Chapter 6
Constant Creation with meta-Grammars

In the previous chapter, the utility of different variations of the Digit Concatenation method for constant creation were examined. Digit Concatenation was also combined with and compared to, Persistent Random Constants. In these experiments, the Digit Concatenation method was seen to produce the most fit individuals more regularly than the other methods examined. In this chapter, the Digit Concatenation method is further explored by examining Digit Concatenation with the use of a meta-Grammar based approach using Grammatical Evolution by Grammatical Evolution ($(GE)^2$) [156]. A meta-Grammar is employed in a dual-chromosomal structure, where one chromosome describes the solution as usual and the second chromosome is the individual's own grammar that maps the solution chromosome. The meta-Grammar is used to map the grammar chromosome for each individual. Providing GE with the ability to evolve its own grammar, specific to each solution, presents an extra layer of adaptability.

This added feature may have benefits for evolution in dynamic environments in particular, as it allows GE to evolve the very vocabulary it uses to describe phenotypic solutions, potentially incorporating biases into the grammar learned through feedback from the environment. This extra layer is a feature that is unique to GE over the fixed function and terminal sets allowed in GP. The inclusion of meta-Grammars also matches with the adaptability requirement for constant generation techniques for dynamic environments, outlined at the beginning of the previous chapter. Added to this, the solution grammar itself has the potential to form a representational memory where the type of change in a dynamic problem is Deterministic oscillatory. The solution grammar can achieve this through biasing the grammar towards evolving the targets that are being switched between.

In this chapter, the performance of the meta-Grammar approach is compared with that of the grammars in the previous chapter. The aim of the chapter is to determine whether the incorporation of a meta-Grammar and $(GE)^2$ produces a boost in performance over the standard GE approach. The next section begins with a brief description of the Grammatical Evolution by Grammatical Evolution paradigm to refresh the reader's memory. This is followed by two sections of experiments that examine the utility of $(GE)^2$ and seek to further understand the behaviour of the Digit Concatenation approach to constant creation.

6.1 Grammatical Evolution by Grammatical Evolution

A description of the Grammatical Evolution by Grammatical Evolution (GE^2) approach is provided in Chapter 4. A brief summary of the approach is provided here as a reminder. In order to allow evolution of a grammar, for $(GE)^2$, another grammar (meta-Grammar) must be provided to specify the form a grammar can take. This is an example of the richness in the expression of grammars that make the GE approach so powerful. By allowing an EA to adapt its representation (in this case through evolution of a solution's grammar), it provides the population with a mechanism or an extra layer of adaptability to survive in dynamic environments in particular, and also to automatically incorporate biases into the search process.

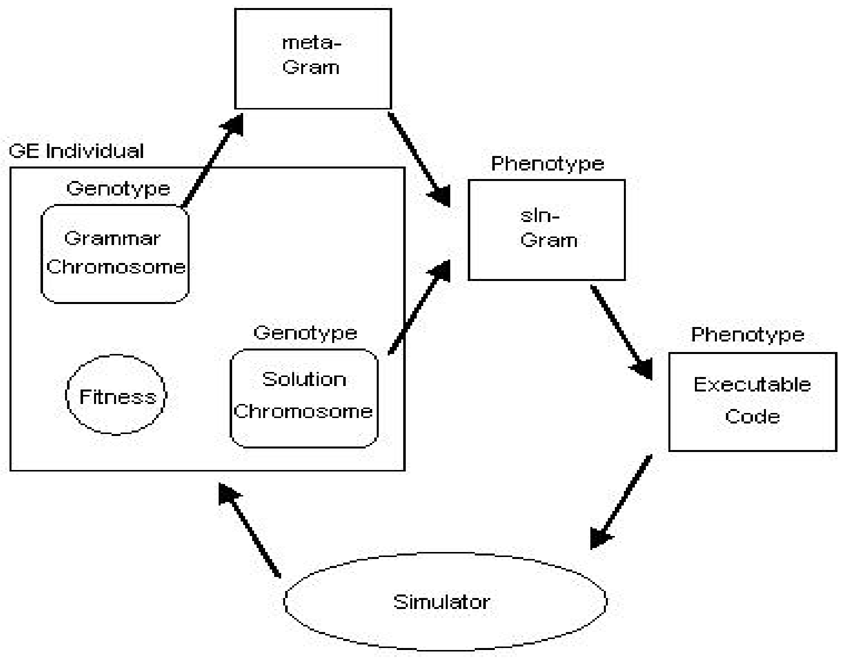


Fig. 6.1 The grammar genotype chromosome is mapped to its BNF phenotype using the meta-Grammar. The solution genotype is mapped to executable code using the solution grammar. This is evaluated in the simulator and a fitness returned to the GE individual that is made up of two genotypic chromosomes and a fitness score.

