Space Partition based Gene Expression Programming for Symbolic Regression

Qiang Lu*, Shuo Zhou, Fan Tao, Zhiguang Wang
China University of Petroleum-Beijing
Beijing, China
luqiang@cup.edu.cn

ABSTRACT

For the problem of symbolic regression, we propose a novel space partition based gene expression programming (GEP) algorithm named SP-GEP, which helps GEP escape from local optimum and improves the search accuracy of GEP by letting individuals jump efficiently between segmented subspaces and preserving population diversity. It firstly partitions the space of mathematical expressions into k subspaces that are mutually exclusive. Then, in order for individuals to jump efficiently between these subspaces, it uses a subspace selection method, which combines multi-armed bandit and ϵ-greedy strategy. Through experiments on a set of standard SR benchmarks, the results show that the proposed SP-GEP always keeps higher population diversity, and can find more accurate results than canonical GEPs.

CCS CONCEPTS

• **Theory of computation → Evolutionary algorithms**.

KEYWORDS

Gene expression programming, multi-armed bandit, symbolic regression, evolutionary computation.

ACM Reference Format:
Qiang Lu*, Shuo Zhou, Fan Tao, Zhiguang Wang. 2019. Space Partition based Gene Expression Programming for Symbolic Regression. In *Proceedings of the Genetic and Evolutionary Computation Conference 2019 (GECCO '19).* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3319619.3322075

1 INTRODUCTION

The problem of symbolic regression (**SR**) is to discover a formula that best fits a given dataset in the space of mathematical expressions. Although many genetic programming methods are proposed to address the problem, they are easy to fall into premature convergence owing to the decline in population diversity. For preserving population diversity, we propose a space partition based gene expression programming (**SP-GEP**) to avoid the premature convergence. It firstly partitions the space of mathematical expressions into k subspaces (called **local spaces**) based on individual chromosome coding and initializes individuals in one of the k subspaces.

GECCO '19, July 13–17, 2019, Prague, Czech Republic

ACM ISBN 978-1-4503-6748-6/19/07.
https://doi.org/10.1145/3319619.3322075

Then, SP-GEP selects a suitable subspace to search for individuals with better fitness based on multi-armed bandit (**MAB**) and ϵ-greedy strategy. The method firstly utilizes MAB to choose one from these subspaces because MAB can balance the exploration that chooses other subspaces and the exploitation that chooses the subspace in which previous generation individuals located. However, MAB will be invalid when the number of visiting subspaces is greater than a certain value. So, in order to preserve population diversity, the method then switches to an ϵ-greedy strategy to choose a subspace according to a proposed time formula that can decide when to use the ϵ-greedy strategy.

At last, SP-GEP uses a proposed crossover method to make individuals jump from the original subspace to the selected subspace, and to search results.

2 SPACE PARTITION BASED GENE EXPRESSION PROGRAMMING

2.1 Space Partition

According to the coding of individuals in GEP, i.e., encodes an individual to a linear structure with the fixed length l and the head length h, the space of mathematical expression is denoted as $\Omega_{l,h} =' \underline{*...*} * ...*'$. l is the length of individuals (the total number of '$*$'); h is the head length (the number of '$\underline{*}$'); '$*$' can be anyone symbol from a symbol set S that consists of a function set F and a terminal set T. In the head, if front '$*$'s are replaced by special symbols s, it can generate a **subspace** ω_s, such as $\omega_+ =' \underline{+*} * **'$ in Figure. 1.

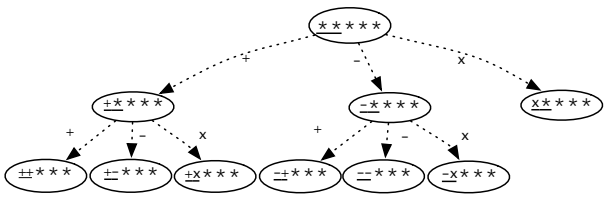

Figure 1: A space-partition tree. The symbol set $T = \{+, -, x\}$

According to the above subspace encoding, these subspaces and their relationships can be represented as **a space-partition tree**, where the root node is $\Omega_{l,h}$, each other node is a subspace of $\Omega_{l,h}$, and a branch represents a containment relationship between two subspaces, e.g., $\omega_{++} \subset \omega_+$ as shown in Figure.1. From the tree, a lot of space-partition sets can be found based on the above two conditions. For example, $\{\omega_+, \omega_-, \omega_x\}$ and $\{\omega_{++}, \omega_{+-}, \omega_{+x}, \omega_+, \omega_x\}$ both are space-partition sets.

2.2 Subspace Selection

SP-GEP firstly uses modified UCB1 to select a subspace from a space-partition set, as shown the equation 1.

$$UCB_{\omega_i} = \frac{1}{f^*_{\omega_i} + 1} + \lambda \sqrt{\frac{2lnt}{n_{\omega_i}}} \quad (1)$$

where ω_i is a subspace, $f^*_{\omega_i}$ is the fitness of the best individual in ω_i, t is the number of visiting Ω until a certain time, and n_{ω_i} is the number of times assesses to the subspace ω_i.

