Feature Encapsulation by Stages Using Grammatical Evolution

Darian Reyes Fernández de Bulnes
University of Limerick
Limerick, Ireland
Darian.ReyesFernandezDeBulnes@ul.ie

Allan de Lima
University of Limerick
Limerick, Ireland
Allan.DeLima@ul.ie

Aidan Murphy
University College Dublin
Dublin, Ireland
aidan.murphy@ucd.ie

Douglas Mota Dias
University of Limerick
Limerick, Ireland
Douglas.MotaDias@ul.ie

Conor Ryan
University of Limerick
Limerick, Ireland
Conor.Ryan@ul.ie

ABSTRACT

This paper introduces a novel mechanism, Feature Encapsulation by Stages (FES), to encapsulate and transfer features as knowledge in a staged manner within the evolutionary process. Encapsulation happens via input space expansion in one or more stages by adding the best-of-run individual as an additional input. This input space expansion is managed by augmenting the grammar. We study the feasibility of dynamically modifying the grammar and re-initialising the population to make way for new individuals which quickly evolve to a better fitness level. Five different approaches to stage management are examined. In addition, three different selection processes, namely, Tournament, Lexicase and Lexi^2, are used to investigate which is best suited to use with our encapsulation procedure. We benchmark our procedure on two problem domains, Boolean and Classification, and demonstrate these staging strategies lead to significantly better results. Statistical tests show our FES outperforms the standard baseline in all Boolean problems, with a 4-stage version performing best, obtaining significant differences in all Boolean problems.

CCS CONCEPTS

• **Computing methodologies** → **Genetic programming**; *Supervised learning*; Discrete space search.

KEYWORDS

feature encapsulation, grammatical evolution, multi-target, multi-output

ACM Reference Format:
Darian Reyes Fernández de Bulnes, Allan de Lima, Aidan Murphy, Douglas Mota Dias, and Conor Ryan. 2024. Feature Encapsulation by Stages Using Grammatical Evolution. In *Genetic and Evolutionary Computation Conference (GECCO '24 Companion), July 14–18, 2024, Melbourne, VIC, Australia.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3638530.3654097

GECCO '24 Companion, July 14–18, 2024, Melbourne, VIC, Australia

ACM ISBN 979-8-4007-0495-6/24/07.
https://doi.org/10.1145/3638530.3654097

1 INTRODUCTION

Grammatical Evolution (GE) [12] is a grammar-based variant of Genetic Programming (GP) [5] that leverages the power and flexibility of grammars to define and constrain the syntax of potential solutions. Inspired by this, we propose a novel evolutionary strategy that uses GE grammars to extend the input space of the problems by adding one or more new inputs. These inputs are the outputs of the best-so-far discovered individual. Once added to the grammar, the population is reinitialised, and the next stage of evolution starts with a new population. We hypothesise that dividing the GE evolutionary process into stages, with the feature encapsulation and the re-initialisation of the population using new grammars, leads to significantly better results as they mitigate ripple effects while limiting the destruction of many grammar updates.

The subsequent sections of this paper are structured as follows: Section 2 establishes connections between this study and the prior research literature. Section 3 presents an in-depth exposition of the proposed methodology. Following that, Section 4 details the experimental results, while Section 5 contains the discussion. Finally, the paper is summarised, and potential avenues for future research are outlined in Section 6.

2 RELATED WORK

Koza et al. [6] were among the first researchers in the field of GP to understand the feasibility of substructure reuse. Although that research is specific to the circuit problem domain, it is based on the idea that a solution to a problem needs several uses of the same sophisticated submodules that must be reusable. Again, using GP, Keijzer et al. [4] introduced the notion of Run Transferable Libraries, a mechanism to pass knowledge acquired in one GP run to another instead of generations. This was mainly intended for high scalability problems. Within GE, Murphy and Ryan [10] proposed a way for GE mapping to identify valuable modules that make up a solution and add them to the grammar.

Apart from the field of GP, stacked Machine Learning (ML) methods for Classification or regression problems with multiple outputs have been introduced in the last two decades. Godbole and Sarawagi [3] demonstrate the importance of combining features for multi-label Classification text problems indicating relationships between classes. In [14], a strategy based on input space expansion is proposed for multi-target regression problems. Multi-output regression problems were also analysed in [8].

