Chapter 4
GE in Dynamic Environments

In the previous Chapter we highlighted the fact that very little research has been conducted into the area of Genetic Programming (GP) in dynamic environments. In this book we outline the foundations of research to date with Grammatical Evolution (GE) for these kinds of non-stationary environments. As described earlier, GE possesses a number of features that differentiate it significantly from GP and it is these features that present the most interesting avenues for exploration in relation to dynamic environments, more so than in their application to static problems.

In this chapter we start out by detailing in Section 4.1 the very first steps which we have taken with GE into the domain of non-stationary environments. Following this, in Section 4.2, we discuss the potential strengths of GE for the challenges presented by a dynamic world. Finally outline in Section 4.3 how we build the foundations upon which GE can be developed for application in these formidable environments.

4.1 The First Steps

In an investigation examining the possibility of evolving the grammar that GE uses [156] the Grammatical Evolution by Grammatical Evolution (GE^2) algorithm was detailed. This study focused on the utility of such an approach for dynamic symbolic regression problems where the target function was changed after a number of generations. The approach was initially inspired by an earlier study by Keller and Banzhaf [103], which examined whether or not it might be useful to evolve the genetic code for their linear form of GP. The net effect of their approach was to evolve biases for individual symbols of their programming language, including both functions and terminals. To this end the study was a success on the static problem examined.

By evolving the grammar that GE uses to specify a solution, one can effectively permit the evolution of the *genetic code*. The ability to evolve genetic code is important when one has little or no information about the

I. Dempsey et al.: Foundations in Gramm. Evol. for Dyn. Environ., SCI 194, pp. 55–68.
springerlink.com

problem being solved, or the environment in which a population exists is dynamic in nature and adaptability is essential for survival. A more adaptive and/or open-ended representation that can facilitate progression to different environments may be required to successfully tackle non-stationary domains. We now describe the $(GE)^2$ algorithm in detail and describe how it was applied to a dynamic environment.

4.1.1 Grammatical Evolution by Grammatical Evolution

When we have a set of production rules for a non-terminal, such as, for example, `<op>::= + | -`, a codon is used to select the rule to be applied during the development of a solution. In a similar manner to a biological genetic code, the productions above represent a degenerate genetic code by which a codon is mapped to a symbol in the output language [154]. A brief overview of the biological genetic code now follows.

In biology, a codon (on mRNA), which is comprised of a group of three nucleotides from the set $\{A, U, G, C\}$, is mapped to an amino acid from the set of 20 naturally occurring amino acids. In nature, the code is encoded in transfer RNA (tRNA) molecules, which have a domino like structure, in that one end matches (with a certain affinity dubbed the *wobble hypothesis*) to a codon, while the amino acid corresponding to this codon is bound to the other end of the tRNA molecule [118]. In this sense, the above productions are equivalent to two such tRNA molecules, one matching a set of codons to $+$ while the other matches a different set of codons to $-$. By modifying the grammar, we are changing the types of tRNA molecules in our system. To put it another way, we are directly modifying the genetic code by changing the mapping of codon values to different rules (amino acids).

In order to allow evolution of a grammar, $(GE)^2$, we must provide a grammar to specify the form a grammar can take. This is an example of the richness of the expressiveness of grammars that makes the GE approach so powerful. By allowing an EA to adapt its representation (in this case through the evolution of the genetic code or grammar) it provides the population with a potential mechanism to survive in dynamic environments. Such a representation also allows the automatic adaptation of biases during the search process.

In this approach we therefore have two distinct grammars, the *meta-grammar* (or grammars' grammar) and the *solution grammar*.[1]

[1] In the original study [156] we adopted the term *Universal Grammar* instead of meta-grammar. The notion of a universal grammar is adopted from linguistics and refers to a universal set of syntactic rules that hold for spoken languages [41]. It has been proposed that during a child's development the universal grammar undergoes modifications through learning that allows the development of communication in their parents' native language(s) [172]. We now prefer the use of the term meta-grammar as it is more firmly rooted in the Computer Science discipline of formal grammars.

