Inclusive Genetic Programming

Francesco Marchetti[0000−0003−4552−0467] and Edmondo Minisci[0000−0001−9951−8528]

Intelligent Computational Engineering Laboratory, University of Strathclyde, Glasgow, UK
{francesco.marchetti,edmondo.minisci}@strath.ac.uk

Abstract. The promotion and maintenance of the population diversity in a Genetic Programming (GP) algorithm was proved to be an important part of the evolutionary process. Such diversity maintenance improves the exploration capabilities of the GP algorithm, which as a consequence improves the quality of the found solutions by avoiding local optima. This paper aims to further investigate and prove the efficacy of a GP heuristic proposed in a previous work: the Inclusive Genetic Programming (IGP). Such heuristic can be classified as a niching technique, which performs the evolutionary operations like crossover, mutation and selection by considering the individuals belonging to different niches in order to maintain and exploit a certain degree of diversity in the population, instead of evolving the niches separately to find different local optima. A comparison between a standard formulation of GP and the IGP is carried out on nine different benchmarks coming from synthetic and real world data. The obtained results highlight how the greater diversity in the population, measured in terms of entropy, leads to better results on both training and test data, showing that an improvement on the generalization capabilities is also achieved.

Keywords: Genetic Programming · Population Diversity · Entropy · Benchmarks · Symbolic Regression

1 Introduction

Genetic Programming [12] is known to be a powerful approach to perform Symbolic Regression (SR) , capable of autonomously finding a symbolic expression, which explicitly models a distribution of observed data, starting from no previous knowledge of such data. In the past years it was used for several kinds of applications, from pure symbolic regression [18], to optimization [11] and control system design [10, 16, 21]. Despite this abundance of applications, GP still suffers from several issues that affect different aspects of a GP algorithm and undermine its performances. In this work, one of such issues is tackled, namely the promotion and maintenance of diversity in a GP population.
The importance of the population diversity in an evolutionary algorithm was treated by several publications in the past: in [5] diversity is addressed as a method to control exploration and exploitation during the evolutionary process,

describing also different diversity measures; in [20] a survey of methodologies for promoting diversity in evolutionary optimization is presented; while in [4] different diversity measures are analyzed.
This work aims to contribute to the landscape of publications about diversity maintenance and promotion by performing a deeper analysis of a heuristic proposed in a previous work [15]: the Inclusive Genetic Programming. This heuristic is based on a different formulation of the evolutionary operations, such as crossover, mutation and selection, aimed to promote and maintain the diversity in a GP population. These evolutionary operations are based on the partition of the population into different niches according to the genotypic diversity of the individuals. Therefore, such heuristic pertains to the class of niching techniques [14]. Traditionally, the individuals are divided according to their genotypic (individual structure) or phenotypic (fitness value) characteristics [19], although recently also a niching approach based on the computational time necessary to execute the individuals was suggested by De Vega et. al. [8]. Nonetheless, all these methods consist in dividing the population into different niches and evolve them separately to find different local optima. On the contrary, the approach proposed in this work aims to subdivide the population into niches and then take into account the individuals from different niches when performing the evolutionary operations so to have a flow of genes between the different niches. The concept of combining diverse individuals (e.g. individuals from different niches) rather than well performing ones was already explored in the past, for example by Aslam et. al. [2], although their approach is based on the phenotypic diversity of the individuals and did not considered a subdivision of the population into niches. Instead they classified the individuals based on a measure called Binary String Fitness Characterisation, originally introduced by Day and Nandi [6], and also explored the concept of good and bad diversity.
In this work, the aim of the flow of genes between the different niches is to preserve also more complex structures, which would otherwise be lost due to bloat control operators, and to exploit them in order to avoid losing well performing genes which are contained in them. It could be argued that bloat control operators could be removed for this purpose, but nonetheless, they are useful to avoid having too big individuals which would become of difficult interpretation for the user. Indeed, bloat control and the GP population diversity are related, but it was not aim of this work to investigate this relation. Some examples of this relation are presented by De Jong et. al. [7], where a multi-objective formulation of GP is used to avoid bloat and promote diversity; and by Alfaro-Cid et. al. [1] where several bloat control operators and their influence on the population diversity were analyzed.
The reminder of this paper is organized as follows: in Section 2 the Inclusive Evolutionary Process is described; in Section 3 the benchmarks used in this work, the settings of the algorithms employed in the comparison and the produced results are presented. Finally in Section 4, conclusions and future work directions are discussed.