3 METHODS

The transfer of prior knowledge via feature encapsulation and the re-initialisation of the population, destroying previously evolved individuals, is done in stages. Unlike the procedure proposed in [9], where the best solution is a combination of the best individual of each period creating a large structure, our methods maintain a hall of fame, saving the best individual by generations across the stages without the need to artificially create a phenotype at the end of the evolutionary process. Furthermore, unlike [14], where there are only two stages managed offline, our proposal makes the number of stages flexible and does so automatically within the same evolutionary process. In our approach, the inputs are expanded only once in the entire evolution, and there is no increase in training cases. Therefore, the input space expansion does not represent a significant increase in the computational cost.

We used the following approaches during experimentation:

- 2 stages 25%: we use two stages, and the second commences after 25% of the generations have passed;
- 2 stages 50%: the same as above, but the second stage commences after 50% of the generations have passed;
- 3 stages: we use three stages. The second commences after one-third of the generations have passed, while the third commences after two-thirds have passed;
- 4 stages: we use four stages. The second stage commences after 25% of the generations have passed, the third stage after 50% of the generations, and the last one after 75% of the generations;
- multi-stage: we use a dynamic number of stages, where a new stage commences every time the best fitness value changes from generation to generation.

To avoid excessive triggering of stages with the Multi-stage approach in early generations, we always wait until at least 25% of the total generations have advanced after the previous stage ended before a new stage can begin. Given that constraint, a multi-stage setup can consist of one to four stages. These five methods were studied with three different selection methods: Tournament, Lexicase [13], and Lexi^2 [2].

4 EXPERIMENTAL SETTING AND RESULTS

The fitness function for all the problems studied is the sum of differences in the output vectors divided by the total number of cases. The baseline for comparing these approaches is the standard evolutionary process.

We conduct experiments on three Boolean problems with different levels of complexity, namely, Parity 3, Multiplier 3×3, and Multiplexer 11. These are important in various electronic engineering applications, such as memory devices and communication systems. We also address the Classification problems of Heart disease (one output) and Car evaluation (two outputs). For this problem domain, we divide the total cases into two subsets: 70 % for training and 30 % for testing.

In all variants of our system, when the second stage is triggered, the standard grammar for the problem in question is automatically replaced by an extended grammar (Extended Grammar Version 1 and Extended Grammar Version 2) that contains the encapsulated features. The experiments are conducted with the GRAPE library [1], and the GE's parameters are presented in Table 1. In the table, the population size values refer to Parity 3, Multiplier 3×3, and Multiplexer 11, respectively.

Table 1: Experimental parameters.

Parameter type	Parameter value
Population size	30/400/4000
Number of generations	200
Initialisation method	Sensible [11]
Mutation probability	0.01
Crossover probability	0.80

4.1 Results

Firstly, Table 2 summarises the results for the Boolean problems and the three selection processes. These results show the total hits out of 30. One hit means that the fitness of the best-of-run individual is 0, which completely solves the problem.

Figure 1 shows three plots for the problem Multiplier 3×3 with Extended Grammar Version 2 using Lexi^2 selection: (a) the fitness of the best individual across generations, (b) the average fitness of the population by generations, and (c) the structural diversity [2] (different phenotype structures in the population) by generations for standard Lexi^2 and two stage-based Lexi^2 approaches (4 stages and multi-stage).

Table 3 shows the results for the Classification problems Heart disease and Car evaluation for the unseen test subsets. The best individual in the last generation is evaluated on the unseen test subset.

