Some Multiobjective Optimizers are Better than Others

David Corne[1]
Department of Computer Science
University of Exeter
Exeter EX4 4QF, UK
d.w.corne@ex.ac.uk

Joshua Knowles[2]
IRIDIA, CP 194/6
Université Libre de Bruxelles
1050 Bruxelles, Belgium
jknowles@ulb.ac.be

Abstract- The No-Free-Lunch (NFL) theorems hold for general multiobjective fitness spaces, in the sense that, over a space of problems which is closed under permutation, any two algorithms will produce the same set of multiobjective samples. However, there are salient ways in which NFL does *not* generally hold in multiobjective optimization. Previously we have shown that a 'free lunch' can arise when comparative metrics (rather than absolute metrics) are used for performance measurement. Here we show that NFL does not generally apply in multiobjective optimization when absolute performance metrics are used. This is because multiobjective optimizers usually combine a *generator* with an *archiver*. The generator corresponds to the 'algorithm' in the NFL sense, but the archiver filters the sample generated by the algorithm in a way that undermines the NFL assumptions. Essentially, if two multiobjective approaches have different archivers, their average performance may differ. We prove this, and hence show that we can say, without qualification, that some multiobjective approaches are better than others.

1 Introduction

The now well-known 'No Free Lunch' theorem (NFL) refers to a small family of theorems which characterize the average behaviour of optimization algorithms over spaces of optimization problems. The seminal works (Wolpert & Macready, 1995; 1997), showed that, when performance is averaged over all possible optimization problems defined over some search space *X*, no algorithm has a performance advantage over any other. This means, for example, that no algorithm is better than random search. This may seem surprising, but can be intuitively understood on the basis that when we consider *all possible* optimization problems over the search space *X*, it is clear that overwhelmingly many of these have no consistent structure that could be exploited successfully by any particular algorithm. Nevertheless, the NFL result remains surprising and interesting, partly since it speaks to the relationship between algorithm design and domain knowledge.

NFL casts clearly into focus the fact that any 'improvement' to an algorithm is really a *re-alignment* of that algorithm towards an altered collection of problems. There has been and continues to be much debate over the importance and applicability of NFL (Radcliffe & Surry, 1995; Christensen & Oppacher, 2001; Droste et al, 1999; Igel & Toussaint, 2001). However, the main consequence for the majority of researchers in optimization is to recognize that the NFL result automatically precludes any claim of the form 'algorithm *A* is better than algorithm *B*', for *any* pair of algorithms *A* and *B*. For example, the statement 'Particle Swarm Optimization outperforms Simulated Annealing' is false. Any such statement can only be true if qualified in terms of the set of problems under consideration, and can then only be true if that set is not 'closed under permutation' (c. u. p.) (Schumacher et al, 2001). If we define an optimization problem as a mapping which assigns a fitness to every point in a search space, then a set of such problems is c.u.p. if and only if any permutation of the fitnesses of a problem in the set produces some other problem in the set. The set of 'all problems' over some given search space is an example of such a set.

In the remainder of the article, we first provide, in section 2, some necessary background and notation, and briefly relate some previous relevant work. Then, in section 3, we outline the essential 'generator plus archiver' structure of most modern multiobjective algorithms, a fact which is salient to the failure of NFL to sensibly apply as regards algorithm performance in the multiobjective case. In section 4, we prove (twice) a theorem that shows that two multiobjective optimizers can differ in performance over all problems. We conclude with general discussion in section 6.

2 Preliminaries

Consistent with Wolpert & Macready (1997), we talk of a search space *X* and a set of 'fitnesses' *Y*. We will also allow a fitness *y* in *Y* to be either single or multiobjective. That is, we will not, unless it seems absolutely necessary, introduce extra notation for multiobjective fitness vectors.

[1] This work was done while the first author was at The Dept. of Computer Science, University of Reading, UK

[2] The second author now holds a David Phillips Fellowship at the new Manchester Interdisciplinary BioCentre, Manchester, UK

What is said in the following holds equally for single or multiobjective optimisation, unless otherwise indicated.

