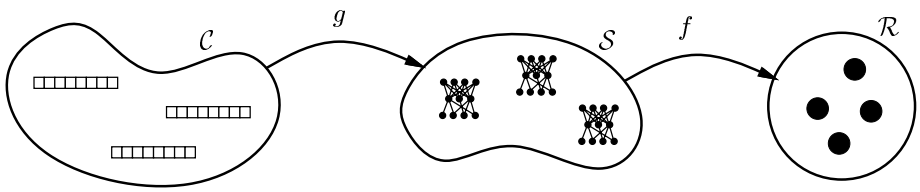

Fig. 1. The *growth function* g maps the *representation space* $\mathcal{C}$ onto the *search space* $\mathcal{S}$, which is mapped to the space of *objective function values* $\mathcal{R}$ by the *objective function* f. If g is invertible, the representation is said to be *faithful.*

2.4 Permutations

Most of the "work" in the following theorems is achieved by using a permutation, i.e. a re-labelling of the elements of one of the spaces (usually $\mathcal{C}$). A permutation of a set $\mathcal{A}$ is a relabelling of the elements of the set, i.e. an *invertible* mapping

$$\pi : \mathcal{A} \longrightarrow \mathcal{A}. \tag{6}$$

The set of all permutations of $\mathcal{A}$ objects will be denoted $\mathcal{P}(\mathcal{A})$. The changes to a faithful representation of a search space $\mathcal{S}$ by a representation space $\mathcal{C}$ under all possible permutations of the elements of $\mathcal{C}$ are shown in figure 2.

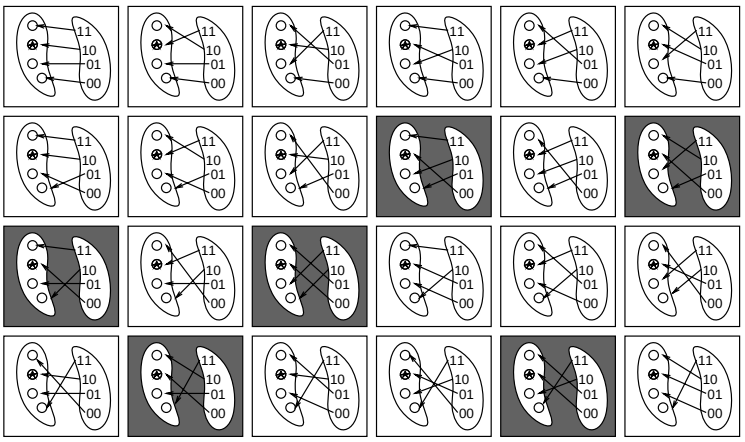

Fig. 2. This figure shows the $4!$ invertible mappings between a representation space $\mathcal{C}$ (in this case binary strings) and an arbitrary search space $\mathcal{S}$ of the same size. Note that $1/4$ of the mappings (each shown in grey) map the global optimum (shown with a star) to the string 00.

2.5 Search Algorithms

For the purposes of the present paper, a *deterministic search algorithm* will be defined by a function which, given a sequence of points $\langle x_i \rangle$, with each $x_i \in \mathcal{C}$, and the corresponding sequence of objective function values for those points under the composed objective function $f \circ g$, generates a new point $x_{n+1} \in \mathcal{C}$. Thus, a search algorithm is defined by a mapping A

$$A : \mathbb{S}(\mathcal{C}) \times \mathbb{S}(\mathcal{R}) \longrightarrow \mathcal{C}. \tag{7}$$

The first point in the search sequence $\langle A_i \rangle \in \mathbb{S}(\mathcal{C})$ associated with the algorithm defined by A is

$$A_1 \triangleq A(\langle\rangle, \langle\rangle), \tag{8}$$

with subsequent points being recursively defined by

$$A_{n+1} \triangleq A(\langle A_i \rangle_{i=1}^{n}, \langle f \circ g(A_i) \rangle_{i=1}^{n}). \tag{9}$$

A *stochastic search algorithm* will be taken to be exactly the same as a deterministic search algorithm except that it also uses a seed for a pseudo-random number generator used to influence its choice of the next point x_{n+1}. Formally, a stochastic search algorithm is then defined by a mapping A' of the form

$$A' : \mathbb{S}(\mathcal{C}) \times \mathbb{S}(\mathcal{R}) \times \mathbb{Z} \longrightarrow \mathcal{C}. \tag{10}$$

(The seed is of course a constant, the same at every application of A'.) This process is illustrated in figures 3 and 4.

