Appears in "Parallel Problem Solving from Nature IV", Springer-Verlag LNCS 1141, pp366-375, 1996.

Formal Algorithms + Formal Representations = Search Strategies

Patrick D. Surry[a,b] & Nicholas J. Radcliffe[a,b]
{pds,njr}@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. Most evolutionary algorithms use a fixed representation space. This complicates their application to many problem domains, especially when there are dependencies between problem variables (e.g. problems naturally defined over permutations). This paper presents a method for specifying algorithms with respect to *abstract* representations, making them completely independent of any *actual* representation or problem domain. It also defines a procedure for generating a concrete representation from an explicit *characterisation* of a problem domain which captures beliefs about its structure. This allows arbitrary algorithms to be applied to arbitrary problems yielding well-specified search strategies suitable for implementation. The process is illustrated by showing how *identical* algorithms can be applied to both the TSP and real parameter optimisation to yield familiar (but superficially very different) concrete search strategies.

1 Introduction

The traditional genetic algorithm (e.g. the Simple Genetic Algorithm of Goldberg, 1989) is defined over a fixed representation space, namely that of binary strings. A common perception is that to employ such an algorithm for a new problem, one need only define a fitness function. (Indeed, standard software packages exist which literally require only computer code for a fitness function, e.g. GENESIS; Grefenstette, 1984.) For problems defined explicitly over binary strings (one counting, royal road, etc.) this does not present any difficulty. For others, such as real-parameter optimisation, some encoding from the problem variables into binary strings must be formulated, in order that the fitness of binary chromosomes can then be calculated by decoding them. However, such "shoe-horning" may make much of the structure of the search problem unavailable to the algorithm in terms of heritable allele patterns (see for example the discussion of meaningful alphabets in Goldberg, 1990). For problems in which candidate solutions are more complicated objects (e.g. the travelling sales-rep problem) a direct binary encoding may be unnatural or even infeasible. A particular case is when the "natural variables" of the problem are not orthogonal, in that the valid settings of one variable depend on the value of another (e.g. permutations). Faced with such a situation, practitioners typically adopt an *ad hoc* approach, drawing on evolutionary "concepts" to define pragmatic new move operators perceived as appropriate in the new domain (e.g. operators such as subtour inversion, partially matched crossover and order crossover have been devised for the TSP; Oliver *et al.*, 1987).

Although the use of a fixed representation space is widespread, it is unsatisfactory for a number of reasons. Algorithms based on this idea fail to be truly independent of problem (particularly for problems with naturally non-orthogonal representations). As

such, it is not possible to transfer a given algorithm to an arbitrary problem domain. Because such algorithms have only limited domains of applicability, it becomes difficult to make meaningful comparisons between different algorithms. (Attempting to compare, for instance, "genetic algorithms" and "simulated annealing" is futile until both a problem domain and set of move operators are specified, to define each algorithm precisely.) Finally, a fixed representation space makes it much more difficult to incorporate knowledge about the structure of the problem—one is forced to change either the growth function (the "genotype-phenotype" mapping) or the move operators, making the algorithm even less problem-independent.

In this paper we discuss a methodology by which these difficulties can be overcome. We show how (evolutionary) algorithms can be precisely specified independently of any particular representation or problem domain. We demonstrate that this allows us to instantiate such an algorithm for *any* appropriate representation of *any* particular problem class, and illustrate this with a compelling example.