An optimisation problem *f* is identified with a mapping $f : X \to Y$ and $F = Y^X$ is the space of all problems. Its size is clearly $|Y|^{|X|}$. A black-box optimisation algorithm *a* generates a time-ordered sequence of points in the search space, associated with their fitnesses, called a *sample*. A sample of size *m* is denoted $d_m \equiv \{(d_m^X(1), d_m^Y(1)), ..., (d_m^X(m), d_m^Y(m))\}$, where, in general, $d_m^X(i)$ is the *i*th distinct point in the search space *X* visited by the algorithm, and $d_m^Y(i)$ is the fitness of this point. We may use the term d_m^X to denote the vector containing only the $d_m^X(i)$, and similarly d_m^Y for the time-ordered set of fitness values in the sample

We can now characterise what we mean by an optimisation *algorithm*. We think of an optimisation algorithm as a process which generates a sequence of points from the search space – i.e. it visits points in the search space one by one in a given order. Which point is visited next depends only on the points visited previously. It is therefore a mapping from samples to points in *X*, with the restriction that $a(d_m) \notin d_m^X$. That is, it does not revisit previously visited points in *X*. This restriction, plus that of the algorithm being *deterministic*, is imposed for simplicity but does not affect the generality of the resulting theorems.

It is of interest to consider the probability of a particular sample (i.e. a particular sequence of points) being generated by algorithm *a* after *m* iterations on problem *f*. This is denoted with $P(d_m^y \mid f, m, a)$. The standard NFL theorem (Wolpert & Macready, 1995; 1997) is essentially characterised by the following, which holds for any pair of algorithms a_1 and a_2:

$$\sum_f P(d_m^Y \mid f, m, a_1) = \sum_f P(d_m^Y \mid f, m, a_2) . \quad (1)$$

So, when averaged over all problems, the probabilities of obtaining a particular sample using a_1 amount to the same as that of using a_2. In fact, it turns out that if a particular sample is generated by a_1 on each of *m* different problems, then a_2, and indeed every other algorithm, generates that same sample on a set of *m* different problems, which may or may not overlap with a_1's set (Schumacher et al, 2001).

The well-known consequences concerning algorithm *performance* stem from the fact that, in single objective optimisation at least, the performance of an algorithm is a straightforward function of the sample it generates. For example, the result returned by an algorithm in single objective optimisation is usually $\max(d_m^y)$ (assuming maximisation, of course). It follows from equation (1) that:

$$\sum_f P(\max(d_m^Y) \mid f, m, a_1) = \sum_f P(\max(d_m^Y) \mid f, m, a_2) . \quad (2)$$

and of course more generally that:

$$\sum_f P(\Phi(d_m^Y) \mid f, m, a_1) = \sum_f P(\Phi(d_m^Y) \mid f, m, a_2) \quad (3)$$

where $\Phi : d_m^Y \to \Re$ measures performance as a scalar valued function of the sample.

We showed in Corne & Knowles (2003) that although equation (3) holds for multiobjective optimisation, and indeed holds for certain interesting subsets of multiobjective problem spaces, it is actually not as *applicable* in the multiobjective case as it is in the single objective case. In Corne & Knowles (2003) we focussed on the fact that in multiobjective optimisation we often use *comparative* metrics (Hansen & Jaszkiewicz, 1998; Knowles & Corne, 2002; Zitzler et al, 2002) which score the relative performance of two algorithms by considering the results of both. This is nontrivially different from simply comparing the difference of two absolute metrics, and it turns out that, in some tangible sense, that when the comparison between some pair of algorithms is averaged over a c.u.p. space of problems, one of these algorithms may show an overall advantage.

In this paper, we focus on a simpler and more accessible line of reasoning which again shows that equation (3) is not generally applicable to the multiobjective case, even if we only use absolute performance metrics.