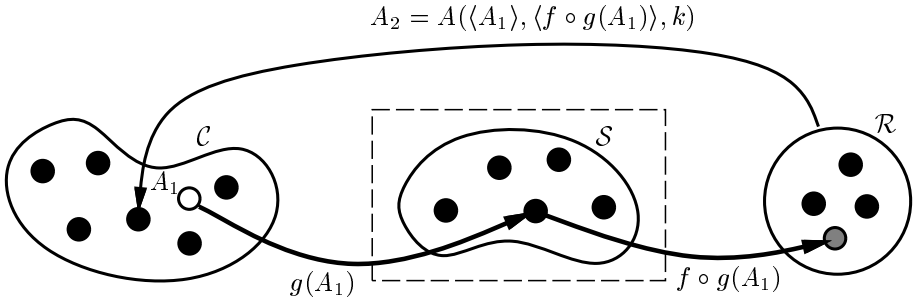

Fig. 3. The first point in the search is determined by the algorithm and the seed, $A_1 = A(\langle\rangle, \langle\rangle, k)$. The next point, A_2 is determined by these values together with the new information $f \circ g(A_1)$. Note that the algorithm has no information about the point $g(A_1)$ in $\mathcal{S}$ which was used to generate the observed function value: this is entirely controlled by the representation g.

Note that a stochastic search algorithm is perfectly at liberty to ignore the given seed, and in this sense will be taken to include the case of a deterministic search algorithm.

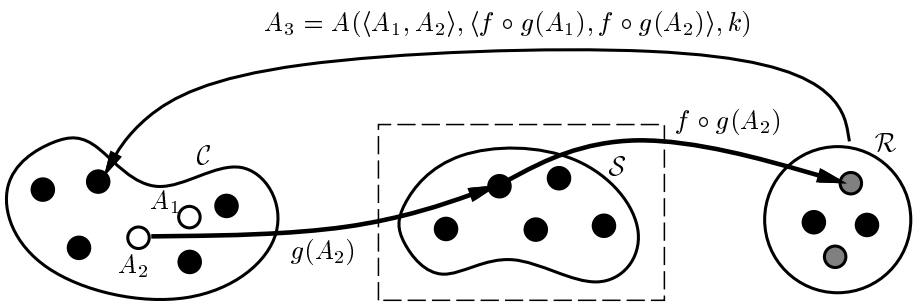

Fig. 4. The next point in the search is simply obtained by iterating the mapping. Notice that if $f \circ g(A_2) = f \circ g(A_1)$, the algorithm cannot "know" whether or not it visited a new point in $\mathcal{S}$ (i.e. whether $g(A_2) = g(A_1)$).

Because of this, results derived for the stochastic case also apply to the deterministic case. These definitions of algorithm are certainly broad enough to include all familiar classes of stochastic search algorithms as actually used (e.g. hill climbers, evolutionary algorithms, simulated annealing, tabu search). The issue of "insisting" on use of a representation is discussed further in section 5.

We shall abuse the notation slightly by identifying a search algorithm with the function that defines it, i.e. the search algorithm defined by the mapping A will also be denoted A.

Note particularly that in both cases, the sequence of points $\langle x_i \rangle$ is *not* assumed to be non-repeating except when this is explicitly stated. Further, search algorithms will always be assumed to run until they have searched the entire representation space $\mathcal{C}$, though performance measures will be at liberty use only some smaller number of function values (such as the first N).

2.6 Valid Search Sequence in $\mathcal{S}$

It will sometimes be convenient to refer to a *valid search sequence* in $\mathcal{S}$ in the context of some representation $g \in \mathcal{S}^{\mathcal{C}}_{>}$ and some non-repeating search algorithm A. A valid search sequence is simply one that can be generated as the image under g of a (non-repeating) search sequence in $\mathcal{C}$. For example, if $\mathcal{C}$ contains one more element that $\mathcal{S}$, each (complete) valid search sequence in $\mathcal{S}$ contains one member of $\mathcal{S}$ twice and all others exactly once.

2.7 Performance Measures

All the results below will refer rather cavalierly to "arbitrary performance measures", so it will be useful to make clear exactly what the assumed scope of such a performance measure actually is. A performance measure for a search algorithm will be taken to be

any measure that depends only on the sequence of objective function values of the images under g of the points in $\mathcal{C}$ chosen by the algorithm. Formally, a performance measure μ will be taken to be any function

$$\mu : \mathbb{S}(\mathcal{R}) \longrightarrow \mathcal{M} \tag{11}$$

where $\mathcal{M}$ is any set used to measure the "quality" of some sequence of objective function values. Thus, abusing the notation further, we will define the performance of an algorithm A to be

$$\mu(A) \triangleq \mu(\langle f \circ g(A_i) \rangle). \tag{12}$$

There is an important distinction between algorithms that revisit points in $\mathcal{C}$ and those that do not. The performance measures defined here exclude those that can distinguish between objective function values deriving from newly visited points in $\mathcal{C}$ and points previously visited. Rather, we regard the non-revisiting algorithm derived from an algorithm that *does* revisit points in $\mathcal{C}$ as a different algorithm with potentially different performance.

We will also consider the performance of an algorithm over an (unordered) set of functions. By an *overall performance measure* we will mean any measure of the quality of a collection of sequences of objective function values. Given that two different objective functions may generate the same sequence of values, this collection will, in general, be a multiset (bag) of such sequences (i.e. sequences may appear more than once in the collection). Notice also that the collection is taken to be *unordered*, so that an overall performance measure is insensitive to the order in which the functions are presented. A simple example of such an overall performance measure is the average number of function evaluations required to find a global optimum.

3 No Free Lunch Theorems

The first task in formulating limitations on search is to define what it means for two algorithms to be *isomorphic*. Very simply, the idea is that two algorithms are isomorphic if one can be obtained from the other by a permutation of the representation space. This is what the following definition says.

Definition (Isomorphism for search algorithms). Let A and B be non-repeating stochastic search algorithms with an objective function $f \in \mathcal{R}^{\mathcal{S}}$, a growth function $g \in \mathcal{S}^{\mathcal{C}}_{>}$ and (fixed) seeds k_A and k_B respectively. Then A and B will be said to be *isomorphic with respect to f and g* if and only if there exists a permutation $\pi \in \mathcal{P}(\mathcal{C})$ such that

$$\langle g(A_i) \rangle = \langle g \circ \pi(B_i) \rangle. \tag{13}$$

■

The idea of isomorphism between search algorithms is important because the following results will demonstrate that isomorphic algorithms perform equally in some extremely strong senses.

Theorem (Isomorphism for non-repeating algorithms). *Let A and B be non-repeating stochastic search algorithms using the same objective function f and the same representation space $\mathcal{C}$. Then for every representation $g \in \mathcal{S}^{\mathcal{C}}_{>}$, A and B are isomorphic with respect to f and g.*

Proof. Let g, g' be two different growth functions in $\mathcal{S}^{\mathcal{C}}_{>}$. Clearly, A must visit a different sequence of points in $\mathcal{S}$ under g and g'. (This may be clarified by looking again at figures 3 and 4. Since the first point, A_1, chosen by A is independent of the growth function, it is easy to see inductively that if A visited the same sequence of points in $\mathcal{S}$ under g and g', then g and g' would in fact be equal, as every point in $\mathcal{C}$ will eventually be visited by a non-terminating, non-repeating algorithm.) This shows that any valid search sequence in $\mathcal{S}$ under some $g \in \mathcal{S}^{\mathcal{C}}_{>}$ defines that g uniquely (for the algorithm A). Conversely each growth function g defines a unique search sequence in $\mathcal{S}$, namely $\langle g(A_i)\rangle$.

It is thus clear that for every $g \in \mathcal{S}^{\mathcal{C}}_{>}$, yielding search sequence $\langle g(A_i)\rangle$, there exists a $g' \in \mathcal{S}^{\mathcal{C}}_{>}$ such that $\langle g'(B_i)\rangle = \langle g(A_i)\rangle$. Further, it is easy to see that some permutation $\pi \in \mathcal{P}(\mathcal{C})$ exists such that $g' = g \circ \pi$—specifically, $\pi(B_i) = A_i$, which is a permutation of $\mathcal{C}$ since both $\langle A_i\rangle$ and $\langle B_i\rangle$ are non-repeating enumerations of $\mathcal{C}$. This is exactly the permutation required to demonstrate isomorphism of A and B with respect to f and g. □

Theorem (Isomorphic algorithms perform equally over permutations of $\mathcal{C}$). *Let A and B be stochastic search algorithms that are isomorphic with respect to some objective function $f \in \mathcal{R}^{\mathcal{S}}$ and some set of growth functions $\mathcal{G} \subset \mathcal{S}^{\mathcal{C}}_{>}$ that is closed under permutation of $\mathcal{C}$. Then their overall performance on the ensemble of search problems defined by all growth functions in $\mathcal{G}$ is identical, regardless of the performance measure chosen.*

Proof. For any valid search sequence $\langle g(A_i)\rangle \in \mathbb{S}(\mathcal{S})$, we can find a permutation $\pi \in \mathcal{P}(\mathcal{C})$ such that $\langle g \circ \pi(B_i)\rangle = \langle g(A_i)\rangle$ from the definition of isomorphism. By symmetry, the converse is also true, so every search sequence in $\mathcal{S}$ generated by either algorithm is also generated by the other. Since it was shown in the proof of the previous theorem that different growth functions define different search sequences, this suffices to show that the set of search sequences in $\mathcal{S}$ generated by $\mathcal{G}$ is the same for the two algorithms, so their performance over $\mathcal{G}$ must be equal. □

Perhaps the most interesting consequence of this theorem is that if the entire ensemble of representations in $\mathcal{S}^{\mathcal{C}}_{>}$ is considered, the only factor that differentiates algorithms is the frequency with which they revisit points. This is because, over this ensemble of representations, visiting one new point is no different from visiting any other.