In $(GE)^2$, the meta-grammar dictates the construction of the solution grammar. Given below are examples of these grammars for solutions that generate expressions, which could be used for symbolic regression type problems.

meta-Grammar (Grammars' Grammar)

```
<g> ::=
      ‘‘<expr> ::= <op> <expr> <expr> | <var>’’
      ‘‘<op> ::=’’ <ops>
      ‘‘<var> ::=’’ <vars>

<ops> ::= <opt> ‘‘|’’ <ops>
      | <opt>

<opt> ::= + | - | * | /

<vars> ::= <vart> ‘‘|’’ <vars>
       | <vart>

<vart> ::= m | v | q | a
```

Solution Grammar

```
<expr> ::= <op> <expr> <expr>
         | <var>

<op> ::= ?

<var> ::= ?
```

In the example meta-grammar, a grammar, `<g>`, is specified such that it is possible for the non-terminals `<var>` and `<op>` to have one or more rules, with the potential for rule duplication. These are the rules that will be made available to an individual during mapping, and this effectively allows bias for symbols to be subjected to the processes of evolution. The productions `<vars>` and `<ops>` in the meta-grammar are strictly non-terminals, and do not appear in the solution grammar. Instead they are interim values used when producing the solution grammar for an individual.

The hard-coded aspect of the solution grammar can be seen in the example above with the rules for `<op>` and `<var>` as yet unspecified. In this case we have restricted evolution to occur only on the number of productions for `<var>` and `<op>`, although it would be possible to evolve the rules for `<expr>` and even for the entire grammar itself. It is this ability that sets this form of genetic code/grammar evolution apart from previous studies in GP. Notice that each individual has its own solution grammar.

In this study two separate, variable-length, genotypic binary chromosomes were used, the first chromosome to generate the solution grammar from the meta-grammar and the second chromosome the solution itself. Crossover operates between homologous chromosomes, that is, the meta-grammar chromosome from the first parent recombines with the meta-grammar chromosome from the second parent, with the same occurring for the solution chromosomes. In order for evolution to be successful it must co-evolve both the genetic code (otherwise known as the solution grammar) and the structure of solutions based on the evolved genetic code.

4.1.2 Experiments in GE^2 and Dynamic Environments

An instance of a symbolic regression problem was tackled in order to verify that it is possible for the co-evolution of a genetic code (or grammar) to occur

along with a solution. A target function of $f(m, v, q, a) = a + a^2 + a^3 + a^4$ was chosen, with the three input variables m,v, and q introducing an element of noise. 100 randomly generated input vectors are created for each call to the target function, with values for each of the four input variables drawn from the range [0,1]. Runs were conducted with a population size of 100, for 100 generations. The other evolutionary parameters were as follows: pairwise tournament selection, generational replacement, bit mutation probability 0.01, one-point crossover probability 0.3, codon duplication probability 0.01. Wrapping is turned off, and codon lengths are initialised in the range [1,10], with a codon size of 8-bits. Fitness is minimisation of the sum of errors over the 100 test cases, and a protected division operator is adopted that returns one in the event of a division by zero. The progress of evolution toward the target solution can be seen in Fig. 4.1 with ever decreasing error at successive generations.

Fig. 4.1 shows the increasing frequency of occurrence of the target solution symbols a, $+$ and $*$. Curiously, after 50 generations the frequency of $*$ is dramatically less than a and $+$, and even less than $/$, even though there are double the number of multiplication symbols in the target solution as there are addition operators. It is not until after this point that we begin to see an increase in the frequency of $*$, which, although it finishes considerably lower than the other two symbols, finishes higher than all others. This could have implications as to how a solution to this problem is constructed, suggesting that firstly terms are added together with the use of multiplication not occurring until much later, perhaps replacing some of the addition operators, or secondly, through expansion of terms with the multiplication of a by itself.

The above results demonstrate that it is possible to co-evolve the solution grammar and solution specification with GE^2. Two experiments were then conducted where GE^2 was applied to two instances of dynamic symbolic regression.