3 Generators and Archivers

For reasons argued in Knowles & Corne (2002; 2003) and elsewhere, applied multiobjective optimisation algorithms commonly incorporate an archiving process. This involves a datastructure (the 'archive') which maintains an *a priori* bounded number of non-dominated points. The archive plays the role of the 'best so far' placeholder in single objective optimisation. However, whereas in single-objective optimisation it is entirely trivial to maintain this best-so-far record, the story is quite different in multiobjective optimisation. The problems arise from the fact that non-dominated sets can be arbitrarily large, and for reasons of time and space complexity we therefore need a bound on the size of the archive. The question then arises as to what happens when the archive is full, but a new non-dominated point is generated. The archiving process must decide whether or not to place the new point in the archive, and, if it does incorporate the new point, it must decide which existing point(s) to remove. In this way, although ideally we would want the archive to maintain the 'best so far' record of the multiobjective search, it can only in general maintain an approximation to that. As we shall see, this essentially leads to 'free lunch' results for archived multiobjective search.

To bring out the results, we will first formalise a simple version of a multiobjective algorithm which uses an archive. Note that this includes the vast majority of published evolutionary multiobjective approaches (which either explicitly maintains an archive – e.g. Knowles & Corne (2000) – or exhibits a *de facto* archive in its management of the population – e.g. Deb et al (2000)). We can think of a multiobjective algorithm as a combination of a *generator* and an *archiver*. The generator produces a series of points from the search space *X*, and we will model it in the way familiarly used for algorithms in the NFL literature. That is, *m* iterations of the generator yields a sample d_m as introduced in section 2, and the generator itself is a mapping from a sample d_m to some point in *X* which was not in d_m^X. We will simplify the notation here and set $z^t \equiv (d_m^X(t), d_m^Y(t))$, i.e. z^t is the *t*th point produced by the generator.

On the other hand, an archiver is a mapping $r : \mathbf{A}_N \times Z \to \mathbf{A}_N$, where $\mathbf{A}_N$ is the set of all archives containing no more than *N* points, and *Z* is simply $X \times Y$, the elements of which are points in the search space associated with their (multiobjective) fitnesses. The archiver takes successive points from the generator, in each case taking archive A^{t-1} and point z^t, and outputting archive A^t, the new notation having the obvious meaning. Both to emphasise practical relevance in the following, and to simplify some of the discussion and formalities, we will only consider a subset of archiving algorithms which we call *precise archivers*. A precise archiver is one which ensures, in its local archiving steps, that as many non-dominated points as possible are kept in the archive, and *only* non-dominated points are kept. A precise archiver with bound *N* has the following properties:

- SA1: if z^t dominates one or more points in A^{t-1}, then z^t is incorporated into A^t and the points it dominated (and no others) are removed.
- SA2: if z^t is dominated by one or more points in A^{t-1}, then the archive remains unchanged (i.e. z^t is not incorporated, and no points are removed).
- SA3: if A^{t-1} contains fewer than *N* points, and neither SA1 nor SA2 apply, then z^t is incorporated into the archive and no points are removed.

These properties are clearly necessary if we wish the archive at time A^t to contain (subject to the bound *N*) the non-dominated subset of $A^{t-1} \cup z^t$, which is an obvious requirement if we aim to return the best approximation we can to the set of Pareto optimal points produced by the generator. Property SA1 is straightforward, ensuring that a precise archive always accepts a point that dominates any point within it, and then removes the dominated points. Property SA2 ensures that an archive will always *reject* a point that is dominated by something already present in it. Property SA3 simply includes a new non-dominated point in the archive if space is available. This reflects a general assumed desire, which is usually felt to be consistent with the idea of maintaining a good approximation to the Pareto front, to fill as many slots in the archive as possible. Put another way, given that we have a bound of *N* on the archive size, and assuming the Pareto set of the points produced by the generator ends up being either *N* or greater, then we would hope that the archive contains *N* non-dominated points at the end of the run. In the case that the generator has produced $s < N$ Pareto points, we would hope and expect that the archive contains precisely these *s* points. Where precise archivers can vary is entirely in what happens when A^{t-1} is full (with *N* points) but z^t is non-dominated with respect to all points in A^{t-1}.