Corollary (Overall Performance Measures). *Consider search algorithms in which the representation space $\mathcal{C}$ is the same as the search space $\mathcal{S}$. The overall performance of all search algorithms that do not revisit points in $\mathcal{C}$ on the ensemble of search problems defined by all objective functions in $\mathcal{R}^{\mathcal{S}}$ is identical, regardless of the overall performance measure chosen.*

■

Proof. For all functions in $\mathcal{R}^{\mathcal{S}}_{>}$, the result follows directly from the theorems by a trivial relabelling (figure 5) where we take $\mathcal{S} = \mathcal{R}$, f to be the identity mapping and $\mathcal{G}$ to be the set of all functions from $\mathcal{S}$ to $\mathcal{R}$. Similarly, for any set of functions surjective onto a subset of $\mathcal{R}$, the theorem still clearly holds taking $\mathcal{R}$ as the given subset of $\mathcal{R}$. Since the set of all functions in $\mathcal{R}^{\mathcal{S}}$ is clearly just the union over such surjective subsets, we have the required result.
■

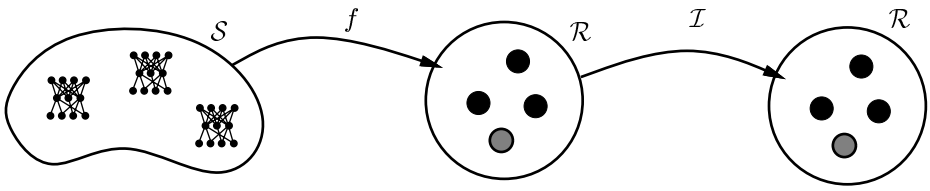

Fig. 5. The relabelling above, with $\mathcal{I}$ as the identity mapping, establishes that all isomorphic algorithms have identical overall performance over all functions in $\mathcal{R}^{\mathcal{S}}$ against any overall performance measure. If f is not surjective, the problem has to be reformulated using the image $f(\mathcal{S})$ in place of $\mathcal{R}$ for the theorem to apply directly, but since the theorem applies to each such image, it clearly also applies to their union. A further corollary to this is the "No Free Lunch" Theorem (Wolpert & Macready, 1995).

Corollary (The No Free Lunch Theorem). *The mean performance of all isomorphic search algorithms is identical over the set of all functions in $\mathcal{R}^{\mathcal{S}}$ for any chosen performance measure.*

Proof. This follows immediately from the previous corollary. □

Wolpert & Macready (1995) conjecture that there may be what they term "minimax" distinctions between pairs of algorithms. Here they have in mind "head-to-head" comparisons on a "function-by-function" basis, where, for example, one algorithm (A) might outperform another (B) on some performance measure more often than the reverse is the case. This is clearly true, as table 1 shows. Notice, however, that such comparisons may be non-transitive, i.e. given a third algorithm C, it might be that case that

$$(C > B \text{ and } B > A) \not\Rightarrow C > A, \tag{14}$$

where $>$ means "outperforms" in the minimax sense described above. An example of exactly this situation, where $C > B$ and $B > A$ but $A > C$ is also shown in the table.

Corollary (Enumeration). *No non-repeating search algorithm can outperform enumeration over the ensemble of problems defined by $\mathcal{R}^{\mathcal{S}}$.*

Proof. This is also immediate from the theorem, noting that no search can sample any proportion of the search space faster than enumeration. □

Function $f((1,2,3))$	Time to Minimum A	B	C	Winner A vs. B	Winner B vs. C	Winner C vs. A
(0,0,0)	1	1	1	tie	tie	tie
(0,0,1)	1	1	2	tie	B	A
(0,1,0)	1	2	1	A	C	tie
(0,1,1)	1	3	2	A	C	A
(1,0,0)	2	1	1	B	tie	C
(1,0,1)	2	1	3	B	B	A
(1,1,0)	3	2	1	B	C	C
(1,1,1)	1	1	1	tie	tie	tie
Overall winner				B	C	A

Table 1. Consider all functions from $\mathcal{S} = \{1,2,3\}$ to $\mathcal{R} = \{0,1\}$ and three search algorithms A, B and C, each of which simply enumerates $\mathcal{S}$. In particular, take $\langle A_i \rangle = \langle 1,2,3 \rangle$, $\langle B_i \rangle = \langle 2,3,1 \rangle$ and $\langle C_i \rangle = \langle 3,1,2 \rangle$. The table lists all eight functions in $\mathcal{R}^{\mathcal{S}}$ and compares the algorithms pairwise with respect to the particular performance measure 'number of steps to find a global minimum'. This illustrates the "minimax" distinction conjectured by Wolpert & Macready (1995), but also shows that this is non-transitive. Indeed, the overall performance of each algorithm is identical, regardless of what overall performance measure is chosen. (Columns 2, 3 and 4 are permutations of each other, illustrating this.)

4 Search Algorithms that Revisit Points

Before considering representational issues in more detail—which is the main theme of this paper—it is first useful to consider the implications of the restrictions of the earlier theorems to non-revisiting searches. In the form that they state the No Free Lunch Theorem, Wolpert & Macready only consider algorithms that do not revisit points already considered, and seem to regard the issue of revisiting as a trivial technicality. In fact, most searches do not have this property, and it is worth noting that there are good reasons for this. Perhaps the most obvious is finite memory of real computers, but this is not the most important explanation. For while there are undoubtedly searches carried out for which it would be impossible today to store every point visited, as main memory sizes increase, this becomes ever less of an issue. At the very least it is typically feasible to store a much larger history than even most population-based algorithms choose to do. Much more significant is the time needed to process large amounts of data—even merely to perform a check against every point so far evaluated requires, at best, log time. Depending on the details of the algorithm, other functions may take even longer. As an example, ranking schemes used in some genetic algorithms (Baker, 1987) require their populations to be sorted.

