Chapter 2

Evolutionary Algorithms

Evolutionary computation has a long history. Unfortunately, this history is virtually unknown to many practitioners in the field.

David Fogel, 1998

This chapter includes a brief overview and historical perspective on evolutionary algorithms (EAs), an introduction to genetic algorithms (GAs), as well as three literature surveys. These provide the context for the evolutionary aspects of the author's research, as well as establishing the foundation on which it was built.

The chapter is divided into six sections: §2.1 is a brief overview of EAs. A tutorial introduction to GAs, Goldberg's Simple Genetic Algorithm and GA/heuristic hybrids is given in §2.2. A near-exhaustive survey of evolutionary telecommunications, *i.e.* the application of EAs to the telecommunications problem domain, up to December 1998 is presented in §2.3. Then §2.4 and §2.5 provide more limited surveys of prior work on GA/heuristic hybrids and operator-probability adaptation, respectively, in areas other than evolutionary telecommunications. Finally, the chapter is summarised in §2.6.

2.1 Overview of Evolutionary Algorithms

Early work

The history of EAs is longer than most practitioners in the field realise. However, this has been remedied in recent times by both a concise history and summary of the state of the art by Bäck *et al.* [38] in 1997, as well as by David Fogel's 1998 edited volume of both early and important papers in evolutionary computation [4]. The origin of EAs dates back to pioneering work in the 1950s, even before adequate computing power for them was available [38]. In particular, David Fogel [4] has highlighted early research by Friedman [39], Fraser [40], Box [41] and Friedberg [42,43].

In his 1956 master's thesis, Friedman [39] described how control circuits employing 'basic neural elements' could be evolved for autonomous robots using random selection and mutation. His work anticipated by decades more recent efforts in robot control, evolving artificial neural networks, and evolving electronic circuits. Friedman's research was theoretical—there was no practical implementation—but by analysis of fitness landscapes, he endeavoured to show the superiority of his suggested approach over purely random search.

Then, as part of a computer-based simulation study of genetic systems in 1957, Fraser [40] introduced an approach which clearly anticipated the bit-string GA later studied by Holland [44], Goldberg [2] and others. Fraser's algorithm employs diploid bit-strings and a multi-point crossover operator for recombination, with different crossing probabilities at different points along the genotype so as to vary linkage between the genes. His algorithm is generational, with a selection mechanism that produces more offspring than required and then removes the excess according to some function of their phenotype fitness. This foreshadowed the (μ, λ) selection method of evolution strategies (ESs).

Box [41], also in 1957, introduced evolutionary operation as an online approach to improving the productivity of an industrial process. During an agreed period, a systematic set of mutations is applied to the current production settings, with performance data gathered from the running plant. At the end of the period the best settings found become the new current production settings, and systematic experimentation recommences. While Box's approach has been criticised by others (*e.g.* Goldberg [2]), it nevertheless represents a successful early application of both mutation and fitness-based selection.

Finally, Friedberg *et al.* [42, 43] in 1958 developed some of the earliest research on the evolution of computer programs. These are represented as binary instructions, and a credit-assignment mechanism is used to apportion credit to individual instructions. The approach suffered from a number of limitations motivated and exacerbated by the restricted computing power available to Friedberg *et al.*. Nevertheless, the work clearly had similarities to the later development of both GAs and classifier systems (CSs). Moreover, concepts such as Holland's implicit parallelism and schemata [44] were anticipated, the importance of incorporating domain knowledge into the algorithm was recognised, and a number of insights obtained that were subsequently rediscovered by the evolutionary computation (EC) community [4].

Genetic algorithms (GAs)

GAs were introduced by Holland in 1962 [44,45], originally intended as a general model of adaptive processes [38], but subsequently widely adopted as optimisers [2,3,9]. They are inspired by natural genetic processes, and thus have a population of chromosomes, each analogous to the DNA in a natural organism [46]. The population evolves by simulated evolution, employing genetic operators modelled on those in nature. The overall process is driven by fitness, determined by decoding the chromosome and applying a problem-specific objective function, analogous to the reproductive success of an individual organism matured from its DNA.

Their popularity can arguably be traced back to two extensively cited books by Goldberg [2] and Davis [9] in 1989 and 1991 respectively. As well as a clear description of the basics of GAs, Goldberg [2] provides a complete Pascal implementation, a survey of applications, advanced operators and techniques, and an introduction to CSs. In his book, Davis [9] describes the step-by-step development of a GA/heuristic hybrid from a simple GA, and then includes a wide-ranging edited collection of application case studies of hybrid GAs.

While the historical distinctions between EAs have blurred [4], GAs are now, by far, the strongest research area. However, to make this assertion clearly, GA/heuristic hybrids [9] must also be regarded as GAs. Some researchers would argue [47], though, that such algorithms bear little resemblance to the canonical GA, and should simply be described as EAs

Evolutionary programming (EP)

EP originated with Lawrence Fogel in 1962 [48–50] as an attempt to create artificial intelligence [38]. The population of individuals being evolved are typically finite state machines (FSMs), and the emphasis is on the evolution of behaviour from parents to offspring, rather than on modelling genetic operations as in GAs [46]. The inspiration for EP is an abstraction of evolution at the species level, consequently recombination (crossover) is not usually included, although even in the early days, this was not always so [47]. Although originally more focused, EP does not now impose any restriction on the representation of an individual in the population [4, 46]: rather a ‘natural’ representation for the problem is selected (*cf.* hybrid GAs). Mutation is stochastic, usually determined such that small variations are common and large variations less so; in addition, adaptation of mutational variation is now often applied during a run. Selection, although originally deterministic, is now typically tournament-based [4].

Evolution strategies (ESs)

ESs were developed by Rechenberg and Schwefel in 1965 [51, 52], with the aim of solving mainly experimental parameter optimisation problems [38]. An individual in the population is an array of real-valued parameters: there is no encoding. ESs were originally based on a 'population' of size one, with the parent competing each time for survival with a single offspring: the (1+1)-ES. Now, however, ESs typically use a larger population, size μ. From this, λ offspring are obtained by mutation (and sometimes recombination). The best μ individuals survive deterministically, selected from either the combined pool of parents plus offspring, termed $(\mu + \lambda)$, or just from amongst the offspring, (μ, λ). Mutation of an individual's parameters is Gaussian (normal) about the current position. In addition, strategy parameters are usually included in the individual, one for each problem parameter; each of these determine the mutation rate (standard deviation) for the corresponding parameter. The strategy parameters are themselves mutated, although according to a global log-normal distribution.

While ESs have achieved a measure of success, particularly within civil engineering [46], its popularity has been limited by much of the early literature being in German. This was remedied in 1995 with a book by Schwefel [52].

Genetic programming (GP)

As has already been noted, efforts to evolve computer programs date back to the earliest days of EC, with the work of Friedberg *et al.* in 1958 [42, 43]. However, these early machine-code based efforts were not particularly fruitful. Better results awaited more powerful computers, and also more expressive representations. In particular, the FSMs of early EP (see above), the if-then rules of CSs (see below), and the LISP s-expressions of what came to be known as GP have all had a measure of success. While several researchers in the 1980s suggested the use of LISP s-expressions for program evolution, including Hicklin, Fujiki and Cramer [4], it was the work of Koza from 1989 [53] which established the GP field.

In GP [54], an individual in the population is usually a single LISP s-expression (*i.e.* a parse tree), composed from a user-specified, problem-specific set of functions and terminals. GP crossover, usually the only genetic operator, is based on the exchange of sub-trees. Fitness of the individuals is established by evaluating an individual program, either directly or indirectly, as a solution to the problem. Selection was originally stochastic and fitness-proportional [54], but subsequently tournament-based selection has been preferred [55].

In the last decade, GP has been developed by Koza [55, 56] and others [57–59], and has also diversified to include a wide-range of representations and operators [60]. It has even returned to the successful evolution of machine-code programs [61].

Classifier systems (CSs)

CSs[1] were introduced by Holland and Reitman in 1978 [62] as an attempt to develop adaptive behaviour by evolving a set of classifiers (*i.e.* condition-message pairs). In a CS, a set of detectors (*i.e.* the program input) posts messages to a message list [2]. Each classifier (in parallel) is activated when it detects a matching condition, and competes to post its message. While messages are binary, the classifier conditions can include 'wild cards' (usually represented as '#'), allowing an individual classifier to respond to a range of conditions. The message list in turn can trigger effectors, resulting in program output. Evolution of the set of classifiers is achieved by assigning credit to successful classifiers; they in turn pass some of this credit on to other classifiers that contributed to their success (the 'bucket brigade'). Periodically, accumulated credit is used as fitness to replace a limited subset of the population with a modified GA, hopefully resulting in an improvement of the overall adaptive behaviour of the system as a whole.

Subsequent development of CSs have included evolving entire sets of classifiers (the 'Pittsburgh' approach) rather than individual classifiers (the 'Michigan' approach) [4], and a wide variety of different credit-assignment mechanisms [46].

Evolutionary hardware (EH)

A relatively recent development in EAs has been the emergence of the EH field. This includes EAs where the evolution occurs in purpose-built processors (rather than in software on general-purpose processors), EAs where the fitness evaluation occurs in hardware (rather than *e.g.* software simulation), or EAs that combine both of these. It could be argued that EH is not really a distinct field, as the only difference is in the implementation of the algorithms or their fitness evaluation. Indeed, much of the work on ESs could be considered EH. Nevertheless, there is a growing community focusing on this specialised aspect of EAs, and the interested reader is referred to the recent books by Sipper [63], Thompson [64] and Mange & Tomassini [65].

[1] CSs have been given a variety of names over the years, including learning classifier systems (LCSs) [46].

Conference	Years
Congress on Evolutionary Computation (CEC)	1999, 2000
Genetic and Evolutionary Computation Conf. (GECCO)	1999, 2000
Intl. Conf. on Genetic Algorithms (ICGA)	1985, 1987, 1989, 1991, 1993, 1995, 1997
IEEE Intl. Conf. on Evolutionary Computing (ICEC)	1994, 1995, 1996, 1997, 1998
Parallel Problem Solving from Nature (PPSN)	1991, 1992, 1994, 1996, 1998, 2000
Conf. on Evolutionary Programming (EP)	1992, 1993, 1994, 1995, 1996, 1997, 1998
Genetic Programming Conf. (GP)	1996, 1997, 1998
Foundations of Genetic Algorithms (FOGA)	1991, 1993, 1995, 1997, 1999
Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA)	1995, 1997
Intl. Conf. on Artificial Neural Networks and Genetic Algorithms (ICANNGA)	1993, 1995, 1997, 1999

Table 2.1: Major evolutionary computation conferences

Conferences & books

In completing this overview, it is worth noting that the growth in popularity of EAs has been accompanied by a growth in the number and size of the conferences in the area and in the number of books published. Some of the major conferences, and the years they took place, are recorded in Table 2.1. In addition, based on the bibliography assembled by Koza [66], Table 2.2 records EC books published in chronological order[2], whereas Table 2.3 shows them by subject area.