It is worth noting that not all published archivers are precise. This arises for good reasons, which are illustrated by Figure 1.

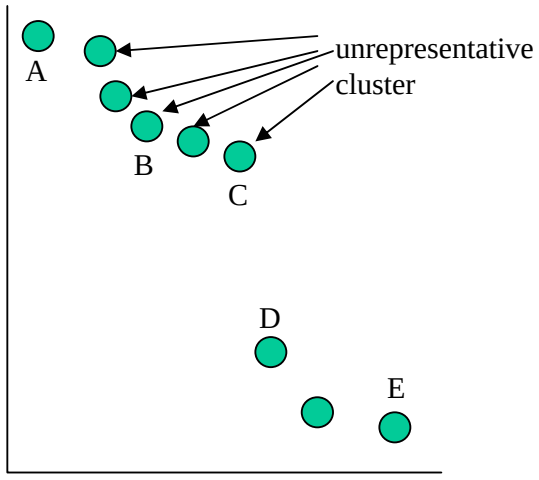

Figure 1. A contrived Pareto front containing nine points. If an archive was bounded to contain at most 5 points, then the set of points labeled A—E would be a preferable archive to the set of points labeled 'unrepresentative cluster'. This illustrates the motivation behind archiving algorithms which eschew all the properties of a 'precise' archiver in favour of promoting the maintenance of a diverse set of points in the archive.

In the figure, a contrived Pareto front is illustrated which contains nine points, and we imagine the case of generator producing a series of points of which these nine are the Pareto set, and an archiver whose bound is $N = 5$. Depending on the order in which points are generated by the generator (and hence the order they are presented to a precise archiver), the contents of a precise archiver at the end of the process may be any subset of these nine points.

Note in particular that it may be the 'unrepresentative cluster' in the figure, or it may be the well-spread-out collection of points A—E. For most practical purposes, a well-spread-out set of points would be more favoured. However, it turns out that the design of archivers to maintain such good spreads of points (in arbitrary numbers of dimensions) is very difficult (Knowles & Corne, 2002; 2003; Laumanns et al, 2002; 2002a). For example, in order to guarantee maintenance of an $\in$-approximate archive, which has good properties in terms of spread across the (approximated) Pareto front, the archiver of Laumanns et al (algorithm 2 in Laumanns et al (2002a)) sacrifices precision in the sense we have defined it here. To elaborate, in $\in$-approximate archiving a special notion of dominance is used, called $\in$-dominance, where we say that, for multiobjective vectors f and g, f $\in$-dominates g iff $(1+\in)\cdot f$ dominates g (assuming maximization). Laumann et al's archiver for maintaining $\in$-approximate Pareto sets violates (for example) property SA3, since a non-dominated point which may be a candidate for inclusion in a non-full archive will not be accepted if it is (as it may well be, depending on the value of $\in$) $\in$-dominated by a point already in the archive; that is, such a point would be rejected, even though it would lead to a larger approximation to the Pareto set, because it would be deemed too 'close' to an existing point in the archive.

Finally, we note that the idea of unbounded archives has been studied in the recent literature. In particular, Fieldsend et al (2003) describe specialized datastructures and algorithms which enable fast update and hence speedy maintenance of such archives. This approach, in which all non-dominated points are always accepted into the archive and none is ever removed, does not succumb to the results in this paper. It is also currently a very uncommon approach, which is impractical in the many cases where memory is strictly limited.

4 The GRATIS Theorem

The NFL theorem tells us that, when averaged over all possible problems, any function of the *samples* produced by two algorithms will be the same. This is equation (3), and it stands for multi-objective samples as well as single-objective samples. In the single-objective case it is a profound result, since all of the overwhelmingly typical measures of algorithm performance, such as the highest fitness obtained during the run, are covered by the theorem. However, the situation is quite different for multiobjective optimisation, because the typical way to measure absolute performance (we deal with comparative performance measures in Corne & Knowles (2003)) is via a function of the *archive*. For example, this may be the hypervolume of the archive, its extent, and any of several other metrics (Hansen & Jaszkiewicz, 1998; Van Veldhuizen, 1999; Knowles & Corne, 2002; Zitzler et al, 2002). But, the archive will not generally be a correct reflection of the performance achievable from the *sample*.