Because most algorithms used do revisit points, there is the potential, in principle, to

improve real search techniques without reference to the particular problem being tackled. It should be clear that the kinds of performance measures relevant in this case concern the average number of points sampled, *including* points visited multiple times, to reach a solution of a certain quality, or the total "wall-clock" time to achieve the same goal. (Indeed, this "best-so-far" graph as a function of the number of function evaluations is precisely the graph shown in most empirical papers on stochastic search.) Thus, for example, when Whitley (1989) and Davis (1991) advocate barring duplicates in the population, it is perfectly possible that this will indeed prove beneficial over an entire ensemble of problems $\mathcal{R}^{\mathcal{S}}$. The suggested benefits of enforcing uniqueness of solutions are that it leads to more accurate sampling (with respect to target sampling rates, Grefenstette & Baker, 1989), increases diversity in the population and makes better use of the limited "memory" that the population represents. On the other hand, there is a clear cost associated with maintaining uniqueness, in that it requires checking each new (tentative) solution generated. The point here is not *whether* enforcing uniqueness *does* improve evolutionary search in general, but that in principal it *could* do so. The same applies to any change that affects the diversity in the population, and which is therefore likely to affect the frequency with which points are revisited.

5 Representations and Algorithms

This paper has concentrated rather heavily on representational aspects of search, and in doing so has established some rather powerful, general results. However, the motivation for focusing on representation goes far beyond this. For present purposes, we shall ignore the almost philosophical debate about whether it is *possible* to perform a search without a representation, and simply accept that a search may be conducted "directly" in the search space, by-passing the representational issue. Nevertheless, it is important to emphasize that the representation is *not* being introduced in order to facilitate storage (or coding) of solutions on a computer or any other device, which we regard as an auxiliary issue. The "point" of our representation is that we assume that move operators are actually formulated (defined!) in the representation space $\mathcal{C}$, rather than the search space $\mathcal{S}$, and that some benefit is potentially conferred by this.

Consider the concrete example of a three-dimensional euclidean space $\mathcal{S}$ and the choice of representations between cartesian coordinates (x, y, z) and spherical polars (r, θ, ϕ). We would consider the representation used by an algorithm to be defined *not* by the coordinate system in which the points in $\mathcal{S}$ were stored in a computer (whether in terms of high-level language constructs or actual discretised bit-patterns in memory chips), but rather by whether the move operators manipulate polar or cartesian components. There are clearly problems for which each representation has advantages. For example, the polar representation might be expected to offer benefits in spherically symmetric problems.

It should further be pointed out that representation is often given an even more significant rôle than the central one discussed above (Hart *et al.,* 1994). Davis (1991), for example, has long used devices such as "greedy decoders" to map a "chromosome" manipulated by a genetic algorithm to the solution that it represents. While this can formally be viewed as a radically different kind of search, in which the search objects are really

"hints" or starting points for a full local search, the more intuitive picture of a particularly complex growth function has some merit. Of course, fascinating new issues arise in this context, as there is typically no particular reason to suppose that the decoder is surjective (indeed, it would almost defeat its motivation if it were!), nor even that it can generate points of interest (such as global optima) for *any* input. Moreover, the decoder may well contain a stochastic element, further complicating matters. There is a need to try to generalise our theories of search to begin to accommodate some of these less conventional, but manifestly powerful, complex representations.

Returning to more prosaic representations, consider two different algorithms A and B, operating on the same problem f from $\mathcal{R}^{\mathcal{S}}$ and using the same representation space $\mathcal{C}$, but with different growth functions $g, g' \in \mathcal{S}^{\mathcal{C}}$. It is clear that there are some situations in which even though $A \neq B$, and $g \neq g'$, the differences "cancel out" so that they perform the same search. (Indeed, an obvious case arises when the algorithms are isomorphic, and $g' = g \circ \pi$ for some suitable permutation π.) This might lead us to suspect that there is a *duality* between representations and algorithms, so that any change that can be effected by altering the representation could equally be achieved by a corresponding alteration to the algorithm, and any change made to the algorithm could equally be effected by a corresponding change to the representation. In fact, it has been shown (Radcliffe, 1994) that while the first of these statements is true, the second is not, in the general case of algorithms that are allowed to revisit points, or if the stochastic element is not "factored out" by fixing the seed. The duality—if that it be—is only partial.

If, for example, we consider a canonical genetic algorithm (say the Simple Genetic Algorithm—Goldberg, 1989c), but allow either one-point or uniform crossover, then there is clearly no change of representation that transforms one of these algorithms into the other, either over all possible seeds for the random number generator, or even for a particular seed when revisiting effects are considered. Moreover, there is no reason to assume that their performance over the class of all representations in $\mathcal{S}^{\mathcal{C}}_{>}$ (or indeed, all objective functions $f \in \mathcal{R}^{\mathcal{S}}$) will be the same. In terms of the theorems described earlier, this merely amounts to the observation that revisiting algorithms are not, in general, isomorphic. However, this formally trivial statement has rather strong practical implications given that, as noted earlier, establishing whether a point has been previously sampled requires significant computational effort.

