М.Е. Семенкина

Самоадаптивные эволюционные алгоритмы проектирования информационных технологий интеллектуального анализа данных[1]

Аннотация. Для генетического алгоритма и алгоритма генетического программирования предложены новые операторы равномерного скрещивания, реализующие селективное давление на этапе рекомбинации. На тестовых задачах показано, что предложенная модификация повышает эффективность обоих алгоритмов. Разработан и реализован метод самоконфигурирования эволюционных алгоритмов в ходе решения задачи, основанный на подстройке вероятности применения генетических операторов. Предложен способ автоматического генерирования нейронных сетей при помощи алгоритма генетического программирования. Выполнено сравнение с известными аналогами, показавшее высокий уровень эффективности разработанных алгоритмов.

Ключевые слова: генетический алгоритм, алгоритм генетического программирования, равномерное скрещивание, самоконфигурация, символьная регрессия, искусственные нейронные сети классификация.

Введение

Искусственные нейронные сети (ИНС) являются методом обработки данных, который решает большой спектр сложных и важных задач в различных областях, таких как аппроксимация, кластеризация, классификация, управление, прогнозирование и т.д. [9]. Они успешно используются во многих реальных приложениях [17]. В то же время, использование ИНС для конечных пользователей, которые не являются экспертами в области нейросетевого моделирования, является трудным из-за сложности процесса разработки ИНС. Этот процесс состоит из трех частей: определения структуры ИНС (выбор количества нейронов и слоев, соединений, функций активации и т.д.), настройка весовых коэффициентов (обучения ИНС) и проверки. Широкое применение ИНС затрудняется необходимостью разработки каждой ИНС и ее обучения человеком-экспертом, который должен выбрать архитектуру, наиболее подходящую для конкретной задачи. Это приводит к снижению производительности ИНС при моделировании, хотя современные методы разработки ИНС позволяют автоматизировать этот процесс. Наиболее перспективным является эволюционный подход к автоматизации проектирования ИНС. Существует множество интересных и перспективных идей в этой области, подробнее о них можно узнать в [25]. Тем не менее, необходимо подчеркнуть, что в этом случае конечный пользователь вместо сложностей с разработкой и обучением ИНС получает проблемы с применением эволюционных алгоритмов.

В данном исследовании рассматривается адаптивный эволюционный подход для автоматизированного проектирования ИНС, не требующий усилий для настройки самого алгоритма. Этот алгоритм был назван самоконфигурируемым алгоритмом генетического программирования (ГП). Согласно определению, данному организаторами семинара " Self-tuning, self-configuring and self-generating evolutionary algorithms" (Self*EAs) в PPSN XI [18], самоконфигурация представляет собой процесс

[1] Работа выполнена при поддержке ФЦП "Исследования и разработки по приоритетным направлениям развития научно-технологического комплекса России на 2007-2013 годы", НИР 2011-1.9-519-005-042 (ГК № 11.519.11.4002), и ФЦП "Научные и научно-педагогические кадры России", НИР 2011-1.2.1-113-025 (ГК № 16.740.11.0742).

выбора и использования существующих алгоритмических компонентов (таких как варианты операторов). Вместе с тем, в предлагаемом подходе вероятности выбора того или иного оператора (параметры) изменяются в ходе работы в зависимости от эффективности этого оператора, что позволяет говорить, что алгоритмы, описываемые в данной статье, являются частично саморегулирующимися (self-tuning [18]).

Для того чтобы более точно определить место наших алгоритмов, следуя классификациям из [1-3, 7], можно сказать, что они используют динамическую адаптацию на уровне популяции [3] в ходе решения задачи и централизованную форму управления процессом настройки параметров, хотя и несколько отличную от обычных подходов. Вероятности для операторов быть выбранными для порождения очередного потомка адаптируются, исходя из успешности этих операторов на последнем поколении независимо от предыдущих результатов работы. Таким образом снимается типичная для централизованных подходов проблема больших затрат памяти [7]. Данные отличия иногда приводят к ситуациям, когда ни один из операторов не произвел улучшения и, тем не менее, один из них, работавший плохо, но лучше других, будет вознагражден. Вероятности применения операторов не включены в хромосому и не являются объектом эволюции. Все операторы могут быть использованы в течение одного поколения для создания потомков. Время выполнения операторов не возрастает от поколения к поколению, но увеличивается вместе с размером популяции.

Описанные алгоритмы прошли исчерпывающую проверку на тестовых задачах и апробированы на ряде практических задач. Полученные результаты численных экспериментов позволяют сделать вывод о том, что дальнейшее развитие и исследование данного подхода является целесообразным.

1. Оператор равномерного скрещивания с селективным давлением для генетического алгоритма

Оператор равномерного скрещивания является одним из наиболее эффективных операторов рекомбинации в стандартном генетическом алгоритме (ГА) [4, 24]. Так как ГА использует хромосому, представляющую решение задачи, имеющую вид бинарной строки, описать его работу можно следующим образом. Каждый ген хромосомы потомка выбирается из соответствующих генов его родителей с определенной вероятностью. Давно известно, что настройка вероятности передачи гена родителя потомку в равномерном скрещивании может существенно повысить его эффективность [4], а также позволяет эмулировать другие операторы скрещивания (одноточечного, двухточечного). Известно также, что использование оператора равномерного скрещивания позволяет применять так называемую многородительскую рекомбинацию, когда для порождения одного потомка используется более двух родителей. Несмотря на это, в большинстве исследований используются только два родителя и фиксированная вероятность передачи гена, равная 0.5 [5, 8].