Formalisation will now be helpful. Since we are dealing with absolute performance metrics over archived multiobjective optimisation, the question which arises is whether the following holds:

$$\sum_f P(A \mid f, m, a_1) = \sum_f P(A \mid f, m, a_2). \qquad (4)$$

where **A** is the set of all archives and $P(A \mid f, m, a)$ is the probability of obtaining archive A after m iterations of multiojective algorithm a on problem f. We now state and prove a GeneratoR/Archiver Theorem In Search (GRATIS).

Theorem: GRATIS

There are pairs of multiobjective algorithms a_1, a_2 for which the following holds

$$\sum_f P(A \mid f, m, a_1) < \sum_f P(A \mid f, m, a_2). \qquad (5)$$

and consequently, some multiobjective algorithms are better than others.

Proof:

Let a_1 and a_2 use precise archiving algorithms with different bounds, with the archiver of a_1 restricted to size N = 1, and the archive of a_2 restricted to size N = 2. Further, let archive A contain two non-dominated points. The term on the LHS of equation (5) must be zero in this case, since, being restricted to hold just 1 point in its archive, a_1 can never produce archive A as a result. However, a_2 will sometimes produce archive A. For example, consider the problem f where the multiobjective fitness space contains just two fitnesses, corresponding to those in A. We can engineer a_2's generator for some problem f such that these two fitnesses are those of z^1 and z^2, and since a_2's archiver is precise, this will remain the archive's contents throughout. The RHS of equation (5) is therefore nonzero, and the theorem is proved.

The proof is straightforward and simple, but may seem to suggest that the Free Lunch in this scenario is contingent on the precise values of memory bounds. That is, for example, an algorithm which can maintain (and hence return as its result) a set of 100 points is naturally better than an algorithm that can maintain and return only

10 points. We stress that this result is not so trivial as that by way of the following alternative, but necessarily longer proof.

Alternative Proof:

Without loss of generality, we will work in the space of two-objective problems, where the aim is to maximise both objectives, *x* and *y*. Let a_1 and a_2 use precise archiving algorithms with the same bound $N = 2$, but differ in response to a non-dominated z^t when A^{t-1} is full as follows: a_1's archiver always accepts z^t if its *x* value or *y* value is larger than that of any other point in A^{t-1}, and a point whose objective values are not extremal is removed (such a point must exist in this 2-objective case with 3 or more points in consideration). However a_2's archiver *never* accepts z^t. That is, a_2's archiver is precise, but never alters the archive unless it has to.

Let the space of multiobjective fitnesses *Y* under consideration contain 3 Pareto optimal fitnesses. Let the two extremal fitnesses in objectives *x* and *y* be *q* and *r* respectively. That is, the *x* value of point *q* is the largest among the 3 Pareto optimal fitnesses in *Y*, and similar for the *y* value of *r*. Let *c* be the non-extremal Pareto optimal point.

We will prove that a_1 and a_2 have different overall performance by showing that, given an archive *A* containing *q* and *r*, a_1 is more likely to produce it than a_2, even when that likelihood is summed over all possible problems with fitnesses in *Y* over the search space *X*. So, let *A* contain just the two points *q* and *r*, and let $m = 3$. That is, the generator has produced a sequence of 3 points. There are several cases as follows:

The generator has not produced both *q* and *r*. In such cases (corresponding to a portion of the problems in the summation in equation (5)), there is no chance of producing archive *A*, and the probabilities on the LHS and RHS of equation (5) are zero.

The next case is that *q* and *r* are the first two points generated (in either order). By definition of a_1 and a_2's archiving algorithms, and given that they are precise archivers, in this case both will produce archive *A* whatever the third point generated happens to be. For each in the collection of problems in the summation of equation (5) which lead to such a sequence, both the LHS and RHS will therefore contribute 1.