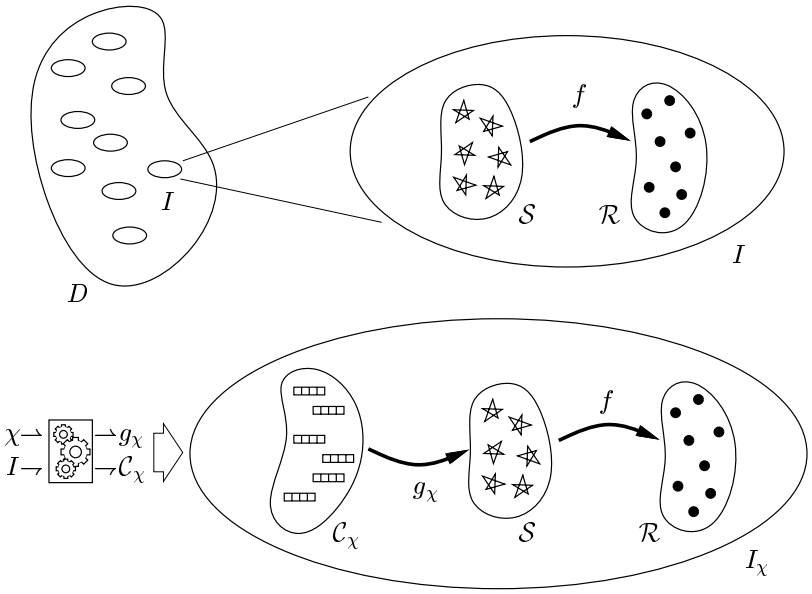

Fig. 1. A *problem domain* D consists of a set of *problem instances*. Each instance I defines a *search space* $\mathcal{S}$ of candidate solutions, a *fitness function* f and a set of objective values $\mathcal{R}$. A *characterisation* χ of the domain specifies a set of equivalences among the solutions for any instance I. These equivalences induce a *representation* made up of a *representation space* $\mathcal{C}_\chi$ (of *chromosomes*) and a *growth function* g_χ mapping chromosomes to the objects in $\mathcal{S}$. A chromosome x is a string of *alleles*, each of which indicates that x satisfies a particular equivalence on $\mathcal{S}$. Algorithms can be completely specified by their action on the alleles of these generalised chromosomes, making them totally independent of the problem domain itself.

In order to fix terminology, a *search problem* is taken to be the task of solving any problem instance from a well-specified problem domain, where by solve we mean finding some optimal or near-optimal solution from a set of candidate solutions. A *problem domain* is considered here to consist of a set of *problem instances*, each of which takes the form of a *search space* (of candidate solutions) together with some fitness function defined on that search space, as illustrated in the top part of figure 1. For instance, "symmetric travelling sales-rep problems" is a problem domain, with a particular set of $n(n-1)/2$ inter-city distances defining an instance (yielding a search space of $(n-1)!/2$ tours and an associated fitness function). A *search strategy* is then simply a prescription by which, for any problem instance, we successively sample candidate solutions from the search space, typically biasing the samples depending on the observed quality (fitness) of previously sampled points. Any such strategy can be viewed as utilising one or more *move operators* which produce new candidate solutions from those previously visited.

Because the objects in the search space can be arbitrary structures (e.g. real-valued vectors, TSP tours, neural-network topologies, etc.), it is often helpful to define search algorithms with respect to an abstract *representation* of the search space, allowing the transfer of search algorithms between problem domains. In general, a representation consists of a *representation space* and a *growth function*. The representation space defines a set of *chromosomes* which will be manipulated (by the move operators) during search, and the growth function defines a mapping between chromosomes and solutions.

The move operators used to construct a search algorithm can now be defined on the representation space, and the quality of any chromosome can be determined using the growth function in conjunction with the fitness function. One obvious method to achieve the goal of problem-independent search is to fix the representation space, and then to choose an appropriate growth function for each new problem domain. Such a search algorithm would sample chromosomes from the fixed representation space, using the growth and fitness functions as a "black-box" to evaluate them. This is essentially the traditional viewpoint, and indeed is the standard approach in the theory of computation (in which Turing machines are used to process arbitrary problems by encoding them as binary strings). There are however strong arguments against this approach. In the first instance, it may be extremely difficult or impossible to construct an appropriate growth function. Secondly, it is increasingly recognised that effective search is only possible if search strategies incorporate appropriate *domain knowledge* (of the structure of the function being optimised). Recent work on the so-called "No Free Lunch Theorem" (Wolpert & Macready, 1995; Radcliffe & Surry, 1995) has formalised these ideas, and has shown that true "black-box" optimisation can be at most as efficient as enumerative search, despite the claims of some authors. We argue that by abstracting the definition of a search algorithm away from a fixed representation-space (just as we strove for independence from a particular problem domain), we can realise the goal of a truly problem-independent algorithm while at the same time making the rôle played by domain knowledge much more explicit.

