ABSTRACT
Grammars provide a convenient and powerful mechanism to define the space of possible solutions for a range of problems. While recent work has shed light on the matters of initialisation and grammar design with respect to grammatical evolution (GE), their impact on other methods, such as random search and context-free grammar genetic programming (CFG-GP), is largely unknown. This paper examines GE, random search and CFG-GP on benchmark problems using different initialisation routines and grammar designs. Results suggest that CFG-GP is less sensitive to initialisation and grammar design than both GE and random search: we also demonstrate that observed cases of poor performance by CFG-GP are managed through simple adjustment of tuning parameters. We conclude that CFG-GP is a strong base from which to conduct grammar-guided evolutionary search, and that future work should focus on understanding the parameter space of CFG-GP for better application.

CCS CONCEPTS
• Computing methodologies → Genetic programming; Supervised learning by regression.

KEYWORDS
genetic programming, grammars, grammar design, initialisation

1 INTRODUCTION
Grammars provide a convenient and powerful mechanism to define the space of possible solutions for a range of problems, and the incorporation of grammars into evolutionary computation has a history spanning over three decades. Most research into the use of grammars in evolution has focused upon grammatical evolution (GE) [17]. While GE has been applied to a range of problems from regression to evolutionary design and hyperheuristics, a significant portion of GE research has focused on understanding the interaction between its linear representation, genotype-to-phenotype mapping, and the polymorphic treatment of codons within its mapping process. Previous work has questioned the utility of GE’s linear representation; specifically, it is argued that the linear representation leads to low locality in search, and poor performance characteristics that resemble random search on some problems [15, 19, 20]. Understandably, the response to these criticisms of GE has been an increase in work to understand the underlying characteristics of GE search, and to design extensions to GE that improve its search characteristics. Some of this work has explored modifications of the linear representation so that it preserves structural information about the grammar during evolution [1, 4, 5]. Other work has focused on the initialisation process used in GE, and the design of the grammars that are used to describe the problem, and the resulting extensions to GE demonstrate improved performance on a range of benchmark problems [9, 10]. In this latter body of work, the improvements to GE are typically examined without comparison to other grammar-based methods, so the relative merits and utility of these improvements remaining largely unknown. The goal of this paper is to take some of the results of previous work and compare them to context-free grammar genetic programming (CFG-GP [18]) and random search that has been subjected to the same alterations of initialisation and grammar design. After testing on a range of benchmark problems, results suggest that CFG-GP is largely insensitive to the initialisation method used and is able to adequately recover from a poor initial population. The results align with previous work in that, for several of the examined problems, GE’s performance is roughly equivalent to random search using the same initialisation routines and grammar designs. Additionally, on the two examples where CFG-GP performs relatively poorly, it is shown that potential limitations in the behaviour of CFG-GP may be overcome through more effective parameter tuning, rather than relying upon elaborate modification of the grammar to match the representation. The findings of this paper suggest that future work should place greater emphasis on both the understanding and application of CFG-GP, and that the design of grammars should focus on more effective expression of the problem itself in order to make good solutions easier to discover.

2 RELATED WORK
The use of grammars in evolutionary computation has a long-established track record, and previous work provides a thorough review of this research [7]. More recent work on GGECC has examined potential limitations of grammatical evolution, focusing on
locality and problems with the genotype-to-phenotype mapping. Indeed, some work has gone so far as to suggest that GE’s behaviour on many problems is roughly equivalent to random search. In response to this criticism, new flavours of grammatical evolution have been developed, aimed at creating more direct and structure preserving representations [1, 5]. The resulting solutions incorporate derivation tree information into the representation, creating derivatives of GE that more closely resemble CFG-GP. Other work has provided greater understanding of the mapping process of linear genotypes and has lead to developments in grammar design and initialisation to improve the performance of standard GE [9, 10].

2.1 Initialisation
The limitations of direct initialisation of genotypes was identified early in the history of grammatical evolution. Subsequent work in GE focused on developing initialisation methods operating in the phenotype space and back-fitted the resulting derivations into a linear representation [16]. This ‘sensible’ initialisation essentially re-implemented the ramped-half-and-half method from standard genetic programming into GE, and presented the first work that begins to blur the separation of genotype and phenotype in GE.1

Following this work on sensible initialisation, work in GE has examined the use of probabilistic tree creation (PTC) methods for initialisation. Like sensible initialisation, PTC works in the phenotype (i.e., derivation tree) space and in the context of GE the resulting solution is then back-mapped into a linear representation. In PTC, the leaves of a tree are developed in a more breadth-like manner than the depth-first approach typically used in genetic programming [6]. Results in grammatical evolution have suggested that a strongly-typed version of PTC, named PTC2, can significantly improve the performance of GE [9].

2.2 Grammar Design
The purpose of a grammar is to define the structure and syntactical properties of the language used to express solutions to a given problem. Ideally, grammar design should focus on efficient and elegant expression of language, and preserving modularity in solutions. From the context of evolutionary search, the design of the grammar should be agnostic to the representation used to search the space of solutions. However, it was identified early in GE research that the design of the grammar itself plays a significant role in the effectiveness of GE on a given problem [8]. The polymorphic nature of the genotype-to-phenotype mapping in GE means that careful design of grammar productions is required to avoid search bias. Indeed, even small changes in grammar design can have significant impact on search, making comparisons of results on the same problem across different papers difficult [14].

To reduce the mismatch between linear representations and grammar productions, recent work has suggested the following steps: Balancing; Unlinking; Eliminating non-terminals; Removing grammar biases; Prefix notation; and Compromise grammars [10]. In previous work, grammars resulting from these modifications are labelled g1-g6, with g0 denoting the unmodified grammar [10]. We adopt this notation in this paper.

3 SETUP
Previous work has examined GE’s behaviour on a number of benchmark problems using a range of grammar designs and initialisation methods [10]. To provide a more complete the picture for grammar-guided evolutionary computation in general, we repeat a number of these experiments, but this time include random search (where a initialisation method is run for a given number of times and the best solution found in all of these samples is returned) and CFG-GP. We reuse problems and algorithm settings from previous work to provide a fair comparison [10, 20]. We also examine the Santa Fe Tile problem, as it has been the subject of several examples of previous work on grammar design [3, 11, 13, 14]. We examined four of these grammars from previous work (g0-g3), and in addition, we designed a novel grammar (g4) for this problem that was informed by inspection of the other grammars and the resulting solutions that were evolved through their use. This final grammar was used to demonstrate the benefits of grammar design where the emphasis is on clearer and more effective definition of the language, rather than modification of the grammar to suit algorithmic requirements. The source code used in all experiments is available online.2

4 RESULTS
All results in Figures are presented with 95% confidence intervals around the relevant statistic as shaded regions.

---

1 Although it is essentially a grammar-aware version of ramped-half-and-half initialisation, we use the name ‘sensible initialisation’ throughout this paper to remain consistent with previous work [9, 16].

2 https://github.com/grantdick/libgges
The results of experiments on the Keijzer-6 problem are shown in Figure 1. Confirming the results of previous work, GE is particularly sensitive to both initialisation and grammar design on this problem. Interestingly, the behaviour of random search is characteristically similar to that of GE, while the behaviour of CFG-GP on this problem appears to be largely unaffected by the choice of initialisation. CFG-GP also appears to be somewhat more robust to changes to the grammar than both GE and random search.

The results of experiments on the Shape problem are shown in Figure 2. The results here present an interesting behaviour for CFG-GP, which struggles to find solutions when using the g5 and g6 grammars. This appears to be an interaction between the depth limiting used in CFG-GP and depths of the derivation trees that these grammars need to describe highly fit solutions. In contrast to CFG-GP, initialisation seems to play an important role in the behaviour of random search and GE on this problem.