В связи с этим представляет интерес модификация оператора равномерного скрещивания, которая повысила бы его эффективность. Прямое применение идеи, предложенной в [4], требует тонкой подстройки вещественного параметра, что может привести к излишним трудностям реализации. Поэтому предлагается ввести селективное давление на этапе скрещивания [22], установив зависимость между вероятностью передачи гена потомку и значением пригодностей родителей. Операторы равномерного скрещивания, основанные на аналогии с обычными операторами селекции - турнирной, пропорциональной и ранговой – были добавлены к обычному оператору равномерного скрещивания, который в данном контексте получил название равновероятного. Очевидно, что турнирное равномерное скрещивание имеет дополнительный смысл только при числе родителей для одного потомка большем, чем 2.

Эффективность ГА с обычными операторами скрещивания и с вновь введенными операторами, а также с многородительской рекомбинацией была оценена на стандартных тестовых задачах оптимизации (Приложение), а также на задачах автоматического проектирования искусственных нейронных сетей (выбор структуры и настройка весовых коэффициентов) и настройки параметров символьных выражений, сгенерированных алгоритмом ГП [22]. После многократных прогонов и статистической обработки полученных численных результатов, были сделаны следующие наблюдения:

- алгоритм с семью родителями превосходит все остальные, причем алгоритм с двумя родителями занимает второе место на тестовых задачах;

- в среднем ранговое равномерное скрещивание является лучшим, равновероятное равномерное скрещивание вторым по эффективности;

- турнирное равномерное скрещивание лучше пропорционального равномерного, одноточечного и двухточечного скрещиваний;

- на двух последних задачах, связанных с нейросетями и генетическим программированием, между операторами скрещивания с семью и двумя родителями нет статистически значимого различия, также как между равновероятным и ранговым равномерным скрещиванием.

Усредненные результаты тестирования приведены в первых шести строках Табл. 1. Она содержит средние, минимальные и максимальные надежности алгоритмов, усредненные по 14 задачам из Приложения, каждая из которых была решена по 1000 раз каждым из вариантов алгоритма. Надежностью называется отношение части запусков алгоритма, в которой была достигнута заданная точность решения, к общему числу запусков. В Табл. 1 заголовки строк обозначают тип скрещивания: одноточечное, двухточечное, равновероятное равномерное, турнирное равномерное, пропорциональное равномерное и ранговое равномерное, соответственно. Колонки «Средняя», «Минимальная» и «Максимальная» означают соответственно среднюю, минимальную и максимальную надежности алгоритмов с данным типом скрещивания и всевозможных сочетаниях остальных операторов (селекции, мутации, замещения). Первое число соответствует значению показателя, усредненного по 14 задачам, числа в скобках показывают разброс значений показателя на различных задачах.

В данной работе при проведении дальнейших исследований используются 3 или 7 родителей для турнирного равномерного скрещивания и 2 или 7 родителей для всех остальных операторов. Также можно заметить, что нельзя исключить из рассмотрения ни один оператор скрещивания, так как каждый из них может быть полезным.

2. Оператор равномерного скрещивания с селективным давлением для алгоритма генетического программирования

В алгоритме генетического программирования используется представление решения в виде дерева, в связи с чем прежде, чем добавлять селективное давление на этапе рекомбинации, необходимо разработать соответствующий оператор равномерного скрещивания. Такой оператор был теоретически обоснован и реализован в [12, 13], позже также была предложена модификация этого оператора для конкретной задачи [15, 16]. Описание такого оператора можно также прочесть в [14].

Согласно [13], выполнение равномерного скрещивания в ГП начинается с вершины дерева и продолжается вниз до нахождения функциональных вершин с различной арностью в одном положении для разных родителей. Каждый узел, находящийся в получившейся общей зоне, может быть передан каждым из родителей с некоторой вероятностью (обычно родителей два и вероятность равна 0.5). Поддеревья, лежащие ниже узлов с разной арностью, передаются вместе со своим верхним узлом в случае, если был выбран данный родитель.

Предлагаемый в данной работе оператор равномерного скрещивания выполняется немного по другому: с заданной вероятностью передачи узлов создается только один потомок, поддеревья узлов с разной арностью соревнуются между собой за право передать потомку свою вершину.

Табл. 1. Надежность ГА на тестовых задачах оптимизации

Скрещивание	Средняя	Минимальная	Максимальная
Одноточечное	0.760/ [0.507,0.915]	0.696/ [0.411,0.856]	0.822/ [0.591,0.978]
Двухточечное	0.479/ [0.132,0.821]	0.413/ [0,0.754]	0.534/ [0.167,0.871]
Равновероятное равномерное	0.819/ [0.442,0.953]	0. 780/ [0.589,0.887]	0.878/ [0.669,0.999]
Турнирное равномерное	0.627/ [0.354,0.967]	0. 587/ [0.299,0.917]	0.697/ [0.38,**1**]
Пропорциональное равномерное	0.647/ [0.276,0.935]	0.622/ [0.232,0.888]	0.718/ [0.3,0.976]
Ранговое равномерное	**0.833/** [0.598,0.974]	0.771/ [0.578,0.935]	0.888/ [0.635,0.999]
SelfCGA	**0.928/** [0.83**, 1**]		

В остальном равномерное скрещивание выполняется так же, как описано в [12]. Описанные модификации добавляют алгоритму гибкости и позволяют надеяться на изменение поведения алгоритма. Напомним также, что равномерное скрещивание позволяет производить многородительскую рекомбинацию.

Имея соответствующий оператор равномерного скрещивания, можно ввести селективное давление тем же способом, как это было сделано выше для генетического алгоритма. Потомок может получить каждый узел от одного из родителей не только с равными, но и с различными вероятностями, определяемыми в этом случае пригодностью родителей одним из трех способов: пропорционально пригодности, согласно рангам или после проведения турнира.

Описанный подход был реализован и протестирован для оценки эффективности разработанных операторов. Так как набор общепринятых тестовых задач для алгоритма генетического программирования все еще относится к "открытым вопросам" [11], то для предварительной оценки были использованы задачи символьной регрессии – 17 тестовых функций, приведенных в Приложении. Каждый алгоритм получал 100 индивидов и 300 поколений для поиска решения и был запущен по 100 раз на каждой функции.

Результаты экспериментов представлены для стандартного ГП и модифицированного алгоритма ГП с новыми операторами равномерного скрещивания (МГП) в двух первых строках Табл. 2. Первые три столбца содержат информацию о надежности алгоритмов, усредненной по 17 тестовым задачам. В первом из них приведены надежность, усредненная по всем настройкам и задачам и в скобках - разброс по задачам надежностей, усредненных по различным вариантам наборов операторов алгоритма. Во втором и третьем столбцах приведены надежности лучшей и худшей из настроек, соответственно, усредненные по 17 задачам, и в скобках - разброс надежностей по задачам. Последняя колонка содержит информацию о количестве ресурсов, необходимых для нахождения первого подходящего по точности решения, усредненную по всем настройкам алгоритма и по 17 задачам, а в скобках - разброс на 17 задачах. Под надежностью в задачах символьной регрессии понимается доля прогонов алгоритма ГП, в которых было найдено символьное выражение со средней квадратической ошибкой, меньшей заданного порога.

Табл. 3 содержит информацию о качестве решений, получаемых алгоритмами ГП. Первая колонка показывает процент точных решений, символьно идентичных тестовым функциям, вторая - процент условно точных решений, для которых необходимо выполнить некоторые элементарные преобразования и округления для достижения символьной идентичности с тестовой функцией. В третьей представлен процент решений, которые не могут быть сведены к символьно идентичным, хотя и дают ошибку меньше заданного порога.

В Табл. 2 и Табл. 3 можно видеть, что ГП с модифицированным равномерным скрещиванием (МГП) немного лучше обычного ГП. Из этого следует, что применение новых операторов имеет смысл. Дополнительные наблюдения практически такие же, как и для ГА: лучшим числом родителей является 7, вторым - 2; лучшим оператором скрещивания является ранговое равномерное, вторым - равновероятное равномерное.

Хотя предложенные новые операторы равномерного скрещивания повышают эффективность ГА и ГП, но, в тоже время, они увеличивают число вариантов настроек алгоритма, что

Табл. 2. Надежность алгоритмов ГП на тестовых задачах символьной регрессии

Алгоритм	Средняя	Минимальная	Максимальная	Среднее поколение
ГП	0.43/ [0.13, 0.77]	0.12/ [0.00, 0.41]	0.60/ [0.27, 0.91]	[33, 289]
МГП	0.53/ [0.31, 0.88]	0.31/ [0.11, 0.56]	0.75/ [0.43, 1.00]	[27, 243]
SelfCGP	0.69/ [0.42, 1.00]			[49, 201]

Табл. 3. Качество решений, получаемых алгоритмами ГП, на тестовых задачах символьной регрессии

Алгоритм	% точных решений	% условно точных решений	% приближенно точных решений
ГП	50	16	34
МГП	58	20	22
SelfCGP	58	16	26

усложняет его применение для конечного пользователя. Поэтому необходимо предложить способы автоматизации выбора генетических операторов, чтобы избежать излишних усилий по настройке алгоритмов, без снижения эффективности их применения.

3. Самоконфигурируемые алгоритмы, основанные на вероятностях применения операторов

Как говорилось выше, предлагается алгоритм с динамической адаптацией вероятностей применения операторов на уровне популяции и с централизованной методикой управления. Чтобы избежать регулирования значений вещественных показателей, будем использовать дискретное множество вариантов алгоритмов – операторов селекции и скрещивания, уровней мутации (высокая, средняя, низкая) и т.п. Каждый из видов операторов имеет свое распределение вероятностей. Например, используются 5 типов селекции (пропорциональная, ранговая и турнирная с тремя размерами турнира). Тогда на старте все вероятности равны 0.2 и меняются согласно некоторому правилу в процессе выполнения алгоритма таким образом, чтобы сумма вероятностей была равна единице и ни одна из вероятностей не была меньше предопределенного минимального значения. В список операторов скрещивания должен быть включен пустой оператор (нет скрещивания), для того чтобы вероятность скрещивания могла быть меньше 1, что используется в стандартных ГА и ГП для моделирования "бездетных пар".

Для создания каждого потомка алгоритм сначала выбирает настройки из списка возможных операторов согласно распределению вероятностей, затем - родителей при помощи выбранного оператора селекции, генерирует потомка отобранных родителей при помощи выбранного оператора скрещивания, мутирует потомка согласно выбранному уровню мутации и помещает его в промежуточную популяцию. После того, как промежуточная популяция заполнена, производится оценка значений функции пригодности и обновляются значения вероятностей для каждого оператора согласно его эффективности на данном шаге. После чего при помощи выбранного оператора замещения формируется новая популяция. Алгоритм останавливается, если кончились выделенные ресурсы или достигнута заданная точность. Псевдокод предлагаемого алгоритма представлен в Табл. 4.

Табл. 4. Псевдокод *для SelfCGA(GP)*

1.	Установить равные вероятности для всех вариантов настройки для каждого вида оператора (исключение - «пустое» скрещивание)
2.	For k=1 to N
3.	For i=1 to NInd
4.	Выбрать вариант селекции, рекомбинации и мутации
5.	Выбрать родителей выбранным оператором селекции
6.	Скрестить родителей выбранным оператором рекомбинации для создания потомка
7.	Мутировать потомка с выбранной вероятностью мутации
8.	Оценить потомка
9.	Выбрать выживших с выбранным оператором замещения
10.	Обновить вероятности для операторов, используя среднюю пригодность потомков, полученных при помощи данного оператора
	Пояснение для шага 1 (установка стартовых вероятностей)
1.	Стартовые вероятности для всех операторов, кроме скрещивания, задаются следующим образом $P_{ij} = \frac{1}{Z_i} \forall\ i = \overline{1,\ 3}\ \forall\ j = \overline{1,\ Z_i}$, где Z_i – число операторов i-го типа, P_{ij} – вероятность применения j-го вида i-го типа оператора
2.	Для скрещивания стартовая вероятность «пустого» скрещивания равна 0.1, 0.9 делится на остальные 6 операторов (одноточечное, двухточечное, равновероятное равномерное, турнирное равномерное, пропорциональное равномерное, ранговое равномерное скрещивания) поровну

Псевдокод метода выбора конкретного оператора для генерации потомка (шаг 4)

1. $\forall\ k = \overline{1,\ N}\ \forall\ i = \overline{1,\ Ind}\ \ \forall\ l = \overline{1,\ 3}$, где Ind – число индивидов
2. Вычисляем границы промежутков для каждого варианта оператора согласно их вероятностям $PP_{l0} = 0,\ \ PP_{l(j+1)} = PP_{lj} + P_{lj}$, где PP_{lj} – границы промежутка для l-го типа оператора
3. Генерируем случайное вещественное число от 0 до 1: $T = random\left(\sum_{j=1}^{Z_l} P_{lj}\right)$
4. Ищем промежуток, в который попало данное случайное число, и номер соответствующего данному промежутку оператора: $T \in \left\lfloor PP_{l(j-1)},\ \ PP_{lj} \right\rfloor$ при $j = \overline{1,\ Z_l}$
5. Выбранный оператор применяется для генерации потомка

Пояснение для шага 10 (перерасчет вероятностей)

1. Для каждого оператора проверить превышают ли его вероятности $\overline{P_{ij}}$ – пороговую вероятность для данного типа оператора. Если:
2. $P_{ij} = \overline{P_{ij}}$ | $P_{lj} < \overline{P_{ij}} + \dfrac{1}{z_l \cdot N}$ и $P_{ij} > \overline{P_{ij}}$ | $P_{lj} > \overline{P_{ij}} + \dfrac{1}{z_l \cdot N}$
3. Тогда вероятность оператора изменяется следующим образом:
4. $P_{lj}^{new} = P_{lj}^{old}$ | $P_{lj}^{new} = \overline{P_{lj}}$ | $P_{lj}^{new} = P_{lj}^{old} - \dfrac{1}{z_l \cdot N}$
5. Выбор оператора получившего самую большую среднюю пригодность сгенерированных с его участием потомков $\max\limits_{j=\overline{1,Z_l}}\left(\dfrac{\sum_j Fit_{ki}}{\sum_j 1}\right) \Rightarrow \bar{j}$, при $k \in \left\lfloor 1,\ \ N \right\rfloor$, где N – число поколений, Fit_{ki} –пригодность i-го индивида на k-м поколении
6. К вероятности «выигравшего» варианта оператора прибавляем все, что было вычтено из вероятностей на шаге 4

Пояснение для шага 10.2(значение пороговых вероятностей)

1. Пороговые вероятности для всех видов операторов, кроме «пустого» скрещивания, задаются следующим образом: $\overline{P_{ij}} = \dfrac{3}{10Z_i} \forall\ i = \overline{1,3}$
2. Пороговая вероятность для «пустого» скрещивания равна 0

Описанный подход может использоваться как для ГА, так и для ГП, а также и для других эволюционных алгоритмов [20, 21].

Результаты оценки эффективности самоконфигурируемого ГА (SelfCGA) на 14 тестовых функциях даны в последней строке Табл. 1. Так как данный алгоритм не имеет дополнительных настроек, то все его результаты приведены в одной колонке.

Худшая надежность по 1000 прогонам равна 0.830, лучшая - 1.000. Надежность SelfCGA, усредненная по 14 функциям, выше, чем лучшая усредненная надежность других алгоритмов. В связи с этим можно сказать, что предложенная модификация полезна, так как дает сопоставимые результаты по надежности, полностью при этом исключая проблему выбора настроек. Т.е., основным преимуществом SelfCGA является отсутствие необходимости в дополнительной настройке алгоритма без потерь в эффективности. Это делает полезным применение алгоритма во многих приложениях, где невозможен выбор настроек перебором или наугад (например, настройка параметров нейронной сети, подбор коэффициентов символьных выражений, сгенерированных ГП) или в случае, когда конечный пользователь не является экспертом в области эволюционных алгоритмов, но применяет ГА для решения своей оптимизационной задачи.

Результаты оценки эффективности самонастраивающегося ГП (SelfCGP) по 17 тестовым функциям даны в последних строках Табл. 2 и Табл. 3. В связи с тем, что данный алгоритм не имеет дополнительных настроек, все его результаты представлены в одной колонке. Худшая надежность по 100 прогонам равна 0.42,

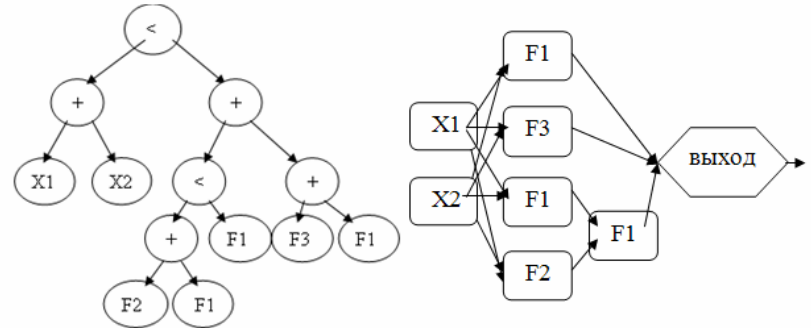

Рис. 1. Пример дерева, сгенерированного представленным алгоритмом генетического программирования, и соответствующей ему нейронной сети

лучшая - 1.00. Надежность SelfCGP, усредненная по 17 задачам, выше, чем лучшая из усредненных надежностей стандартного ГП, и немного меньше, чем лучшая усредненная надежность модифицированного ГП, кроме того, SelfCGP тратит меньше ресурсов, чем альтернативные алгоритмы. Это позволяет рекомендовать SelfCGP для решения задач символьной регрессии как лучшую альтернативу стандартному ГП. Основным достоинством SelfCGP является отсутствие необходимости в настройке его параметров, что делает этот алгоритм весьма полезным в различных приложениях, где конечный пользователь, не будучи специалистом в области эволюционных алгоритмов, тем не менее, применяет ГП для решения своих задач.

Таким образом, в данном разделе была обоснована полезность предложенных модификаций ГА и ГП. Все три варианта ГП применяли при настройке параметров получаемых символьных выражений самонастраивающийся ГА для страховки от последствий неверно выбранных настроек. Однако все эксперименты проводились на тестовых задачах, которые не сопоставимы по сложности с реальными практическими задачами. Теперь проверим предложенные алгоритмы и сравним их с альтернативными при решении сложных практических задач.

4. Алгоритм генетического программирования для автоматического генерирования нейронных сетей

Как правило, алгоритмы генетического программирования работают с представлением в виде дерева. При проектировании алгоритма определяется функциональное и терминальное множества, выбираются конкретные подходящие для выбранного представления операторы преобразования решений (способ селекции, оператор скрещивания, мутации и т.д.), а также критерий остановки [14].

Терминальное множество предлагаемого алгоритма генетического программирования для автоматического генерирования нейронных сетей включает в себя входные нейроны и 15 функций активации. Функциональное множество включает в себя операции по размещению нейронов и установке нейронных связей. Первая операция размещает нейрон или группу нейронов в одном слое, не создавая дополнительных соединений. Вторая операция размещает нейроны или группы нейронов в последовательные слои таким образом, чтобы нейрон (группа нейронов) их левой ветви дерева предшествовал нейрону (группе нейронов) из правой ветви дерева. В этом случае, будут добавлены новые связи, которые соединяют нейроны из левой ветви дерева с нейронами из правой ветви дерева. Входные нейроны не могут принимать сигнал, но должны послать его, по крайней мере, на один скрытый нейрон. В некоторых случаях алгоритм генетического программирования не включает некоторые входные нейроны в результирующие дерево, т. е. структура ИНС может получать решение с высокой эффективностью не используя все входы. Эта особенность данного подхода допускает использование предлагаемого ГП для выбора комбинации наиболее информативных входов задачи [19]. На Рис. 1 показан пример соответствующих друг другу дерева и нейронной сети.

Алгоритм генетического программирования порождает дерево, представляющее собой структуру ИНС. Обучение ИНС осуществляется с целью оценки его пригодности, определяемой

качеством решения имеющейся задачи, например, точность приближения или количество неправильно классифицированных случаев. Весовые коэффициенты ИНС настраиваются при помощи самоконфигурируемого генетического алгоритма (SelfCGA). Лучшая найденная алгоритмом генетического программирования структура ИНС дополнительно обучается при помощи гибридизации SelfCGA с локальным поиском.

5. Сравнение эффективности самоконфигурируемого алгоритма генетического программирования с альтернативными подходами

Сравнение эффективности SelfCGP для проектирования различных интеллектуальных информационных технологий со стандартным ГП на тестовых задачах показало, что они сопоставимы по надежности. В этом разделе предложенные алгоритмы будут апробированы при решении реальных практических задач, по результатам их решения будет проведено сравнение с альтернативными подходами.

Задачи классификации будут решены посредством построения разделяющих поверхностей, представленных символьными выражениями, т.е. с применением алгоритма SelfCGP для символьной регрессии (SelfCGP+SRF), а также при помощи ИНС, автоматически сгенерированных алгоритмом SelfCGP (SelfCGP+ANN).

Во время экспериментов использовались 100 индивидов и максимум 500 поколений на каждом из 20 запусков. Для каждого запуска набор данных делился случайным образом на обучающее и проверочное множество в пропорциях 70 и 30% соответственно. Все результаты были усреднены. SelfCGA использовался для настройки параметров получаемых символьных выражений обоими предложенными методами (МГП и SelfCGP+SRF), а также для обучения ИНС, получаемых алгоритмом SelfCGP+ANN.

Первый эксперимент был проведен для сравнения эффективности подходов, основанных на SelfCGP между собой. В качестве испытательного полигона использовались три задачи (ирисы, рак молочной железы, диабет) из [6]. Материалы для сравнения были взяты из работы [26], где вместе с результатами авторов алгоритма (CROANN) были представлены результаты 15 других подходов на этих трех задачах классификации, причем алгоритм CROANN превосходил их по эффективности. Числа в Табл. 5 рассчитывались как мера ошибки согласно [26].

Табл.5. Сравнение алгоритмов классификации

Классификаторы	Ирисы	Рак	Диабет
SelfCGP+SRF	2.63	1.23	20.01
SelfCGP+ANN	1.30	1.05	19.69
CROANN	1.31	1.06	19.67

По результатам экспериментов можно увидеть, что SelfCGP+ANN превосходит SelfCGP+SRF на всех трех решаемых задачах и сопоставим с CROANN.

В рамках второго численного эксперимента решались две более сложные задачи классификации, а также было произведено сравнение результатов предложенных алгоритмов с альтернативными подходами.

Первый набор данных о кредитах в Германии включает в себя сведения о заемщиках – 20 входных переменных, таких как возраст, пол, кредитная история, работа, цель получения кредита и другая персональная информация. 700 заемщиков являются надежными получателями кредита и 300 ненадежными, т.е. 700 человек должны получить кредит, а 300 не должны. Второй набор данных о кредитах в Австралии включает в себя 307 примеров надежных заемщиков и 383 ненадежных, содержит 14 входов, 6 из которых вещественные и 8 категориальные. Оба набора данных можно найти в свободном доступе на сайте репозитория данных для автоматического обучения от Калифорнийского университета [6]. Они часто используются для сравнения точности различных методов классификации.

Результаты альтернативных подходов были найдены в научной литературе. В [10] приведены результаты оценки эффективности на этих двух наборах данных для двухэтапного ГП (2SGP), обычного ГП, многослойного персептрона (MLP), деревьев классификации и регрессии (CART), деревьев принятия решений C4.5, метода k ближайших соседей (k-NN), линейной регрессии (LR). Дополнительные материалы для сравнения взяты из [23], где представлены результаты оценки для автоматически генерируемого нечеткого классификатора (Fuzzy classifier), байесовского подхода, бустинга, бэггинга, метода случайных подпространств (RSM), кооперативно-коэволюционого обучения ансамблей (CCEL).

Результаты сравнения (часть правильно классифицированных примеров на проверочном множестве) представлены в Табл. 6.

Лучшие результаты в данном сравнении выделены в Табл. 6 жирным шрифтом. Победивший подход 2SGP из [10] был специально разработан для решения задачи банковского скоринга. Второе место на задаче банковского скоринга из Австралии делят предложенные в данной статье SelfCGP для решения задач символьной регрессии и для автоматического генерирования ИНС. При решении задачи банковского скоринга из Германии второе место занял SelfCGP для автоматического генерирования ИНС, а третье - SelfCGP для решения задач символьной регрессии. Другой алгоритм (МГП) занял пятое место на обеих задачах.

Данное сравнение достаточно условно, так как отсутствуют данные о вариациях показателя и затраченных вычислительных ресурсах, причем некоторые результаты являются лучшими для соответствующего метода, а другие – усредненными. Более того, очевидно, что 2SGP и нечеткий классификатор более полезны, так как дают не только вычислительную процедуру, но и понятные для специалистов правила. Однако в задачи данной работы не входило построение оптимальной системы для банковского скоринга. Целью было показать, что предложенный подход может давать результаты, сравнимые по эффективности с аналогами, без затраты дополнительных усилий на адаптацию к конкретной задаче (здесь - банковский скоринг). Именно по этой причине не вносились изменения, призванные помочь алгоритму справиться с данными задачами, например, было выделено только 500 поколений в отличие от 1000 в [10] и т.д.

В результате проделанной работы можно сделать вывод о том, что SelfCGP, использующий SelfCGA для настройки параметров символьных выражений, и SelfCGP, автоматически генерирующий ИНС, предложенные в данной статье, демонстрируют свою конкурентоспособность, имеют стандартный недостаток для универсальных подходов, т.е. проигрывают алгоритмам, предназначенным для конкретных задач, на соответствующих задачах, и, в то же время, определенные преимущества, позволяя избегать затрат интеллектуальных и вычислительных ресурсов на настройку алгоритмов без снижения эффективности.

Табл. 6. Сравнение алгоритмов классификации

Алгоритм	Кредит в Австралии	Кредит в Германии
SelfCGP+SRF	0.9022	0.7950
SelfCGP+ANN	0.9022	0.7954
GP+ANN	0.8969	0.7863
МГП	0.8985	0.7875
2SGP	**0.9027**	**0.8015**
ГП	0.8889	0.7834
Fuzzy classifier	0,8910	0,7940
C4.5	0.8986	0.7773
LR	0.8696	0.7837
Байес. подход	0,8470	0,6790
Бустинг	0,7600	0,7000
Бэгинг	0,8470	0,6840
RSM	0,8520	0,6770
CCEL	0,8660	0,7460
k-NN	0.7150	0.7151
СНГП	0.8744	0.7565

Заключение

В данной статье предложены специальные типы операторов равномерного скрещивания, вводящие селективное давление на этапе рекомбинации для ГА и ГП. На тестовых задачах показано, что данные операторы дают положительный эффект как для ГА, так и для ГП. В частности ранговое равномерное скрещивание в среднем является лучшим среди остальных рассмотренных операторов скрещивания.

Кроме того, был представлен подход для самоконфигурирования алгоритмов, основанный на вероятностном выборе операторов для каждого потомка и обновлении вероятностей на каждом поколении по результатам работы на данном поколении. Данный подход также продемонстрировал высокую работоспособность на тестовых задачах, как для ГА, так и для ГП.

Также был предложен подход для автоматического генерирования ИНС на основе SelfCGP, который был апробирован на ряде задач классификации и продемонстрировал свою высокую эффективность при решении задач анализа данных.

Таким образом, в данной работе предложен эффективный инструментарий, позволяющий автоматически генерировать алгоритмическое ядро информационных технологий интеллектуального анализа данных с помощью адаптивных эволюционных алгоритмов, не требующих выбора настроек и определения оптимальных параметров.

Приложение. Тестовые задачи оптимизации и моделирования

№	Функции	Интервалы
1	$I(x) = 0.05(x-1)^2 + (3 - 2.9 \cdot e^{-2.77257 \cdot x^2})(1 - \cos(x(4 - 50 \cdot e^{-2.77257 \cdot x^2})))$	$x \in [-1;1]$
2	$I(x) = 1 - 0.5\cos(1.5(10x - 0.3))\cos(31.4x) + 0.5\cos(\sqrt{5} \cdot 10x)\cos(35x)$	$x \in [-1;1]$
3	$I(x,y) = 0.1x^2 + 0.1y^2 - 4 \cdot \cos(0.8 \cdot x) - 4 \cdot \cos(0.8 \cdot y) + 8$	$x, y \in [-16;16]$
4	$I(x,y) = (0.1 \cdot 1.5 \cdot y)^2 + (0.1 \cdot 0.8 \cdot x)^2 - 4 \cdot \cos(0.8 \cdot 1.5 \cdot y) - 4 \cdot \cos(0.8 \cdot 0.8 \cdot x) + 8$	$x, y \in [-16;16]$
5	$I(x,y) = 100 \cdot (y - x^2)^2 + (1-x)^2$	$x_1, x_2 \in [-2;2]$
6	$I(x,y) = \dfrac{-10}{0.005 \cdot (x^2 + y^2) - \cos(x) \cdot \cos(y/\sqrt{2}) + 2} + 10$	$x_1, x_2 \in [-16;16]$
7	$I(x,y) = \dfrac{-100}{100(x^2 - y) + (1-x)^2 + 1} + 100$	$x_1, x_2 \in [-5;5]$
8	$I(x,y) = \dfrac{1 - \sin^2(\sqrt{x^2 + y^2})}{1 + 0.001 \cdot (x^2 + y^2)}$	$x_1, x_2 \in [-10;10]$
9	$I(x_1,x_2) = 0.5(x_1^2 + x_2^2)\left[2 * 0.8 + 0.8\cos(1.5x_1)\cos(3.14x_2) + 0.8\cos(\sqrt{5}x_1)\cos(3.5x_2)\right]$	$x_1, x_2 \in [-2.5;2.5]$
10	$I(x_1,x_2) = 0.5(x_1^2 + x_2^2)\left[2 * 0.8 + 0.8\cos(1.5x_1)\cos(3.14x_2) + 0.8\cos(\sqrt{5}x_1)\cos(3.5x_2)\right]$	$x_1, x_2 \in [-5;5]$
11	$I(x_1,x_2) = x_1^2\lvert\sin 2x_1\rvert + x_2^2\lvert\sin 2x_2\rvert - \dfrac{1}{(5x_1^2 + 5x_2^2 + 0.2)} + 5$	$x_1, x_2 \in [-4;4]$
12	$I(x_1,x_2) = 0.5(x_1^2 + x_1x_2 + x_2^2)[1 + 0.5\cos(1.5x_1)\cos(3.2x_1x_2)\cos(3.14x_2) + 0.5\cos(2.2x_1)\cos(4.8x_1x_2)\cos(3.5x_2)]$	$x_1, x_2 \in [0;4]$
13	$I(x_1,x_2) = -z(x_1)z(x_2)$, $z(x) = -\dfrac{1}{(x-1)^2 + 0.2} - \dfrac{1}{2(x-2)^2 + 0.15} - \dfrac{1}{3(x-3)^2 + 0.3}$	$x_1, x_2 \in [0;4]$
14	$I(x_1,x_2) = z(x_1) + z(x_2)$, $z(x) = -\dfrac{1}{(x-1)^2 + 0.2} - \dfrac{1}{2(x-2)^2 + 0.15} - \dfrac{1}{3(x-3)^2 + 0.3}$	$x_1, x_2 \in [0;4]$
15	$I(x_1,x_2) = (x_1 - 2)^2 + (x_2 - 1)^2$	$x_1, x_2 \in [-5;5]$
16	$I(x_1) = \sin(x_1)x_1^2$	$x_1 \in [-5;5]$
17	$I(x_1) = \sin(x_1) + x_1$	$x_1 \in [-5;5]$

Литература

1. Angeline P.J. Adaptive and self-adaptive evolutionary computations. In: Palaniswami M. and Attikiouzel Y. (Eds.) Computational Intelligence: A Dynamic Systems Perspective, pages 152–163. IEEE Press, 1995.
2. Eiben A.E., Hinterding R., and Michalewicz Z. Parameter control in evolutionary algorithms. IEEE Transactions on Evolutionary Computation, 3(2):124–141, 1999.
3. Meyer-Nieberg S., Beyer H.-G. Self-Adaptation in Evolutionary Algorithms. In F. Lobo, C. Lima, and Z. Michalewicz (Eds.) Parameter Setting in Evolutionary Algorithm, pp. 47-75, 2007.
4. De Jong K.A., Spears W. On the Virtues of Parameterized Uniform Crossover. In: Proceedings of the 4th International Conference on Genetic Algorithms, Morgan Kaufmann, July, 1991.
5. Eiben A.E., Smith J.E. Introduction to evolutionary computing. Springer-Verlag, Berlin, Heidelberg, 2003.
6. Frank, A., Asuncion, A. (2010). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.
7. Gomez J. Self Adaptation of Operator Rates in Evolutionary Algorithms. In Deb K. et al. (Eds.) GECCO 2004, LNCS 3102, pp. 1162–1173, 2004.
8. Haupt R.L., Haupt S.E. Practical genetic algorithms. John Wiley & Sons, Inc., Hoboken, New Jersey, 2004.
9. Haykin S. Neural Networks (2nd ed.). Prentice Hall, 1999.
10. Huang J.-J., Tzeng G.-H., Ong Ch.-Sh. Two-stage genetic programming (2SGP) for the credit scoring model // Applied Mathematics and Computation, 174 (2006): 1039–1053.
11. O'Neill M., Vanneschi L., Gustafson S., Banzhaf W. Open issues in genetic programming. In: Genetic Programming and Evolvable Machines (2010) 11:339–363.

12. Poli R., Langdon W.B. On the Ability to Search the Space of Programs of Standard, One-Point and Uniform Crossover in Genetic Programming. – Technical Report CSRP-98-7. - The University of Birmingham (UK), 1998. – 21 p.
13. Poli R., Langdon W.B. On the search properties of different crossover operators in genetic programming. In J. R. Koza, et al., editors, Genetic Programming 1998:Proceedings of the Third Annual Conference, pages 293–301, University of Wisconsin, Madison, Wisconsin, USA, 22-25 July 1998.
14. Poli R., Langdon W.B., McPhee N.F. A Field Guide to Genetic Programming. - Published via http://lulu.com and freely available at http://www.gp-field-guide.org.uk, 2008. (With contributions by J. R. Koza).
15. Poli R., Page J., Langdon W.B. Smooth uniform crossover, sub-machine code GP and demes: A recipe for solving high-order boolean parity problems. In W. Banzhaf, et al., editors, Proceedings of the Genetic and Evolutionary Computation Conference, volume 2, pages 1162-1169, Orlando, Florida, USA, 1999.
16. Poli R., Page J. Solving high-order boolean parity problems with smooth uniform crossover, sub-machine code GP and demes. In Genetic Programming and Evolvable Machines, 1 (1/2):37-56, 2000.
17. Rabunal J.R., Dorado J. (Eds) Artificial Neural Networks in Real-Life Applications. Idea Group Inc., 2005.
18. Schaefer R., Cotta C., Kołodziej J., Rudolph G. (Eds.) Parallel Problem Solving from Nature – PPSN XI 11th International Conference, Kraków, Poland, September 11-15, 2010.
19. Semenkin, E., Semenkina, M. Artificial Neural Network Design with Self-Configuring Genetic Programming Algorithm. In: Filipic, B., Silc J. (eds.). Bioinspired Optimization Methods and their Applications: proceedings of the 5th International Conference. – BIOMA 2012, Bohinj, Slovenia. – Jozef Stefan Institute, 2012. – P. 291-300.
20. Semenkin, E., Semenkina, M. Self-configuring Genetic Algorithm with Modified Uniform Crossover Operator. In: Y. Tan, Y. Shi, and Z. Ji (Eds.): ICSI 2012, Part I, LNCS 7331, pp. 414–421, 2012. - Springer-Verlag Berlin Heidelberg 2012.
21. Semenkin, E., Semenkina, M. Self-Configuring Genetic Programming Algorithm with Modified Uniform Crossover. In: Proc. of IEEE Congress on Evolutionary Computation. IEEE World Congress on Computational Intelligence, Brisbane, Australia, 2012.
22. Semenkin E.S., Semenkina M.E. Application of GA with modified uniform recombination operator for automated implementation of intellectual information technologies. In: Vestnik. Scientific Journal of Siberian State Aerospace University named after academician M. F. Reshetnev. – 2007. – Issue 3 (16). – Pp. 27-32. (In Russian, abstract in English).
23. Sergienko R., