УНИВЕРСИТЕТ ИТМО
НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ
январь–февраль 2015 Том 15 № 1 ISSN 2226-1494 http://ntv.ifmo.ru/
SCIENTIFIC AND TECHNICAL JOURNAL OF INFORMATION TECHNOLOGIES, MECHANICS AND OPTICS
January–February 2015 Vol. 15 No 1 ISSN 2226-1494 http://ntv.ifmo.ru/en

УДК 004.855

ИССЛЕДОВАНИЕ ВЛИЯНИЯ ЯЗЫКА ПРЕДСТАВЛЕНИЯ РЕШЕНИЙ НА ЭФФЕКТИВНОСТЬ ПРЕДСКАЗАНИЯ ЦЕЛОЧИСЛЕННЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ

А.С. Потапов[a], В.В. Батищева[b], А.Ю. Воропай[c]

[a] Университет ИТМО, 197101, Санкт-Петербург, Россия
[b] Санкт-Петербургский государственный университет, Санкт-Петербург, 199034, Российская Федерация
[c] ИП «ТраффикДНА Украина», Запорожье, 69001, Украина
Адрес для переписки: pas.aicv@gmail.com

Информация о статье
Поступила в редакцию 10.11.14, принята к печати 25.12.14
doi: 10.17586/2226-1494-2015-15-1-107-114
Язык статьи – русский
Ссылка для цитирования: Потапов А.С., Батищева В.В., Воропай А.Ю. Исследование влияния языка представления решений на эффективность предсказания целочисленных последовательностей // Научно-технический вестник информационных технологий, механики и оптики. 2015. Том 15. № 1. С. 107–114

Аннотация. Предметом исследования в работе являются методы генетического программирования при решении задач экстраполяции целочисленных последовательностей. В целях проверки гипотезы о влиянии выразительности языка представления программ на эффективность предсказания был реализован метод генетического программирования на основе нескольких ограниченных языков для представления рекуррентных последовательностей. На одной выборке последовательностей реализованный метод с использованием наиболее полного языка показал результаты, существенно превосходящие один из представленных в литературе современных методов на основе искусственных нейронных сетей. Анализ результатов экспериментального сравнения разработанного метода с использованием различных языков показал, что расширение языка, делая доступными для предсказания новые классы последовательностей, в то же время увеличивает сложность поиска закономерностей для последовательностей, доступных для предсказания в более простом языке. Данный эффект может быть ослаблен, но не устранен полностью, при расширении языка конструкциями, делающими решения наиболее компактными. На основе проведенных исследований сделан вывод, что, хотя с использованием генетического программирования могут быть получены практически применимые методы, одного только выбора адекватного языка представления решений недостаточно для полного решения задачи предсказания целочисленных последовательностей (и, тем более, проблемы универсального предсказания).
Ключевые слова: генетическое программирование, универсальное предсказание, целочисленные последовательности, алгоритмическая сложность.
Благодарности. Работа выполнена при поддержке Министерства образования и науки Российской Федерации (проект №2323 в рамках базовой части государственного задания) и частично при государственной финансовой поддержке ведущих университетов Российской Федерации (субсидия 074-U01).

STUDY OF SOLUTION REPRESENTATION LANGUAGE INFLUENCE ON EFFICIENCY OF INTEGER SEQUENCES PREDICTION

A.S. Potapov[a], V.V. Batishcheva[b], A.Yu.Voropay[c]

[a] ITMO University, Saint Petersburg 197101, , Russia
[b] Saint Petersburg State University, Saint Petersburg, 199034, Russian Federation
[c] FE «TrafficDNA Ukraine», Zaporizhzhya, 69001, Ukraine
Corresponding author: pas.aicv@gmail.com

Article info
Received 10.11.14, accepted 25.12.14
doi: 10.17586/2226-1494-2015-15-1-107-114
Article in Russian
Reference for citation: Potapov A.S., Batishcheva V.V., Voropay A.Yu. Study of solution representation language influence on efficiency of integer sequences prediction. *Scientific and Technical Journal of Information Technologies, Mechanics and Optics*, 2015, vol. 15, no. 1, pp. 107–114 (in Russian)