In this approach then, there are two distinct grammars: the *meta-Grammar* (or grammars' grammar) and the *solution grammar*. The meta-Grammar dictates the construction of the solution grammar. In the experiments described in this chapter, two separate variable-length, genotypic chromosomes were used: the first chromosome to generate the solution grammar from the meta-grammar, and the second chromosome to generate the solution from the solution grammar. Crossover operates between homologous chromosomes, that is, the solution-grammar chromosome from the first parent recombines with the solution-grammar chromosome from the second parent, with the same occurring for the meta-grammar chromosomes. In order for evolution to be successful it must co-evolve both the grammar and the structure of solutions based on the evolved genetic code. Figure 6.1 presents a diagram illustrating the structure of a $(\text{GE})^2$ individual as well as the mapping process.

Other researchers have also indirectly examined the evolution of grammars with GE by evolving Lindenmayer-systems. Ortega [169] employed GE to evolve Lindenmyer grammars for the construction of fractal curves, while Hemberg and O'Reilly [87] also evolve Lindenmyer grammars for the generation of curved surfaces in Genr8. Outside of GE, Keller and Banzhaf [103] investigated the evolution of genetic code by allowing each individual to evolve its own mapping table for the genotype-to-phenotype mapping.

The inclusion and evolution of an individual's own solution grammar provides two potential benefits for conducting evolution in dynamic environments:

- **Adaptability:** By allowing an individual to evolve its own vocabulary an extra dimension of adaptability is provided. Once the phenotypic grammar has changed as a result of genetic operators, whole new areas of the phenotypic solution space can be opened up due to the provision of new terminals in the solution grammar.
- **Representational Memory:** The fitness of an individual in $(\text{GE})^2$ is dependent upon an effective grammar being evolved that allows the solution chromosome to produce a fit phenotype. Where the problem domain is dynamic, this phenotype will have to change over time. For an individual to survive and maintain a good fitness, it is therefore incumbent upon the solution-grammar chromosome, to provide a grammar that can allow the solution chromosome to express fit solutions effectively. Where the type of change being conducted is a Deterministic oscillating problem, a solution grammar may include potential representations for the previously visited targets.

A consequence of including an extra chromosome for each individual in the population is that the utilisation of computational resources for storing and evaluating the population is effectively doubled. A criticism of explicit memory, laid out in Section 3.3, was that it negated part of the efficiency gain of maintaining and evolving a population across changes in the environment by adding storage space for individuals that were deemed useful enough to be included in memory. The evaluation process was also effectively increased, due to the necessity of evaluating each of the stored individuals whenever a change in the environment occurred. In this case, similar criticism can be made against the incorporation of a second chromosome for each individual. While the grammar chromosome is not explicitly evaluated itself, an extra computational overhead is experienced in storing the extra chromosome and also in conducting the GE mapping process a second time for each individual. Therefore, experiments in this chapter will seek to maintain an equilibrium in the utilisation of computational resources by reducing the size of the population by 50% in the case of the $(\text{GE})^2$ setup.

6.2 Evolving Constants Using a meta-Grammar with Digit Concatenation

This section analyses the use of a meta-Grammar that incorporates the use of pure Digit Concatenation, that is Digit Concatenation without the ability to form expressions. To place perspective on the relative performance of the meta-Grammar, the results in this section will be compared with those in Section 5.4 that used a combination grammar incorporating the Traditional method, Digit Concatenation without the ability to form expressions and Persistent Random Constants.

6.2.1 Problem Domain and Experimental Approach

The constant generation problems tackled are: Finding a Static Constant, Finding Dynamic Real Constants, and the Logistic Difference Equation. This enables the comparison of results of the meta-Grammar approach with results achieved using earlier grammars and to incorporate Markov and Deterministic types of change in the dynamic experiments. The meta-Grammar used for constant generation is provided below.

```
<g> ::= "<SlnCatR> ::=" <catRs>
        "<SlnCat> ::=" <cats>
        "<SlnDigit> ::=" <digit>
<catRs> ::= <catRt> "|" <catRs> | <catRt>
<catRt> ::= "<SlnCat>" | "<SlnCat>"."<SlnCat>"
<cats> ::= <catT> "|" <cats> | <catT>
<catT> ::= "<SlnDigit>"<catT> | "<SlnDigit>"
        | <digit>
<digit> ::= <digitT> "|" <digit> | <digitT>
<digitT> ::= 0|1|2|3|4|5|6|7|8|9
```

A simple example of this meta-Grammar in action is seen in evolving the static target 50. In one such experiment, the meta-Grammar produced the solution grammar displayed below.

```
<SlnCatR> ::= <SlnCat>
<SlnCat> ::= <SlnDigit>0
<SlnDigit> ::= 5
```

This solution grammar then makes it very easy to produce the target, as the only mapping available produces 50. This underlines the strength of the dual structure and use of a meta-Grammar, as it allows the grammar itself to specialise towards the solution.