Despite the fact that more power is available by changing the algorithm than from merely changing the representation, the separation between representation and search algorithm remains extremely valuable. In section 6 we shall examine how good representations can be exploited and constructed.

6 Restricted Problem Classes

One of Wolpert and Macready's central messages, and one with which the authors agree strongly, is that if algorithms are to outperform random search, they must be matched to the search problem at hand. ("If no domain-specific knowledge is used in selecting an appropriate representation, the algorithm will have no opportunity to exceed the performance of an enumerative search"—Radcliffe, 1994.) It will be useful to consider the implications of the limitations on search algorithms from the theorems in two stages. First,

consideration will be given to what can be expected if it is known that the representation used does—in an appropriately defined sense—"match" some restricted problem class under consideration (section 6.1). These considerations apply even if nothing more than this is known. After this case has been discussed, the focus will turn to how knowledge of a problem class can be harnessed to select algorithms that might be expected to perform well (section 6.2).

6.1 Assuming a Good Representation

It will first be useful to formalise what is meant by a "good" representation. The following definition is suggested.

Definition (Good representation, relative algorithmic performance). Let A be a search algorithm that can be applied to a set of search problems defined by $\mathcal{R}^{\mathcal{S}}$, using a representation space $\mathcal{C}$. Let $\mathcal{D}$ be a (proper) subset of $\mathcal{R}^{\mathcal{S}}$ (a problem "domain"). Then a representation $g \in \mathcal{S}^{\mathcal{C}}_{\geq}$ will be said to be a *good* representation for $\mathcal{D}$ with respect to some performance measure μ if and only if A achieves better average performance with respect to μ on problems in $\mathcal{D}$ than on problems in the whole of $\mathcal{R}^{\mathcal{S}}$.

Similarly, algorithm B will be said to be better than algorithm A over $\mathcal{D}$, with representation g, if it achieves a better average performance over $\mathcal{D}$ with respect to μ.

■

This is clearly a rather minimalist definition of "good", but it will serve the present purpose. In particular, the notion of a quality of a representation is interesting in the context of "representation-independent" algorithms, (effectively algorithms that can be instantiated for any specified representation g—Radcliffe & Surry, 1994b, 1994a). These are discussed in section 6.2.

Focusing once more on evolutionary algorithms, the kinds of representation spaces that are typically used are high dimensional

$$\mathcal{C} = \mathcal{C}_1 \times \mathcal{C}_2 \times \cdots \times \mathcal{C}_n \tag{15}$$

and the operators used normally manipulate the components of members of $\mathcal{C}$ to some extent independently. The "ideal" representation that we probably typically have in mind is one that induces a linearly separable mapping $f \circ g$ from $\mathcal{C}$ to $\mathcal{R}$, (Mason, 1993) so that

$$f \circ g((c_1, c_2, \ldots, c_n)) = \sum_{i=1}^{n} \phi_i(c_i) \tag{16}$$

for some functions

$$\phi_i : \mathcal{C}_i \longrightarrow \mathcal{R}. \tag{17}$$

While we would expect that most reasonable evolutionary algorithms would outperform enumeration in such circumstances, using any reasonable performance measure, ironically, we would *not* expect an evolutionary algorithm to compare well with simple hill climbers in this case. In particular, any greedy hill climber will have optimal performance against typical performance measures on such problems. The territory in which we actually expect sophisticated adaptive search schemes to excel is some intermediate *régime* between random representations and completely linearly separable (i.e. non-epistatic) ones.

6.2 Implications for Tackling Problem Classes

The observations above provide a motivation for considering ways in which both good representations and good algorithms can be designed and recognised for particular problem classes. In the case of representations, some preliminary steps have been taken, including work by Davidor (1990), Kauffman (1993) and Radcliffe & Surry (1994a), and from a different perspective, there is an extensive literature on deception, a particular form of linear non-separability (Goldberg, 1989a, 1989b; Whitley, 1991; Das & Whitley, 1991; Grefenstette, 1992; Louis & Rawlins, 1993). All of these examine some form of questions about what precisely "some degree of linear separability" means, how it can be measured and how it may be exploited by algorithms.

One approach that seems particularly fruitful to the authors is to measure properties of representations over well-defined problem classes, and then to seek to construct a theory of how such measured properties can be exploited by algorithms. In Radcliffe & Surry (1994a), we measured the variance of objective function value for four different representations for the travelling sales-rep problem (TSP) as a function of schema order to obtain the graph shown in figure 6. (A schema—or forma—fixes certain components of a member of $\mathcal{C}$, with order measuring the number of such fixed components.) The lower variance for the "corner" representation indicates that objective function value is more nearly linearly separable with respect to the components of this representation than is the case with the other representations. Although the four representations are all highly constrained, well-defined move operators were specified solely in terms of manipulations of $\mathcal{C}$: this is what is meant by representation-independent move operators. We were therefore able to compare the performance of identical algorithms with these four representations over a range of problem instances. The results, as expected, indicated that on standard performance metrics, most of the algorithms used worked better with the lower variance representations than their higher variance counterparts. This suggests strongly that developing further measurements of representation quality, together with further ways of characterising representation-independent operators (such as the notions of respect, assortment and transmission, developed by Radcliffe, 1991, 1994) and a theory linking those together, is an urgent task.