Abstract. Methods based on genetic programming for the problem solution of integer sequences extrapolation are the subjects for study in the paper. In order to check the hypothesis about the influence of language expression of program representation on the prediction effectiveness, the genetic programming method based on several limited languages for recurrent sequences has been developed. On the single sequence sample the implemented method with the use of more complete language has shown results, significantly better than the results of one of the current methods represented in

literature based on artificial neural networks. Analysis of experimental comparison results for the realized method with the usage of different languages has shown that language extension increases the difficulty of consistent patterns search in languages, available for prediction in a simpler language though it makes new sequence classes accessible for prediction. This effect can be reduced but not eliminated completely at language extension by the constructions, which make solutions more compact. Carried out researches have drawn to the conclusion that alone the choice of an adequate language for solution representation is not enough for the full problem solution of integer sequences prediction (and, all the more, universal prediction problem). However, practically applied methods can be received by the usage of genetic programming.
Keywords: genetic programming, universal prediction, integer sequences, algorithmic complexity.
Acknowledgements. The research has been supported by the Ministry of Education and Science of the Russian Federation (project #2323 within the basic component of the government assignment) and partially financially supported by the Government of the Russian Federation (grant 074-U01).

Введение

Экстраполяция последовательностей символов является принципиальным компонентом моделей универсального искусственного интеллекта [1]. Для этой проблемы существует общее теоретическое решение – модель универсального предсказания, основанная на понятии алгоритмической вероятности [2]. К сожалению, данное решение является невычислимым, в связи с чем на его основе невозможно непосредственно построить практически применимые методы. Даже вычислимые аппроксимации модели универсального предсказания, в частности, основанные на сложности по Л.А. Левину [3], вместо алгоритмической вероятности, без использования каких-то дополнительных эвристик требуют слишком большого объема вычислений.

В этой связи представляет интерес исследование связи теоретической модели универсального предсказания с эффективными специализированными методами, работающими в конкретных проблемных областях. В качестве такой области может выступать предсказание целочисленных последовательностей, представляющее самостоятельный интерес. Действительно, такое предсказание является традиционным компонентом IQ-тестов [4], в связи с чем его выполнение нередко требуется и от моделей искусственного интеллекта [5–7].

Некоторые авторы [5] указывают на существование всего двух подходов к решению этой проблемы – на основе искусственных нейронных сетей (ИНС) и антиунификации, противопоставляя их собственному когнитивному подходу (к которому, однако, можно отнести и иные работы [6, 7]). Другие авторы [7] выделяют большое число прочих подходов, включая подходы на основе скрытых марковских моделей, динамического программирования, обучения с подкреплением, графовых моделей, моделей эвристического поиска, эволюционных вычислений, символьного планирования и продукционных систем. Кроме того, существуют и модули предсказания числовых последовательностей в промышленном программном обеспечении, например Mathematica, Maple, Wolfram|Alpha и др. Однако в некоторых работах отмечается их слабость [6, 7].

Стоит отметить, что указанные подходы не являются взаимоисключающими, и их можно классифицировать, по крайней мере, по двум сравнительно независимым компонентам – способу представления решений и методу поиска оптимального решения (в качестве дополнительного компонента можно рассмотреть и критерий оптимальности). Действительно, модели эвристического поиска или эволюционных вычислений имеют прямое отношение к компоненту поиска оптимального решения и могут сочетаться с другими подходами, преимущественно выделяемыми по способу представления решений (задающему пространство решений). Теоретически оптимальная (по вероятности правильного предсказания) модель универсального предсказания характеризуется тем, что в ней используется Тьюринг-полный язык представления решений и исчерпывающий поиск [1, 2], в отличие от практически реализуемых методов. Среди последних выразительная сила языка представления решений может быть различной, что влияет как на скорость поиска решения, так и на множество доступных для предсказания последовательностей. В частности, в работе [7] отмечается, что некоторые коммерческие системы предсказания численных последовательностей ограничиваются рекуррентными соотношениями определенного вида, для которых известны эффективные алгоритмы вычисления коэффициентов по заданной последовательности. В ряде же работ выбор способа представления решений не имеет содержательного обоснования, а ограничения на возможность предсказания последовательностей, накладываемые выбранным способом, в явном виде не описываются. Примером такой работы нам представляется [8], где используются ИНС, оказавшиеся принципиально неспособными предсказывать некоторые из последовательностей, выбранных авторами.