5 DISCUSSION

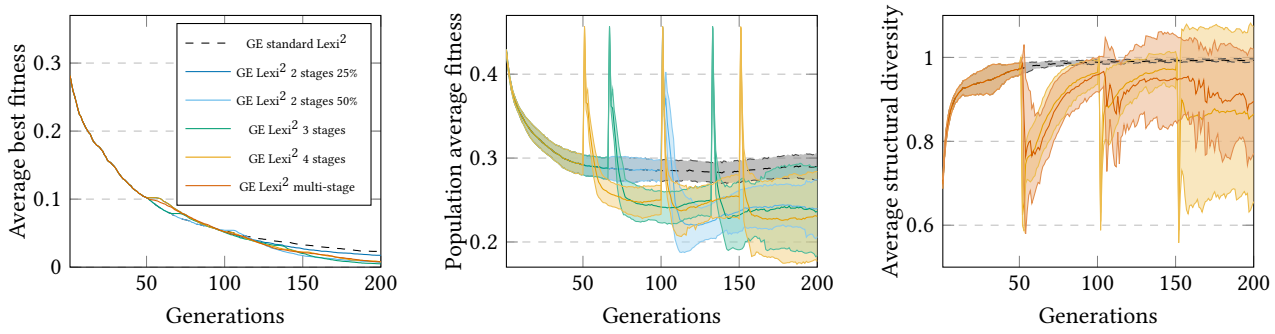
We conducted experiments on Boolean problems to evaluate the effectiveness of the proposed approaches in achieving zero errors, which corresponds to successfully solving these problems. The results are presented in Table 2. The observed trend remains consistent across a range of problems, from the Parity 3 problem (with a smaller population size of 30) to the Multiplier 3×3 problem (which requires a larger population size of 4000). Notably, when employing Tournament selection, the number of successful outcomes is low, indicating its unsuitability for these specific problem types.

Lexicase and Lexi^2 fare better, with Lexi^2 tending to perform slightly better. For instance, in the case of the Multiplier 3×3 problem, standard Lexicase fails to achieve any successful outcomes, whereas all stage-based approaches have successful outcomes, particularly the four-stage approach utilising Extended Grammar Version 2, which achieved 17 successful runs.

To conclude this experimental analysis of the evolutionary processes, results for the Classification problems of Heart disease and Car evaluation are shown in Table 3, measuring the error on the test subsets. While no single method outperforms all others, it can be observed that some stage-based approaches improve the standard baseline, in particular, 4 stages with Extended Grammar Version 1, which outperforms the test error in five out of six experiments presented in the Table.

Table 2: Successful runs (out of 30). Results for stages approaches include both grammars (Version 1 and Version 2). The best result for each row is in bold.

Problem	Selection	Standard	2 stages 25%		2 stages 50%		3 stages		4 stages		Multi-stage	
Parity 3	Tournament	0	0	0	0	0	0	0	0	0	**1**	0
	Lexicase	7	20	16	20	16	22	22	20	**23**	18	10
	Lexi^2	5	**9**	6	**9**	8	6	**9**	**9**	4	6	4
Multiplier 3 × 3	Tournament	0	0	0	0	0	0	0	0	0	0	0
	Lexicase	0	1	5	3	7	9	9	7	**17**	3	13
	Lexi^2	2	4	2	8	7	9	**15**	10	**15**	6	**15**
Multiplexer 11	Tournament	0	0	0	0	0	0	0	0	0	0	0
	Lexicase	11	14	13	21	**22**	19	19	15	21	15	13
	Lexi^2	21	13	15	**22**	21	17	**22**	15	18	10	17



(a) Average best fitness **(b) Population average fitness** **(c) Average structural diversity**

Figure 1: Convergence plots for Multiplier 3 × 3 using Lexi^2 selection and Extended Grammar Version 2.

Table 3: Average fitness on the test set for two Classification problems. Results for stages approaches include both extended grammars (Version 1 and Version 2). The best result for each row is in bold.

Problem	Selection	Standard	2 stages 25%		2 stages 50%		3 stages		4 stages		Multi-stage	
Heart disease	Tournament	0.205	0.209	0.209	0.200	0.198	0.198	0.201	0.194	**0.193**	0.208	0.212
	Lexicase	0.205	0.203	0.215	**0.200**	0.206	0.208	0.203	0.201	0.200	0.206	0.207
	Lexi^2	**0.196**	0.112	0.206	0.210	0.206	**0.196**	0.198	0.204	0.198	0.202	0.209
Car evaluation	Tournament	0.139	0.141	0.152	0.130	0.135	**0.126**	0.146	0.133	0.157	0.131	0.157
	Lexicase	0.106	0.108	0.111	0.099	0.100	0.106	0.099	0.098	0.101	**0.097**	0.101
	Lexi^2	0.106	0.099	0.107	0.100	0.102	0.096	0.096	0.096	0.096	**0.093**	0.101

5.1 Statistical tests

Friedman-Nemenyi non-parametric statistical tests were performed for the Boolean problems to check for significant differences between the mean population fitness in the final generation of our proposed methods and the baselines. With $\alpha = 0.05$, if the *p-value* is less than α, the alternative hypothesis (there are significant differences) is activated, and the null hypothesis H_0 (no significant differences) is rejected. A Nemenyi plot is created for each problem to check all pairwise test results at a glance. One example (Parity 3) is shown in Figure 2, and equivalent figures for all other problems assessed can be found in our supplementary material.

Each bold line measures the Critical Distance (CD), wrapping methods when H_0 is activated. Since the objective function for all problems is minimisation, in the Nemenyi rank, a higher value is better, ordering the methods from left to right in order of good performance in the Friedman tests. This example (Parity 5) shows that the standard approaches (baselines) are outperformed. We obtained similar results for the rest of the problems (8 out of 10), except for two problems (Adder 2+3 and Multiplier 3×3), where the standard Tournament shows competitive results without significant differences with the stage-based methods.

The best method, which obtains first place in the Nemenyi ranking on three problems (Parity 3, Parity 5, and Multiplexer 11), is Lexicase 4 stages with Extended Grammar Version 2. With these statistical results, the hypothesis stated at the beginning of the manuscript is confirmed; clearly, the best results are produced by

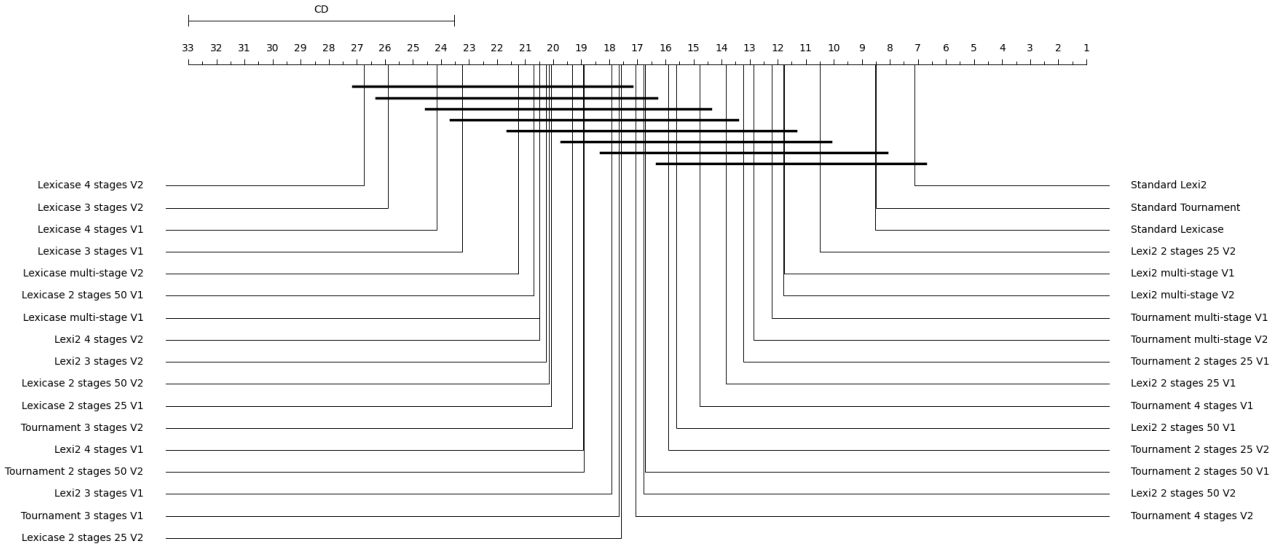


Figure 2: Friedman-Nemenyi non-parametric statistical test result for Parity 3.

stage-based methods. No significant difference exists for the unseen test data on studied classification problems.

6 CONCLUSIONS

We demonstrated the possibility of obtaining substantial performance improvement by dividing the evolutionary process into stages with the semantic encapsulation of features and the re-initialisation of the population. In the Boolean problems domain, the approach with four stages with Extended Grammar Version 2 is the best performing stage-based variant, obtaining significant differences. In Classification, a 4-stage version is also the best, outperforming the test error in five out of six experiments.

Although these proposals are presented in the context of Boolean and Classification problems and GE algorithm, they can also be applied to other problem families and evolutionary computation techniques. We plan future research with other benchmarks, such as symbolic regression employing ϵ-Lexicase selection [7].

7 ACKNOWLEDGMENTS

This publication has emanated from research conducted with the financial support of Science Foundation Ireland under Grant number 16/IA/4605.