However, as visit times increase in a subspace, the size of the confidence interval ($\sqrt{\frac{2lnt}{n_i}}$ in equation 1) tends to be zero so that UCB_{ω_i} falls back to a greedy method with the subspace value ($f^*_{\omega_i}$) and becomes invalid in the balance between exploration and exploitation. So, when the modified UCB1 becomes invalid in most of the subspaces, SP-GEP then uses the ϵ-greedy method to select a subspace.

2.3 Exploitation with Crossover

After SP-GEP selected a subspace ω_i, suppose that the current all individuals are at ω_j, it firstly let them jump from ω_j to ω_i by replacing their encodings of ω_j into encodings of ω_i. As an example, given two individuals $++/-\times xxxxxx$ and '$+++\times\times xxxxxx$' in ω_{++}, the two individuals are changed to '$+--\times xxxxxx$' and '$+-+\times\times xxxxxx$' after the encoding ++ in them is replaced by '+−' that is the encoding of ω_{+-}. Then, it exploits ω_i by recombining each of transferred individuals with the best individual in ω_i.

3 EXPERIMENTS

In this paper, the dataset that consists of 20 test problems are derived from GP benchmarks[1]. The parameters of algorithms are set as follows: the size of population is 100; the head length is 20; the maximal number of generations is 10000; the mutation rate is 0.03; the recombination rate is 0.7; the number of subspaces k is in the scope [144,1000].

To obtain the performance metrics of algorithms: GEP, SP-GEP, GEP-ADF, and SP-GEP-ADF, each of them runs 10 times on the 20 test problems. Moreover, their results show in Table 1. Observing the comparative data between GEP and SP-GEP from Table 1, we conclude that SP-GEP finds more correct results than GEP ("6" in row "+" and "3" in row "-") so that it obtains better average fitnesses than GEP ("12" in row "+"). Comparing data between GEP-ADF and SP-GEP-ADF, SP-GEP-ADF can find more correct results and better average fitnesses than GEP-ADF.

4 CONCLUSION

For dealing with the SR problem, we have proposed a novel algorithm–SP-GEP based on partitioning the space of mathematical expression. By the subspace selection method, it can guide the population effectively jump between segmented subspaces so that it can maintain population diversity. Experimental results show that the proposed SP-GEP can overcome the problem of falling into local optimum, and has a better performance than canonical GEPs.

ACKNOWLEDGMENTS

This work was supported by National Science & Technology Major Project (no.2017ZX05018-005), National Natural Science Foundation of China (no. 61402532) and Science Foundation of China University of Petroleum-Beijing (no. 01JB0415).

REFERENCES

[1] James McDermott, David R. White, Sean Luke andLuca Manzoni, et al. 2012. Genetic programming needs better benchmarks. In *Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference - GECCO '12*. ACM, 791–798.

Table 1: Performance Metrics

F[1]	GEP		SP-GEP		GEP-ADF		SP-GEP-ADF	
	Suc[2]	RMSE[3]	Suc	RMSE	Suc	RMSE	Suc	RMSE
Ny4	10%	0.0206	20%	0.0290	10%	0.0375	60%	0.0153
Kz1	50%	0.0116	70%	0.0068	50%	0.0156	60%	0.0091
Kz2	90%	0.0084	100%	0.0078	100%	0.0083	100%	0.0075
Ny5	80%	0.0071	100%	0.0069	80%	0.0100	100%	0.0073
Ny6	100%	0.0058	100%	0	30%	0.0149	100%	0.0031
Ny7	80%	0.0094	90%	0.0092	40%	0.0137	60%	0.0114
Ny10	20%	0.0146	100%	0.0024	50%	0.0141	90%	0.0021
Kr11	10%	0.1095	0%	0.3451	20%	0.0418	0%	0.2875
Kr12	80%	0.0105	20%	0.0157	80%	0.0088	80%	0.0098
Kr12	0%	0.9459	0%	0.9383	0%	0.9440	0%	0.9380
Kj1	0%	0.0296	0%	0.0544	10%	0.0225	60%	0.0087
Kj7	0%	0.0518	0%	0.0451	0%	0.0384	0%	0.0398
Kj10	0%	0.0285	0%	0.0430	0%	0.1668	0%	0.0264
Kj12	40%	0.5369	0%	1.1224	80%	0.3302	90%	0.1257
Kj15	0%	0.5557	0%	0.5838	30%	0.0051	20%	0.2640
Vl1	0%	0.1003	0%	0.0652	0%	0.0659	0%	0.0556
Vl2	0%	0.0422	0%	0.0395	0%	0.0380	0%	0.0323
Vl3	0%	0.4622	0%	0.4243	0%	0.4687	0%	0.4086
Vl7	0%	1.3266	0%	1.4819	0%	1.2740	0%	1.0940
Vl8	0%	0.3206	0%	0.2801	0%	0.3161	0%	0.2627
=[4]			11	0			10	0
+			6	12			7	16
−			3	8			3	4

[1] Ny, Kz, Kr, Kj, and Vl represent "Nguyen", "Koza", "Korns", "Keijzer", and "Vladislavleva" in [1], respectively.

[2] "Suc" represents the proportion of finding the correct result whose fitnesses computed by RMSE is less than 0.01.

[3] "RMSE" shows the average fitness obtained by running one of these algorithms 10 times.

[4] "=, +, −" represents comparison results: same, good and poor. And the numeric in the row represents the comparison result between GEP and GP-GEP or between GEP-ADF and GP-GEP-ADF.