Более глубокими на этом фоне нам представляются работы, в которых выбор языка представления присутствует как явно поставленная задача. Естественно, постановка этой задачи может проистекать из особенностей конкретного приложения. Однако в рамках исследований по искусственному интеллекту интерес представляет создание системы, способной выполнять предсказание, по крайней мере, тех же последовательностей, что и человек [5–7]. При этом структура языка представления также зачастую выбирается, исходя из аналогий с человеческим мышлением, и разрабатывается лишь один язык представления.

Таким образом, актуальной является задача более детального сравнения и анализа влияния выбора языка представления решений на эффективность предсказания целочисленных последовательностей. В рамках настоящей работы рассмотрена последовательность из нескольких усложняющихся языков для представления выражений, задающих числовые последовательности. Для поиска выражения, восстанавливающего заданную последовательность, реализован вариант метода генетического программирования. Проведено исследование эффективности поиска различных выражений (возможности и трудоемкости их обнаружения) в зависимости от языка.

Модель универсального предсказания

Основы теории универсального предсказания заложены полвека назад [9, 10]. Основное понятие в этой теории – понятие алгоритмической вероятности, которая для некоторой символьной (например, бинарной) строки α определяется как

$$P_U(\alpha) = \sum_{\mu:U(\mu)=\alpha} 2^{-l(\mu)},$$

где U – некоторая универсальная машина (например, универсальная машина Тьюринга), $l(\mu)$ – длина программы μ для этой машины (при представлении μ как битовой строки). Суммирование ведется по всем программам, которые при выполнении на машине U на выходе печатают α (в целях решения задачи предсказания следует требовать, чтобы программа μ печатала некоторую строку, не обязательно точно совпадающую с α, но строку, префиксом которой является строка α).

Предсказание осуществляется на основе условной алгоритмической вероятности, определяемой на основе выражения

$$P_U(\beta \mid \alpha) = P_U(\alpha\beta)/P_U(\alpha),$$

где αβ – конкатенация двух строк. $P_U(\beta \mid \alpha)$ оказывается вероятностью того, что строка β будет напечатана после строки α.

Крайне привлекательным свойством алгоритмической вероятности является то, что предсказание при ее использовании в качестве критерия асимптотически (причем зачастую достаточно быстро) сходится к оптимальному при увеличении длины экстраполируемой строки [2] для любого вычислимого источника данных, даже если никакой априорной информации об этом источнике или его классе нет, в связи с чем такое предсказание и получило название модели универсального предсказания. При этом в предсказании используются все возможные модели источника информации, не противоречащие имеющимся данным (т.е. не происходит выбора лучшей модели). Однако таких моделей бесконечно много.

Очевидным упрощением, позволяющим избежать суммирования по бесконечному числу слагаемых, является использование только наибольшего из них. Таковым является слагаемое $2^{-K_U(\alpha)}$, где $K_U(\alpha) = \min_{\mu:U(\mu)=\alpha} l(\mu)$ – алгоритмическая сложность строки α по А.Н. Колмогорову (или ее модификации, предполагающая порождение программой μ строки α в качестве префикса).

Данное упрощение, не являющееся адекватным в моделях общего искусственного интеллекта [11], приемлемо при решении задачи предсказания, поскольку также обеспечивает сходимость [12] (хотя и более медленную). Более того, в IQ-тестах зачастую предполагается однозначный выбор лучшей модели. Действительно, такие последовательности, как 1, 2 или 6, 7, 8, 10, представляются для теста некорректными, поскольку допускают несколько близких по сложности решений с разными предсказаниями. Таким образом, достаточно вести поиск наикратчайшей программы, согласованной с заданной строкой, после чего выполнять предсказание только на ее основе.

Однако такой поиск также алгоритмически нереализуем в силу неразрешимости проблемы останова. Использование сложности по Левину, $C_U(\alpha) = \min_{\mu:U(\mu)=\alpha} \left[l(\mu) + \log_2 t(\mu,\alpha) \right]$, где $t(\mu, \alpha)$ – число операций, совершаемых программой μ для печати α, решает данную проблему. Величина $C_U(\alpha)$ является не только вычислимой, но также может считаться адекватнее и с точки зрения задания априорных вероятностей программ. В частности, предпочтение моделям, наиболее простым как структурно, так и вычислительно, отдается и человеком [7].