2.2 Tutorial on Genetic Algorithms

This section provides a tutorial introduction to GAs. Its main purpose, following some background on both GAs in general and Goldberg's Simple Genetic Algorithm in particular, is to introduce GA/heuristic hybrids. This latter approach is a central theme of the thesis. However, the tutorial does not give sufficient detail to fully appreciate the literature surveys later in this chapter (*cf.* §1.1).

[2] In Tables 2.2 and 2.3, only English language books are included, and books are recorded only in their most recent editions, so *e.g.* Holland's *Adaptation in Natural and Artificial Systems* is represented by the 1992 edition [45] rather than the 1975 [44].

Year	Count	Books
1966	1	[49]
1983	1	[67]
1986	1	[68]
1987	1	[69]
1989	2	[2], [70]
1990	1	[71]
1991	3	[9], [72], [73]
1992	3	[74], [45], [54]
1993	1	[75]
1994	5	[76], [77], [78], [57], [55]
1995	5	[79], [80], [81], [82], [52]
1996	11	[83], [84], [85], [86], [87], [3], [88], [89], [90], [58], [91]
1997	15	[92], [93], [94], [95], [96], [97], [98], [99], [100], [101], [102], [103], [104], [61], [63]
1998	10	[105], [4], [106], [107], [60], [108], [109], [110], [64], [65]
1999	15	[111], [112], [113], [114], [115], [116], [117], [118], [119], [120], [121], [56], [59], [122], [50]
2000	1	[123]
Total	76	

Table 2.2: Evolutionary computation books – chronological order

Subject	Count	Books
EC	14	[67], [79], [80], [83], [92], [93], [94], [95], [96], [105], [4], [111], [112], [113]
GAs	38	[69], [2], [70], [71], [9], [74], [45], [75], [76], [77], [78], [81], [82], [84], [85], [86], [87], [3], [88], [89], [90], [97], [98], [99], [100], [101], [102], [103], [106], [107], [114], [115], [116], [117], [118], [119], [120], [121]
EP	4	[49], [72], [109], [50]
ESs	1	[52]
GP	13	[54], [57], [55], [58], [91], [104], [61], [60], [108], [56], [59], [122], [123]
CSs	3	[68], [73], [110]
EH	3	[63], [64], [65]
Total	76	

Table 2.3: Evolutionary computation books – by subject area

2.2.1 Genetic Algorithms: An Introduction

Goal of optimisation

When engineers attempt to optimise the parameters of a problem, or search for the global maximum (or minimum) of an objective function, what is their actual goal? In a large and complex problem space, where it may never be possible to find the true global optimum, surely a 'good' solution is sufficient? In thinking about optimising a problem, it is possible to be biased into thinking that only the true, global optimum is sufficient. However, in complex systems, as in the natural world, a solution that is 'good enough' may suffice. The goal is not so much to find the global optimum solution, but rather a solution that is the best that can be achieved with the time and resources available. It is with complex problems such as these, that the robustness and generality of modern heuristic search methods, such as EAs, becomes apparent. As Wasserman has observed [124]: "Sam and Ed are on a camping trip when they see a charging bear. Sam immediately starts to lace up his running shoes. Ed says, 'Sam, don't bother; you'll never outrun that bear.' Sam replies, 'I don't have to outrun the bear. I only have to outrun you.'"

Evolutionary algorithms

EAs are based on a simplification of the mechanisms of natural evolution *i.e.* natural selection and genetics. As has already been noted, a variety of different EAs have been proposed [4]. However, they all share a common basis of simulating the evolution of individual structures using *selection*, *reproduction* and *mutation*. These processes in turn depend on the apparent *fitness* of the individuals in the context of their environment. Thus EAs maintain a *population* of structures that reproduce under the influence of selection and other genetic operators, such as mutation and, in some cases, *recombination*. Reproduction focuses attention on high-fitness individuals, thus exploiting the available fitness information. Recombination and mutation modify those individuals, thus promoting exploration of new possibilities. Although EAs are simplistic from a biologist's point of view, nevertheless, they are still sufficiently complex to act as robust and powerful adaptive search techniques [46].

Genetic algorithms

Against this general background of EAs, GAs can be distinguished by two key features. First of all, they operate in a search space using genetic operators on fixed-length *strings* that encode points in the problem space, rather than on the problem space directly. In addition, they use a particular recombination operator on these fixed-length strings that is unique to GAs, called *crossover*.

Chromosomes

The use of fixed-length strings in GAs is inspired by the encoding of the genetic inheritance of an individual biological organism in the DNA that makes up its chromosomes. The strands of DNA in the cells of, say, a human being, encode its entire genetic heritage in a four-'letter' alphabet. Thus, in a GA, each individual in the population contains a (fixed-length) string (also sometimes called a chromosome), usually, but not always, using a binary alphabet. The use of such a binary representation follows from schema theory [2], which suggests that a two-symbol encoding promotes more effective search.

Fitness

The fitness of a biological organism is measured by its interaction with the environment. Its fitness determines (to some extent) how long it will survive, what opportunities it will have to reproduce, and consequently, how many offspring (or children) it will have. In a GA there is not usually an environment as such, but rather the fitness of an individual is determined by decoding its chromosome (*i.e.* the fixed-length binary string) and thereby interpreting it as a point in the problem space (Fig. 2.1). The (possibly modified) value of the objective function for the problem is then fed back to the GA as the fitness of that individual. The fitness value is then used to determine the extent to which that individual is allowed to reproduce into the next generation.

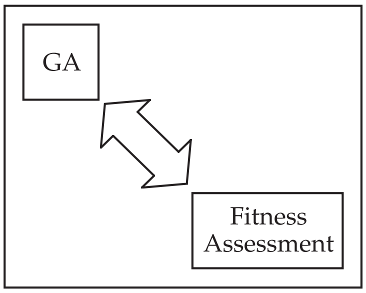


Figure 2.1: Genetic algorithm and fitness assessment

Crossover

What then of the evolution of the population from one generation to the next? As well as reproduction as a function of fitness (which in a GA may simply be *fitness-proportional*), many biological organisms use sexual reproduction to allow the blending of genetic characteristics from two organisms to be reflected in their offspring, such that at least some of the children will be 'fitter' than either of their parents. Thus in GAs, parents that are selected to reproduce will (with a certain probability) not simply duplicate their chromosomes in their children (*i.e.* cloning), but instead two parents will recombine their chromosomes using the crossover operator such that the chromosomes of their children are a combination of both their parents.

Mutation

In the natural world, another factor that operates on biological organisms' genetic heritage is mutation—the apparently random modification of some of the genetic material inherited by an organism. Most such mutations would appear to be harmful[3], but occasionally one will be beneficial, resulting in increased fitness for the organism, and consequently the mutated genetic material will be passed on to more and more organisms as the generations pass. Inspired by this, GAs include the possibility of the mutation of a child's chromosome during reproduction by randomly changing (at low probability) each of the bits (or letters for a non-binary alphabet) of its chromosome. For GAs though, this operator is regarded as much less important than fitness-proportional reproduction and crossover, and so the probability of any single bit being mutated is usually kept very low [2].

Population and generations

In contrast with natural populations, the population of individuals in a GA is often maintained at a constant size (for simplicity). The initial starting population (which has no analogue in the natural world) is usually randomly generated, although it may be seeded with a few individuals encoding better-than-average attempts at the problem. In addition, many GAs are *generational*—a new population of individuals is created from the old by reproduction, crossover and mutation, and then the old generation is discarded (thankfully this is not done with human populations!).

Distinctive qualities

What then are the main differences between GAs and other search algorithms? Goldberg [2] identifies four. GAs work with a coding of the problem's parameters, not with the parameters directly. They employ the objective function values themselves, rather than derivatives. They

[3]Unless there is considerable redundancy in the chromosome, as there often is in nature, in which case the majority will actually be neutral, *i.e.* no observable impact on fitness; it is the majority of *expressed* mutations that are actually harmful.

search using a population of points, instead of a single point. Finally, GAs incorporate probabilistic choices, rather than purely deterministic.

Because the problem parameters are coded as finite-length (binary) strings, and the GA then operates on these directly using only objective function values, *i.e.* without problem-specific information (at least for a 'pure' GA) or even derivatives, a very broad range of problems can be tackled successfully. Although it should be noted that considerable thought can still be required to develop a suitable binary encoding for a complex problem domain. Clearly, using a 'database' of points in the problem space (*i.e.* the population of strings), rather than a single point, immediately reduces the chance of getting trapped in a local optimum. Finally, the probabilistic choices in the heart of the GA, far from indicating directionless search, are actually used to guide the algorithm to explore areas of the search space most likely to lead to improvement. Further, by re-running a GA several times with the same parameters (apart from the random seeds), the probability of finding a solution of a given quality can be greatly enhanced.

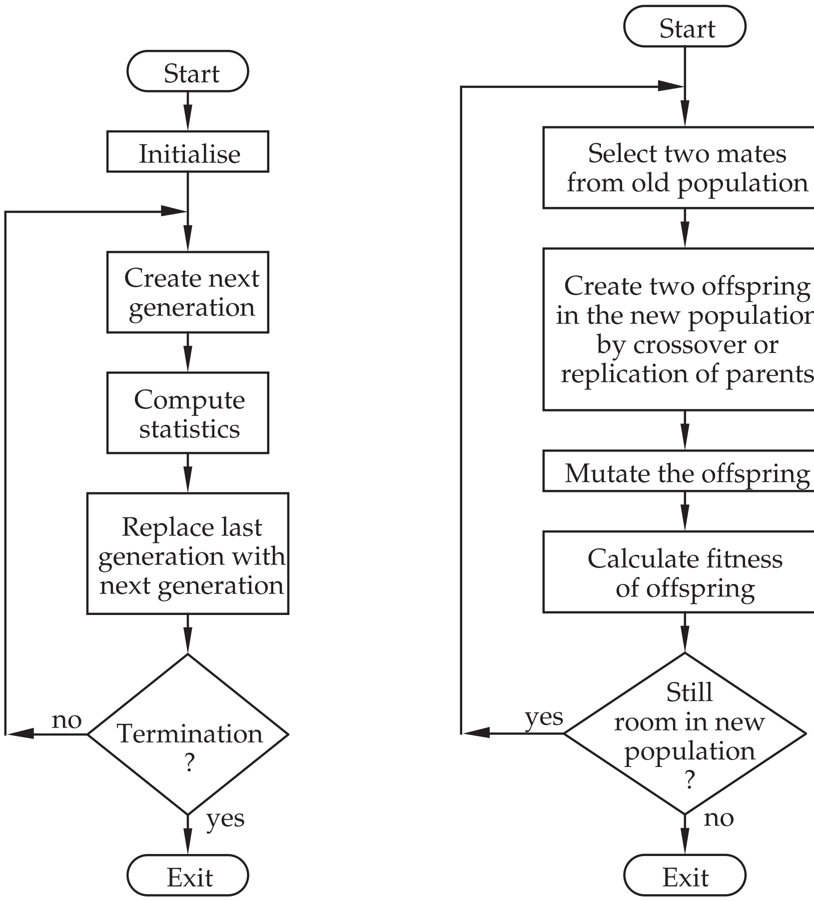
2.2.2 The Simple Genetic Algorithm

Overall algorithm

To make the description of the genetic algorithm above a little more concrete, it is perhaps beneficial to examine Goldberg's Simple Genetic Algorithm (SGA) [2]. An overall flowchart for the algorithm is given in Fig. 2.2(a). After creating the initial population (random binary strings), assessing the fitness of each individual and calculating overall population statistics, the algorithm enters its main loop. Each generation is created from the one before, using selection, reproduction, crossover and mutation. The population statistics are computed again, including calculating average fitness and identifying if a new best-of-run individual has been found. The old generation is then replaced by the new. The algorithm continues to loop until a termination condition, usually a maximum number of generations, is met.