As the meta-Grammar's performance is being compared to that of the combination grammar of the previous chapter, the grammar is presented below.

```
<exp> ::= <value>
<value> ::= <trad> | <catR> | <persistent>
<op> ::= + | - | / | *
<trad> ::= <trad> <op> <trad> | <tradT>
<tradT> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<catR> ::= <cat> <dot> <cat> | <cat>
<cat> ::= <cat> <catT> | <catT>
<catT> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<dot> ::= .
<persistent> ::= <persistent> <op> <persistent> | <persistentT>
<persistentT> ::= ‘‘150 randomly generated real constants’’
```

6.2.2 Results

The experimental parameters adopted are the same as those outlined in Section 5.3.4 with the exception of population size. In the case of the meta-Grammar runs population sizes of 250 were adopted in order to maintain an equal computational effort with prior experiments. For convenience, Table 6.1 provides a summary of the experimental parameters adopted. A description of the results is given next, followed by a discussion of same.

Table 6.1 Experimental parameters adopted

Parameter	$(GE)^2$	Combo Grammar
Population Size	250	500
Crossover	0.9	0.9
Mutation	0.1	0.1
Selection	Roulette	Roulette
Replacement	25%	25%
Wrapping	8 events	8 events
Codon Size	8 bits	8 bits

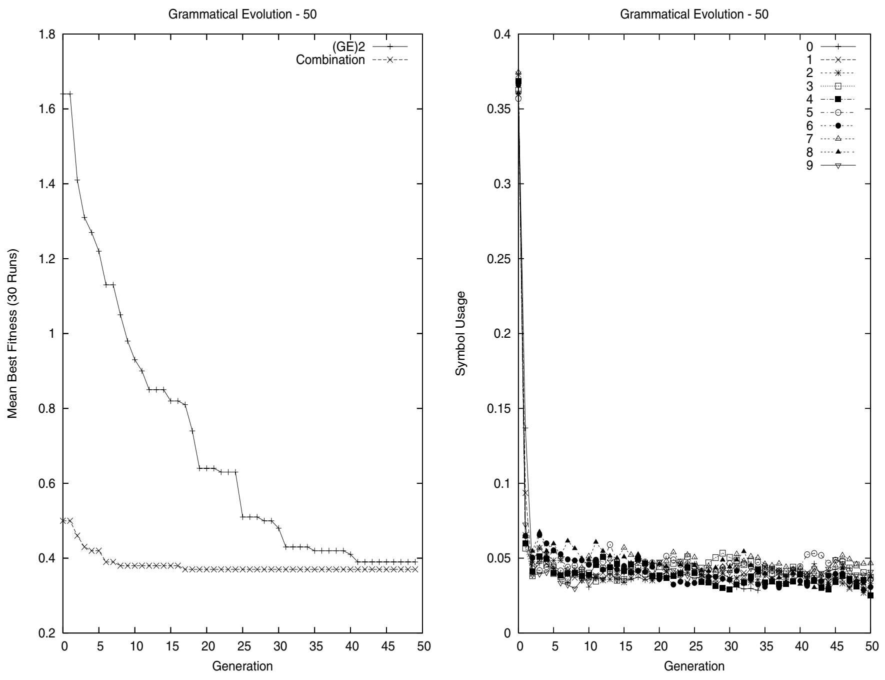


Fig. 6.2 Plot of the mean best fitness values for each constant generation method (left) for the static target 50 and the mean symbol usage at each generation (right) on the static constant problem instance

Finding a Static Constant

The results presented in Figure 6.2 display a comparison of the average best fitness of each of the grammars over the 30 runs. As can be seen the meta-Grammar begins with a poorer fitness in comparison to the combination grammar but quickly evolves a comparable fitness over the 50 generations. A t-test and a bootstrap t-test reveal that there is no significant difference in the results by the final generation. The average best performance of the combination grammar by the final generation was 0.373499 with 6 runs evolving the exact target. In comparison the meta-Grammar produced an average best performance of 0.391333 and reached the target exactly in 14 of the 30 runs.