4.1.3 Dynamic Symbolic Regression I

In addition to learning symbol bias, dynamic problems are another area in which one could expect to derive some benefit from using evolvable grammars by adapting these biases over time. In this case, one could reasonably expect a system with an evolvable grammar to be able to react more quickly to a change in the environment than a static one could, as a single change in a grammar can reintroduce lost genetic material. The target functions for the first instance are:

i. $f(m, v, q, a) = a + a^2 + a^3 + a^4$
ii. $f(m, v, q, a) = m + m^2 + m^3 + m^4$
iii. $f(m, v, q, a) = v + v^2 + v^3 + v^4$
iv. $f(m, v, q, a) = q + q^2 + q^3 + q^4$
v. $f(m, v, q, a) = a + a^2 + a^3 + a^4$

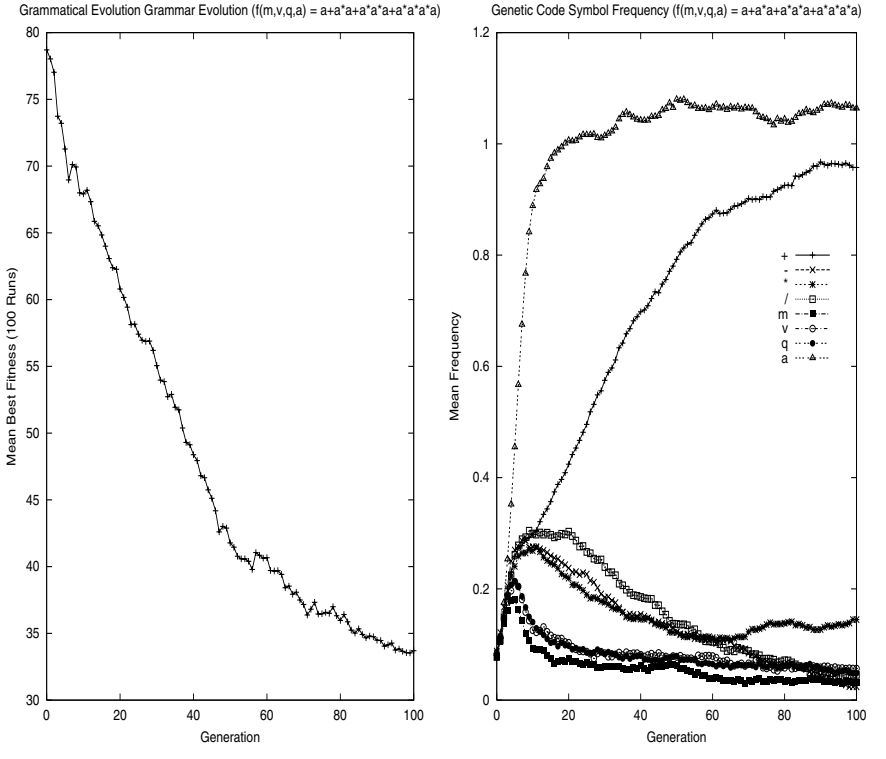

Fig. 4.1 A plot of the mean best fitness (left) and mean symbol frequency (right) from 100 runs of the quartic symbolic regression problem

The target changes between the functions above every 20 generations. The only difference between each successive function is the variable used. 100 randomly generated input vectors are created for each call to the target function, with values for each of the four input variables drawn from the range [0,1]. The symbols −, and / are not used in any of the target expressions. Runs were conducted with a population size of 500, for 100 generations, with all other parameters as reported earlier. A plot of the average best fitness and average symbol frequencies can be seen in Fig. 4.2. A sample of evolved grammars from one of the runs is given below, where in each case the grammar selected is the best solution from the generation just prior to a change in target.