The final case is where both *q* and *r* are generated, but one of the first two points generated is neither *q* nor *r*. If the non-extremal point is dominated by *q* or *r*, then both archivers will produce the same result (archive *A*). If not, then a_1's archiver will produce *A* after iteration 3, but a_2's archiver will not. This finally leads to an imbalance in equation (5), and we have exhausted all cases. Hence, overall, when averaged over all possible problems, multiobjective algorithm a_1 will produce archive *A* more often than will multiobjective algorithm a_2. It generally follows from this, since absolute performance measures are functions of the archives returned, that some Multiobjective algorithms are better than others.

Despite two proofs, there may still seem to be an air of practical irrelevance to this result, which we can give shape to in the following observations and question: it seems that what makes one multiobjective algorithm better or worse than another is that the two algorithms use *different* archivers. Also, although 'precise', the archivers which have led to 'beaten' algorithms so far have been clearly deficient in some way (smaller in the first case, indifferent to Pareto extremes in the second case). Is there not one 'super-precise' archiver which all multiobjective algorithms should use anyway, in which case the GRATIS theorem would not arise (save for the case when different algorithms use different archive bounds)?

Current practice strongly suggests this is not the case, with many variations being explored for precise archivers (Zitzler, 1999; Knowles & Corne, 2000; Laumanns et al, 2002). More to the point, when we consider what properties a 'super-precise' archiver might have, it would seem clearly desirable for it to contain (if its bound allows) the Pareto front of the points it has 'seen', or for it to be filled up with points contained in that Pareto front. However, our recent result (Knowles & Corne, 2003), shows that such cannot in general be guaranteed by any archiving algorithm. More precisely, we showed that no bounded (by *N*) archiver can guarantee to maintain $\min(N, |F^*|)$ points, where F^* is the Pareto subset of the generated points. In other words, there is no 'super-precise' archiver which ensures, for sequences of points which contain a Pareto set of at least size *N*, to return an archive containing *N* Pareto points, and there is not even any archiver which can generally guarantee to always return the Pareto set of the generated points even if the size of this set is within the archive bound. To intuit the essential background to such
results, consider figure 2.

In the figure, in which we assume objectives are being minimized towards the bottom left corner, we see four points and imagine they are generated in the order A, B, C, then D. For simplicity, we also imagine that our archive is bounded with $N = 2$. By the time A and B are generated, the archive is precisely: {A, B}. When *C* is generated, the archiver can either incorporate it (and hence must eliminate either A or B), or it can choose to reject C whether to eliminate A, eliminate B, or reject C. Let's assume it decides to reject C. At the next step, when D is generated, our precise archiver will incorporate D and eliminate both A and B, and will hence be {D}. Note that the size of the archive here is smaller than the archive bound, despite the fact that the number of Pareto points generated so far is equal to the archive bound. Naturally,

things would have been different in this example if we had chosen to accept C and eliminate A or B, in which case the archive following the generation of D would be equal to the Pareto set generated so far. But, notice that we could have simply contrived D to be a point which dominated B and C, and the same issue would have arisen. The fact that we cannot see into the future (deciding how to update the archive on the basis of points not yet generated) leads to this basic limitation on archivers.

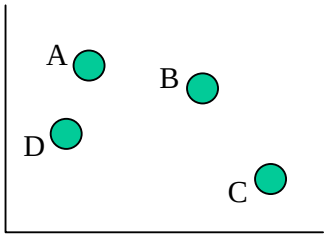

Figure 2. Assuming minimization, four points generated in the order A, B, C, then D. We imagine an archive with bound $N = 2$, which is first {A} and then {A, B}. When *C* is generated, the archiver must choose whether to eliminate A, eliminate B, or reject C. If C is rejected, then, after the next step, the archive is only {D}.

Since there seems to be no *ideal* archiver (we also prove this in Knowles & Corne (2003) with respect to a definition of 'ideal' based on $\in$ -approximate sets), there can only be competing approximations to the ideal. The state of the art in multiobjective optimisation at the moment comprises in part a thriving research theme whereby different researchers are considering the design of archiving algorithms, and consequently there are several different archivers being used. The consequence of this, which follows from the Gratis theorem, is that some modern multiobjective optimisation approaches are fundamentally better than others.