Finding Dynamic Real Constants

In Figure 6.3 graphs are presented for the dynamic Markov experiments where the set of numbers to be evolved over the course of a run are: 192.47, 71.84, 71.83, 173.59, and 192.47, changing at every 10^{th} generation. Here the Combination grammar again begins with a good fitness. Once generation 10 is reached and the target changes to 71.84, it quickly attains a very good fitness. In these

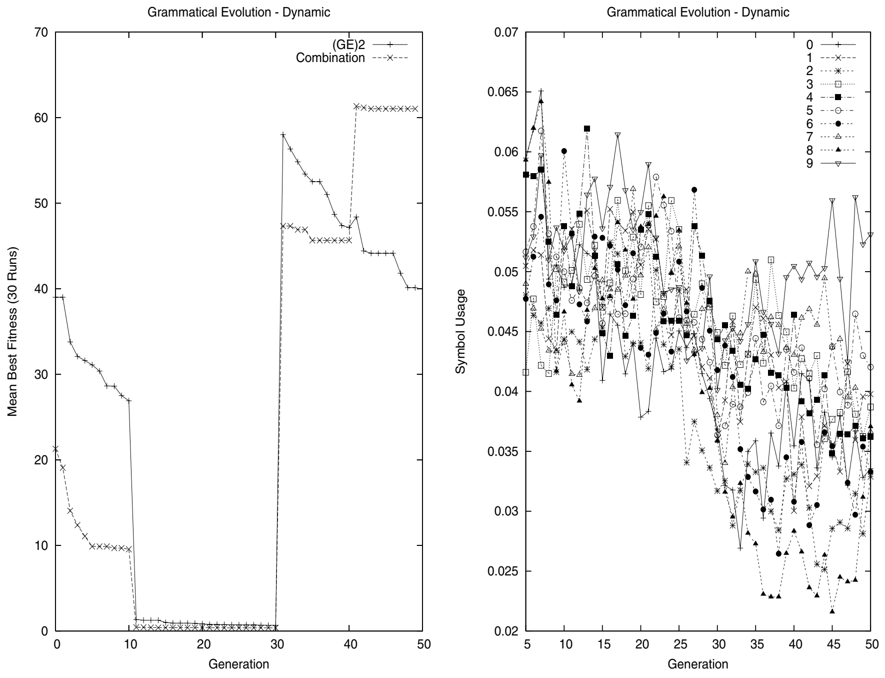


Fig. 6.3 Plot of the mean best fitness values for each constant generation method (left) and the mean symbol usage at each generation (right) on the first dynamic problem instance

experiments, the meta-Grammar method again starts off with a poorer fitness but attains a fitness similar to the combination grammar by generation 10. When the target changes to 173.59, it too experiences a strong deterioration in fitness. However, unlike the combination grammar it takes large leaps in fitness as the generations progress on this target. It then goes on to continue improving fitness when the target again shifts at generation 40 to 192.47.

Results for the Deterministic oscillating non-stationary problem instance are presented in Figure 6.4. In the second instance of this problem where the target oscillates from 192.47 to 71.84 every 10 generations, a similar trend is noticed. For the meta-Grammar method, it again begins with a poorer fitness but quickly catches up with the combination grammar by generation 10. Once the target changes to 192.47, a similar story to the previous dynamic Markovian experiments is observed. The meta-Grammar begins with a poor fitness but quickly evolves more fit individuals over the 10 generations. Interestingly, this trend is emphasised when the target hits 192.74 for the second time where the meta-Grammar begins with a fitness that is worse than at the start the last time, but ends on a fitness that is better than at generation 30.

An interesting aside at this point is to examine the changes in the best-performing solution grammar over the course of an oscillation experiment.