2 Inclusive Evolutionary Process

In this Section, the components of the evolutionary process that is at the foundation of the IGP are described. Such evolutionary process is based on a modified version of the evolutionary strategy $\mu + \lambda$ [3], as described by Algorithm 1. The differences from the standard version consist in: 1) the creation of the niches at the beginning of the evolutionary process and every time after a new offspring is generated; 2) the use of the Inclusive Reproduction; 3) the use of the Inclusive Tournament.

Algorithm 1 Pseudocode of the Inclusive $\mu + \lambda$

```
Evaluate fitness of the individuals in the population
Update Hall of Fame of best individuals
while Generation < Maximum Generation Number do
    Create n niches according to the maximum and minimum length of the individuals in the population and allocate the individuals to their respective niche
    Perform Inclusive Reproduction to produce λ offspring
    Evaluate fitness of the individuals in the obtained offspring
    Update Hall of Fame of best individuals
    Create n niches according to the maximum and minimum length of the individuals considering both the parents and the offspring and allocate the individuals to their respective niche
    Perform Inclusive Tournament Selection to select μ new parents
end while
```

2.1 Niches Creation

At the core of the Inclusive Evolutionary Process, there is the niching creation mechanism. The niches are created in an evenly distributed manner (linearly divided) between the maximum and minimum length (number of nodes) of the individuals in the population, then the individuals are assigned to the respective niche according to their length. The same number of niches is kept during the evolutionary process, but their size (the interval of individuals lengths that they cover) and the amount of individuals inside them change at every generation. The variation of size of the niches allows for a shifting of the individuals between contiguous niches every time maximum and minimum lengths of the individuals in the population changes. Once the niches are created, both the reproduction and selection are performed considering individuals from different niches in order to maintain the population diversity. Figure 1 depicts the rationale behind the niches creation mechanism. In this example, a population composed by 10 individuals of length 1, 1, 2, 2, 3, 4, 4, 5, 7, 8 (without considering the root node) is considered and 10 niches are created. These niches span the lengths depicted in the figure and contain the individuals depicted inside them.

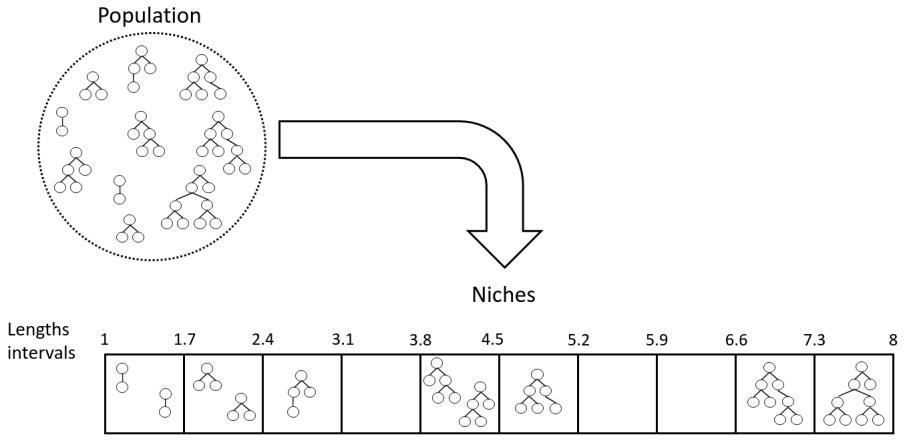


Fig. 1: Illustration of niches creation rationale.

2.2 Inclusive Reproduction and Inclusive Tournament

Algorithm 2 describes the mechanism behind the Inclusive Reproduction. Such mechanism consists in applying either crossover, mutation or 1:1 reproduction (the individual is passed unaltered to the offspring) using the individuals in the different niches. If the crossover is selected, a one point crossover is applied between two individuals which are selected from two different niches. About the two individuals chosen, one is the best of the considered niche, in order to pass the best performing genes to the future generation and the other is selected randomly in order to maintain a certain degree of diversity in the population. Moreover, a mechanism to avoid breeding between the same or very similar individuals is used (lines 16-20 in Algorithm 2). Here n_l is a preset constant defining the maximum number of loop iterations, needed to avoid possible infinite loops. If the mutation is selected, a mutation operator among those listed in Table 1 is applied to an individual randomly chosen from the chosen niche. Finally, if the 1:1 reproduction is selected, a randomly chosen individual from the chosen niche is passed to the offspring. The niches selected in all three previously described operations (crossover, mutation and 1:1 reproduction) are picked from a list of exploitable niches, which is continuously updated in order to avoid selecting always from the same niches.

The Inclusive Tournament consists in performing a Double Tournament [13] on each niche as in Algorithm 3. For the Inclusive Tournament the niches are selected in a sequential manner and the double tournament on each niche is performed at most t times where t is the number of individuals inside the considered niche, to avoid having clones in the final population.