Create next generation

The 'Create next generation' subroutine can be further expanded, as shown in Fig. 2.2(b). First two individuals are selected from the old population. From these, two new offspring are created which, with a certain probability, are recombined using crossover. Then the new individuals, whether simple clones of their parents or not, are subject to mutation. Finally, the fitness of the new



(a) Overall (b) Create next generation

Figure 2.2: Simple Genetic Algorithm flowcharts

individuals is determined by decoding them and applying the objective function. The subroutine continues to loop until the new population has been completely filled with offspring.

Another way of looking at the details of the SGA is in terms of the relationship of one generation to the next (the new population to the old). The SGA is generational, *i.e.* it uses non-overlapping populations. Each generation is created entire from the one before, and then the old one is discarded. In contrast with natural populations, the population size is kept constant: each generation has the same number of individuals as the preceding one. This is illustrated in Fig. 2.3, which shows generation t of N individuals transformed into generation $t+1$ by the genetic operators: selection, reproduction, crossover and mutation.

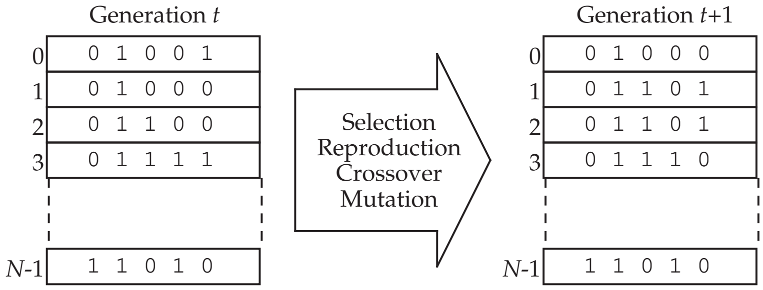


Figure 2.3: From one generation to the next (after [2])

Selection & reproduction

By way of illustration, consider a tiny example population of six strings, each of length 5 bits. The strings encode unsigned integers (x), and the objective function (fitness) is $f(x) = 2x^2 + x$ (Table 2.4). The simplest form of selection is fitness-proportional. The probability of a string being selected is simply its fraction of the sum-of-fitness of the whole population (Fig. 2.4). Fitness-proportional selection can be imagined to operate by analogy with a roulette wheel (or a 'wheel of fortune'). For every vacancy in the new population, say, the roulette wheel is spun, and where the ball lands (by randomly selecting a number between 0.0 and 1.0, representing the circumference of the wheel), that individual is reproduced in the new population. The chance an individual has of being selected and reproduced is thus proportional to its share of the wheel, and hence to its fitness. Selection is done 'with replacement' *i.e.* just because an individual is selected once does not mean it cannot be selected again. The wheel remains the same, and is simply spun again until the new population is filled. Therefore, the number of offspring each individual in the old population can expect to have in the new, is just that fraction of the new population that corresponds to its proportion of the total fitness in the old. In practice though, because of stochastic effects, an individual may have more (or less) than the expected number of offspring—in the limit, a very fit individual might have no offspring, or a very unfit individual might have many!

Crossover

For every pair of individuals in the new population (which are just clones of their parents at this stage), the crossover operator is applied with a certain probability (typically 0.6). Simple crossover operates by selecting a crossover point at random somewhere along the strings, *e.g.* if the strings are length k, a crossover point j would be selected between 1 and $k-1$. All the bits between 0 and $j-1$ inclusive would remain unaffected, but the bits between j and $k-1$ would be swapped between strings (Fig. 2.5). Between two strings from the tiny example population, say:

```
0 1 1 0|0
0 1 1 1|1
```

with $k = 5$ and $j = 4$ (indicated by the '`|`'s), the resulting crossover would yield:

```
0 1 1 0 1
0 1 1 1 0
```

Mutation

For GAs, fitness-proportional reproduction and crossover are the primary genetic operators, but something more is still needed. As a result of selection and crossover, the population will gradually converge on more promising areas of the search space. However, in any given generation, all members of the population may come to have the same bit value (say `1`) at a particular bit position—perhaps all those individuals with a `0` at that location proved unfit and died out. But what if later in the evolution of the population, better fitness could be found by incorporating a `0` at that bit position (as other locations have changed over time)? Selection and crossover alone can never bring back a `0` at that bit position—hence the need for mutation. In a binary string alphabet, mutation simply consists of, at low probability (say 0.001), toggling the bit at each position in a new individual—a `0` to a `1`; a `1` to a `0`. For non-binary alphabets, some variation must be used—perhaps selecting a letter at random from the alphabet, excluding the letter currently at that location. Although mutation brings needed diversity back into a population that might otherwise converge prematurely on less than ideal points in the search space, it must be used sparingly [2]. Just as mutation in

No.	String	x	Fitness $2x^2 + x$	% of Total
0	`01001`	9	171	5.2
1	`01000`	8	136	4.2
2	`01100`	12	300	9.2
3	`01111`	15	465	14.2
4	`10100`	20	820	25.1
5	`11010`	26	1378	42.1
Total			3270	100.0

Table 2.4: Sample problem strings and fitness values

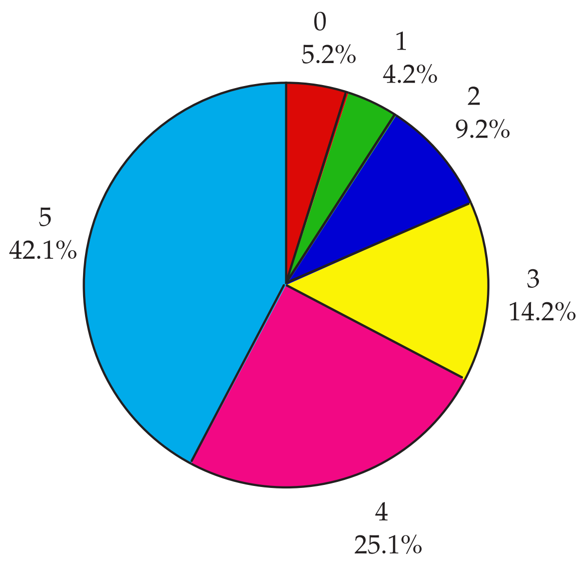


Figure 2.4: Roulette-wheel selection

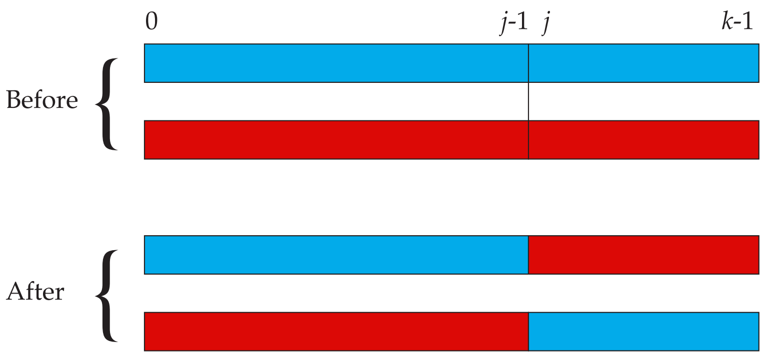


Figure 2.5: Crossover

the natural world is often harmful, so mutation in GAs often reduces the fitness of the individuals affected—it is after all just a form of random walk.

Improving the SGA

Although the SGA is a reasonable starting point, a wide variety of ways have been suggested to extend it [2]. These are usually inspired by either additional aspects of natural evolution, or by integrating other search algorithms or domain knowledge into the GA.

Instead of fitness-proportional selection, for example, fitness *ranking* can be used, where an individual's rank in the fitness-sorted population determines its selection probability, rather than its absolute fitness value. Another possibility is *tournament selection* where, whenever an individual is to be selected, a small number of structures, chosen randomly, but *not* according to fitness, compete directly in terms of their fitness values for the right to mate.

Alternatively, rather than abandoning fitness-proportional selection, the fitness values themselves can be suitably modified to maintain adequate selection pressure and avoid premature convergence. Variations include *windowing*, *i.e.* rescaling fitness values using the history of the last few generations, or using *linear*, *sigma truncation* or *power-law rescaling* mechanisms.

The SGA is strictly generational, but this too can be altered. One, or a few, of the better individuals can be passed unchanged into the new population (called *elitism*). At the other extreme, the population as a whole can be left largely unchanged, and only a few (usually the worst) individuals replaced (known as *steady-state reproduction*).

Modified or additional genetic operators can also be introduced. Instead of single-point crossover, used in the SGA, *multi-point* (usually two-point) *crossover* can be employed. In the limit, *uniform crossover* can be used to create an offspring by taking a bit at random, for each position along the string, from either parent.

The possibilities for extending the SGA are considerable, of which just a few have been mentioned, but some caution should be exercised. While for any particular problem, some of these changes will be beneficial, others may prove counter-productive. The next section describes a more promising alternative.

2.2.3 Genetic-algorithm/Heuristic Hybrids

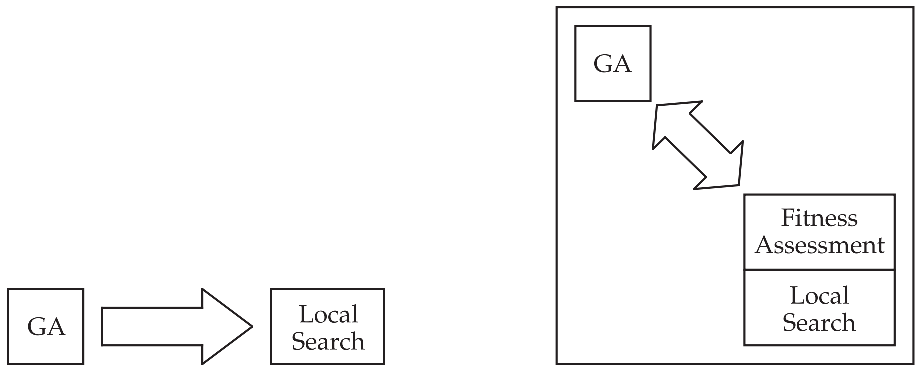
Incorporating other search algorithms

How else then can the SGA be improved? Most of the extensions to the SGA mentioned above were biologically inspired. What about making use of other optimisation techniques or domain-specific knowledge? One straightforward approach is based on the recognition that GAs are good for generalised search, but having located optimal or near-optimal regions in the search space, can find it difficult to locate the optimal point itself. Calculus, greedy algorithms and problem-specific heuristic search can, however, be very good at local search. By combining GAs and local search algorithms, the overall hybrid algorithm can outperform its individual components—"the genetic algorithm finds the hills and the hill-climber goes and climbs them" [2].