Target 1

```
<op>::= +
<var>::= a
<expression>::= + a a
fitness: 34.6511
```

Target 2

```
<op>::= +
<var>::= m
<expression>::= + m m
fitness: 34.2854
```

Target 3

```
<op>::= + | -
<var>::= v
<expression>::= + v v
fitness: 36.6667
```

Target 4

```
<op>::= + | *
<var>::= q
<expression>::= + + q q * * q q * q q
fitness: 22.8506
```

Target 5

```
<op>::= + | *
<var>::= a
<expression>::= + * a + a a * a a
fitness: 7.85477
```

The results presented suggest that, when using dynamic grammars, it is possible to successfully preserve and improve solution structure, while still being able to learn appropriate terminal values. This is reflected in the fitness plot where, when the fitness function changes, in most cases there is a decrease in solution fitness for a short period when solutions adjust to the new variable

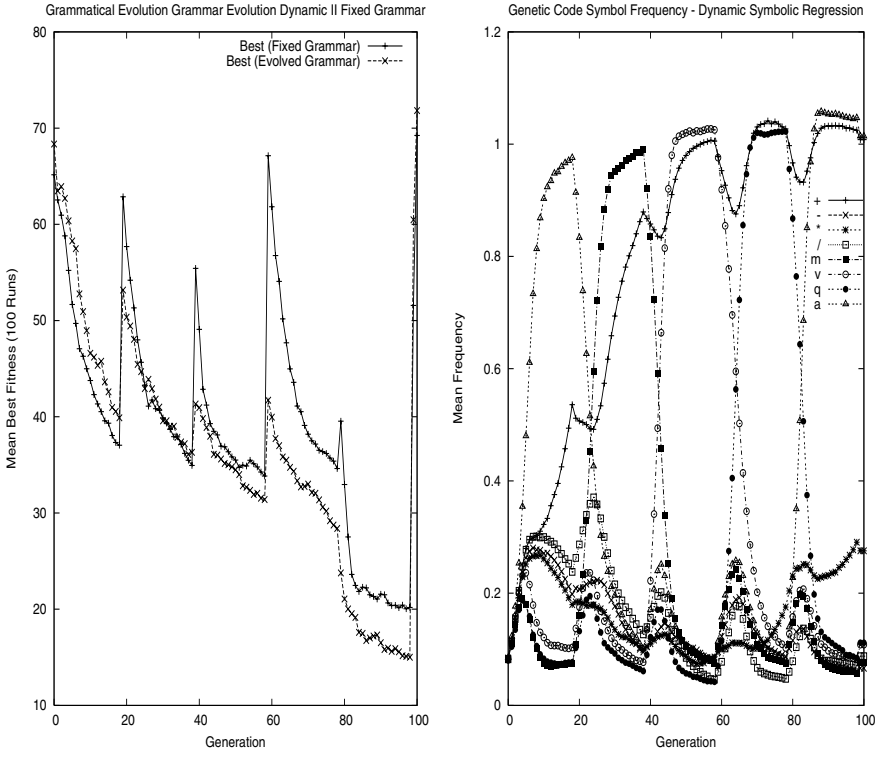

Fig. 4.2 Plot of the mean best fitness over 100 generations on the first dynamic symbolic regression instance with both static and dynamic grammars (left). Symbol frequency plot (right).

Table 4.1 Statistics for both the static and evolvable grammars on the first dynamic problem instance. Lower scores indicate better performance.

Fitness Case	mean fixed(dynamic)	median fixed(dynamic)	std. dev fixed(dynamic)	signif.
1	37.33 (40.55)	37.75 (38.22)	7.81 (10.082)	Yes
2	35.48 (36.08)	37.1 (36.57)	6.35 (8.73)	No
3	34.26 (31.53)	36.6 (36.48)	7.54 (10.79)	Yes
4	35.39 (28.74)	37.2 (35.08)	7.96 (12.46)	Yes
5	20.05 (15.1)	22.00 (20.54)	5.99 (10.17)	Yes

adopted. Later on in the simulations we reach the point where the structure becomes closer to the target and changes in variables alone no longer confer as much damage to fitness, which is again illustrated in the fitness plot (Figure 4.2).