The formulation presented in section 2 postulates a problem-dependent characterisation χ which captures knowledge about a problem domain. This characterisation mechanically generates a *formal representation* (representation space and growth function) for any instance of the problem, by defining a number of equivalences over the search space, as shown in the bottom part of figure 1. These equivalences induce subsets of the search space thought to contain solutions with related performance; possibly as partitions[1] generated by equivalence relations, or simply as groups of solutions sharing some

[1] A partitioning of a set is a collection of disjoint subsets (partitions) covering the set.

characteristic. For a given solution, the pattern of its membership in the specified subsets is used to define its alleles (and possibly genes). Although in some problems the search space can be partitioned *orthogonally* (informally, meaning that all combinations of alleles represent legal solutions), this is not always the case. For example, in the traveling sales-rep problem, natural representations involve characterising tours by the edges (links) that they share, and it is clear that an arbitrary collection of edges does not always represent a valid tour.

In section 3 we show how *formal algorithms* can be precisely specified—completely independently of any particular representation—whose effectiveness is a direct function of the quality of the domain knowledge captured by the allele structure generated by the characterisation. All of the move operators used in such an algorithm are defined to manipulate solutions only in terms of their abstracted subset-membership properties (alleles). Because many problem domains are naturally characterised using non-orthogonal representations, the traditional operators are seen not to be fully general (e.g. consider one-point crossover between permutations). We provide examples of generalisations of N-point and uniform crossover, and of mutation and hill-climbing operators, and later show how they reduce to traditional forms in familiar problem domains.

We proceed in section 4 to show how *any* formal algorithm can be instantiated with *any* suitable representation of a problem domain of interest to produce a concrete *search strategy*. This is illustrated by defining a simple representation-independent evolutionary algorithm and instantiating it on the one hand to solve the TSP and on the other to solve a real-parameter optimisation problem. The resulting search strategies for the two problems look very different from each other but are both similar to evolutionary algorithms commonly applied in their respective domains. We thus prove the surprising result that two apparently quite different algorithms, in two completely different problem domains, are in fact *identical*, in a strong mathematical sense.

In section 5 we summarise the implications of this more formal approach. We see that by separating algorithm and representation, we achieve the goal of truly problem-independent algorithms. This separation also makes the rôle of domain knowledge in the search process much more explicit, allowing us to pose more carefully questions such as "What is a good algorithm given certain properties of the characterisation?" and "What is a good characterisation of a given problem domain?" Although this formalism might be argued to contain a certain degree of circularity, it is seen to yield practical benefits. For instance, we are able to transfer such algorithms between arbitrary problem domains and to compare different algorithms fairly, independent of a particular problem.

2 Formal Representations

In tackling a domain of search problems, we often prefer to search over a set of structures (chromosomes) representing the objects in the search space rather than directly over the objects themselves. Use of such a representation (made up of a representation space and associated growth function) makes the search algorithm much more generic. A general method for defining a representation is to classify subsets of solutions according to characteristics which they share.

In his seminal work, Holland (1975) proceeded using exactly this approach. He identified subsets of a search space of binary strings using *schemata*—sets of strings that share particular bit values. His Schema Theorem shows how the *observed* fitness of any schema in a population can be used to bound the *expected* instantiation of the same schema in the next generation, under the action of fitness-proportionate selection. Several authors then generalised the notion of a schema and have shown that the theorem applies

to arbitrary subsets of the search space, provided that suitable disruption coefficients are chosen (Radcliffe, 1991a; Vose & Liepins, 1991).