There are two main approaches. Either run the GA for a reasonable period and then run the local search algorithm on the best (or better few) individuals (Fig. 2.6(a)), or incorporate the local search into the overall algorithm (Fig. 2.6(b)). In the latter case, the local search can itself be used as an operator (alongside crossover and mutation, say), with the results of the local search re-encoded (if necessary) as an offspring—essentially acting like an enhanced mutation operator (*i.e.* Lamarkian evolution). Alternatively, local search can be applied before function evaluation, but no change made to the string itself (*i.e.* Baldwinian evolution). The fitness of each individual is simply its fitness after the application of local search each time. Because the results of the local searches are only fed back into the population in the form of fitness values, not actual strings, this approach (at the expense of considerable additional processing) helps to reduce the risk of a super individual being found by local search, whose genetic material would then dominate future generations, leading to premature convergence.

Domain-specific knowledge

As discussed above, one way of incorporating problem-specific knowledge into a GA is by, say, adding an enhanced mutation operator which decodes a string to a point in the problem space, applies a local search heuristic, and then re-encodes the result. This approach is a comparatively minor extension to a traditional GA, that nevertheless can bring considerable performance enhancement. However, it is possible to go a good deal further! There is a pragmatic 'school of thought' in GA research [9] that sees GAs solely as a means to the solution of engineering problems. Here, the approach is to adopt a problem-specific encoding, and then develop and employ problem-specific



(a) After the GA (b) Within the GA

Figure 2.6: Incorporating local search

operators that together combine the best existing heuristics for the problem with an overall GA framework—a hybrid algorithm. The motivation for this approach is simple, according to Davis [9]: "Although genetic algorithms using binary representation and single-point crossover and binary mutation are robust algorithms, they are almost never the *best* algorithms to use for any [specific] problem". It is almost always possible to develop a problem-specific GA/heuristic hybrid that will outperform either the heuristic(s) or a traditional GA alone.

Domain-specific encoding

Rather than using a binary string, the encoding chosen is the same as the current algorithm (*i.e.* the best known algorithm used on the problem before the development of the hybrid) [9]. However, if the problem is a new one, then a 'natural' encoding would be used. For example, if the problem is to optimise the ten real-valued parameters that control the overall design of an aeroplane, then an individual in the GA would simply consist of a list of ten real numbers, perhaps constrained to the physical limits of the parameters. By contrast, in a traditional GA, each parameter would be quantised and encoded as a binary string, with an individual consisting of the concatenation of the ten encoded individual parameters. In other problems the contrast might be even more marked, especially where the natural encoding was, say, two-dimensional or some complex pointer-based structure in computer memory.

Domain-specific operators

The method then proceeds to developing domain-specific versions of the traditional crossover and mutation operators [9]. For the aeroplane example, one offspring might be constructed by selecting parameters at random (with equal probability) from either parent; the other offspring would get the complementary set (uniform crossover). Another possibility might be *average crossover*, with one offspring generated by averaging the values of the corresponding parameters in the two parents. Possible mutation operators include simply choosing a single parameter at random and giving it a random value (within established limits); or each of the parameters on an individual (with a certain probability) could be modified by a small random amount—*real-number creep*. Finally, operators based on existing domain- or problem-specific heuristics can be developed and incorporated as enhanced mutation operators, as was described previously for the traditional GA. In terms of the aeroplane example, these might include well-established relationships between, say, wing and tail area, or overall engine power and fuselage length—whatever could be obtained from domain experts and existing design software. However, the choice of encoding and the development of problem-specific operators requires considerable ingenuity. It is an iterative process, involving online experimentation, consultation with domain experts, and experience drawn from the growing body of literature on hybrid algorithms. Clearly, efforts to improve a hybrid algorithm should only continue as long as the fitness of the problem solutions discovered by the hybrid remain less than required. Remember, as Sam said, "'I don't have to outrun the bear. I only have to outrun you.'" [124]

2.3 Evolutionary Telecommunications

This section provides a (nearly) exhaustive survey of evolutionary telecommunications, *i.e.* the application of EAs to the telecommunications problem domain. To the best of the author's knowledge, there has been no prior publication of a thorough survey in this area, although there is a brief prior survey by Proestaki & Sinclair [125]. In addition, two (unpublished) technical reports, by Smith [126] and Smith *et al.* [127], were consulted. A summary of the survey was published at the GECCO'99 Workshop on Evolutionary Telecommunications: Past, Present and Future [8]. It should be noted that the survey was completed in December 1998, and has not been updated subsequently.

2.3.1 Survey Scope

In considering the scope of the survey, it was decided to restrict it to the application of purely evolutionary approaches to problems that are unique (or nearly unique) to telecommunications, such as network design.

As a result, not only other modern heuristic methods, such as simulated annealing (SA) [5] and tabu search (TS) [6, 7], but also other natural algorithms were excluded. An example of the latter is ant colony optimisation (ACO) [128], a search and optimisation procedure inspired by the cooperative problem-solving behaviour of natural ant colonies. This has been applied, amongst other areas, to static routing (load balancing) in circuit-switched telecommunications networks [129] and dynamic routing in packet-switched networks [130].

Limiting the problem domain to those applications unique to telecommunications also removed many papers. Examples of these include areas as diverse as signal processing [131,132], ground-water monitoring network design [133], distributed computing [134, 135] and distributed databases [136].

Furthermore, it was decided to focus exclusively on published papers, rather than include project reports, theses or technical reports. These are not necessarily of the same standard as published material, and where they are, this is best demonstrated by subsequent peer-reviewed publication.

A final limitation is that there are some papers, of which the author is aware, that could not be located in the timescales and with the means available. As a result, these too have been omitted from both the survey and bibliography of the thesis.

In total, 112 papers were judged within the scope of the survey, and these are listed (in alphabetical order of first author surname) in Appendix D, Table D.1. For each paper, as well as its own reference number in the thesis bibliography, those papers it cites and which in turn cite it, from within the survey, are also recorded. In addition, classification of each paper in terms of application area (and problem element), solution representation and genetic operators (including population dynamics) are provided.

2.3.2 Historical Growth

Fig. 2.7 shows the chronological distribution of the papers (see also Table D.2 on p. 242). From the earliest in 1987, there has been a long-term pattern of steady growth, overlaid by the two-year cycle of important conferences for the area, particularly the Intl. Conf. on Genetic Algorithms (ICGA) series. The slightly depressed total in 1998 is believed to be partially due to this latter effect, as well as to the relative difficulty of locating recent conference papers.

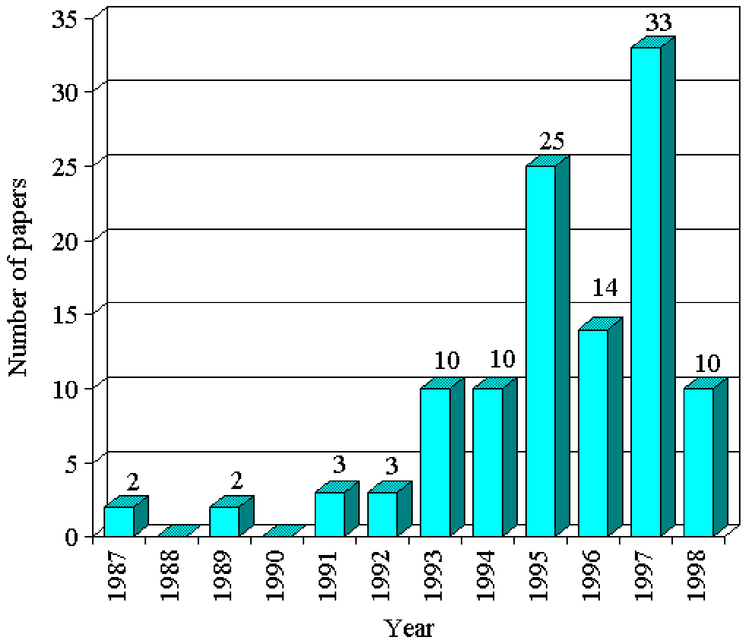


Figure 2.7: Evolutionary telecommunications papers – historical growth

2.3.3 Early Papers

The ten earliest papers in evolutionary telecommunications addressed a variety of application areas and employed a wide range of representations and operators. Apart from citing their own work, none of them cites other evolutionary telecommunications papers (in common with many later authors), and so each represents an independent entry into the area. While some have been cited by several subsequent papers [137–141], others have received comparatively little attention [142–144] and some have not been rediscovered at all [145, 146].

In their three papers from 1987 and 1989, Davis & Coombs [137–139] selected link capacities for wide-area packet-switched networks using a hybrid GA with both an integer representation and several problem-specific operators. They also applied efficient approaches to incorporate their numerous design constraints. Then in 1992, Potter *et al.* [142] combined both a hybrid GA and a bit-string GA for the design of military networks, including network element selection, dimensioning and battlefield location. Michalewicz [143] evolved minimum-spanning-tree topologies for computer networks in 1991 using a two-dimensional binary adjacency matrix representation and problem-specific versions of mutation and crossover. The next year, Kumar *et al.* [145] tackled three constrained computer network topology problems, aiming for maximum reliability, minimum network diameter or minimum average hop count. Their GA was bit-string, but used problem-specific crossover, as well as a repair operator to correct for redundancy in their network representation.

Also in 1992, Sobotka [146] augmented a survivable military communications network with a fixed number of satellite links selected by GA. He employed elitism, problem-specific recombination and fitness based on network damage assessment. For the Steiner tree problem, Hesser *et al.* [144] used a simple bit-string GA in 1989 for arbitrary Steiner point locations, although applying several heuristics during decoding. For network routing, Cox *et al.* [140] used a permutation-based GA in 1991 for the bandwidth-packing component of an heuristic for dynamic-anticipatory routing in circuit-switched networks. Also that same year, Pan & Wang [141] employed a GA for minimum-delay bandwidth allocation, *i.e.* link capacity assignment and routing, in an ATM network. Their GA encodes individuals as a string of indices into a combined look-up table of both link capacities and routes.

2.3.4 Applications

Fig. 2.8 illustrates the classification of papers into application groups (see also Table D.3 on p. 243). While papers have usually only been included in one overall application area (*e.g.* Computer networks), they have been assigned to as many problem elements as necessary (*e.g.* Dimensioning, Node location, *etc.*). Although there is reasonable overall coverage, with every feasible 'cell' containing some papers, some applications have resulted in more publications than others. For example, while there are many similarities between wavelength allocation in optical networks and frequency assignment in radio networks, the latter has almost four times the number of papers. Such imbalances may well indicate areas with potential for further research.

Dimensioning

Fully 33 papers consider dimensioning of telecommunications networks, although usually alongside another aspect of network design, such as node location, topology or routing. (The discussion of many of these [34, 147–168] has therefore been deferred to other sections below.) Papers addressing dimensioning are amongst some of the earliest evolutionary telecommunications papers, including Davis & Coombs [137–139], Potter *et al.* [142] and Pan & Wang [141] (see §2.3.3 above).