A performance comparison of the dynamic and static equivalent of the grammar (given below) for this problem is presented in Fig. 4.2 and corresponding statistics can be found in Table 4.1.

```
<expr> ::= <op> <expr> <expr> | <var>

<op> ::= + | - | * | /

<var> ::= m | v | q | a
```

4.1.4 Dynamic Symbolic Regression II

The target functions for the second dynamic symbolic regression problem instance are:

i. $f(m, v, q, a) = a + a^2 + a^3 + a^4$
ii. $f(m, v, q, a) = m + a^2 + a^3 + a^4$
iii. $f(m, v, q, a) = m + m^2 + a^3 + a^4$
iv. $f(m, v, q, a) = m + m^2 + m^3 + a^4$
v. $f(m, v, q, a) = m + m^2 + m^3 + m^4$

The target changes between the functions above every 20 generations. The transition used in this problem differs from the previous in that only one term changes each time. However, the change is larger each time (because the power that the new term is raised to increases). 100 randomly generated input vectors are created for each call to the target function, with values for each of the four input variables drawn from the range [0,1]. The symbols q, v, $-$, and / are not used in any of the target expressions. As in the previous dynamic symbolic regression problem instance runs are conducted with a population size of 500, for 100 generations, with all other parameters as per the standard values reported earlier. A plot of the average best fitness and average symbol frequencies can be seen in Figure 4.3.

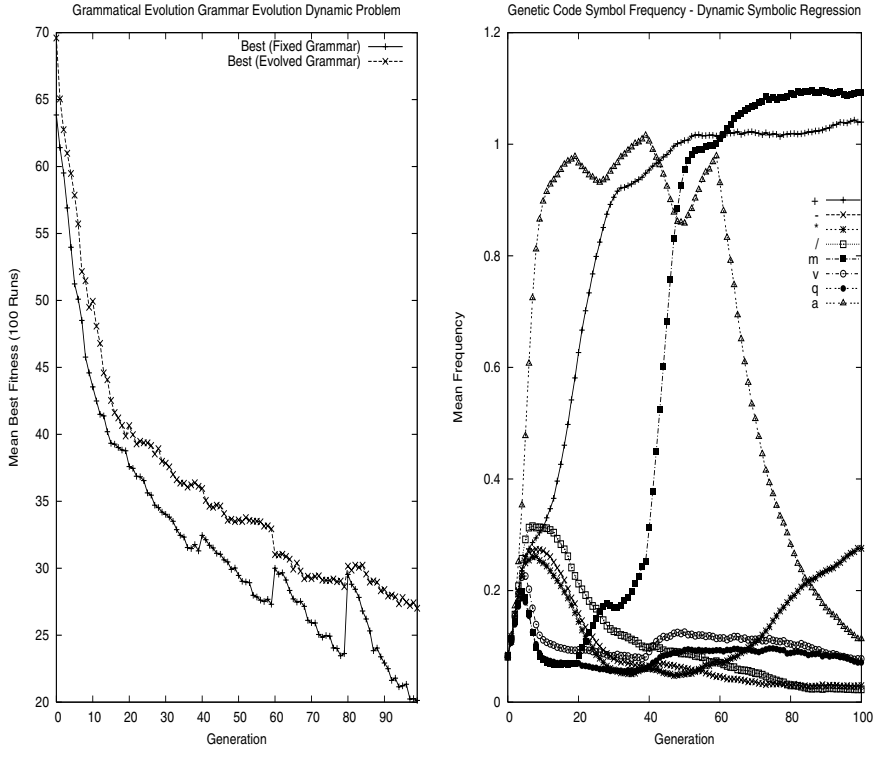

Fig. 4.3 Plot of the mean best fitness over 100 generations on the second dynamic symbolic regression instance with both dynamic and static grammars (left), and the mean symbol frequency (right)

It is interesting to note that fitness keeps improving over time for the evolvable grammar, with an occasional deterioration corresponding with a change in the fitness function. Notice how the disimprovement is more pronounced later in the runs, particularly for the static grammar, which is due to higher powers being exchanged. These results suggest that the evolvable grammar is more adaptable in scenarios with larger changes facilitating smoother transitions to successive targets. Also evident from Figure 4.3 is the manner in which the quantity of a in the population decreases over time while that of m increases. The two plots intersect at around generation 42, shortly after the target has changed to $f(m, v, q, a) = m + m^2 + a^3 + a^4$. However, the plots remain very close until around generation 60, at which time m^3 becomes part of the solution.

A sample of evolved grammars from one of the runs is given below, where the grammars presented represent the best solution at the generation just prior to each fitness change.