Algorithm 2 Pseudocode of Inclusive Reproduction

```
1: if Crossover probability < 0.8 then
2:    Mutation probability = Mutation probability - 0.01
3:    Crossover probability = Crossover probability + 0.01
4: end if
5: Good Indexes ← Indexes of filled niches
6: List of exploitable niches ← Good Indexes
7: while Size offspring < λ do
8:    Choice ← Random number between [0, 1]
9:    if Choice < Crossover Probability then
10:      if List of exploitable niches is empty then
11:         List of exploitable niches ← Good Indexes
12:      end if
13:      Select randomly two different niches from List of exploitable niches
14:      Remove chosen niches from List of exploitable niches
15:      Select the best individual from the first niche and select a random individual
         from the second niche
16:      n = 0
17:      while The selected individuals have the same fitness and n < n_l do
18:         Repeat lines 10 to 15
19:         n = n+1
20:      end while
21:      Apply crossover to the chosen individuals
22:      Add first child to offspring
23:      if Size of offspring < λ then
24:         Add second child to offspring
25:      end if
26:   else if Choice < Mutation probability + Crossover Probability then
27:      Repeat lines 10 to 15 but selecting only one category
28:      Select randomly one individual from the chosen category
29:      Perform mutation of the chosen individual
30:      Add mutated individual to the offspring
31:   else
32:      Repeat line 27, 28
33:      Add chosen individual to the offspring
34:   end if
35: end while
```

3 Algorithms Setup and Results

In this section the results obtained from a comparison between the IGP and the Standard Genetic Programming (SGP) are presented. The SGP is a standard formulation of GP.

3.1 Benchmarks

The benchmarks selected for the comparison were taken from [22]. These were selected since comprehend both synthetic and real world data, hence they cover

Algorithm 3 Pseudocode of Inclusive Tournament

```
1: while Number of selected individuals < μ do
2:    for i in number of niches do
3:       if Number of selected individuals from i-th niche < total number of individuals
         in i-th niche then
4:          Select one individual in i-th niche with Double Tournament selection
5:       end if
6:    end for
7: end while
```

a sufficient variety of problems. More on the benchmarks can be found in [22]. In this work the same number of samples and sampling technique to produce them as in [22] were used, except for the benchmarks *korns11*, where 5000 samples were used instead of 10000 to reduce the computational time, both on training and testing samples; and for the benchmark *S2* where the same number of training samples used for the x variable were also used for the y one.

3.2 SGP and IGP settings

Both algorithms were implemented in Python 3 relying on the open source library DEAP [9]. The experiments were run on a Laptop with 16GB of RAM and an Intel® Core™ i7-8750H CPU @ 2.20GHz × 12 threads and multiprocessing was used. The code developed in this work is open source and can be found at https://github.com/strath-ace/smart-ml.
For both algorithms the bloat control mechanism implemented in the DEAP library was used with the limit height and size set as in Table 1. The mutation operators listed in Table 1 refers to the homonym functions in the DEAP library. Regarding the primitives listed in Table 1, *add3* and *mul3* are respectively a ternary addition and multiplication, while *plog* and *pexp* are respectively a protected natural logarithm and protected exponential, to avoid numerical errors. Regarding the crossover and mutation probability for the IGP, they changed dynamically during the evolutionary process as shown in lines 1-4 of Algorithm 2. This mechanism was introduced to incentivate the exploration at the beginning of the evolutionary process and the exploitation at the end of it. It was not adopted for the SGP since it was tested but did not introduce any improvement. 300 generations were chosen as stopping criteria to have a good compromise between the goodness of the results and a reasonable computation times.

3.3 Results

Both the SGP and IGP algorithms were run on nine different benchmarks consisting of synthetic and real world data as described in Section 3.1. On each benchmark the evolutionary process was repeated 100 times in order to obtain statistically significant results.

Table 1: Settings for the SGP and IGP algorithms. The percentages near the mutation mechanisms refers to the probability of that mutation mechanism to be chosen when the mutation is performed.

	SGP	IGP
Population Size	300 individuals	
Maximum Generations	300	
Stopping criteria	Reaching maximum number of generations	
Crossover probability	0.8	$0.2 \rightarrow 0.8$
Mutation probability	0.2	$0.8 \rightarrow 0.2$
Evolutionary strategy	$\mu + \lambda$	
μ	Population size	
λ	Population Size $\times$ 1.2	
Number of Ephemeral constants	1	
Limit Height	15	
Limit Size	30	
Selection Mechanism	Double Tournament	Inclusive Tournament
Double Tournament fitness size	2	
Double Tournament parsimony size	1.2	
Tree creation mechanism	Ramped half and half	
Mutation mechanisms	Uniform (50%), Shrink (5%), Insertion (25%), Mutate Ephemeral (20%)	
Crossover mechanism	One point crossover	
Primitives Set	$+$, $-$, $*$, $add3$, $mul3$, $tanh$ $square, plog$, $pexp$, sin, cos	
Fitness measure	RMSE	

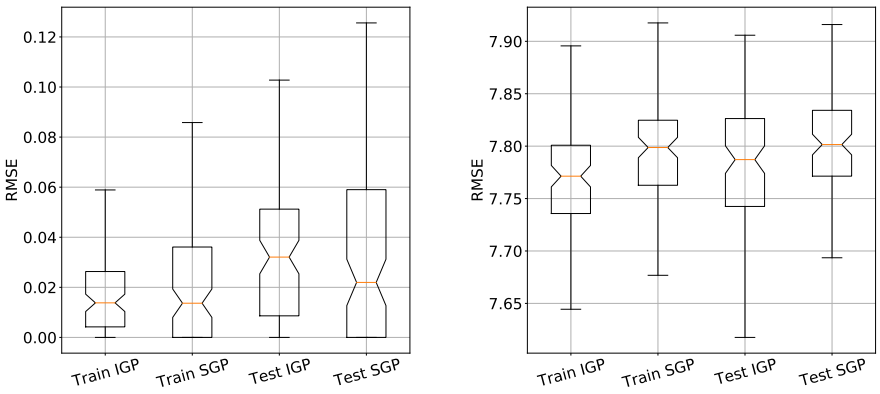
Figure 7 depicts the evolution of the RMSE median values of the SGP and IGP algorithms during the evolutionary process on the nine benchmarks. The shaded areas represent the standard deviation interval. From these results it is clear how the IGP is always capable of finding better or equally performing individuals to those found with SGP and also it converges faster than SGP to a minimum. Moreover, the Figures from 2 to 6 show the RMSE median and standard deviation on the different benchmarks on both train and test data. As can be seen, IGP always outperforms the SGP in terms of generalization capabilities, achieving a lower (better) RMSE, except for the *koza1* benchmark were the IGP performs worse than the SGP. A possible explanation for this could be that the key feature of IGP, which is the greater diversity in the population, leads to bigger individuals than those obtained with SGP. Such bigger individuals could be overkill solutions for a simple test case like *koza1*, which could be solved more efficiently by smaller individuals.

To assess how and if the population diversity was maintained throughout the evolutionary process, the entropy of the population was considered as proposed in [17]. Figure 8 shows the entropy median and standard deviation values of the population of both the SGP and IGP during the evolutionary process for all the benchmarks. As for the fitness values, also the entropy median values were

computed over the results obtained from 100 different runs.
The entropy was measured as in Equation 1

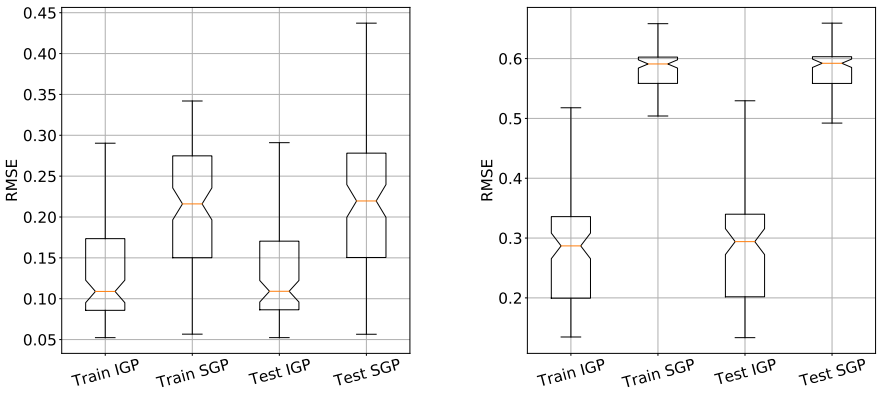
$$entropy = -\sum(d * log(d)) \tag{1}$$

where d is the distribution of the individuals across the different niches without considering the empty ones. Assuming the example illustrated in Figure 1, d would be the array [0.2, 0.2, 0.1, 0.2, 0.1, 0.1, 0.1] and the corresponding entropy would be 1.89. For sake of the comparison carried out in this work, also the population in the SGP algorithm was subdivided into niches to evaluate the entropy in the same way as was done in the IGP, although these were not used during the evolutionary process.
As can be seen in Figure 8 the IGP is able to maintain an entropy value around 2.30 during the whole evolutionary process, while the SGP tends to decrease towards the end of the evolutionary process, meaning that the population tends to have more similar individuals, i.e. lose diversity, while the evolution proceeds. Now by looking at both Figure 7 and 8, the results suggest that indeed the diversity maintenance plays an important role during the evolutionary process, resulting in a GP which can converge faster to the minimum, is more precise and has better generalization capabilities than a standard GP implementation.



(a) Benchmark Koza1 **(b)** Benchmark Korns11

Fig. 2: RMSE of training and test data on synthetic benchmarks Koza1 and Korns11. The median and standard deviations values were evaluated over the results produced on 100 different runs.



(a) Benchmark S1 **(b)** Benchmark S2

Fig. 3: RMSE of training and test data on synthetic benchmarks S1 and S2. The median and standard deviations values were evaluated over the results produced on 100 different runs.

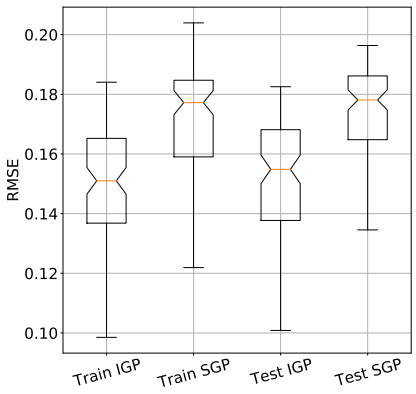
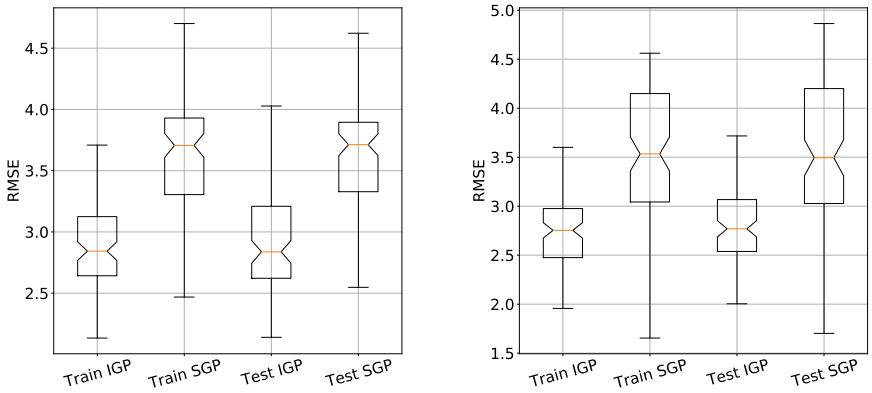
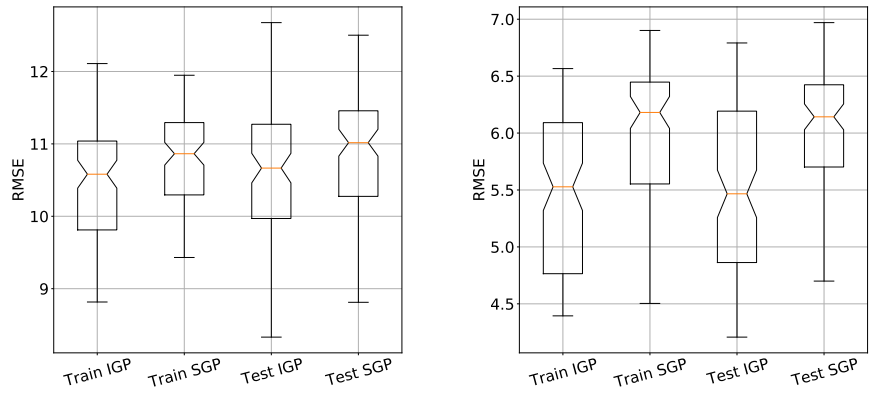


Fig. 4: RMSE of training and test data on synthetic benchmark UB. The median and standard deviations values were evaluated over the results produced on 100 different runs.



(a) Benchmark ENC **(b)** Benchmark ENH

Fig. 5: RMSE of training and test data on real world benchmarks ENC and ENH. The median and standard deviations values were evaluated over the results produced on 100 different runs.



(a) Benchmark CCS **(b)** Benchmark ASN

Fig. 6: RMSE of training and test data on real world benchmarks CCS and ASN. The median and standard deviations values were evaluated over the results produced on 100 different runs.