5 Concluding Discussion

The NFL Theorem is a profound result which cuts across all forms of optimization. In its 'sharp' form (Schumacher et al, 2001), it reveals that, given all possible problems defined over some search space *X*, every black-box optimization algorithm (i.e. every algorithm that can be viewed as visiting points in the search space one by one – essentially including all local search and population-based search algorithms) produces the same set of *samples*. That is, if a particular algorithm is shown to visit points in a particular order on precisely 47 problems, then *all* algorithms visit points from *X* in that same order on precisely 47 problems. This is as true for multiobjective problem spaces (Radcliffe & Surry, 1995; Corne & Knowles, 2003) as it is for single-objective problem spaces (Radcliffe & Surry, 1995; Wolpert & Macready, 1995; 1997). However, when we come to consider the *performance* of algorithms, there are important differences between single objective and multiobjective search.

In single-objective search, there is a straightforward link between a sample – the sequence of points visited (or generated) by an algorithm, and the performance of that algorithm. Mostly, for example, performance is taken to be the highest (assuming maximization) fitness found during the run. Hence, performance is simply the highest ftness in the sample. There may occasionally be other performance measures in play, but invariably the link between sample and performance is straightforward and computationally trivial. Another point, which almost goes without saying, is that when two algorithms are compared in terms of performance, we always use the *same* performance measure (such as Φ on both the LHS and RHS of equation (3)). What follows from all this is that, in single objective optimisation, given the common and precise ways in which performance is measured and compared, the average performance (over all problems) of every algorithm is the same.

Despite the NFL results holding for multiobjective problem spaces at the level of *samples*, the follow-on considerations as regards *performance* are problematic. We have already seen that a 'free lunch' can arise when *comparative* performance is measured by comparative metrics (Corne & Knowles, 2003). In this paper we have shown a more direct free lunch result concerning the simpler case of absolute performance metrics. The result arises because, in multiobjective optimization, we have no reliable space-efficient way to keep track of the 'best so far' (the Pareto set of the points generated so far). Consequently, we are forced to maintain an *estimate* of this set, and there are different ways of maintaining this estimate (different archivers), and it turns out that multiobjective approaches which use different archivers can differ in overall performance (i.e. performance averaged over all possible problems). The essential reason for the latter is that two archivers can produce different archives after seeing the same sequence. We would have the same things to say about single objective optimization if it were the case that two correct 'best-so-far'-trackers could produce different results on the same sequence, but this is of course not the case. In fact, the 'equivalent' result for single objective optimisation would be tantamount to saying that a free lunch can arise between algorithms a_1 and a_2 if we used *different* performance measures for each. However, just as this is true but not sensibly applicable in single-objective optimisation, we would claim, backed up by the results herein, that the standard NFL result is true but not sensibly applicable in multiobjective optimisation.

The Gratis Theorem applies, strictly, only to multiobjective optimization approaches which follow the model in which a generator interacts with an archiver in the simple way described in section 3. It is not clear that all multiobjective approaches which use archiving are appropriately covered by this interaction model, and this will be explored in later work. In particular, it is notable

that the Gratis theorem depends entirely on the archivers of algorithms a_1 and a_2 being different. We think that a more interesting result could be proven concerning multiobjective approaches in which the generator and archiver are more closely linked.

Acknowledgments

The first author is grateful to Evosolve (UK Registered Charity number 1086384) for partial support of this work, and to British Telecommunications Plc and BT Exact Plc, who have funded previous projects concerning multiobjective optimization, out of which arose the idea for this study. The second author gratefully acknowledges the support of a CEC Marie Curie Fellowship, contract number HPMF-CT-2000-00992 (to September 2003) and a David Phillips Fellowship (from October 2003).

Bibliography

Christensen, S. and Oppacher, F. (2001) What can we learn from No Free Lunch? A First Attempt to Characterize the Concept of a Searchable Function, in L. Spector et al (eds), *Proc. of GECCO 2001*, Morgan Kaufmann, pp. 1219–1226.