Building on their earlier work Potter *et al.*, in three further papers [169–171], describe improvements to their battlefield communications network configuration work including efficiency gains, enhancements of the objective function(s) and the incorporation of additional heuristics into the hybrid GA. Davis *et al.* [172] apply a hybrid GA to obtain both link capacities and traffic routes in a network subject to link failures; their aim is minimum cost within a reliability constraint. Their

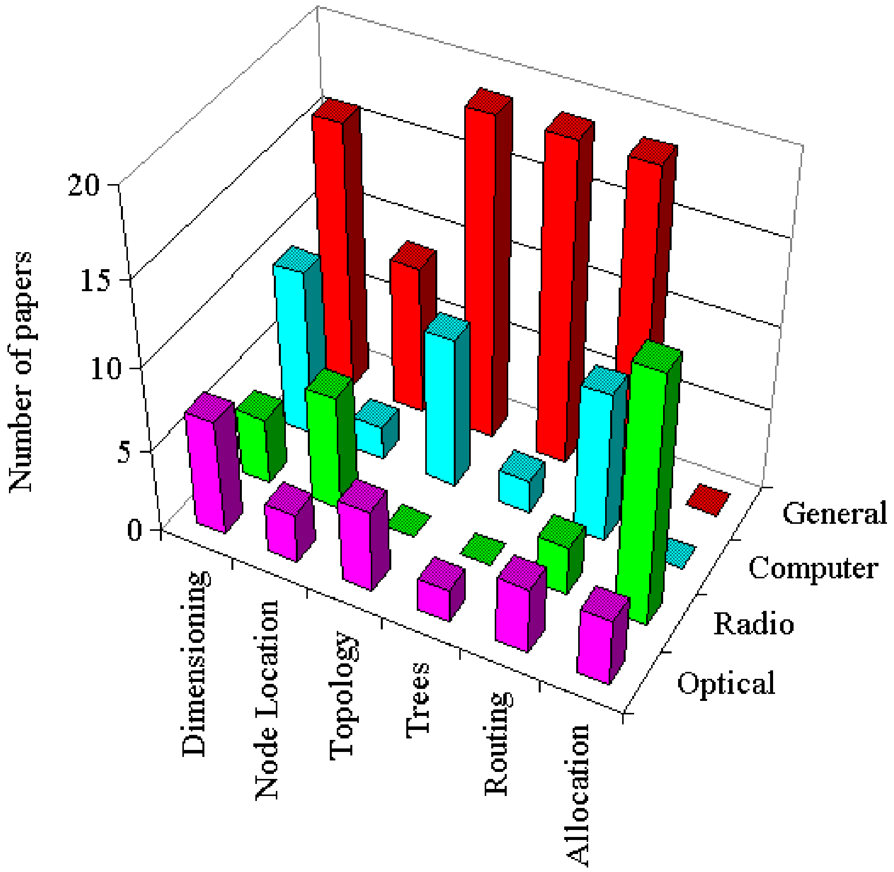


Figure 2.8: Evolutionary telecommunications papers – application groups

GA uses a combination of integer and order-based representations, problem-specific operators and a local greedy repair strategy. More recently, Garcia *et al.* [173] address the multi-period capacity expansion problem *i.e.* when and how much to increase network link sizes to meet forecast demand. They employ a steady-state GA with a hybrid integer-real representation combined with a variety of local search techniques.

Node Location

Network node location is tackled by 17 papers. Of these, a few also address topology design, and are discussed in the next section [161,162,174]. In addition, the work of Potter *et al.* [142,169–171] on battlefield location has already been discussed.

In a 1994 paper, Routen [175] considered the concentrator assignment, multipoint line topology[4] (*i.e.* constrained minimum spanning tree) and concentrator location problems in local access network design. His GAs use integer representation and some problem-specific operators. As well as reviewing earlier work on Steiner trees [176] and frequency assignment [177], Chardaire *et al.* [178] also tackled the concentrator location problem. They combined a bit-string GA for concentrator choice with a greedy heuristic for terminal assignment. Celli *et al.* [179] selected suitable multiplexor sites for metropolitan area networks (MANs) using a parallel GA with bit-string representation. Kumar *et al.* [180] developed their earlier topology work [145] (restated in [181]) to consider computer network expansion. Their GA both selects the new nodes and also determines their links to the existing network. To illustrate their distributed GA, Srivastava *et al.* [182] revisited Kumar *et al.*'s earlier work on both network topology design [145] and network expansion [180]. More recently, Webb *et al.* [183, 184] employed a bit-string GA with heuristic repair for backbone node selection in a ring-star transport network. They contrasted topologies resulting from both distance- and tariff-based cost models.

For radio networks, Gondim [185] has applied GAs to the location area problem in GSM cellular networks *i.e.* clustering cells around mobile switching centres. They used an elitist, integer-based GA with problem-specific recombination. Calégari *et al.* [186] used a multi-population GA to select basestation sites for a cellular network. Their GA, which is generational and employs a bit-string representation, clearly out-performed a uni-population variant. Finally, Tang *et al.* [187] applied a multi-objective GA to obtain both the number and location of basestations in a wireless local area network. They combined a hierarchical binary-real representation with appropriate genetic

[4] Denoted 'terminal layout' by Routen [175].

operators and Pareto ranking.

Topology

Network topology design is the focus of some 33 papers. Of these, three approach topology via routing [151–153], and are discussed below. Another three are early papers by Michalewicz [143], Kumar *et al.* [145] and Sobotka [146] (see §2.3.3 above). In addition, papers by Routen [175], Kumar *et al.* [180–182] and the more recent work of Webb *et al.* [183, 184] have already been considered in the context of node location.

For general networks, in 1997 Deeter & Smith [148] addressed minimum-cost topology design for small network sizes subject to an all-terminal reliability constraint. They determined not only which links to include, but also the level of reliability (and hence cost) associated with each. Their GA uses an integer representation, uniform crossover, mutation and rank-based selection. Dengiz *et al.*'s slightly earlier paper [188] also focused on minimum-cost topology design subject to a reliability constraint. In contrast, here all links shared a common reliability and their costs, while differing, were fixed at the outset. Consequently, only a bit-string GA was required, and they were also able to tackle far larger network designs by employing an approximate fitness function. Dengiz *et al.* developed their work in four further papers. In the first two [189, 190], they refined their estimate of fitness to give increasingly accurate assessment to the fitter individuals. As a result, they were able to attempt even larger networks. Then in [191, 192], while still retaining their focused fitness calculation, they obtained superior results by adopting a hybrid GA. This employs a variable-length integer representation, ranked selection, local-search mutation and repair of individuals with poor connectivity. Furthermore, although their major focus has been network trees, in 1994 Abuali *et al.* [193] tackled minimum-cost terminal assignment using a GAs with two different permutation encodings combined with different initialisation approaches, permutation crossovers and both fixed and adaptive mutation probabilities. The same authors [194] later applied a bit-string GA to the subset interconnection problem. This is a constrained minimum-cost network topology design problem, whereby every one of the given node subsets also has to be internally connected, as well as the network as a whole. Again they applied a variety of operators, demonstrating the superiority of their GA over problem-specific heuristics. Tanaka & Berlage [167] tackled the design of video-on-demand distribution networks, determining not only the network topology, but also which nodes would provide video storage. Their GA uses a binary matrix representation, as well as problem-specific crossover and repair. Meanwhile, Saha & Chakraborty [195] employed a bit-string GA to address the link enhancement problem *i.e.* adding new links to an existing network topology (*cf.* Kumar *et al.* [180]). However, their approach is somewhat simplistic, as they assume not only the cost, but also the profit resulting from the addition of any link is known in advance. Nakamura & Oda [196] also tackled link enhancement, but over multiple planning periods, however their focus was limited to installation costs. They combined an heuristic routing and restoration algorithm with a simple GA to select from a range of possible additional links.

For optical networks, Sinclair has worked on multi-wavelength all-optical transport networks, applying a simple bit-string GA [166], a hybrid GA [34] and, with Aiyarak & Saket, three different GP approaches [147] to mesh network design. While Sinclair *et al.*'s EAs only select which links to include in the network topology, routing and link dimensioning are accomplished by their fitness function, which also includes a connectivity constraint. Mikac & Inkret [157] developed Sinclair *et al.*'s work with bit-string GAs [166] to consider both minimum cost and minimum unavailability. Their algorithm employs two fitness functions: in odd generations it focuses on minimum cost, applying an unavailability penalty; whereas in even generations unavailability is the focus, with an associated cost penalty. In contrast, both Paul *et al.* [161, 162] and Brittain *et al.* [174] address constrained minimum-cost passive optical network (PON) topology design for local access[5]. However, while problem-specific representations are used by both, as well as problem-specific genetic operators by Paul *et al.*, only Brittain *et al.* provide full details of their algorithm. This employs a two-part chromosome comprising a bit-string and a permutation, with each part manipulated using appropriate standard genetic operators.

For computer networks, Hewitt *et al.* [150] studied minimum-cost mesh network design subject to both connectivity and packet-delay constraints. They applied a hybrid GA with a linear link-data-structure representation and problem-specific crossover and mutation. Later, a three-stage GA for minimum-cost design of a biconnected computer network including topology, routing and capacity assignment was described by Ko *et al.* [154]. Their GAs are tailored to each stage, employing bit-string, integer and hybrid representations, as well as suitable operators, respectively. However, Pierre & Legault employed a bit-string GA for computer mesh network topology design under connectivity and delay constraints [163,164]. Although their GA only directly selected network links, the fitness function included link capacity allocation (*cf.* Sinclair [166]). Finally, Qin *et al.* [165] also used a bit-string GA, this time for ATM mesh network topology design. Their algorithm considers

[5] Strictly, Paul *et al.* consider only *tree* topologies, but [161, 162] are discussed here alongside the other optical network papers.

capacity assignment in the fitness assessment, but makes no explicit calculation of network delay, and applies a biconnectivity constraint only by repair.

Trees

Network trees are the focus of 24 papers. Of these, the early paper by Hesser *et al.* [144] on Steiner trees, as well as the more recent work by Routen [175] on multipoint line topology design, and by Paul *et al.* [161, 162] on PON topology design, have already been discussed. In addition, papers on multipoint routing by Christensen *et al.* [197] and Zhu *et al.* [198] have been deferred to a later section.

While papers in this area address a number of aspects, nevertheless, it would appear that the central issue in successful application of EAs to network tree design is the encoding of the trees themselves.

Several authors have researched minimum Steiner trees. In contrast to Hesser *et al.* [144], who tackled arbitrary Steiner point locations, Kapsalis *et al.* [176, 178] used a bit-string GA to focus on the more limited problem of selecting Steiner nodes from a given graph. Julstrom [199], however, tackled the minimum rectilinear Steiner tree problem with a hybrid integer-binary encoding and problem-specific crossover. His encoding exploited the constraints on Steiner point locations in the rectilinear problem, although empirical performance of the GA was poor compared to problem-specific heuristics. Subsequently, Esbensen & Mazumder returned to the selection of Steiner nodes for an arbitrary given graph [200–202]. Their hybrid GA [200] uses a modified bit-string encoding (allowing inversion) and repair to bias a simple problem-specific heuristic. In a subsequent paper, Esbensen [201] applied their GA to a wide range of large test networks, demonstrating its superiority over both Kapsalis *et al.* [176] and several powerful problem-specific heuristics. Then in [202], Esbensen improved the GA by adopting a steady-state population dynamic and bit-string encoding (hence no inversion), employing uniform crossover and including both adaptation of the mutation rate and local search.