7 Outlook

We have provided in this paper a formal demonstration of various fundamental limitations on search algorithms, with particular reference to evolutionary algorithms. These results establish clearly the central rôle of representation in search, and point to the importance of developing a methodology for formalising, expressing and incorporating domain knowledge into representations and operators, together with a theory to underpin this.

The results demonstrate the futility of trying to construct universal algorithms, or universal representations. For example, neither "binary" nor gray coding (Caruana & Schaffer, 1988) can be said to be superior, either to each other or to any other representation, without reference to a particular problem domain. One immediate practical consequence of this should be a change of methodology regarding test suites of problems and comparative studies. Rather than developing a fixed (small) suite of problems

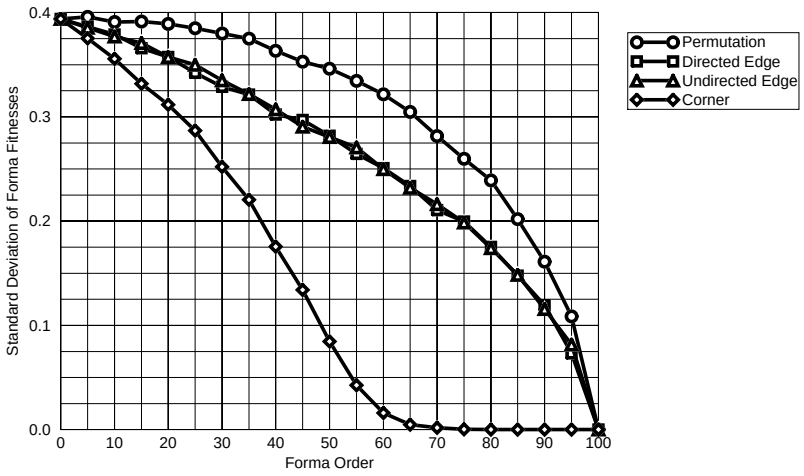

Fig. 6. The graph shows the mean standard deviation of tour length as a function of schema (forma) order for each of four representations of the TSP. These particular results are from a single problem instance generated using 100 samples drawn from 100 randomly generated schemata at each order shown, though the graph would look nearly identical if averaged sensibly over all 100-city TSPs.

for refining algorithms and representations, we believe that a significantly more useful approach consists of developing algorithms for a well-specified *class* of problems. The idea would be to refine algorithms using any small, randomly chosen subset of problems from this class, but to compare performance only against different randomly selected problem instances from the same class. We believe that a move towards this philosophy of comparative studies would allow the development of much more systematic insights.

The issues discussed in this paper seem to the authors central to gaining greater understanding of search methods, yet the questions that they raise receive remarkably little attention within the research community. On this basis, we conclude by suggesting some of the key open questions that we see.

- Given a class of problems about which we have partial knowledge, how can that knowledge be formalised in a way useful to constructing representations algorithms? Ideas from forma analysis (Radcliffe, 1991, 1994) offer some directions here, but there is clearly vastly more to be done.
- Can the quality of a representation be measured in any useful way?
- What kinds of predictive models of performance (if any) can be built given a well-defined problem domain, representation and algorithm?
- What performance metrics for algorithms are useful, given the limitations imposed by the theorems?

– Can we develop a useful methodology for determining appropriate parameters for a stochastic algorithm, given a representation and problem domain?
– Given an algorithm, can we determine the structure of the fitness landscape induced by the representation and move operators that is (implicitly) exploited by that algorithm? (What is a genetic algorithm *really* "processing"?) Such insights might allow us to build more powerful models of landscapes (Jones, 1994).

Acknowledgements

The authors would like to thank Bill Macready and David Wolpert for their comments on a draft of this paper. The ensuing discussion established that we had misunderstood their notion of "minimax" distinctions, and allowed us to refine our ideas in this area and establish their conjecture.

References

T. Bäck and H.-P. Schwefel, 1993. An overview of evolutionary algorithms for parameter optimisation. *Evolutionary Computation*, 1(1):1–24.

J. E. Baker, 1987. Reducing bias and inefficiency in the selection algorithm. In *Proceedings of the Second International Conference on Genetic Algorithms*. Lawrence Erlbaum Associates (Hillsdale).

R. A. Caruana and J. D. Schaffer, 1988. Representation and hidden bias: Gray vs. binary coding for genetic algorithms. In *Proceedings of the 5th International Conference on Machine Learning*. Morgan Kaufmann (Los Altos).

R. Das and D. Whitley, 1991. The only challenging problems are deceptive: Global search by solving order-1 hyperplanes. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 166–173. Morgan Kaufmann (San Mateo).