In particular, Radcliffe (1991a, 1991b) has developed the idea of *forma analysis*, in which general subsets of the search space are termed *formae*. Typically, the formae are defined as the equivalence classes induced by a set of equivalence relations, although this need not be the case. Any solution can then be identified by specifying the equivalence class to which it belongs for each of the equivalence relations (provided the set of relations is sufficiently rich). Loosely speaking, we identify genes with a set of basic equivalence relations and alleles with the corresponding equivalence classes. For instance, in a search space of faces, "same hair colour" and "same eye colour" might be two basic equivalence relations, which would induce the formae "red hair", "brown hair", "blue eyes", etc. Higher order formae are then constructed by intersection, e.g. "brown hair and blue eyes". Chromosomes made up of strings of alleles can then be used to *represent* the original structures of the search space (faces in our example). These chromosomes make up the representation space and the objects they encode define the growth function.

In certain cases, the genes are orthogonal, meaning that any combination of allele values represents a valid solution, but in many cases this is not so (certain alleles are incompatible). It is also not always easy to define equivalence relations, so we simply identify particular subsets of the search space that share some characteristic. In such cases, genes are not defined and a chromosome consists simply of a set of "alleles". For instance, in the TSP, we could identify $n(n-1)/2$ subsets of the search space, each containing all tours in which city i is linked to city j. Then a particular tour would be represented by the set of (undirected) edges it contained. Although here each chromosome would have the same number of alleles, this need not be so, and the ideas generalise easily to variable-length chromosomes (Radcliffe, 1992).

A problem-dependent *characterisation* procedure is used to generate the required equivalences for any instance of a given problem (see figure 1), and captures all of the structure that will be exploited by a search algorithm. The selection of an appropriate characterisation for a particular problem domain is an open problem. However, several *design principles* have been previously proposed (Radcliffe, 1991a). The most important of these is that the generated formae should group together solutions of related fitness (Radcliffe & Surry, 1994a), in order to create structure which can be exploited by the move operators. (It must also be possible to find a member of any given formae in "reasonable" time without resorting to enumeration, but this is true of most "reasonable" characterisations.)

Examples of several representations designed for various problem domains are shown in table 1. These include the traditional binary representation for real parameters, the "Dedekind" representation[2] for real parameters introduced in Surry & Radcliffe (1996), two natural representations for the traveling sales-rep problem (for comparisons of these and others see Radcliffe & Surry, 1994a), and two representations for subset-selection problems (used in neural-network topology optimisation), one in which only set membership is considered to be important and one in which both membership and non-membership is used (Radcliffe, 1992).

[2] A Dedekind cut is a partitioning of the rational numbers into two non-empty sets, such that all the members of one are less than all those of the other. For example, the irrationals are defined as Dedekind cuts on the rationals (e.g. $\sqrt{2} \triangleq \langle \{x \mid x^2 > 2\}, \{x \mid x^2 < 2\} \rangle$).

Representation	Basic formae	Genes	Degeneracy		
			Orthogonal	Redundancy	
Binary-coded reals	value has ith bit equal to j	yes	yes	none	none
Dedekind real parameters	value above/below cut at i	yes	no	none	huge
TSP: City positions	city i in position j	yes	no	$\times 2n$	low
TSP: Undirected edges	contains link ij	no	no	none	low
Subset-selection: inclusive	includes ith element	no	yes	none	none
Subset-selection: incl/excl	incl/excl ith element	yes	yes	none	none

Table 1. This table summarises the characteristics of several representations for different problem domains. Basic formae indicates the way in which basic subsets of the search space are identified, and the existence of genes is noted. Orthogonal representations are those in which any combination of alleles define a valid solution. Degeneracy occurs when multiple chromosomes represent the same solution, and redundancy is the amount of excess information in the chromosome.