Target 1

```
<op>::= + | +
<var>::= a
<expression>::= (+ a a)
fitness: 37.4525
```

Target 2

```
<op>::= +
<var>::= m | a
<expression> = (+ a m)
fitness: 33.8423
```

Target 3

```
<op>::= + | *
<var>::= m | a
<expression> = (+ a (+ m (* a m)))
fitness: 22.9743
```

Target 4

```
<op>::= + | *
<var>::= m
<expression> = (+ m (* (+ (* m m) m ) m) )
fitness: 15.6311
```

Target 5

```
<op>::= + | *
<var>::= m
<expression>::= (+ (* (+ m (* m (+ m (* m m) m) m) ) ) )
fitness: 4.57967e-15
```

Table 4.2 Statistics for the second dynamic problem instance. Lower numbers indicate a better fitness.

Fitness Case	mean fixed(dynamic)	median fixed(dynamic)	std. dev fixed(dynamic)	signif.
1	39.27 (41.63)	37.98 (38.65)	9.18 (12.59)	No
2	31.55 (36.06)	31.93 (36.60)	6.77 (3.84)	Yes
3	27.62 (33.46)	25.82 (34.52)	6.3 (4.1)	Yes
4	24.05 (29.2)	22.62 (32.17)	5.83 (6.39)	Yes
5	21.34 (27.47)	18.74 (35.2)	11.42 (14.94)	Yes

A performance comparison of the dynamic and static equivalent of the grammar (static grammar as per earlier dynamic problem instance) for this problem is presented in Figure 4.3 and corresponding statistics can be found in Table 4.2. In this case the static grammar outperforms the evolving grammar in terms of best fitness values achieved for all targets but the first. With the evolving grammar there is, as usual, a warm up period where a suitable grammar must be adopted before good solutions can be found. When successive targets are very similar to previous ones this almost negates the potential benefits that a more adaptive representation can bring, as in the case of the evolvable grammars. Clearly, some dynamic problems are more dynamic than others as discussed in Chapter 3, especially in terms of the degree of change. Previous work (e.g., [144]) with GAs applied to dynamic problems has shown that, when the change is relatively small, a standard GA with high mutation can handle those types of problems. We believe it is likely to be the same for GP.

These results would also lend support to the idea of introducing different operator rates on the grammar chromosome to the solution chromosome,

allowing the population to converge towards a similar grammar, facilitating the exploration of solutions based on a similar grammar. If these rates were adaptable, then it may be possible to allow grammars to change more often if the target changes are large, and vice versa.

This first study in the application of GE to dynamic environments was encouraging as it suggested that there may be benefits to representational flexibility, which can be provided with a GE approach. The following section describes some of the potential strengths of GE for dynamic problems.

4.2 Strengths

In [148] O'Neill described eight desireable features, inspired by Molecular Biology, that evolutionary algorithms could incorporate as researchers attempt to improve their algorithms' performance. Seven of these, implemented in GE, are as follows:

i. **Generalised encoding that can represent a variety of structures**
Through the decoupling of search and solution spaces, a mapping process can afford the opportunity to generate phenotypes in an arbitrary language.

ii. **Efficiency gains for evolutionary search**
Improvement of the evolutionary search is generally positive. By incorporating degenerate genetic code, a medium is provided for neutral mutations to occur giving rise to neutral evolution. As described in Sections 3.3.5 and 3.3.6, this yields efficiency gains in evolution.

iii. **Maintenance of genetic diversity within an evolving population**
Diversity was highlighted as being one of the key approaches for evolutionary algorithms in dynamic environments in Chapter 3. An ability to maintain a diverse dispersed population allows the algorithm to conduct wide coverage of the search space, equipping it with the ability to quickly discover new optima when the environment changes.

iv. **Preservation of functionality while allowing continuation of search** Providing a mechanism to conduct neutral mutations allows an algorithm to preserve a functioning, fit, phenotype while continuing the evolutionary search. This feature gives rise to the potential of Engelhart's neutral networks being evolved. Individuals aligned along neutral networks in the genotypic space may then easily evolve to produce alternate phenotypes.

v. **Reuse of genetic material**
Reusing genetic material presents efficiency gains in an implemented algorithm. When individuals in a population are of variable length, the possibility for *bloat* arises whereby the amount of genetic material in the individuals may grow over the course of the evolutionary process.

vi. **A compression of representation**
Again regarding an implemented algorithm this feature bears benefits at execution.

vii. **Positional independence**
By decoupling the expressed functionality of a gene from its position on the chromosome a mechanism is provided to preserve the functionality of genes after crossover. This may lead to more productive recombination events.