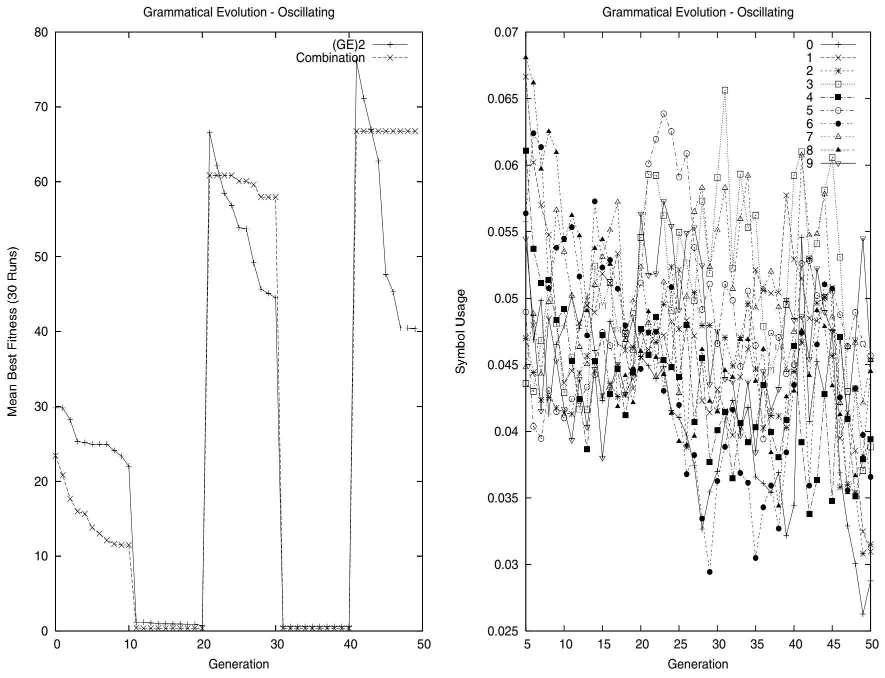


Fig. 6.4 Plot of the mean best fitness values for each constant generation method (left) and the mean symbol usage at each generation (right) on the oscillating dynamic problem instance

In one such experiment, the meta-Grammar was able to produce a solution grammar that helped in closely approximating the 71.84 target. This grammar is described below.

```
<SlnCatR> ::= <SlnCat> | <SlnCat>.<SlnCat>
<SlnCat> ::= <SlnDigit>2
<SlnDigit> ::= 3 | 7
```

A solution yielded by this grammar produced a phenotype of 72.32. The system was then able to maintain this individual in the population when the target swung to 192.47, and recall it again when the target returned to 71.84, demonstrating a population-based memory. A form of representational-based memory is also demonstrated in the grammar below.

```
<SlnCatR> ::= <SlnCat>.<SlnCat>
<SlnCat> ::= <SlnDigit><SlnDigit>
           | <SlnDigit><SlnDigit>
           | <SlnDigit><SlnDigit>6
<SlnDigit> ::= 5 | 6 | 9 | 1 | 7 | 4 | 6
```

Here, a grammar was evolved which allowed the solution chromosome to form phenotypic solutions that were good approximations of both targets. The grammar yielded 196.97 as the solution for target 192.47 and could obtain a good fitness for the target 71.84 by combining the terminals 7 and 1, followed by a pair of floating point digits.

These experiments also saw $(GE)^2$ steadily produce improving fitnesses when the target switched to 192.47 at a better rate than standard GE, highlighting an ability to adapt more easily. Right down to the final generation, better grammars are produced when one best-performing grammar transitioned from:

```
<SlnCatR> ::= <SlnCat>
<SlnCat> ::= <SlnDigit>0
           | <SlnDigit><SlnDigit><SlnDigit>
<SlnDigit> ::= 0 | 1 | 0
```

and a solution of 101 to:

```
<SlnCatR> ::= <SlnCat>.<SlnCat>
<SlnCat> ::= <SlnDigit><SlnDigit><SlnDigit>
<SlnDigit> ::= 0 | 2 | 9 | 1
```

and a solution of 190.021, achieving a good approximation of the target. This is in contrast to the combination-grammar method that generally saw little or no evolution towards the target when it swung to the largest number outside the range of the PRC range.

The Logistic Difference Equation

Here both methods present good fitnesses. Table 6.2 shows average best fitnesses for the different values of α.

Table 6.2 Average best fitness for different values of α for each grammar

α	**Combo**	**$(GE)^2$**
3.59	0.000061	0.00032
3.80	0.00045	0.00041
3.84	0.00024	0.0002468

As can be seen in the table meta-Grammar performs well in comparison to the combination grammar with close results in all but $\alpha = 3.59$. Figure 6.2 presents a sample of the results for $\alpha = 3.84$. It once more follows trends seen in the previous experiments, where the meta-Grammar begins with a poorer fitness but rapidly takes the evolutionary steps to reach a fitness similar to the combination grammar.

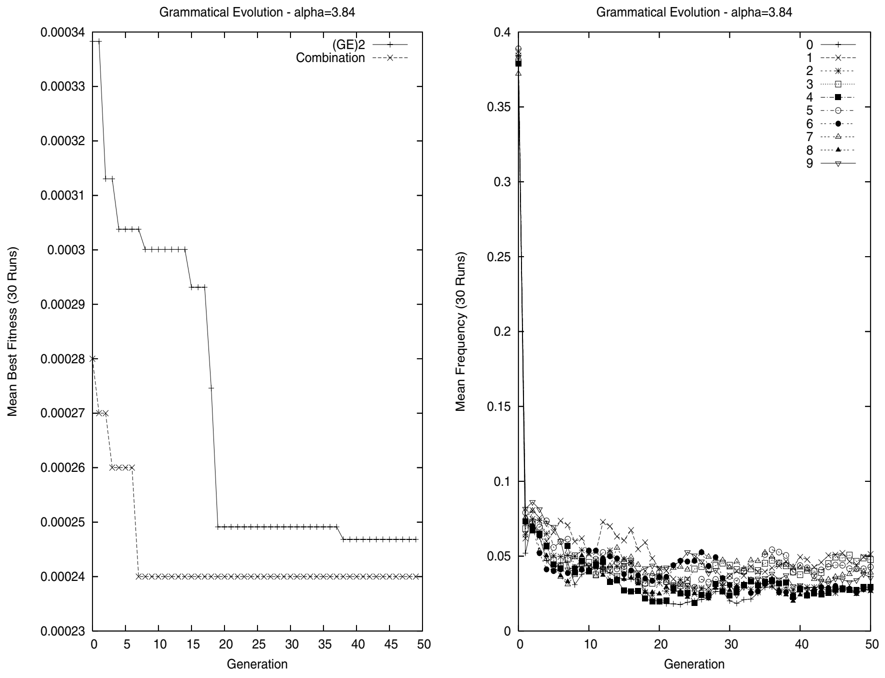


Fig. 6.5 Plot of the mean best fitness on the logistic difference equation problem instance where α=3.84

Discussion

In this section, the combination grammar used in Section 5.4 was compared with a meta-Grammar that adopted the pure Digit Concatenation method, and advantages of the meta-Grammar approach over the combination grammar were seen. These advantages were largely seen in the dynamic experiments where the meta-Grammar has the advantage of being able to evolve to a new target with large evolutionary steps. This is due to its dual-chromosomal structure, where both the grammar and the solution are evolved simultaneously and favourable biases are quickly built upon.

Among the static experiments, the meta-Grammar is able to hold its own with t-tests highlighting there was no significant difference between the methods for evolving the static constant 50. In the logistic equation the meta-Grammar achieved a better fitness in one out of three values for α, and only marginally underperformed at one setting. When considering these results, it should also be recalled that the system was operating with half the population size compared to that of the previous chapter. This demonstrates that efficiency gains can be brought about in the search without increasing utilisation of technical resources.

One of the interesting features of these experiments was the high rate of evolution produced by the meta-Grammar. In all the problem instances, the meta-Grammar began the early generations with a far inferior fitness due to the larger search space presented by the dual chromosome structure. However, over a small number of generations this disadvantage is quickly overcome and fitnesses are attained that are comparable to the combination grammar and its smaller search space, highlighting the utility of the meta-Grammar method.

6.3 Analysis of $(GE)^2$ Using Digit Concatenation with Expressions

Section 6.2 explored the use of a meta-Grammar incorporating pure Digit Concatenation and found that it had advantages over the combination grammar. This section continues the exploration of meta-Grammars by using a meta-Grammar that allows for the creation of expressions incorporating numbers created through Digit Concatenation, utilising the best-explored mechanism from the previous chapter. This method was found to produce significantly better fitnesses over pure Digit Concatenation in Section 5.5.