3 Formal Algorithms

Traditional evolutionary algorithms are typically defined using a set of move operators which assume a particular form of the representation space. For example, many genetic algorithms assume chromosomes are binary strings, and most evolution strategies assume chromosomes are strings of real parameter values. Although some of the operators used by such algorithms can be generalised straightforwardly to related representation spaces (for example N-point crossover between binary strings is easily generalised to k-ary chromosomes), they typically are not general enough to handle arbitrary representations. In particular, variable-length genomes and non-orthogonal representations both present difficulties, and have generally led in the past to *ad hoc* construction of problem-specific move operators (for example in the traveling sales-rep problem).

We seek to define formal algorithms using move operators which manipulate the subset membership properties of chromosomes, as generated by *any* representation. Such algorithms are completely independent of representation, and can be applied to any problem domain by instantiating them with a representation appropriate to that domain.

A number of design principles have been proposed to facilitate the development of simple structure-preserving move operators. This has led to the definition of a number of *representation-independent* recombination and mutation operators, permitting the construction of truly representation-independent algorithms. These design principles (Radcliffe, 1991a; 1994) and associated operators include:

1. *Respect.* Respect requires that children produced by recombination are members of all formae to which both their parents belong. For example, if our representation included equivalence relations about hair colour and eye colour, then if both parents had red hair and green eyes, so should all of the children produced by a respectful crossover operator.

R^3: Random respectful recombination is defined as that operator which selects a child uniformly at random from the set of all solutions which share all characteristics possessed by both parents (their *similarity set*).

2. *Transmission.* A recombination operator is said to be *strictly transmitting* if every child it produces is equivalent to one of its parents under each of the basic equivalence relations (loosely, every gene is set to an allele which is taken from one or other parent). Thus, if one parent had red hair and the other had brown hair, then transmission would require that the child had either red or brown hair.
   RTR: The random transmitting recombination operator is defined as that operator which selects a child uniformly at random from the set of all solutions belonging only to basic formae present in either of the parents (their *dynastic potential*).
3. *Assortment.* Assortment requires that a recombination operator be capable of generating a child with any compatible characteristics taken from the two parents. In our example above, if one parent had green eyes and the other had red hair, and if those two characteristics are compatible, assortment would require that we could generate a child with green eyes and red hair.
   RAR: The random assorting recombination operator, a generalised form of uniform crossover, has been previously defined (Radcliffe, 1992). It proceeds by placing all alleles from both parents in a conceptual bag (possibly with different multiplicities), and then repeatedly draws out alleles for insertion into the child, discarding them if they are incompatible with those already there. If the bag empties before the child is complete, which can happen if not all combinations of alleles are legal (so that the representation is *non-orthogonal*) remaining genes are set to random values that are compatible with the alleles already present in the child.
   GNX: A generalised version of N-point crossover has also been defined (Radcliffe & Surry, 1994a). This proceeds in much the same way as standard N-point crossover, dividing the two parents with N cut-points, and then using genetic material from alternating segments. The alleles within each segment are tested in a random order for inclusion in the child, and any remaining gaps are patched by randomly selecting compatible alleles first from the unused alleles in the parents, and then from all possible alleles.
4. *Ergodicity.* This demands that we select operators such that it is possible to move from any location in the search space to any other by their repeated action. (Typically a standard mutation operator is sufficient.)
   BMM: Binomial minimal mutation, a generalisation of standard point-wise mutation, has been proposed in Radcliffe & Surry (1994a). Minimal mutations are defined to be those moves which change the fewest possible number of alleles in a solution (in non-orthogonal representations it may be necessary to change more than one allele at a time to maintain legality). BMM performs a binomially-distributed number (parameterised by the genome length and a gene-wise mutation probability) of minimal mutations, and does not forbid mutations which 'undo' previous ones.
   Hill-climbers: the definition of representation-independent "minimal mutation" allows us to define a number of representation-independent hill-climbing operators, and to define *memetic algorithms* based on the idea of searching over a sub-space of local-optima (Radcliffe & Surry, 1994b).