Standard GE, as described in [148], integrated the first six features with further research leading to the implementation of the seventh feature in πGE [157]. Considering the various approaches for evolution in dynamic environments identified in Section 3.3, some of these features bear special relevance when applied to dynamic environments.

The first feature of adopting a generalised encoding is achieved in GE through the use of the BNF grammar plug-in. With regard to dynamic environments, the development of the grammar is of particular importance as it affords the modeller the opportunity to incorporate some fundamental domain knowledge into the system while maintaining flexibility. Recent developments in GE have lead to the development of meta-Grammars and Grammatical Evolution by Grammatical Evolution. In this case a meta-Grammar or grammar's grammar is defined, which allows evolution to evolve its own vocabulary for the expression of phenotypes. Chapter 6 will deal with this in more detail.

Regarding efficiency gains for evolutionary search, O'Neill refers to the presence of degenerate genetic code and its relationship with Kimura's neutral theory of evolution [108]. The degenerate or redundant genetic material is brought about in GE through the many-to-one mapping from genotype to phenotype, allowing neutral mutations to occur. This caters directly to dynamic environments as it allows for the development Engelhart and Newman's neutral networks [140] as already outlined in Section 3.3.5 and provides a representation that is more evolvable than a direct encoding as outlined in Section 3.3.6.

With the employment of the genotype-to-phenotype mapping and the presence of degenerate genetic code, features three and four are simultaneously addressed. Due to the unconstrained search in the genotype space with a many-to-one mapping back to the phenotype solution space, there is the potential for a high degree of diversity to be maintained continuously over the life cycle of the algorithm. Whether this is in a static or dynamic environment, it places this feature in the category of preemptive approaches used to maintain diversity.

The possibility of neutral evolution brought about by degenerate genetic code is also responsible for a robustness in the solution space to changes in the genetic search space. This is because as mutations can occur with the presence of degenerate code and still produce the same phenotype. This mechanism then allows GE to preserve the expressed functionality of the phenotype

while continuing with a neutral evolutionary search in the genotypic space dispersing across a neutral network.

Features five and six are implemented in GE through the use of the wrapping operator. In other words, if upon reaching the end of the genome, an individual is still not a fully formed phenotype, mapping will continue again at the beginning of the genetic string. Though O'Neill found the use of this operator to provide a better success rate than with the operator turned off, the operator does not address directly any of the extra issues involved in dynamic environments. Table 4.3 presents a summary of the implemented features in GE.

Table 4.3 Implementation of features in GE

	Feature	Implementation
1	Generalised Encoding	BNF Grammar
2	Efficiency Gains	Degenerate Genetic Code
3	Diversity	Many to One Mapping
4	Preservation of Functionality During Search	Many to One Mapping
5	Re-use of Genetic Code	Wrapping Operator
6	Compression of Representation	Wrapping Operator
7	Positional Independence	πGE

4.3 Extending GE for Dynamic Environments

Aside from [156], no work has been done in exploring the use, and analysing the performance, of GE in dynamic environments prior to the research outlined in the remainder of this book. In light of this, shortfalls specific to GE's proficiency in these environments have not been identified. Considering that GE has undergone little analysis in dynamic environments, a number of the features built-in to the design of the paradigm bear particular significance when placed in a dynamic environment. The addition of BNF grammars as an input to GE present an extra level of representational flexibility that may be exploited in terms of evolvability. With BNF grammars, the researcher may incorporate a greater level of adaptability in the vocabulary available to the search process while ensuring syntactic correctness. Through the design of the grammar, a structure may be created that aids in the evolution towards fitter solutions. The ability to allow the evolutionary process to evolve its own grammars takes this feature even further. Reflecting upon the approaches highlighted in this book so far, with the inclusion of degenerate genetic code and many-to-one mapping, standard GE is equipped with an ability to form neutral networks in the genotypic space through neutral evolution and to preemptively maintain a high degree of dispersed diversity unconstrained by a phenotypic search-space boundary.