Однако прямой поиск оптимального по критерию $C_U(\alpha)$ решения μ^* требует числа операций, возрастающих экспоненциально с $l(\mu^*)$, и становится практически неосуществимым уже для весьма коротких программ (длиной несколько десятков бит) [13]. Может возникнуть впечатление, что неэффективность такого подхода связана с его чрезмерной общностью, проистекающей из универсальности (полноты по Тьюрингу) опорной машины U. Однако отказ от универсальности U в пользу ограниченных машин (языков представления решений), допускающих эффективные алгоритмы поиска, приводит к принципиальной невозможности предсказания тех или иных последовательностей, доступных для предсказания человеку. При менее жестких ограничениях, даже в случае неуниверсальности U, поиск оптимального решения вполне может оказаться NP-полной задачей (к примеру, при использовании в качестве U опорной машины, эквивалентной контекстно-зависимым грамматикам).

Таким образом, разработка эффективных методов поиска для специализированных *U*, хотя и является практически важной (и даже, может быть, значимой компонентой общего интеллекта), не решает проблемы в целом. Общим приемом облегчения поиска оптимального решения здесь может служить уменьшение сложности самого этого решения за счет выбора подходящей опорной машины. Способ представления программ на языке этой машины и наличие или отсутствие в нем тех или иных операций в качестве базовых может существенно сказаться на длине искомого решения и, как следствие, на скорости его обнаружения [13]. В частности, в случае классической универсальной машины Тьюринга реализация одной только операции сложения чисел в битовом представлении будет достаточно длинной, чтобы задача экстраполяции числовых последовательностей оказалась практически нерешаемой даже в самых простых случаях. Таким образом, выбор представления оказывается одним из ключевых моментов для достижения эффективности в методе универсальной индукции.

Выбор языков представления числовых последовательностей

Программа μ, являющаяся моделью источника некоторой последовательности α, должна порождать строку, префиксом которой оказывается α. Вместо таких программ несколько удобнее рассматривать функции, которые возвращают следующий элемент последовательности, получая на вход предыдущие элементы. Имея такую функцию, несложно построить программу, которая будет в цикле порождать всю последовательность.

В качестве таковых функций мы будем рассматривать функции из разных подмножеств языка Racket (диалект Lisp/Scheme – http://racket-lang.org/). Одно из простейших подмножеств для задачи предсказания можно определить следующим образом:

(define (f xs) t).

Здесь f – функция, возвращающая следующий элемент последовательности по списку предыдущих элементов xs; t – терм, определяемый рекурсивно: цифры 0, ..., 9 – термы; (length xs) – терм; если t1 и t2 – термы, то (o2 t1 t2), где o2 – одна из двуместных библиотечных функций (+, –, *), также терм. (length xs) рассматривается как элементарный терм, а не как одноместная операция, применяемая к произвольному терму, поскольку в данном подмножестве языка нет других списков, к которым эту операцию можно было бы применить, и такой подход позволяет избавиться от необходимости контроля типов (все термы имеют тип целого числа). Для явного указания на элементарность этого терма можно множество функций задать в следующем виде

```
(define (f xs)
  (let ([n (length xs)]) t))
```

и при генерации t вместо (length xs) использовать n.

Данный язык, который мы обозначим S_1, задает множество многочленов с целочисленными коэффициентами. Например, вместо t может быть подставлено выражение (+ n 1), что даст последовательность '(1 2 3 4…), или (* n (+ n 1)), что даст последовательность '(0 2 6 12…), и т.д. Естественно, этот язык весьма ограничен, и в нем не может быть описано много простых для человека последовательностей, например, последовательность Фибоначчи.

Для человека естественным является задание последовательностей в рекуррентном виде [7]. В этой связи введем расширение языка S_1, включив в него символы x1 и x2, обозначающие последний, (n–1)-й, и предпоследний, (n–2)-й, элементы списка xs соответственно. Обращение к этим элементам требует особой обработки при n<2. Типичным для рекуррентного задания последовательностей является введение базовых случаев. В этой связи может быть рассмотрено специальное подмножество функций, которое мы будем обозначать через S_2:

```
(define (f xs)
  (let ([n (length xs)])
    (cond [(equal? n 0) a0]
          [(equal? n 1) a1]
          [#t (let ([x1 (list-ref xs (- n 1))]
                    [x2 (list-ref xs (- n 2))])
                t)])))
```

Здесь a0 и a1 заменяются первыми двумя элементами последовательности, а t определяется, как и ранее, но с добавлением символов x1 и x2. Стоит отметить, что в данном специальном подмножестве программ значения a0 и a1 не ищутся перебором, что само по себе является существенной ad hoc оптимизацией универсальной индукции. Также структура cond является априорно заданной и может не учитываться при определении сложности программы, за исключением вариативной части, т.е. заменяемых элементов a0 и a1. Для функций, не использующих символы x2 и (или) x1, задание a1 и (или) a0 не требуется, и их сложность может считаться более низкой.

Введение указанных двух символов существенно расширяет выразительную силу языка S_2. Вместо этих двух символов можно было бы добавить общую операцию взятия элемента последовательности с

заданным номером *k*, которое бы работало как (list-ref xs k). Тогда вместо x1 можно было бы использовать выражение (list-ref xs (- n 1)). Очевидно, однако, что вероятность породить такое выражение в ходе перебора весьма мала, в отличие от выражения x1. В этой связи может быть полезным ввести данную операцию не как замену, а как дополнение к символам x1 и x2. Кроме того, поскольку обращение к предыдущим элементам последовательности может быть более типичным, чем к первым элементам, более эффективной представляется операция вида (xs-ref k), эквивалентная (list-ref xs (- n k)). Язык, расширенный данной операцией, а также операцией целочисленного деления, обозначим как S_3.

Полезным является сравнение этого языка с языком S_{3-}, в котором вместо x1, x2 и xs-ref используется list-ref, первым аргументом которого всегда является xs, так как других списков в этом языке нет, а вторым аргументом является номер элемента в списке (который, в отличие от xs-ref, отсчитывается не с конца списка, а с его начала).

Специальной обработки при вызове функций list-ref и xs-ref требует случай выхода за границы списка. Как и при использовании термов x1 и x2, оказывается необходимым вводить базовые случаи a0 и a1, так и для этих функций обращение к несуществующим элементам является индикатором базового случая. Такая программа считается корректной, но введение дополнительных базовых случаев штрафуется за сложность (прибавляется к длине программы).

Рассмотренные языки могут быть расширены и дальше до алгоритмически полных языков, однако для достижения целей нашего исследования этого не требуется.

Реализация генетического программирования

Эффективным средством поиска в пространстве программ (в различных формализмах – от клеточных автоматов до нейронных сетей) считается генетическое программирование (ГП) [14, 15], идея использования которого для реализации универсальной индукции также высказывалась ранее [2]. Естественно, нельзя ожидать, чтобы ГП целиком решало проблему поиска в пространстве программ, однако, по крайней мере, оно существенно практичнее полного перебора или случайного поиска (что, в частности, было проверено в наших экспериментах: с помощью полного перебора оказалось практически невозможным искать программы длиною более 5 термов).

Программы в случае рассмотренных языков представляются деревьями (вложенными списками), чем и определяется реализация метода ГП для них. ГП традиционно включает следующие шаги:

- генерацию начальной популяции некоторого размера N_{pop};
- порождение последующих поколений (в базовом случае число поколений является установочным параметром N_{gen}):
  - порождение kN_{pop} потомков, где k – установочный параметр, больший либо равный единице (k=1 может использоваться в случае, когда особи из предыдущего поколения участвуют в отборе вместе с особями из нового поколения; в противном случае типично использование k≥2), для каждого из которых выполняются действия:
    - выбор родительской пары;
    - скрещивание;
    - мутация;
  - отбор лучших N_{pop} особей.

В нашей реализации каждая особь начальной популяции порождалась с использованием процедуры, работающей в соответствии с рекурсивным определением терма со случайным выбором между возможными вариантами, вероятности которых брались разными. Например, в языке S_1 в 30% случаев в качестве терма порождалась случайная цифра (сами цифры были равновероятными), в 30% случаев – символ n, а в 40% случаев – (op2 t1 t2), где op2 с равной вероятностью оказывалась одной из операций +, –, *, а t1 и t2 далее порождались рекурсивно. Конкретный выбор этих вероятностей может существенно сказываться на трудоемкости нахождения решения для продолжения той или иной последовательности, однако этот вопрос детально не исследовался в связи с тем, что нас интересовало не построение максимально эффективной системы предсказания последовательностей, а относительная эффективность разных языков.