Using these operators, we can define algorithms which are independent from any particular representation or problem, such as the example shown below. Note that every step of the algorithm is precisely defined, and that given a representation of a problem domain, we can mathematically derive a concrete search strategy suitable for implementation on a digital computer (see section 4). This is different from traditional evolution-

ary algorithms, in which steps 4 and 5 would have to be modified for any problem domain which required a new representation space.

A representation-independent evolutionary algorithm

1. Generate an initial population by randomly sampling p times from the space of chromosomes.
2. Evaluate the p members of the initial population via the growth and fitness functions.
3. Select two parents using binary-tournament selection.
4. Recombine the parents using RAR.
5. Mutate the resulting child using BMM.
6. If the child does not exist in the population, evaluate it and replace the member with the worst fitness.
7. Repeat to step 3 until termination criterion.

4 Search Strategies

In order to construct a practical search strategy for a given problem domain, we simply combine a formal algorithm with an appropriate representation of the problem domain. There is no need to construct new move operators, as we simply instantiate those defined in the formal algorithm of choice. Since *exactly* the same formal algorithm (for example, that shown above) can be instantiated for two different representations (of either the same or different problem domains), one can make much more definite statements about the quality of the algorithm itself (as it is defined independently of any problem). It is also possible to fix the representation and vary the algorithm, allowing more meaningful comparisons between algorithms.

For several of the representations shown in table 1, the generalised operators defined in section 3 reduce to traditional forms. For example, for any orthogonal representation, R^3, RTR and RAR all reduce to uniform crossover (Syswerda, 1989), GNX reduces to N-pt crossover, and BMM becomes simple gene-wise point mutation. For the Dedekind real representation, R^3, RTR and RAR reduce to blend crossover with parameter $\alpha = 0$, as defined by Eshelman & Schaffer (1992) and widely used in evolution strategies (Baeck *et al.,* 1991), and BMM is equivalent to gaussian creep mutation (Surry & Radcliffe, 1996). If we consider the undirected edge representation for the traveling sales-rep problem, RAR becomes a variant of edge recombination (Whitley *et al.,* 1989) and R^3 reduces to a weaker version of the same operator. BMM here involves a binomially distributed number of sub-tour inversions, whereas for the city-position representation, BMM reduces to a binomially distributed number of city exchanges (Radcliffe & Surry, 1994a).

Such reductions imply that formal algorithms defined using these operators reduce to commonly used search strategies in the relevant problem domains. To illustrate, the algorithm shown above is instantiated below for both the traveling-sales rep problem using the undirected-edge representation, and for a real-parameter function optimisation problem using the Dedekind representation. This results on the one hand on a strategy based on edge-recombination and sub-tour inversions, and on the other in one based on blend-crossover and gaussian creep-mutation. Both of these strategies have been widely used in their respective domains, but it was not clear before now that they were exactly the same formal algorithm.

Search strategy as algorithm plus representation		
Problem domain:	**TSP**	**Real-parameter opt.**
Representation:	**Undirected-edges**	**Dedekind**
Choose initial population:	of random tours	of random vectors
Evaluate each solution:	by measuring tour length	using provided $f(\mathbf{x})$
Select two parents:	using binary-tournament selection	
Recombine parents using:	variant of edge-recomb.	BLX-0
Mutate the child with :	binomial number of sub-tour inversions	gaussian creep-mutation for each parameter
Evaluate, replace worst:	if the child does not exist in the population	
Repeat:	until termination criterion	

Note that both the representation and algorithm are mathematical constructions and need not be directly related to the actual way in which the data structures and computer code for the resulting search strategy is implemented on a digital computer[3]. Thus, rather than simply plugging together different bits of computer code, we plug together different bits of mathematics from which we can formally *derive* an actual implementation in a well-specified way. For example, the (formal) Dedekind representation for real numbers has (in the limit) an infinite number of genes, yet it is a simple matter to mathematically derive forms of the various operators suitable for (finite!) implementation.