Considering trees more generally, Abuali *et al.* have published some five papers [203–207] in addition to the two on network topology design already reviewed. In the first [203] in 1993, they considered multipoint line topology design *i.e.* the constrained minimum spanning tree problem, investigating eight different GAs. These were formed from combinations of two encodings (integer and permutation), with two different crossover operators for each encoding, and both steady-state and generational approaches. Their permutation-encoded GAs performed better than the well-known Esau-Williams heuristic [10, 11], particularly for larger networks. Others applying GAs to this problem (independently of Abuali *et al.*) include Routen [175] in 1994, who provided only a summary of his disappointing results using an integer-based encoding, and Zhao *et al.* [208] with a more successful hybrid GA in 1995. This used an object-oriented representation, as well as both problem-specific recombination and mutation incorporating repair. Once again, they were often able to obtain better results than either the Prim or Esau-Williams heuristics [10,11]. Meanwhile, in 1994 Abuali *et al.* [204] moved to consider the probabilistic minimum spanning tree problem *i.e.* finding the minimum-expected-cost spanning tree in a network where each subset of nodes only require to be connected with a certain probability. They adopted a Prüfer number encoding and tested a wide variety of operators. However, that same year Palmer & Kershenbaum [159, 160] argued against Prüfer encoding (as well as predecessor-vector encoding), instead using a GA to select node-cost biases for a standard minimum spanning tree heuristic in a capacitated network (*cf.* Esbensen & Mazumder [200] above). They obtained results superior to those of a problem-specific heuristic, particularly for more unusual test problems. Abuali *et al.* [205] also later abandoned Prüfer encoding, instead comparing two graph-theoretical integer encodings, determinant factorisation and cycle basis, for the probabilistic minimum spanning tree problem. In a subsequent paper [207], Abuali *et al.* tested their determinant-factorisation encoding against both Prüfer encoding and Palmer & Kershenbaum's node-cost bias encoding [159, 160]. Overall, both determinant factorisation and node-cost bias encoding performed well, although the latter appears superior for larger networks, as well as having the important advantage of conceptual simplicity. A further paper by Abuali *et al.* [206] addressed the more constrained problem of three-star trees, where all tree branches consist of a chain of three nodes attached to the root. Once again, their determinant encoding performed well, although an alternative permutation-based encoding used to bias a greedy heuristic appears to be superior. The minimum spanning tree problem in a capacitated network was tackled independently by Berry *et al.* [209] in 1995. They adopted the predecessor-vector encoding deprecated by Palmer & Kershenbaum [159], although avoiding its limitations through heuristic recombination and mutation. However, their paper offers little experimental evidence in support of their GA. Also that year, Walters & Smith [168] independently developed a hybrid GA for (directed) minimum spanning trees in capacitated networks using a problem-specific integer encoding combined with heuristic recombination and mutation (*cf.* Dengiz *et al.* [191, 192]). Again, it is unclear how their approach would compare to those of other researchers. Finally, Elbaum & Sidi [210, 211] considered the topological design of a local area network (LAN) for minimum average network delay. As the

LAN is divided into clusters interconnected by transparent bridges, the network topology is limited to a spanning tree. Individuals are encoded using a hybrid three-part chromosome to define the network tree (based on a 'virtual Huffman tree'), match clusters to tree nodes, and assign users to clusters.

Routing and Restoration

Routing and restoration is tackled by 32 papers. Two of these are early papers by Cox *et al.* [140] and Pan & Wang [141] (see §2.3.3 above). Two more, by Davis *et al.* [172] and Ko *et al.* [154], have already been discussed in the contexts of dimensioning and topology, respectively. In addition, three papers on optical routing and wavelength allocation, by Tan & Sinclair [29] and Sinclair [35, 36], are discussed in a later section.

As well as Davis *et al.* [172], other routing papers from 1993 were by Munakata & Hashier [158], Sinclair [27] and Shimamoto *et al.* [212]. To address the maximum flow problem in a directed capacitated graph, Munakata & Hashier [158] combined an integer representation of the flow matrix with both a problem-specific recombination and an adaptive problem-specific mutation. While their hybrid GA obtained good results, its computational complexity is relatively poor compared with conventional heuristic algorithms. Meanwhile, to design static-routing tables for minimum network grade-of-service (NGOS) in an unreliable circuit-switched network, Sinclair [27] used a compact bit-string encoding of the routing tables with conventional operators, but employed a penalty function to eliminate infinite-routing loops. As with Munakata & Hashier [158], his algorithm also proved uncompetitive with an earlier heuristic. In contrast, Shimamoto *et al.* [212] tackled dynamic routing in reliable circuit-switched networks. An individual in their algorithm was represented as a string of indices into a look-up table of the k-shortest paths for each node pair (*cf.* Pan & Wang [141]). They used a form of steady-state GA to exploit the relatively slow changes in traffic distribution, and obtained grade-of-service that compared well with a fixed routing algorithm (although in both cases only a single route was available between node pairs).

In 1995, Mann *et al.* [213] addressed static routing, using a representation very similar to that of Shimamoto *et al.* [212], but encoded as bits rather than integers. Their aim was to minimise network cost while also balancing the traffic load on the network links. Further detail of their work was given in [214]; they made a detailed comparison with a variety of algorithms, both GA and SA, obtaining similar results from the best variant of each, although with much less computational effort by SA. A subsequent study by Mann & Smith [156] focused on routing in ring networks, aiming for balanced load and hence a minimal ring capacity requirement for rerouting in the event of failure. Their representation consists of one bit per node pair, with each bit simply indicating whether the traffic is routed clockwise or anti-clockwise. They again applied a wide range of algorithms, both GA and SA, this time finding GA to be more robust to algorithm and parameter choice.

Two papers by Hoelting *et al.* [215,216] in 1996 addressed variations on more standard routing problems[6]. Their first study [215] considered finding an investigator tour for a network, *i.e.* the route taken by a network management agent attempting to locate a failed link by travelling to and interrogating an appropriate subset of the network nodes. Their permutation-based GA was shown to obtain superior tours to an heuristic approach. In the second study [216], Hoelting *et al.* tackled the minimum broadcast time problem, *i.e.* where a message must be distributed to all the nodes in a network in a minimum number of time steps, but at any one time step each node can only transmit the message to one of its neighbours. They applied a permutation-based GA, and once again showed it to be superior to a problem-specific heuristic. Almost the same group of authors went on to tackle the point-to-multipoint routing problem, *i.e.* routing a set of circuit-switched connection requests, each of which has a single source and one or more destinations [197,198]. First Christensen *et al.* [197] extended the earlier work of Cox *et al.* [140] to provide dynamic-anticipatory routing for multiple point-to-multipoint connections. They combined a permutation-based GA to order the requests and an heuristic Steiner tree algorithm to find their minimum cost routing trees. Subsequently, Zhu *et al.* [198] extended their work to also consider subsets of the outstanding requests where routing the full set is infeasible.

An unusual variation on applying GAs for routing was considered by Servet *et al.* [217], who used both binary and real-valued GAs with a variety of fitness functions to invert a single-moment traffic model of a network and so estimate the size of traffic streams starting from only the offered (or carried) traffic on the links.

Several authors in recent years have applied EC to network restoration rather than just primary routing. In 1996, Huang *et al.* [151] evolved two minimum-cost node-disjoint routes between one or more node pairs in a communications network, and used these to guide both topology selection and link dimensioning. Their algorithm used a node-index representation with heuristic recombination and mutation. In two later papers [152,153], they explored a variety of parallel computing approaches to enhance the speed of their GA. In 1997, Bentall *et al.* [218] used a GA with a two-

[6]These two papers share common authors, Schoenfeld and Wainwright, with the papers by Abuali *et al.* on network trees [203–207] and network topology [193,194].

dimensional order-based representation to obtain restoration paths for a heavily-loaded network. Their aim was not so much to apply the GA to real-time restoration, but rather for the results obtained to serve as a benchmark in assessing rapid heuristic algorithms with only local knowledge. Kozdrowski *et al.* [155] used a hybrid GA to allocate traffic streams and link capacities in a network subject to both link failures and a number of possible constraints on restoration paths. Meanwhile, Kirkwood *et al.* [219, 220] applied GP to obtain both primary and restoration paths in a network subject to single-link failures. They evolved programs that determined near-optimal routes from one node to all others under all failure conditions. However, their approach does not scale well to larger networks.

Focusing on computer networks, in two 1995 papers, Carse *et al.* [221, 222] evolved a network routing agent, employing a Pittsburgh-style fuzzy classifier system, for adaptive distributed routing in a datagram packet network. They applied their algorithm to a small network, and obtained reduced mean packet delay, for realistic measurement intervals, compared to a lowest-delay routing algorithm. Then in 1997, Munetomo *et al.* [223] tackled much the same problem, real-time dynamic source routing of packets, but in far larger networks, by employing path-based genetic operators to directly manipulate alternative routes in the node routing tables. In comparison with two established Internet routing algorithms, their scalable approach demonstrated both considerably reduced delay and more balanced link loads, particularly under heavy traffic conditions. In contrast, He [149] addressed static route and link capacity allocation in large packet-switched networks. His GA encodes individuals as a string of indices into look-up tables of link capacities and routes, independently replicating the representation used much earlier by Pan & Wang [141]. Also in 1997, Tanterdtid *et al.* [224, 225] applied Shimamoto *et al.*'s GA for dynamic routing in reliable circuit-switched networks [212] to the related problem of virtual-path routing in ATM networks. Finally, Markaki *et al.* [226] developed a hybrid GA with a two-dimensional binary encoding and local repair for connection-oriented channel selection and reuse in a DQDB MAN. Their GA allowed many more active connections, and obtained higher channel reuse, than an earlier graph-colouring based approach.

Call Admission

A series of three papers by Yener & Rose [227–229] are the only ones to address call admission using EAs, in their case for cellular radio networks. In 1994 [227] they evolved admission policies for a small network using a bit-string GA. The admission policy decides which classes of call to admit to each cell in each network state. Their GA compared well to the 'admit if possible' heuristic, and employed a random codeword based bit-string compression technique to cope with the large bit strings that would otherwise be required. In their two subsequent papers [228,229] they considered admission policies based on local rather than global state information, and were thus able to tackle larger networks. Interestingly, examination of the structure of the evolved policies themselves suggested a novel heuristic local admission policy to Yener & Rose.

Wavelength Allocation

Only four papers consider wavelength allocation in optical networks. The earliest, by Tan & Sinclair [29] in 1995, describes several GA-based solution approaches for multi-wavelength all-optical transport networks. The most successful of these combines a bit-string GA for route selection with an heuristic for wavelength and fibre allocation. Subsequently, Sinclair [35, 36] developed a hybrid GA for routing, fibre choice and wavelength allocation using both an object-oriented representation and problem-specific operators. He applied it first to minimise the number of different wavelengths required [35], and then to minimise a simple model of network cost [36]. In contrast, Abed & Ghanta [230] applied a bit-string GA to design the logical topology of a local area network (LAN) or MAN using wavelength allocation on a single optical fibre.