Finally, Figure 3 shows the evolution of fitness on the Santa Fe Trail problem. Again, an interesting observation is made of the performance of CFG-GP on this problem: when using the g0 grammar, search appears to stagnate quickly resulting in poor fitness. A similar observation is made for random search on this problem. As discussed in the next section, this again appears to be a consequence of parameter choice in CFG-GP, this time in the maximum depth of the trees generated in subtree mutation. For the remaining grammars, there is a clear advantage to using CFG-GP over the other two methods. Most interesting, however, is the g4 grammar that was developed specifically for this paper: this grammar was not designed or adapted to better align with any specific representation, but rather to bias search to encourage the discovery of better solutions. As can be seen, using this grammar leads to effective search regardless of chosen method.

5 DISCUSSION

While CFG-GP appears to offer the overall most stable and effective search performance, its behaviour on the Shape problem (using grammars g5 and g6) and the Santa Fe Trail (using grammar g0) is poor. However, an analysis of the grammars, and the requirements of these grammars in terms of the derivation trees required to produce good solutions yields useful information that can help the performance of CFG-GP on these problems. In the case of the Shape problem, examples of optimal solutions are provided in previous work [12]. The derivation trees for these optimal solutions have a depth of around 20, which is outside the depth limits imposed on CFG-GP in this work. Similarly, for the Santa Fe Trail, the production that ultimately produces if (food_ahead()) ... expressions requires a minimum derivation tree depth of 5 to be fully terminated. Given that CFG-GP operated with a maximum mutation depth of 4, this meant that no such expressions could appear in solutions through mutation, which greatly inhibited search quality on this problem using this grammar.

Given the insights developed from examining the grammars for the Shape and Santa Fe Trail problems, a simple modification to CFG-GP was proposed. First, depth limiting was removed, and then a small modification was made to subtree mutation: rather than setting a hard limit, maximum mutation depth was computed as the larger of the either the depth of the largest production for the non-terminal being mutated, or the depth of the subtree that mutation is replacing. The results of this modified configuration of CFG-GP are shown in Figures 4 and 5. All results are presented using the g6 grammar for Shape and the g0 grammar for Santa Fe Trail: these modifications to CFG-GP have markedly improved performance.
When tested on several benchmark problems, it is shown that the parameters, rather than resorting to more elaborate modifications, may be able to adapt to more problems through simple tuning of its structure and domain knowledge into evolutionary search. Recent work suggests that grammatical evolution, currently the most popular form of grammar-guided evolutionary computation, requires extensive tuning to problems through careful grammar design and is heavily reliant upon accurate initialisation of the population for good performance. This paper attempts to shed light of the relative merit of these modifications to GE by repeating similar experiments using random search and CFG-GP as a reference point. Finally, the results presented here also suggest that CFG-GP is less sensitive to the effects of initialisation and the design of the grammar, rather than resorting to more elaborate modifications to the grammar.

The results presented here suggest several areas of future work. Clearly, the parameter space of CFG-GP is not as fully-understood as it could be, so future work should develop a better understanding of how the parameters of CFG-GP impact its behaviour. Likewise, there would be significant benefit from future work developing new methods for grammar design that emphasise more effective expression of the problem being searched rather than focusing on distorting the grammar to fit the chosen representation.