In placing the features of GE in the context of tables developed in this chapter, the inclusion of the BNF grammar allows researchers to scope the

adaptiveness of the vocabulary. It also presents an opportunity to incorporate useful domain knowledge tailoring it for problems where information can be extracted out of the domain. Added to this, the BNF grammar can aid in the evolvability of individuals thus aiding in the evolution for all types of change. Finally, the maintenance of diversity through many-to-one mapping is a feature that enables wide coverage of the solution space as evolution progresses. While all problems benefit from this coverage of the search space, the maintenance of diversity throughout the life cycle of a run is tailored for Markov and Complex type changes. Though Deterministic problems will also benefit from this diversity early in the search, a level of convergence is sought in uncovering the predictable nature of the change. Therefore the maintenance of diversity is of lesser importance for this type of problem. Table 4.4 summarises these features.

Table 4.4 Features of GE that cater to the types of change faced in dynamic environments

Type	BNF Input (Evolvability)	Many-to-One (Diversity)
Markov	X	X
Complex System	X	X
Deterministic	X	-

A problem identified in GP in general, and also existing in GE, is its inflexible approach to the creation and variance of constants [9]. GE's approach to the variance of constants is through the evolution of expressions operating on a handful of constants that are defined in the BNF grammar prior to the system's execution. This weakness presents particular problems when conducting evolution in a dynamic environment, as for both efficiency and flexibility purposes, it may be desireable to directly adapt old or evolve new constants in the solutions as the optimum shifts or moves on the landscape. Indeed, in many of these problems a continual adaptation of constants is required and improvements in this area will be useful in applying GP and GE to dynamic problems. This weakness will be addressed in Chapter 5. It is also worth noting that even though [156] did apply $(GE)^2$ to dynamic environments it did so with very little analysis pertaining to the dynamic domain. Chapter 6 will endeavour to take this analysis further and examine the effects of the extra dimension of adaptability in the algorithm.

Through examining the extra layer of adaptability in $(GE)^2$ and adapting a population of solutions as time progresses the representation of solutions as well as the solutions themselves are being adapted. Another facet of GE may also be adapted with the progression of time–dynamic adjustment of the parameters of the algorithm. These include, for example, the number of generations of evolution and probabilities for applying genetic operators. Work has been conducted on investigating the effects of adapting these parameters for static environments [141].

4.4 Conclusion

In conjunction with the review of EC for dynamic environments and our knowledge of GE to date we have identified a number of research gaps. These are summarised as follows:

- **Analysis of the level of diversity maintained in GE promoted by the separation of search and solution spaces**
  The maintenance of a dispersed, diverse population is of great importance in dynamic environments; the separation of search and solution spaces may aid this. An examination of this is undertaken in Chapter 8.
- **Exploiting the use of BNF grammars to improve evolvability**
  Not only does GE provide for improved evolvability through potential search efficiencies presented due to its genotype-to-phenotype mapping and subsequent separation of its search and solution spaces. In addition, the opportunity exists to improve evolvability through the use of the BNF grammar used by GE. The majority of the research presented in this book focuses on this form of representational evolvability. Chapters 5 and 6 explore this.
- **Use of a real-world problem as a benchmark, and the application of problem domain specific analytics to determine behaviour of algorithm developments**
  The use of such a benchmark and analytics provide both qualitative and quantitative evidence of algorithm performance and also facilitates comparison with human achievements. Chapter 8 conducts such analysis in the financial domain of trading.
- **Application of GP type paradigm to a dynamic problem**
  The lack of research in the application of GP to dynamic problems is remarkable, this book will endeavour to close that gap. This gap is addressed in a number of the experimental chapters beginning with more simple types of change and progressing to complex types of change in Chapter 8.

In subsequent chapters these gaps shall be examined in greater detail.