Параметры N_{pop} и N_{gen} в каждом случае подбирались таким образом, чтобы правильное решение находилось не менее чем в 90% запусков. При этом для простоты бралось N_{pop}=N_{gen}. Значение k было взято равным двум, но при этом в отборе также участвовали и родительские особи (т.е. отбор на каждой итерации проводился среди $3N_{pop}$ особей).

Родители в паре выбирались равномерно случайно (без учета значения их фитнесс-функции) и независимо. Скрещивание было реализовано как замена одного случайно выбранного поддерева в первом родителе на случайно выбранное поддерево во втором родителе. Мутации были реализованы как замена случайно выбранного поддерева в особи на случайно порожденный терм (процедура порождения термов использовалась та же, что и при генерации начальной популяции) и осуществлялись с вероятностью 10%.

Фитнесс-функция включала штраф за сложность программы (число узлов в дереве программы N_{nodes}) и за использование базовых случаев (или выход за границы списка xs) – N_{base}, а также штраф за суммарную ошибку предсказания (в качестве которой бралась сумма модулей отклонений порожденного значения от заданного E_{mod}). Поскольку во всех случаях предполагалась возможность точного предсказания, детальный баланс между сложностью и точностью решения не требовался, и фитнесс-функция носила отчасти эвристический характер: $F = 0{,}1(N_{nodes} + N_{base}) + E_{mod}$.

Поскольку само ГП не является предметом исследования настоящей работы, исследование различных вариантов реализации генетических операторов и детальная настройка параметров не проводились.

Экспериментальный анализ

На первом этапе экспериментов была проверена возможность находить корректные решения для разных S_i в сравнении друг с другом и другими методами, описанными в литературе. К сожалению, детальный перечень тестовых последовательностей был найден лишь в одной работе [8], тогда как в ряде других работ (например, [5]) были использованы последовательности из коммерческих IQ-тестов, в связи с чем они не приводились. Рассмотрим результаты сравнения на 20 последовательностях из [8] (табл. 1)

Последовательность	человек, P	ИНС, P	S_1	S_2	S_3	S_{3-}
12, 15, 8, 11, 4	0,88	0,63	–	60	100	–
148, 84, 52, 36, 28	0,71	0,82	–	–	600	–
2, 12, 21, 29, 36	0,82	0,64	–	180	250	750
2, 3, 5, 9, 17	0,76	0,23	–	130	370	–
2, 5, 8, 11, 14	0,53	0,79	40	35	100	250
2, 5, 9, 19, 37	0,35	0	–	70	180	–
25, 22, 19, 16, 13	0,94	0,79	35	50	70	200
28, 33, 31, 36, 34	1	0,40	–	45	60	1000
3, 6, 12, 24, 48	0,76	0	–	10	50	500
3, 7, 15, 31, 63	0,71	0	–	35	110	1500
4, 11, 15, 26, 41	0,47	0,03	–	40	50	–
5, 6, 7, 8, 10	0,59	0,14	–	–	700	–
54, 48, 42, 36, 30	0,94	0,45	100	45	50	350
6, 8, 5, 7, 4	0,94	0,26	–	120	130	–
6, 9, 18, 21, 42	0,82	0,12	–	–	700	–
7, 10, 9, 12, 11	0,82	0,48	–	55	60	–
8, 10, 14, 18, 26	0,76	0,03	–	150	250	–
8, 12, 10, 16, 12	1	0,06	–	–	230	–
8, 12, 16, 20, 24	0,88	0,73	30	30	35	200
9, 20, 6, 17, 3	0,94	0,48	–	50	70	–

Таблица 1. Результаты предсказания числовых последовательностей

Здесь данные в столбцах «человек, P», «ИНС, P» приведены на основе информации из работы [8] и означают вероятности нахождения корректного решения человеком и ИНС. Последующие столбцы показывают требуемые значения N_{pop} (N_{gen}), при которых ГП находит решение с вероятностью не менее чем 90%. Чтобы получить эти пороговые значения, метод ГП запускался не менее 10 раз для больших и малых значений N_{pop}. Проводилось увеличение нижней границы N_{pop}, пока количество ошибок оказывалось не более двух на 10 запусков, и уменьшение верхней границы N_{pop}, пока не возникало первого ошибочного результата. Далее между этими границами подбиралось такое значение N_{pop}, при котором вероятность ошибки оказывалась приближенно равной 10% на 20 запусках. Шаг по N_{pop} брался от 5 до 50 в зависимости от его абсолютного значения. Стоит отметить, что время работы метода ГП, реализованного на интерпретируемом языке Scheme, при запуске на компьютере с процессором i7 3,40 ГГц составляло около 1 с при $N_{pop} = N_{gen} = 100$. К сожалению, информации о времени обучения ИНС в работе [8] не приводится.