REFERENCES
[1] Alberto Bartoli, Mauro Castelli, and Eric Medvet. 2020. Weighted Hierarchical Grammatical Evolution. IEEE Transactions on Cybernetics 50, 2 (2020), 476–488.
[2] Grant Dick. 2017. Interval Arithmetic and Interval-Aware Operators for Genetic Programming. arXiv arXiv:submit/4263726
[3] Robin Harper. 2010. GE, explosive grammars and the lasting legacy of bad initialisation. In Proceedings of the 2010 IEEE Congress on Evolutionary Computation. IEEE, Barcelona, Spain, 1–8.
[4] Robin Harper and Alan Blair. 2005. A structure preserving crossover in grammatical evolution. In Proceedings of the 2005 IEEE Congress on Evolutionary Computation, Vol. 3. IEEE, Edinburgh, Scotland, 2537–2544.
[5] Nuno Lourenço, Francisco B Pereira, and Ernesto Costa. 2016. Unveiling the properties of structured grammatical evolution. Genetic Programming and Evolvable Machines 17, 3 (2016), 251–289.
[6] Sean Luke. 2000. Two fast tree-creation algorithms for genetic programming. IEEE Transactions on Evolutionary Computation 4, 3 (2000), 274–283.
[7] Robert I Mekay, Nguyen Huu Hoai, Peter Alexander Whigham, Yin Shan, and Michael O’Neill. 2010. Grammar-based genetic programming: a survey. Genetic Programming and Evolvable Machines 11, 3-4 (2010), 365–396.
[8] Miguel Nicolau. 2004. Automatic grammar complexity reduction in grammatical evolution. In The 3rd Grammatical Evolution Workshop: A workshop of the 2004 Genetic and Evolutionary Computation Conference (GECCO-2004), Seattle, Washington, USA, 26-30 June 2004. Springer, Cham, 12 pages.
[9] Miguel Nicolau. 2017. Understanding grammatical evolution: initialisation. Genetic Programming and Evolvable Machines 18, 4 (2017), 467–507.
[10] Miguel Nicolau and Alexandros Agapitos. 2018. Understanding Grammatical Evolution: Grammar Design. In Handbook of Grammatical Evolution, Conor Ryan, Michael O’Neill, and JJ Collins (Eds.). Springer, Cham, 23–53.
[11] Miguel Nicolau, Michael O’Neill, and Anthony Brabazon. 2012. Termination in grammatical evolution: Grammar design, wrapping, and tails. In Proceedings of the 2012 IEEE Congress on Evolutionary Computation. IEEE, Brisbane, Australia, 8 pages.
[12] Michael O’Neill, John Mark Swafford, James McDermott, Jonathan Byrne, Anthony Brabazon, Elizabeth Shotton, Ciaran McNally, and Martin Hemberg. 2009. Shape grammars and grammatical evolution for evolutionary design. In Proceedings of the 2009 Conference on Genetic and Evolutionary Computation. ACM, Montréal, Canada, 1035–1042.
[13] Michael O’Neill and Conor Ryan. 1999. Evolving multi-line compilable C programs. In Proceedings of the 2nd European Workshop on Genetic Programming. Springer, Göteborg, Sweden, 83–92.
[14] Denis Robilliard, Sebastien Mahler, Dominique Verhaeghe, and Cyril Fonlupt. 2005. Santa Fe Trail Hazards. In Artificial Evolution, 7th International Conference, Evolution Artificielle, EA 2005, El-Ghazali Talbi, Pierre Liardet, Pierre Collet, Evelyne Lutton, and Marc Schoenauer (Eds.). Springer, Lille, France, 1–12.
[15] Franz Rothlauf and Marie Oetzel. 2006. On the Locality of Grammatical Evolution. In Proceedings of the 6th European Conference on Genetic Programming (Lecture Notes in Computer Science, Vol. 3985), Pierre Collet, Marco Tomassini, Marc Ebner, Steven Gustafson, and Aniko Ekart (Eds.). Springer, Budapest, Hungary, 320–330.
[16] Conor Ryan and R. Muhammad Atif Azad. 2003. Sensible Initialisation in Grammatical Evolution. In GECCO 2003: Proceedings of the Bird of a Feather Workshops, Genetic and Evolutionary Computation Conference, Alwyn M. Barry (Ed.). AAAI, Chicago, IL, 142–145.
[17] Conor Ryan, Michael O’Neill, and JJ Collins. 2018. Introduction to 20 Years of Grammatical Evolution. In Handbook of Grammatical Evolution. Springer, Cham, Switzerland, 1–21.
[18] Peter A Whigham. 1995. Grammatically-based Genetic Programming. In Proceedings of the Workshop on Genetic Programming: From Theory to Real-World Applications. University of Rochester, USA, 33–41.
[19] Peter A Whigham, Grant Dick, and James Maclaurin. 2017. On the mapping of genotype to phenotype in evolutionary algorithms. Genetic Programming and Evolvable Machines 18, 3 (2017), 353–361.
[20] Peter A Whigham, Grant Dick, James Maclaurin, and Caitlin A Owen. 2015. Examining the best of both worlds of grammatical evolution. In Proceedings of the 2015 Conference on Genetic and Evolutionary Computation. ACM, Madrid, Spain, 1111–1118.