6.3.1 Experimental Approach

The problems tackled in this section are the same as the previous one. This section, however, uses a meta-Grammar that allows the formation of expressions with the numbers created through Digit Concatenation. Again the same problems are tackled to allow comparisons: finding a static constant and finding dynamic real constants. In this section, the results from experiments using the Digit Concatenation meta-Grammar with expressions are graphed against the results from the previous section, along with the results from a standard GE grammar with expressions and Digit Concatenation from Section 5.5. The new grammar used in this section is shown below.

```
<g> ::= "<SlnExpr> ::= " <catExp>
        "<SlnCat> ::= " <cat>
        "<SlnDigit> ::=" <digits>
        "<SlnOp> ::=" <ops>
<catExp> ::= <catExp> "<SlnOp>" <catExp> | <catRs>
<catRs> ::= <catRt> "|" <catRs> | <catRt>
<catRt> ::= "<SlnCat>" | "<SlnCat>"."<SlnCat>"
<cat> ::= <catT> "|" <cat> | <catT>
<catT> ::= "<SlnDigit>"<catT> | "<SlnDigit>" | <digits>
<digits> ::= <digitT> "|" <digits> | <digitT>
<ops> ::= <op> "|" <ops> | <op>
<op> ::= + | - | * | /
<digitT> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

6.3.2 Results

For every problem instance, the parameters used and number of runs conducted were the same as in Section 6.2.2.

Finding a Static Constant

The results presented in Figure 6.6 display a comparison of the average best fitness of each of the grammars over the 30 runs.

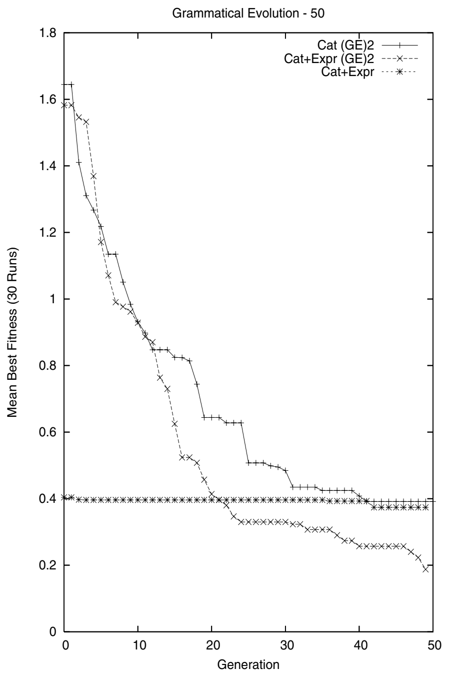
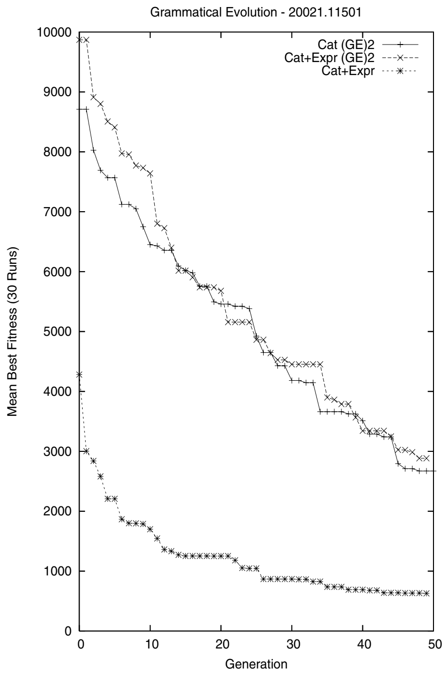


Fig. 6.6 Plot of the mean best fitness values for each constant generation method on the static constant problem instance

The results presented for $(GE)^2$ with the ability to form expressions exhibit similar behaviour to that of the meta-Grammar seen in Section 6.2. However, here an even greater rate of evolution is presented as the generations progress. This strong rate of evolution surpasses standard Cat+Expr average fitness by generation 22, resulting in a final average fitness of 0.187667, versus 0.391333 and 0.373938 for pure Cat $(GE)^2$ and standard Cat+Expr respectively. A t-test and a bootstrap t-test reveal that there is no significant difference in the results by the final generation. In examining the number of runs that evolved the target exactly, $(GE)^2$ with expressions evolved the correct solution 60% of the runs, with Cat $(GE)^2$ on 47% and standard Cat+Expr on 43%.

Fig. 6.7 Plot of the mean best fitness values for each constant generation method on the complex static constant problem instance



In Figure 6.7 the results for evolving 20021.11501 are presented. Here, neither meta-Grammar method is seen to achieve the level of fitness attained by the standard GE grammar. In this case, both meta-Grammars also displaying a similar rate of evolution with neither having a clear edge by the final generation. The difficulty experienced by the meta-Grammar method is likely down to the larger search space it must navigate, due to the dual chromosome structure, in order to improve its fitness. However, it should be noted that the rate of evolution for the meta-Grammar method, does again, continue at a steady pace right to the final generation. Comparatively, evolution appears to stagnate towards the latter generations of the standard GE Cat+Expr method.

Finding Dynamic Real Constants

Figure 6.8 displays a graph of the average best fitnesses for the dynamic Markov problem where the target changed every 10th generation from 192.47 to 71.84, 71.83, 173.59, and 192.47. Here the standard GE method clearly outperforms the two meta-Grammars with the meta-Grammar method using pure Cat outperforming the Cat+Expr though without any statistical significance under t-test and bootstrap t-tests.