4 Conclusions

In this paper an heuristic approach to promote and maintain the diversity in a genetic programming algorithm, the Inclusive Genetic Programming, is described and tested by comparing its performance with a standard genetic programming implementation. The comparison was performed and here presented on nine different benchmarks, composed by both synthetic and real world data and on each benchmark the evolutionary process was repeated 100 times both with IGP and SGP. Then, the obtained best behaving individuals were tested on a set of test data, different from the training data, to assess the generalization capabilities. The results show how the IGP almost always (8/9 benchmarks) outperforms the SGP by converging faster to a minimum, with a better final fitness function value and by possessing better generalization capabilities than the SGP. To assess the importance of the population diversity on this outcome, the population's entropy was analyzed and it was observed that indeed the IGP is capable of maintaining it approximately constant throughout the evolutionary process, while in the SGP it tends to diminish towards the end of the evolutionary process, meaning a decrease in the population diversity.
This work aims to be a positive contribution to the open problem of the exploration vs. exploitation in the GP community, by describing and analyzing a new effective heuristic which highlights the importance of the exploration achieved through the promotion and maintenance of population diversity. Future works could investigate the application of the proposed heuristic to other GP formulations like the MGGP in order to assess its applicability. Moreover, it would be interesting to combine it with other diversity maintenance strategies to further increase the obtained benefits.

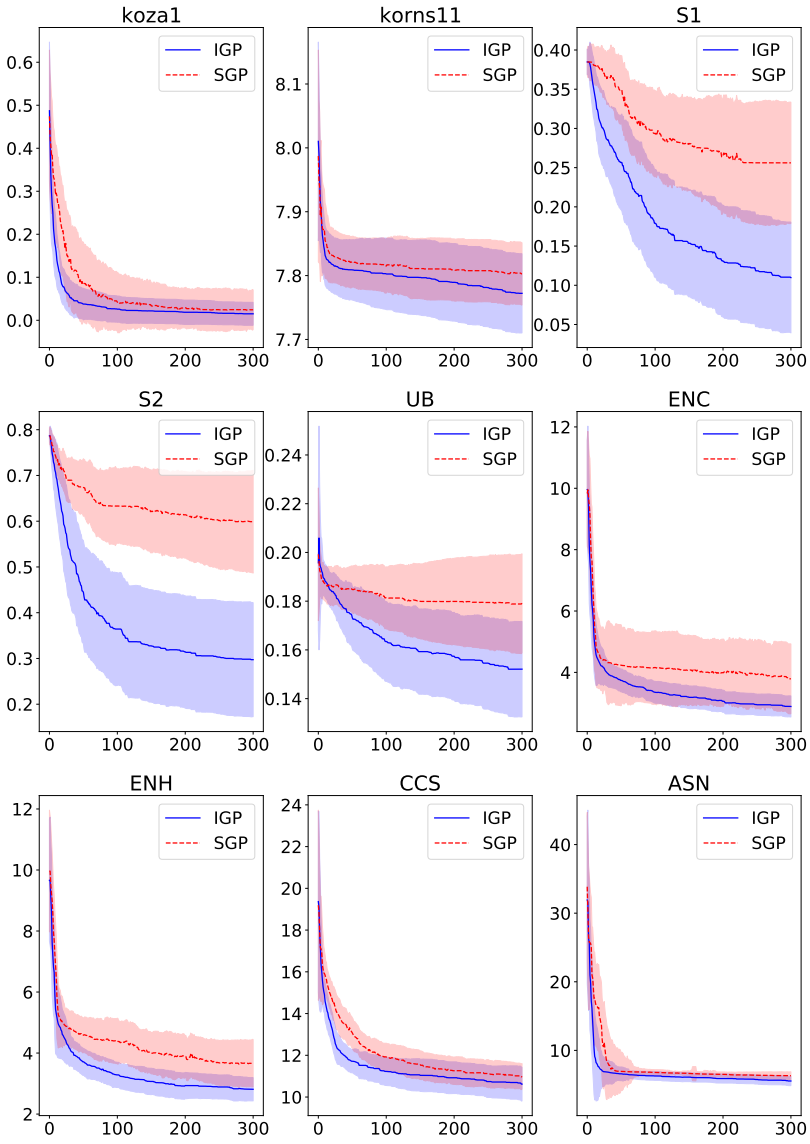


Fig. 7: Fitness function values of the SGP and IGP algorithms on the nine different benchmarks. On the ordinate is the RMSE while on the abscissa the number of generations. The solid and dashed lines represents the median values for the IGP and SGP respectively while the shaded regions are the standard deviation intervals