Frequency Assignment

Frequency assignment in radio networks is the topic of 15 papers. Of these, seven have included Hurley as co-author: with Crompton & Stephens [231–233], with Smith [234] and Thiel & Smith [235], and most recently, with first Valenzuela & Jones [236] and then Valenzuela & Smith [237]. In the first of these, and also the earliest paper in the area, Crompton *et al.* in 1993 [231] applied a multi-population GA to frequency assignment for the air-ground-air problem using an integer representation. Their aim was to minimise the number of constraint violations and hence interference. Then, for the same problem, they reported improved results in [232] with a transmitter-set representation combined with heuristic recombination. Subsequently, in [233] they demonstrated the superiority of their GA to a backtracking heuristic even on an equal execution time basis. However, Hurley & Smith [234] demonstrated an SA [5] approach to be superior to the same GA, primarily because their SA was able to use an incremental fitness function, and hence perform up to 1,000 times more evaluations in the same execution time. Nevertheless, despite sharing the advantage of incremental fitness evaluation, TS [6, 7] was found to be inferior to SA by Hurley *et al.* [235]

(although still better than their GA). Moving to a slightly different problem in 1997, Valenzuela *et al.* [236] aimed to minimise the span (*i.e.* range) of frequencies required to satisfy all constraints, resulting in zero interference, in both regular and irregular cellular networks. Their steady-state GA biases a greedy assignment heuristic[7] by providing an initial ordering of transmitters. In addition, they subsequently demonstrated [237] the superiority of their GA over earlier SA and TS approaches to the same problem (although *cf.* Hurley *et al.* [235] above).

Another early paper on frequency assignment was in 1994 by Cuppini [238], representing an independent entry into the area. However, her bit-string algorithm uses no crossover, although very high rates of mutation, and is only applied to small problems. The following year, Kapsalis *et al.* [177, 178] used a bit-string GA for realistic problem sizes, testing a wide range of heuristic operators and adaptive fitness functions (*cf.* Crompton *et al.* [232]). Their aim was to minimise both interference and the number or span of frequencies. Subsequently, Kapsalis & Smith [239] applied a meta-GA to select from an even wider range of operators, as well as to tune the parameters of their primary GA. Although improved results were obtained, they were discouraged by the extensive execution times required. More recently, Crisan & Mühlenbein [240] applied a breeder GA to digital cellular networks. Their frequency assignments are represented as integers and several heuristic operators employed in evolving a minimum-interference solution. Their problem formulation is more complex than earlier authors (*e.g.* Crompton *et al.* [232] and Kapsalis *et al.* [177]), including not only hard constraints, but also interference probabilities used to define an 'interference cost'. Also in 1998, Ngo & Li [241] used a compact bit-string representation and applied problem-specific operators and local search to minimise interference and hence obtain a feasible assignment (although this is a more limited objective than other recent authors).

In addition, by incorporating a detailed propagation model and fuzzy logic in fitness assessment, Kaminsky [242] was able to tackle the hourly assignment of frequencies to military radio networks using a steady-state GA (*cf.* Potter *et al.* [142, 169–171]). In contract, Sandalidis *et al.* [243] addresses dynamic frequency assignment for cellular mobile radio. They applied a (1,λ) ES with a bit-string representation and mutation based on channel swapping (thereby avoiding the need to repair individuals following GA crossover). Their results were superior to four problem-specific heuristics and a neural-net based approach, although the question of real-time operation of their algorithm was not addressed.

[7] *cf.* Esbensen & Mazumder [200–202] and Palmer & Kershenbaum [159, 160] above

2.3.5 Algorithms

Clearly, evolutionary telecommunications has been addressed using a wide-variety of algorithms (see Table D.1). Solution representations surveyed included binary, integer, real, permutation and tree-based (GP) encodings, as well as a few hybrid approaches, each modifying or combining more straightforward encodings, or in some cases employing object-oriented models. Population dynamics, although dominated by generational and elitist approaches, also included some steady-state, multi-population and breeder algorithms. Authors employed a variety of selection methods, and although favouring the random methods of fitness-proportional, tournament or rank-based selection, some ES-type deterministic selection was also used. Not surprisingly, many different genetic operators were applied, including single- and multi-point crossover, uniform crossover, permutation crossover, heuristic recombination in hybrid algorithms, tree crossover (GP), a wide range of mutation operators, inversion, repair and local search.

Overall, a few observations can be made. Firstly, biasing a greedy heuristic, either by using a GA to select its starting point, or to modify the local fitness landscape seen by the heuristic, seems a promising direction for future work (see, *e.g.* [159, 160, 200–202, 236, 237]). Secondly, while there are arguments for developing EAs elsewhere as general problem-solving tools, for the complex and demanding problems addressed by evolutionary telecommunications, incorporating domain knowledge seems vital, rather than just using, say, a simple GA (see §2.2.3). However, far too few of the researchers surveyed seem to be investigating and employing hybrid GAs. Next, there must be more comparative work. Many, although not all, of the papers examined describe a single EA for the solution of the problem presented, without apparently investigating any alternatives. Authors may well be going through an algorithm development and parameter tuning process, but this is not being reported. Instead, their 'final' algorithm is simply presented as a *fait accomplie*. In addition, there are often no comparisons with other non-EA algorithms. Now, while there are potential difficulties with such comparisons, including only benchmarking against a 'straw man' (*e.g.* random search, or a poorly-implemented modern heuristic search technique), nevertheless, too few authors are making the attempt. Finally, two further problems remain: citations and reproducibility.

Citations & Reproducibility

It is apparent from Table D.1 that papers in evolutionary telecommunications rarely cite earlier work in the field. Indeed, if citations of each author's own prior papers are discounted, remarkably few remain. Examination of the individual papers reveals that many are 'first entries' into the evolutionary telecommunications domain. A typical paper will thus cite earlier research on the specific problem tackled, make reference to one or more books on EAs (such as those by Goldberg [2] or Davis [9]), before describing the particular EA used and reporting results. Little reference, if any, is made to any earlier work in applying EAs to this or similar problems. Clearly, given that there is now more than ten years of research reported in over 100 papers, such an approach fails to develop on the experience of previous researchers. However, this is not an uncommon omission in EC in general, as David Fogel has observed [4]: "Evolutionary computation has a long history. Unfortunately, this history is virtually unknown to many practitioners in the field."

In addition, many of the papers in the survey do not present sufficient information to allow the algorithm to be reproduced. Authors need to include detailed, unambiguous information on solution representation, population dynamics, selection methods, genetic operators and the parameters used. Without these details, it is very difficult to compare one EA with another.

Thus, as well as finding earlier work to cite and compare current work with, evolutionary telecommunications authors must try and ensure the reproducibility of their own work, so that others can cite and compare their work with it in turn. Indeed, it is only as researchers begin to build on previous work in these and other ways, that evolutionary telecommunications will begin to mature as a field.

2.4 Applying Genetic-algorithm/Heuristic Hybrids

To complement the more comprehensive survey of evolutionary telecommunications above, in this section a brief overview is given of some prior work from other areas using GA/heuristic hybrid algorithms, the solution approach adopted in the thesis, and the basics of which were explained in §2.2.3 above.

In 1989, Jog *et al.* [244] tackled the travelling salesman problem (TSP) using a generational GA/heuristic hybrid algorithm, and building on a number of even earlier papers in this area. Their algorithm used the usual permutation representation, but employed both an heuristic crossover operator and two local-search mutation operators applied with fixed operator probabilities. Their experiments focused on the effects of including or excluding operators from the hybrid, as well as varying the population size. They concluded that larger population sizes, using problem-specific crossover and including local-search heuristics of increasing sophistication were all beneficial to the final solution quality.

An early and influential reference in this area, as already noted, is the *Handbook of Genetic Algorithms* edited by Lawrence Davis in 1991 [9]. After a step-by-step development of a GA/heuristic hybrid from a simple GA, the book includes a wide-ranging collection of application case studies of (mainly) hybrid GAs. These include the parametric design of aircraft [245], dynamic anticipatory routing in circuit-switched telecommunications networks [140], robot arm trajectory generation [246], international security modelling [247], aeroplane control strategy design [248], neural network design [249], industrial plant control [250], multiple-fault diagnosis [251], conformational analysis of DNA [252], interpretation of passive sonar traces [253], engineering design optimisation [254], scheduling [255] and the TSP [256]. Between them, the case studies employ a wide variety of representations and operators.

For example, the chapter by Bramlette & Bouchard [245] from [9] concerns the parametric design of a modern fighter aircraft. Their design required eight parameters to be optimised, including fuselage width, height and length, wing loading and engine size. They investigated not only an integer-coded generational GA, but also a wide variety of stochastic approaches including various types of SA, hill climbing and random search, used both individually and in combination. Overall, they found that iterated SA, and a hybrid GA using stochastic hill climbing as the mutation operator, obtained the best results on their test problem.

Also in [9], Davidor [246] developed trajectories for robot arms using a hybrid GA which represents the trajectories as order-dependent variable-length strings of (intermediate) arm positions, with each position specified by the inter-link angles in the arm configuration. A problem-specific two-point crossover is introduced which chooses two random arm configurations as crossover points in one parent (*cf.* Munetomo *et al.* [223]). It then locates the two most similar (*i.e.* analogous) configurations in the other parent. Then the configuration sequences between these points (and including the second crossover point) are exchanged between the parents to form the children. Mutation is guided by problem information to focus on those points in an arm trajectory that are least fit. In addition, two further problem-specific mutations are used that can add (duplicate) and delete arm configurations from the trajectories. In experiments on one particular target trajectory, Davidor's hybrid GA performed well in comparison with both random search and hill climbing, and was found to be further improved by directly incorporating hill climbing within the GA.

As a third example from [9], Grefenstette [248] sought to develop strategies for sequential decision tasks in a dynamic system, specifically evasive maneuvers allowing a plane to avoid a pursuing missile. Within his generational GA, each strategy is represented by a set of condition-action rules that relate the sensor inputs (missile range, missile bearing, missile speed, *etc.*) and the system response (*i.e.* the turning rate of the plane). The problem-specific crossover operator exchanges complete rules between individuals, whilst trying not to separate successful rule sequences. Of the two mutation operators provided, one specialises existing rules, and the other randomly modifies an individual clause within a rule. In addition, the GA can be seeded with rules generated by domain experts that are combined with randomly-generated rules within the individual strategies that form the initial population. In experimental runs, the GA performed well providing a high probability of missile evasion, although no direct comparison was made by Grefenstette with other non-trivial approaches.

More recently, in their 1997 paper, Schnecke & Vornberger [257] address constrained placement problems—both the facility layout problem and VLSI macro-cell layout generation—using a hybrid GA. Their steady-state algorithm uses a binary slicing-tree representation to express the relative placement of the blocks, three problem-specific mutations, and a form of gene-pool recombination. It employs a multi-population approach, with each sub-population using parameter adaptation, including the operator probabilities. Their hybrid GA produced competitive results for VLSI macro-cell layout generation, without the use of sophisticated routing algorithms, and obtained the best results yet reported on large facility layout problems.