Corne, D.W. and Knowles, J.D. (2003) No Free Lunch and Free Leftovers Theorems for Multiobjective Optimization Problems. *Evolutionary Multi-Criterion Optimization (EMO 2003) Second International Conference*, Faro, Portugal, April 2003, Proceedings, pp. 327-341, Springer LNCS.

Deb, K, Agrawal, S., Pratab, A. and Meyarivan, T. (2000) *A fast elitist non-dominated sorting genetic algorithm for multiobjective optimization: NSGA-II*, KanGAL Technical Report 200001, Indian Institute of Technology, Kanpur, India.

Droste, S., Jansen, T. and Wegener, I. (1999), Perhaps Not a Free Lunch But At Least a Free Appetizer, in W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela and R. E. Smith (eds.) *Proc. of GECCO 9)*, Morgan Kaufmann Publishers, Inc., pp. 833–839.

Fieldsend, J.E. Everson, R.M., and Singh, S. (2003) Using Unconstrained Elite Archives for Multi-Objective Optimization, *IEEE Transactions on Evolutionary Computation* 7(3), pp 305-323.

Hansen, M.P. and Jaszkiewicz, A. (1998) *Evaluating the quality of approximations to the non-dominated set*, Tech. Report IMM-REP-1998-7, Technical University of Denmark.

Igel, C. and Toussaint, M. (2001) On Classes of Functions for which No Free Lunch Results Hold, see http://citeseer.nj.nec.com/528857.html.

Knowles, J.D., Corne, D.W. (2000) Approximating the non-dominated front using the Pareto Archived Evolution Strategy, *Evolutionary Computation*, **8**(2): 149–172.

Knowles, J.D. and Corne, D.W. (2002) On metrics for comparing non-dominated sets, in *Proc. 2002 Congress on Evolutionary Comp.*, IEEE Service Center, Piscataway, NJ.

Knowles, J.D. and Corne, D.W. (2003) Bounded Pareto archiving: theory and practice, in *Multiple Objective Meta-Heuristics: Selected Papers*, Springer LNES, to appear.

Laumanns, M., Thiele, L., Deb, K., Zitzler, E. (2002) Archiving with guaranteed convergence and diversity in multiobjective optimization, in *Proceedings of GECCO 2002*, Morgan Kaufmann Publishers, pp. 439–447.

Laumanns, M., Thiele, L., Deb, K., Zitzler, E. (2002a) Combining Convergence and Diversity in Evolutionary Multi-Objective Optimization, *Evolutionary Computation* **10**(3): 263–282.

Radcliffe, N.J. and Surry, P.D. (1995) Fundamental Limitations on Search Algorithms: Evolutionary Computing in Perspective, in *Computer Science Today*, pp. 275–291

Schumacher, C., Vose, M.D. and Whitley, L.D. (2001) The No Free Lunch Theorem and Problem Description Length, in L. Spector et al (eds), Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2001), Morgan Kaufmann, pp. 565–570.

Van Veldhuizen (1999) *Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations*, PhD Thesis, Dept. of Electrical and Computer Eng., Graduate School of Engineering, Air Force Institute of Technology, Wright-Paterson AFB, Ohio.

Wolpert, D.H. and Macready, W.G. (1995) *No Free Lunch Theorems for Search*, Santa Fe Institute Technical Report SFI-TR-05-010, Santa Fe Institute, Santa Fe, NM.

Wolpert, D.H. and Macready, W.G. (1997) No Free Lunch Theorems for Optimization, *IEEE Transactions on Evolutionary Computation* **1**(1): 67–82.

Zitzler, E. (1999) *Evolutionary Algoirthms for Multiobjective Optimization: Methods and Applications*, PhD Thesis, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, November.

Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., Fonseca, V.G. (2002) *Performance assessment of multiobjective optimizers: an analysis and review*, available from the url: http://citeseer.nj.nec.com/zitzler02performance.html.