Y. Davidor, 1990. Epistasis variance: Suitability of a representation to genetic algorithms. *Complex Systems*, 4:369–383.

L. Davis, 1991. *Handbook of Genetic Algorithms*. Van Nostrand Reinhold (New York).

L. J. Fogel, A. J. Owens, and M. J. Walsh, 1966. *Artificial Intelligence Through Simulated Evolution*. Wiley Publishing (New York).

F. Glover, 1986. Future paths for integer programming and links to artificial-intelligence. *Computers and Operations Research*, 13(5):533–549.

D. E. Goldberg, 1989a. Genetic algorithms and Walsh functions: Part I, a gentle introduction. *Complex Systems*, 3:129–152.

D. E. Goldberg, 1989b. Genetic algorithms and Walsh functions: Part II, deception and its analysis. *Complex Systems*, 3:153–171.

D. E. Goldberg, 1989c. *Genetic Algorithms in Search, Optimization & Machine Learning*. Addison-Wesley (Reading, Mass).

J. J. Grefenstette and J. E. Baker, 1989. How genetic algorithms work: A critical look at intrinsic parallelism. In *Proceedings of the Third International Conference on Genetic Algorithms*. Morgan Kaufmann (San Mateo).

J. J. Grefenstette, 1992. Deception considered harmful. In *Foundations of Genetic Algorithms 2*, pages 75–91. Morgan Kaufmann (San Mateo, CA).

W. Hart, T. Kammeyer, and R. Belew, 1994. The role of development in genetic algorithms. To appear in Foundations of Genetic Algorithms 3.

J. H. Holland, 1975. *Adaptation in Natural and Artificial Systems*. University of Michigan Press (Ann Arbor).

T. Jones, 1994. A model of landscapes. Technical Report, Santa Fe Institute.

S. A. Kauffman, 1993. *The origins of order: self-organization and selection in evolution*. Oxford University Press (New York).

S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, 1983. Optimisation by simulated annealing. *Science*, 220(4598):671–680.

J. R. Koza, 1991. Evolving a computer to generate random numbers using the genetic programming paradigm. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 37–44. Morgan Kaufmann (San Mateo).

S. J. Louis and G. J. E. Rawlins, 1993. Pareto optimality, GA-easiness and deception. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*. Morgan Kaufmann (San Mateo, CA).

A. J. Mason, 1993. Crossover non-linearity ratios and the genetic algorithm: Escaping the blinkers of schema processing and intrinsic parallelism. Technical Report Report No. 535b, School of Engineering, University of Auckland.

N. J. Radcliffe and P. D. Surry, 1994a. Fitness variance of formae and performance prediction. Technical report, To appear in Foundations of Genetic Algorithms 3.

N. J. Radcliffe and P. D. Surry, 1994b. Formal memetic algorithms. In Terence C. Fogarty, editor, *Evolutionary Computing: AISB Workshop*, pages 1–16. Springer-Verlag, Lecture Notes in Computer Science 865.

N. J. Radcliffe, 1991. Equivalence class analysis of genetic algorithms. *Complex Systems*, 5(2):183–205.

N. J. Radcliffe, 1992. Non-linear genetic representations. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature 2*, pages 259–268. Elsevier Science Publishers/North Holland (Amsterdam).

N. J. Radcliffe, 1994. The algebra of genetic algorithms. *Annals of Maths and Artificial Intelligence*, 10:339–384.

I. Rechenberg, 1973. *Evolutionstrategie—Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Frommann-Holzboog (Stuttgart).

I. Rechenberg, 1984. The evolution strategy. a mathematical model of darwinian evolution. In E. Frehland, editor, *Synergetics—from Microscopic to Macroscopic Order*, pages 122–132. Springer-Verlag (New York).

J. Shapiro, A. Prügel-Bennett, and M. Rattray, 1994. A statistical mechanical formulation of the dynamics of genetic algorithms. In T. C. Fogarty, editor, *Evolutionary Computing: AISB Workshop*, pages 17–27. Springer-Verlag, Lecture Notes in Computer Science 865.

P. J. M. van Laarhoven and E. H. L. Aarts, 1989. *Simulated Annealing: Theory and Applications*. D. Reidel Publishing Company.

M. D. Vose and G. E. Liepins, 1991. Schema disruption. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 237–243. Morgan Kaufmann (San Mateo).

M. D. Vose, 1992. Modelling simple genetic algorithms. In D. Whitley, editor, *Foundations of Genetic Algorithms 2*. Morgan Kaufmann (San Mateo, CA).

D. Whitley, 1989. The GENITOR algorithm and selection pressure: Why rank-based allocation of reprodutive trials is best. In *Proceedings of the Third International Conference on Genetic Algorithms*, pages 116–121. Morgan Kaufmann (San Mateo).

L. D. Whitley, 1991. Fundamental principles of deception. In G. J. E. Rawlins, editor, *Foundations of Genetic Algorithms*, pages 221–241. Morgan Kaufmann (San Mateo).

D. H. Wolpert and W. G. Macready, 1995. No free lunch theorems for search. Technical Report SFI–TR–95–02–010, Santa Fe Institute.