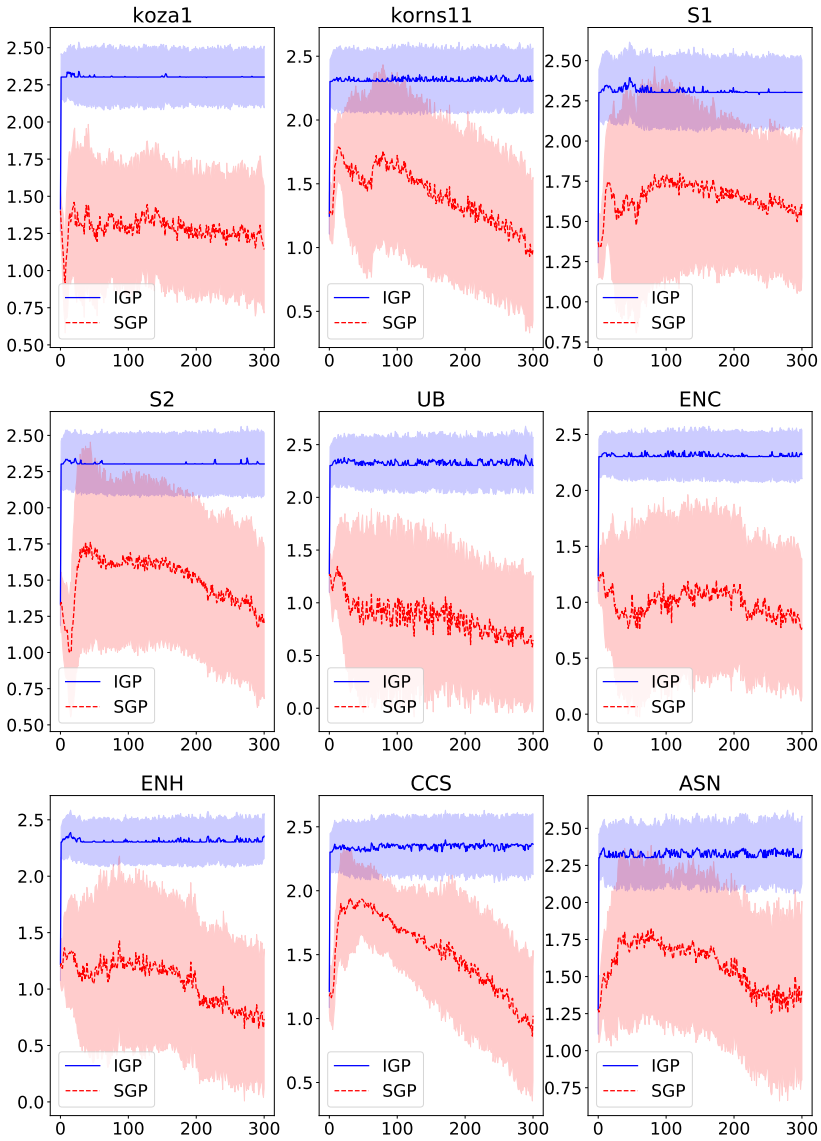


Fig. 8: Entropy values of the SGP and IGP algorithms on the nine different benchmarks. On the ordinate is the Entropy while on the abscissa the number of generations. The solid and dashed lines represents the median values for the IGP and SGP respectively while the shaded regions are the standard deviation intervals