Figure 6.9 displays the graph for the oscillating target problem. Under this experiment, the meta-Grammars perform more competitively against the

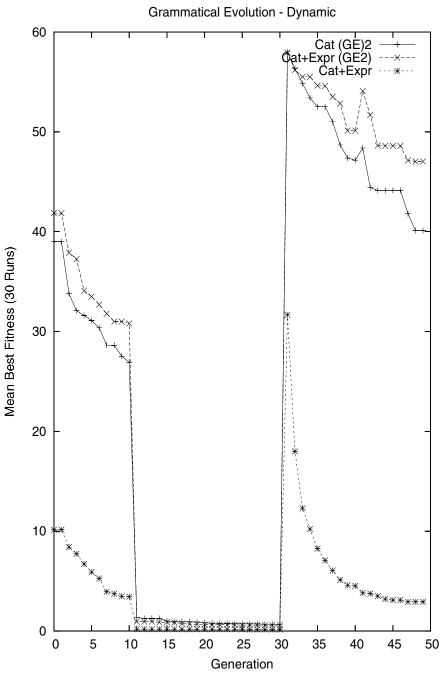


Fig. 6.8 Plot of the mean best fitness values for each constant generation method on the first dynamic problem instance

standard GE approach. For the first generation at each target, the standard GE approach produces better results on average. However, as seen in Section 6.2, this advantage is eroded by $(GE)^2$'s greater rate of evolution. When the target reaches 192.47 for the second time, the Cat+Exp$(GE)^2$ provides the best average performance by the 30^{th} generation. However, when the target returns to 192.47 for the final time, it is the pure Cat$(GE)^2$ that presents the best results that are statistically superior to the Cat+Exp$(GE)^2$ method under t-test and bootstrap t-tests.

Discussion

This section presented a number of interesting results. First, in evolving the static number 50, the Cat+Expr$(GE)^2$ method provided the best results, displaying a strong rate of evolution enabling it to outperform the the standard GE method even with its larger search space and a reduced population size. Evolving 20021.11501 proved to be equally difficult for both meta-Grammars in comparison to standard GE.

In the dynamic experiment, the standard GE method outperforms both the meta-Grammar methods strongly. Here, the smaller search space likely allowed it to gain an upper hand in the transition from 173.59 to 192.47 as it suffers from very little loss of fitness on this change of targets. This performance, however, does not continue in the oscillating experiment where the

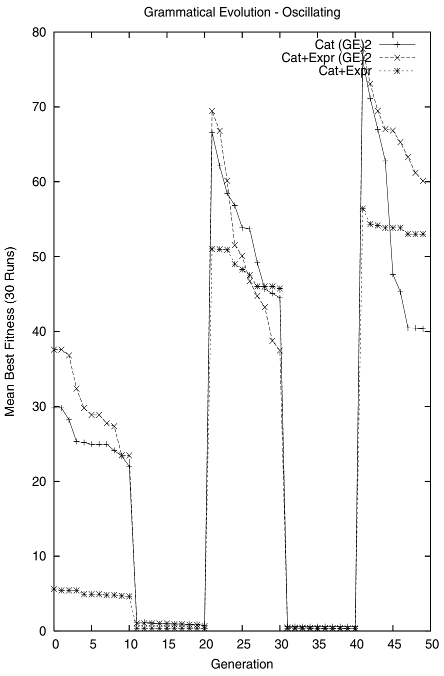


Fig. 6.9 Plot of the mean best fitness values for each constant generation method on the oscillating dynamic problem instance

standard GE approach suffers from a weak rate of evolution when the target transitions to 192.47. The meta-Grammar approach, on the other hand, again displays a steady rate of evolution, with one of the two meta-Grammar methods performing better by the end of each of the final transitions to 192.47. Interestingly, the meta-Grammar that uses the pure Digit Concatenation method for constant creation performed better than the method which incorporated expressions with a statistical significance.

6.4 Conclusions

This chapter uncovered some of the potential of the meta-Grammar approach and $(GE)^2$ for dynamic environments. The addition of the extra layer of adaptability allowed $(GE)^2$ to take large evolutionary steps when a change in the target occurred. Across all problems, a high rate of evolution was observed right through to the latter generations. Evidence was also observed, though anecdotal, of a representational-based memory forming in the solution-grammar chromosome for the Deterministic oscillating problem.

In examining the basic functionality of $(GE)^2$, it was seen to perform comparatively well on the static experiments, presenting a higher ratio of exact matches for the target 50 though with a slightly inferior performance on the large number. For the dynamic experiments, $(GE)^2$ demonstrated a higher rate of adaptability in most experiments. In the Deterministic oscillating problems, $(GE)^2$ was able to recall previously successful grammars when a target was encountered again, offering evidence of an implicit memory.

It should also be highlighted that the addition of the meta-Grammar represents an efficient improvement in GE's search capability. Even though the population size was reduced by 50%, $(GE)^2$ was able to perform comparatively with standard GE and present improvements in performance on dynamic problems. This is particularly significant as the 50% reduction implies that there is half the feedback back to the population from fitness evaluations while the search space itself has grown with the addition of the extra chromosome.