Как видно, с использованием языка S_3 оказывается возможным предсказывать все из указанных последовательностей, что говорит о нецелесообразности решения задач символьной индукции на нейросетевом уровне.

Очевидно, что в языках S_1 и S_2 некоторые закономерности, представимые в языке S_3, оказываются непредставимыми, в связи с чем решение не может быть найдено. Интереснее, однако, вопрос, насколько более сложным оказывается поиск эквивалентных решений в более богатых языках. Ведь, как видно из табл. 1, в языке S_{3-} не удается найти многие последовательности, хотя они все представимы, но в более сложном, чем в S_3 виде. В табл. 2 приведены результаты для последовательностей (взятых из табл. 1 и пополненных), закономерности в которых представимы во всех языках S_i.

Последовательность	Значение N_{pop} (N_{gen})			
	S_1	S_2	S_3	S_{3-}
2, 5, 8, 11, 14	40	35	100	250
25, 22, 19, 16, 13	35	50	70	200
54, 48, 42, 36, 30	100	45	50	350
8, 12, 16, 20, 24	30	30	35	200
0, 1, 4, 9, 16	20	150	200	100
1, 3, 7, 13, 21	60	100	250	750
9, 8, 9, 12, 17, 24	250	200	300	–

Таблица 2. Результаты предсказания числовых последовательностей

Результаты, представленные в табл. 2, могут показаться противоречивыми. В ряде случаев язык S_1 оказывается менее эффективным, чем S_2 или S_3. Это, однако, связано с тем, что одна и та же закономерность может быть представлена разными способами (как функция от n или рекуррентно). Иначе говоря, расширение языка может не только делать представимыми некоторые закономерности, но также и уменьшать сложность уже представимых закономерностей. В то же время, однако, когда не появляется более простого представления некоторой закономерности, сложность ее поиска, естественно, возрастает. Нас интересует, насколько этот рост существенный.

Во многих случаях это возрастание оказывается умеренным и не приводит к принципиальной невозможности нахождения решения в связи с ресурсными ограничениями. На основе сравнения представлений S_3 и S_{3-} видно, что введение избыточных конструкций, «синтаксического сахара», предпочтительно для ГП: оно дает больший выигрыш в возможности решения новых задач, чем проигрыш в усложнении решения тех задач, которые решались в более простом языке. В то же время сравнение S_1 и S_{3-} показывает, что при «неудачном» расширении языка сложность нахождения решения заметно возрастает (или даже превосходит критический уровень).

Заключение

В работе было проведено исследование влияния языка представления программ на эффективность их поиска методом генетического программирования в задаче экстраполяции целочисленных последовательностей. Рассмотренные языки обладали разной выразительной силой, но не были алгоритмически полными. При этом, если некоторая закономерность не выразима в заданном языке, то последовательность на ее основе не может быть точно экстраполирована (таковой для выбранных языков является, например, последовательность простых чисел). Тем не менее, разработанным методом оказалось возможным предсказывать широкий класс последовательностей, в том числе недоступных для некоторых других существующих методов.

Эксперименты показали, что расширение языка повышает сложность поиска простых закономерностей, постепенно делая этот поиск неэффективным. Добавление «синтаксического сахара» – конструкций, компактно выражающих полезные комбинации термов, – ослабляет этот эффект, однако не устраняет его полностью. Таким образом, представимость решения не означает возможность его найти за обозримое время. Хотя использование генетического программирования с подходящим языком представления программ позволяет получить практически приемлемые методы, для полного решения задачи предсказания целочисленных последовательностей (и, тем более, проблемы универсального предсказания) одного лишь выбора языка недостаточно, и требуется развивать также методы поиска, не ограничиваясь «слепым» поиском, направляемым лишь метаэвристиками.