References

1. Alfaro-Cid, E., Merelo, J.J., Fernández de Vega, F., Esparcia-Alcázar, A.I., Sharman, K.: Bloat control operators and diversity in genetic programming: A comparative study. Evolutionary Computation **18**(2), 305–332 (2010). https://doi.org/10.1162/evco.2010.18.2.18206
2. Aslam, M.W., Zhu, Z., Nandi, A.K.: Diverse partner selection with brood recombination in genetic programming. Applied Soft Computing Journal **67**, 558–566 (2018). https://doi.org/10.1016/j.asoc.2018.03.035
3. Beyer, H.G., Schwefel, H.P.: Evolution strategies – A comprehensive introduction. Natural Computing **1**(1), 3–52 (2002). https://doi.org/10.1023/A:1015059928466
4. Burke, E., Gustafson, S., Kendall, G., Krasnogor, N.: Advanced population diversity measures in genetic programming. In: Guervós, J.J.M., Adamidis, P., Beyer, H.G., Schwefel, H.P., Fernández-Villacañas, J.L. (eds.) Parallel Problem Solving from Nature — PPSN VII. pp. 341–350. Springer Berlin Heidelberg, Berlin, Heidelberg (2002)
5. Crepinsek, M., Liu, S.H., Mernik, M.: Exploration and exploitation in evolutionary algorithms: A survey. ACM Computing Surveys **45**(3), 1–33 (2013). https://doi.org/10.1145/2480741.2480752
6. Day, P., Nandi, A.K.: Binary string fitness characterization and comparative partner selection in genetic programming. IEEE Transactions on Evolutionary Computation **12**(6), 724–735 (2008). https://doi.org/10.1109/TEVC.2008.917201
7. De Jong, E.D., Watson, R.A., Pollack, J.B.: Reducing Bloat and Promoting Diversity using Multi-Objective Methods. In: Proceedings of the Genetic and Evolutionary Computation Conference GECCO2001. vol. GECCO-2001, pp. 11–18 (2001)
8. Fernandez De Vega, F., Olague, G., Lanza, D., Chavez De La O, F., Banzhaf, W., Goodman, E., Menendez-Clavijo, J., Martinez, A.: Time and Individual Duration in Genetic Programming. IEEE Access **8**, 38692–38713 (2020). https://doi.org/10.1109/ACCESS.2020.2975753
9. Fortin, F.A., De Rainville, F.M., Gardner, M.A., Parizeau, M., Gagńe, C.: DEAP: Evolutionary algorithms made easy. Journal of Machine Learning Research **13**, 2171–2175 (2012)
10. Grosman, B., Lewin, D.R.: Lyapunov-based stability analysis automated by genetic programming. In: 2006 IEEE Conference on Computer Aided Control System Design, 2006 IEEE International Conference on Control Applications, 2006 IEEE International Symposium on Intelligent Control. pp. 766–771 (2009). https://doi.org/10.1109/CACSD-CCA-ISIC.2006.4776742
11. Khayyam, H., Jamali, A., Assimi, H., Jazar, R.N.: Genetic Programming Approaches in Design and Optimization of Mechanical Engineering Applications. In: Jazar, R., Dai, L. (eds.) Nonlinear Approaches in Engineering Applications: Automotive Applications of Engineering Problems, pp. 367–402. Springer International Publishing, Cham (2020). https://doi.org/10.1007/978-3-030-18963-1_9
12. Koza, J.R.: Genetic programming as a means for programming computers by natural selection. Statistics and Computing **4**(2), 87–112 (1994). https://doi.org/10.1007/BF00175355, https://link.springer.com/content/pdf/10.1007{%}2FBF00175355.pdf
13. Luke, S., Panait, L.: Fighting bloat with nonparametric parsimony pressure. In: Guervós, J.J.M., Adamidis, P., Beyer, H.G., Schwefel, H.P., Fernández-Villacañas, J.L. (eds.) Parallel Problem Solving from Nature — PPSN VII. pp. 411–421. Springer Berlin Heidelberg, Berlin, Heidelberg (2002)

14. Mahfoud, S.W.: Niching methods for genetic algorithms. Ph.D. thesis (1995)
15. Marchetti, F., Minisci, E.: A Hybrid Neural Network-Genetic Programming Intelligent Control Approach. In: Filipič, B., Minisci, E., Vasile, M. (eds.) Bioinspired Optimization Methods and Their Applications. BIOMA 2020. Springer, Cham, Brussels (2020)
16. Oh, C.K., Barlow, G.J.: Autonomous controller design for unmanned aerial vehicles using multi-objective genetic programming. In: Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753). pp. 1538–1545 Vol.2 (2004). https://doi.org/10.1109/CEC.2004.1331079
17. Rosca, J.P.: Entropy-Driven Adaptive Representation. Proceedings of the Workshop on Genetic Programming: From Theory to Real-World Applications pp. 23–32 (1995)
18. Schmidt, M., Lipson, H.: Symbolic Regression of Implicit Equations. In: Riolo, R., O'Reilly, U.M., McConaghy, T. (eds.) Genetic Programming Theory and Practice VII, pp. 73–85. Springer US, Boston, MA (2010). https://doi.org/10.1007/978-1-4419-1626-6_5
19. Shir, O.M.: Niching in Evolutionary Algorithms. In: Rozenberg, G., Bäck, T., Kok, J.N. (eds.) Handbook of Natural Computing, pp. 1035–1069. Springer Berlin Heidelberg (2012). https://doi.org/10.1007/978-3-540-92910-9_32
20. Squillero, G., Tonda, A.: Divergence of character and premature convergence: A survey of methodologies for promoting diversity in evolutionary optimization. Information Sciences **329**, 782–799 (2016). https://doi.org/10.1016/j.ins.2015.09.056
21. Verdier, C.F., Mazo, Jr., M.: Formal Controller Synthesis via Genetic Programming. IFAC-PapersOnLine **50**(1), 7205–7210 (2017). https://doi.org/10.1016/j.ifacol.2017.08.1362
22. Žegklitz, J., Pošík, P.: Benchmarking state-of-the-art symbolic regression algorithms. Genetic Programming and Evolvable Machines (2020). https://doi.org/10.1007/s10710-020-09387-0