A hybrid GA for computer-aided process planning for job shop machining, considering operation selection, machine selection, setup selection, cutting tool selection, and operations sequencing simultaneously, was presented by Zhang *et al.* in [258]. Their representation was a ordered list of operations, each one providing details of the corresponding machine, tool and tool-approach direction choices. A problem-specific form of cyclic crossover and three mutation operators for machines, tools and tool-approach direction were applied using fixed operator probabilities. For the one complex problem examined, Zhang *et al.* found that their hybrid GA was superior to an heuristic approach.

An adaptive evolutionary planner/navigator for mobile robots was described by Xiao *et al.* [259] in 1997. Their system uses the same GA/heuristic hybrid algorithm for both offline planning and online navigation. The steady-state hybrid represents paths as a linked list of knot points (*i.e.* intersection points) between straight line segments. It uses a problem-specific crossover and seven mutation operators, with their probabilities adapted during the course of a run (see below). Their system has been developed over a period of time, and demonstrates robust online and offline performance over a wide range of challenging environments. Their work was subsequently adapted by Śmierzchalski & Michalewicz [260] for modelling ship trajectories in collision situations. The path representation was extended to include ship speeds on the line segments, the concept of time introduced to allow for moving obstacles, due allowance made for dynamic as well as static constraints, and a new speed mutation operator added. However, apart from these, the earlier planner/navigator was largely adopted unchanged, except that for simplicity Śmierzchalski & Michalewicz omitted operator-probability adaptation, preferring instead to use equal fixed operator probabilities.

In 2000, Magyar *et al.* [261] described a GA/heuristic hybrid for the three-matching problem. This problem requires a set of nodes to be clustered in disjoint groups of three (*i.e. triplets*), with the cost of each triplet simply that of the minimum-spanning tree required to interconnect its nodes, and the overall cost taken to be the sum of the triplet costs. Magyar *et al.*'s hybrid uses a permutation representation and incorporates several problem-specific recombination and local-improvement operators. It employs operator-probability adaptation (see below), and the results obtained were better than those of a local-search heuristic, SA and a grouping GA over twenty large problem instances.

Although this section has focused on GA/heuristic hybrids, it should perhaps be mentioned that many authors have hybridised GAs with other modern search heuristics. As one example, a paper in 1999 by Bhandarkar & Zhang [262] addresses the problem of image segmentation using EC. As well as developing a problem-specific generational GA, Bhandarkar & Zhang hybidise this with three different stochastic annealing algorithms in turn: SA, microcanonical annealing (MCA) and the random cost algorithm (RCA). In the GA, a two-dimensional integer representation is used, combined with problem-specific crossover, mutation and a local-search operator. All three operators follow deterministic changes in their operator probabilities during a run, with local search and mutation increasing, and crossover decreasing. The hybridisation is achieved by wrapping a stochastic annealing replacement mechanism around the GA framework (excluding the local search operator). Thus children that had been created using GA crossover and mutation only replace their parents if they satisfy the appropriate annealing condition. Overall, they found that the three hybrids all produced higher quality results than the GA, particularly so for the SA-GA combination.

Overall, as in evolutionary telecommunications, the GA/heuristic hybrid approach has been readily applied to a wide range of problem domains, only a few of which have been considered here. Authors have shown considerable ingenuity in developing problem-specific algorithms for difficult design and optimisation tasks, and the GA/heuristic hybrid approach clearly has considerable promise.

2.5 Operator-probability Adaptation

One aspect of GA/heuristic hybrid algorithms explored in this thesis is their potential use of operator-probability adaptation mechanisms. Consequently, in this section some attempts to classify the many approaches to parameter control in EAs will be mentioned, as well as briefly examining a selection of prior work, taken from areas other than evolutionary telecommunications, employing such mechanisms.

A detailed survey and classification of parameter setting was recently completed by Eiben *et al.* [263]. They based their work in part on their own earlier, although more limited, classification scheme [264]. In [263], they argue that there are only two main criteria for classifying adaptation mechanisms: what is changed, and how the change is made. For the first, they considered parameters controlling representation, evaluation function, mutation operators and their probabilities, crossover operators and their probabilities, parent selection, replacement operator (survivor selection) and the population itself as potential targets for adaptation. As to how the changes are made, they divided *parameter setting* into two main areas: *parameter tuning* and *parameter control*. In the former case, parameters remain fixed throughout the course of the EA run, although they can be tuned between runs by the experimenter. In the latter approach, however, the parameter values change during the run, with this mechanism further divided into three categories of parameter control. In the *deterministic* method, the changing parameter value follows a preset schedule—there is no feedback from the evolutionary process. With *adaptive* control, there is feedback, but this only changes the parameter value using some heuristic, rule-based approach. The final possibility is *self-adaptation*—the parameter values participate in the ongoing evolutionary process; thus the adaptation mechanism is itself evolutionary.

An earlier and more focused classification by Tuson [265, 266], considered only operator-probability adaptation algorithms. In his study, he classifies these into two groups: co-evolutionary methods (equivalent to Eiben *et al.*'s self-adaptive parameter control), that encode operator probabilities into each member of the population; and learning-rule methods (equivalent to adaptive parameter control), that adapt the probabilities based on measurement of the operator productivities.

Turning now from surveys to individual papers employing operator-probability adaptation[8], an early paper by Fogarty [267] in 1989 studied varying mutation probability for an industrial

[8]Here the focus is primarily on the adaptive approach and algorithms from the GA 'tradition', rather than ESs, EP or even GAs which employ self-adaptation to control operator probabilities.

optimisation problem, *i.e.* minimising combustion stackloss in the common flue of multiple burner furnaces. He compared two fixed mutation schemes (one of which applied reduced levels of mutation to more significant bits) with two corresponding deterministic schedules that gradually reduced mutation throughout the run. He found that deterministic variation of the mutation probability improved the GA's online performance in both cases when starting from a conservative initial population.

In the same year, Davis [268] presented an operator-probability adaptation mechanism incorporated into a steady-state GA/heuristic hybrid with an integer representation and five problem-specific operators. The adaptation mechanism was found to provide improved performance over both naive and tuned fixed probabilities, and to be competitive even with a deterministic schedule derived from a substantial number of experimental runs.

Davis subsequently described his approach in more detail in [9]. The operator productivity is computed by crediting operators with any improvement over the best member of the population; a decreasing fraction of this credit is also passed back to the operators responsible for creating the individual's parents, and so on, to a given depth. After a defined interval, the productivity of an operator is taken to be its total credit over the interval divided by the number of individuals it has created. Operator probabilities are then adjusted by reassigning a fraction of their probabilities in proportion to their productivity. Davis also mentions the possibility of preventing an operator's probability falling below an assigned minimum value, as well as the potential of his approach to measure the productivity of individual problem-specific heuristic operators.

Rather than using a credit-assignment approach, in 1994 Srinivas & Patnaik [269] adapted both crossover and mutation probabilities in a generational binary-string GA such that the smaller the gap between the best-in-population fitness and the average fitness, the higher the operator probabilities. However, to preserve the better individuals, the probabilities are also reduced the nearer an individual is to the best-in-population fitness. In particular, the mechanism ensures the best individual is (almost) always preserved unchanged (although a small background level of mutation prevents premature convergence). Their experimental results show that across a range of test functions their adaptive algorithm was in most cases significantly better than a GA with fixed operator probabilities, showing both more rapid convergence and a reduced tendency to get trapped in local optima.

In 1995, Julstrom [270] presented ADOPP, an operator-probability adaptation mechanism for steady-state GAs, in which all operators back to a given depth in an operator tree are credited with a decreasing fraction of any improvement over population median fitness. At all times, actual operator probabilities (and not just a fractional part) are determined by their recently earned credit. However, his work was limited to two-operator GAs, and provided no comparison with fixed operator probabilities.

Tuson & Ross [266, 271] describe an investigation of Corne's COBRA (COst Based operator Rate Adaptation) algorithm, which instead of adjusting individual operator probabilities, assigns operators to a fixed set of probabilities according to their operator-productivity rank over a given interval. The operator productivity is determined by the average improvement of a child individual over its parent(s). Although COBRA has shown itself to be beneficial for timetabling problems, Tuson & Ross found it provided only equal or worse solution quality over a wide range of other test problems, compared with carefully chosen fixed operator probabilities (for both steady-state and generational GAs). They concluded that operator productivity (at least as Corne defines it) may misguide COBRA as, in some cases, although an operator's productivity is high, its preferred probability is low.

In a 1997 paper, Xiao *et al.* [259] describe an adaptive evolutionary planner/navigator for mobile robots (see §2.4 above). Their adaptation algorithm not only takes account of improvements in path fitness over its parent(s) (*cf.* COBRA [271]), but modifies this by the time taken to execute an operator and any change in the number of line segments in the path (as more line segments require both more memory and processing time). Operator probability in one interval of a given number of trials directly reflects the operator's relative figure of merit over the previous interval. Overall, they found this adaptive algorithm gave better results than either equal or pre-tuned fixed operator probabilities.

Most recently, Magyar *et al.* [261] incorporated an operator-probability adaptation method into their GA/heuristic hybrid for the three-matching problem (see §2.4 above). Their GA is generational and the operators obtain credit by improvements of the children over their parents (*cf.* COBRA [271]), but they are also penalised to an equivalent extent if there is no improvement. However, no credit is given to chains of operators (*cf.* Davis [9]), although the credit (or debit) obtained by an operator is moderated by the time taken to execute it. The actual probabilities are adjusted using an exponential moving average similar to Davis' scheme [9]. Whilst Magyar *et al.* did obtain slightly poorer results than with a non-adaptive, but finely-tuned, set of fixed operator probabilities, this was only by a very small margin, and at a considerable saving in computer time over the substantial effort involved in parameter tuning.

Overall, while a wide range of different mechanisms have been proposed for operator-probability adaptation, in general authors seem to consider such a mechanism advantageous for GAs. In addition, although this approach has not always proved superior to carefully pre-tuned probabilities, nevertheless the advantages in computer time are clear, with one run using an adaptive mechanism producing results (almost) equivalent to the many runs required for parameter tuning.

2.6 Summary

This chapter has provided a brief overview and historical perspective on EAs, a tutorial introduction to GAs, as well as three literature surveys. The overview considered early work in the field, GAs, EP, ESs, GP, CSs and EH, as well as conferences and books on the area. The tutorial provided background on both GAs in general and Goldberg's Simple Genetic Algorithm in particular before introducing a central theme of the thesis, GA/heuristic hybrids. The first survey gave near-exhaustive coverage of evolutionary telecommunications, *i.e.* the application of EAs to the telecommunications problem domain, up to December 1998. It concluded with some comments on promising directions for future research and encouragement to future authors in regard to both citation of earlier work and the reproducibility of their research. The two remaining sections gave more limited surveys of prior work on GA/heuristic hybrids and operator-probability adaptation, respectively, in areas other than evolutionary telecommunications. Overall, the chapter has given the context for the evolutionary aspects of the author's research, as well as establishing the foundation on which it was built. Accordingly, in the next chapter, the focus moves to an overview of the author's software toolset for optical network optimisation, modelling and design (NOMaD).