5 Summary

This paper has presented a more formal approach to evolutionary search, by separating a search strategy into a representation and an algorithm. We have introduced a disciplined methodology for attacking new problem domains—instead of simply using evolutionary "ideas" to invent new operators, one need only provide a *characterisation* of the problem that explicitly captures beliefs about its structure, and then instantiate an existing algorithm with the derived representation. This applies equally to problems with non-orthogonal representations where traditional evolutionary algorithms are inapplicable. We have demonstrated, by way of example, that identical algorithms can be applied to both the TSP and real parameter optimisation, yielding familiar (but apparently quite different) concrete search strategies.

Because these formal algorithms are independent of any particular representation, it is possible to transfer them to arbitrary problem domains, and to make meaningful comparisons between them. By making the rôle of domain knowledge more explicit we are also directed to more reasoned investigation of what makes a good representation for a given problem. Further investigations will build on these ideas to construct a more complete taxonomy of representations, and to investigate issues of algorithmic performance and quality of representation.

[3] Thus the title of this paper has been inspired by but differentiated carefully from the prior works by Wirth (1976) and Michalewicz (1992).

References

T. Bäck, F. Hoffmeister, and H.-P. Schwefel, 1991. A survey of evolution strategies. In *Proceedings of the Fourth International Conference on Genetic Algorithms*. Morgan Kaufmann (San Mateo).

L. J. Eshelman and D. J. Schaffer, 1992. Real-coded genetic algorithms and interval schemata. In D. Whitley, editor, *Foundations of Genetic Algorithms 2*. Morgan Kaufmann (San Mateo, CA).

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

D. E. Goldberg, 1990. Real-coded genetic algorithms, virtual alphabets, and blocking. Technical Report IlliGAL Report No. 90001, Department of General Engineering, University of Illinois at Urbana-Champaign.

J. J. Grefenstette, 1984. GENESIS: A system for using genetic search procedures. In *Proceedings of the 1984 Conference on Intelligent Systems and Machines*, pages 161–165.

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

Z. Michalewicz, 1992. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer Verlag (Berlin).

I. M. Oliver, D. J. Smith, and J. R. C. Holland, 1987. A study of permutation crossover operators on the travelling salesman problem. In *Proceedings of the Third International Conference on Genetic Algorithms*. Morgan Kaufmann (San Mateo).

N. J. Radcliffe and P. D. Surry, 1994a. Fitness variance of formae and performance prediction. In L. D. Whitley and M. D. Vose, editors, *Foundations of Genetic Algorithms III*, pages 51–72. Morgan Kaufmann (San Mateo, CA).

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

N. J. Radcliffe and P. D. Surry, 1995. Fundamental limitations on search algorithms: Evolutionary computing in perspective. In J. van Leeuwen, editor, *Computer Science Today: Recent Trends and Developments, Lecture Notes in Computer Science, Volume 1000*, pages 275–291. Springer-Verlag (New York).

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

N. J. Radcliffe, 1991b. Forma analysis and random respectful recombination. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 222–229. Morgan Kaufmann (San Mateo).

N. J. Radcliffe, 1992. Genetic set recombination. In D. Whitley, editor, *Foundations of Genetic Algorithms 2*. Morgan Kaufmann (San Mateo, CA).

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

P. D. Surry and N. J. Radcliffe, 1996. A formalism for real-parameter evolutionary algorithms and directed recombination. In *submitted to Foundations of Genetic Algorithms IV*. (San Diego).

G. Syswerda, 1989. Uniform crossover in genetic algorithms. In *Proceedings of the Third International Conference on Genetic Algorithms*. Morgan Kaufmann (San Mateo).

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).

D. Whitley, T. Starkweather, and D. Fuquay, 1989. Scheduling problems and traveling salesmen: The genetic edge recombination operator. In *Proceedings of the Third International Conference on Genetic Algorithms*. Morgan Kaufmann (San Mateo).

N. Wirth, 1976. *Algorithms + Data Structures = Programs*. Prentice-Hall (Englewood Cliffs, NJ).

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

This article was processed using the LaTeX macro package with LLNCS style