REFERENCES

[1] Allan de Lima, Samuel Carvalho, Douglas Mota Dias, Enrique Naredo, Joseph P. Sullivan, and Conor Ryan. 2022. GRAPE: Grammatical Algorithms in Python for Evolution. *Signals* 3, 3 (2022), 642–663. https://doi.org/10.3390/signals3030039

[2] Allan de Lima, Samuel Carvalho, Douglas Mota Dias, Enrique Naredo, Joseph P. Sullivan, and Conor Ryan. 2022. Lexi^2: Lexicase Selection with Lexicographic Parsimony Pressure. In *Proceedings of the Genetic and Evolutionary Computation Conference* (Boston, Massachusetts) *(GECCO '22)*. Association for Computing Machinery, New York, NY, USA, 929–937. https://doi.org/10.1145/3512290.3528803

[3] Shantanu Godbole and Sunita Sarawagi. 2004. Discriminative Methods for Multi-labeled Classification. In *Advances in Knowledge Discovery and Data Mining*, Honghua Dai, Ramakrishnan Srikant, and Chengqi Zhang (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 22–30.

[4] Maarten Keijzer, Conor Ryan, and Mike Cattolico. 2004. Run Transferable Libraries - Learning Functional Bias in Problem Domains. In *Genetic and Evolutionary Computation - GECCO 2004*, Kalyanmoy Deb (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 531–542.

[5] John R. Koza. 1992. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA.

[6] John R. Koza, Forrest H. Bennett, David Andre, and Martin A. Keane. 1997. Reuse, parameterized reuse, and hierarchical reuse of substructures in evolving electrical circuits using genetic programming. In *Evolvable Systems: From Biology to Hardware*, Tetsuya Higuchi, Masaya Iwata, and Weixin Liu (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 312–326.

[7] William La Cava, Lee Spector, and Kourosh Danai. 2016. Epsilon-Lexicase Selection for Regression. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016* (Denver, Colorado, USA) *(GECCO '16)*. Association for Computing Machinery, New York, NY, USA, 741–748. https://doi.org/10.1145/2908812.2908898

[8] Haitao Liu, Jianfei Cai, and Yew-Soon Ong. 2018. Remarks on multi-output Gaussian process regression. *Knowledge-Based Systems* 144 (2018), 102–121. https://doi.org/10.1016/j.knosys.2017.12.034

[9] David Medernach, Jeannie Fitzgerald, R. Muhammad Atif Azad, and Conor Ryan. 2015. Wave: A Genetic Programming Approach to Divide and Conquer. In *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation* (Madrid, Spain) *(GECCO Companion '15)*. Association for Computing Machinery, New York, NY, USA, 1435–1436. https://doi.org/10.1145/2739482.2764659

[10] Aidan Murphy and Conor Ryan. 2020. Improving Module Identification and Use in Grammatical Evolution. In *2020 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, Glasgow, UK, 1–7. https://doi.org/10.1109/CEC48606.2020.9185571

[11] Conor Ryan and R Muhammad Atif Azad. 2003. Sensible initialisation in grammatical evolution. In *GECCO*. AAAI, Chicago, Illinois, USA, 142–145.

[12] Conor Ryan, JJ Collins, and Michael O. Neill. 1998. Grammatical evolution: Evolving programs for an arbitrary language. In *Genetic Programming. EuroGP (Lecture Notes in Computer Science)*, Wolfgang Banzhaf, Riccardo Poli, Marc Schoenauer, and Terence C. Fogarty (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 83–96.

[13] Lee Spector. 2012. Assessment of Problem Modality by Differential Performance of Lexicase Selection in Genetic Programming: A Preliminary Report. In *Proceedings of the 14th Annual Conference Companion on Genetic and Evolutionary Computation* (Philadelphia, Pennsylvania, USA) *(GECCO '12)*. Association for Computing Machinery, New York, NY, USA, 401–408. https://doi.org/10.1145/2330784.2330846

[14] Eleftherios Spyromitros-Xioufis, Grigorios Tsoumakas, William Groves, and Ioannis Vlahavas. 2016. Multi-target regression via input space expansion: treating targets as inputs. *Machine Learning* 104, 1 (2 2016), 55–98. https://doi.org/10.1007/s10994-016-5546-z