References

1. Hutter M. Universal algorithmic intelligence: a mathematical top→down approach. *Cognitive Technologies*, 2007, vol. 8, pp. 227–290. doi: 10.1007/978-3-540-68677-4_8
2. Solomonoff R. Algorithmic probability, heuristic programming and AGI. *Proc. 3rd Conf. on Artificial General Intelligence, AGI 2010*. Lugano, Switzerland, 2010, pp. 151–157.
3. Schmidhuber J. The new AI: general & sound & relevant for physics. *Cognitive Technologies*, 2007, vol. 8, pp. 175–198. doi: 10.1007/978-3-540-68677-4_6
4. Stern W., Whipple G. *The Psychological Methods of Testing Intelligence*. Baltimore, Warwick & York, 1914, 160 p.
5. Siebers M., Schmid U. Semi-analytic natural number series induction. *Lecture Notes in Computer Science*, 2012, vol. 7526, pp. 249–252. doi: 10.1007/978-3-642-33347-7_25
6. Strannegard C., Nizamani A.R., Sjoberg A., Engstrom F. Bounded Kolmogorov complexity based on cognitive models. *Lecture Notes in Computer Science*, 2013, vol. 7999, pp. 130–139. doi: 10.1007/978-3-642-39521-5_14

7. Strannegard C., Amirghasemi M., Ulfsbacker S. An anthropomorphic method for number sequence problems. *Cognitive Systems Research*, 2013, vol. 22–23, pp. 27–34. doi: 10.1016/j.cogsys.2012.05.003
8. Ragni M., Klein A. Predicting numbers: an AI approach to solving number series. *Lecture Notes in Computer Science*, 2011, vol. 7006, pp. 255–259. doi: 10.1007/978-3-642-24455-1_24
9. Solomonoff R.J. A formal theory of inductive inference. Part I. *Information and Control*, 1964, vol. 7, no. 1, pp. 1–22.
10. Solomonoff R.J. A formal theory of inductive inference. Part II. *Information and Control*, 1964, vol. 7, no. 2, pp. 224–254.
11. Potapov A., Svitenkov A., Vinogradov Y. Differences between Kolmogorov complexity and Solomonoff probability: consequences for AGI. *Lecture Notes in Computer Science*, 2012, vol. 7716, pp. 252–261. doi: 10.1007/978-3-642-35506-6_26
12. Poland J., Hutter M. MDL convergence speed for Bernoulli sequences. *Statistics and Computing*, 2006, vol. 16, no. 2, pp. 161–175. doi: 10.1007/s11222-006-6746-3
13. Potapov A., Rodionov S. Universal induction with varying sets of combinators. *Lecture Notes in Computer Science*, 2013, vol. 7999, pp. 88–97. doi: 10.1007/978-3-642-39521-5_10
14. Tsarev F. Sovmestnoe primenenie geneticheskogo programmirovaniya, konechnykh avtomatov i iskusstvennykh neironnykh setei dlya postroeniya sistemy upravleniya bespilotnym letatel'nym apparatom [Application of genetic programming, finite state machines and neural nets for construction of control system for unnmaned aircraft]. *Scientific and Technical Journal of Information Technologies, Mechanics and Optics*, 2008, no. 53, pp. 42–60.
15. Bondarenko I .B., Gatchin Yu.A., Geranichev V.N. Sintez optimal'nykh iskusstvennykh neironnykh setei s pomoshch'yu modifitsirovannogo geneticheskogo algoritma [Synthesis of optimal artificial neural networks by modified genetic algorithm]. *Scientific and Technical Journal of Information Technologies, Mechanics and Optics*, 2012, no. 2 (78), pp. 51–55.

Потапов Алексей Сергеевич – доктор технических наук, доцент, профессор, Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация, pas.aicv@gmail.com

Батищева Вита Вячеславовна – студент, Санкт-Петербургский государственный университет, Санкт-Петербург, 199034, Российская Федерация, elokkuu@mail.ru

Воропай Алексей Юрьевич – кандидат технических наук, инженер-программист, ИП «ТраффикДНА Украина», Запорожье, 69001, Украина, voropay.o@gmail.com

Alexei S. Potapov – D.Sc., Associate professor, Professor, ITMO University, Saint Petersburg, 197101, Russian Federation, pas.aicv@gmail.com

Vita V. Batishcheva – student, Saint Petersburg State University, Saint Petersburg, 199034, Russian Federation, elokkuu@mail.ru

Alexei Yu. Voropay – PhD, software engineer, FE «TrafficDNA Ukraine», Zaporizhzhya, 69001, Ukraine, voropay.o@gmail.co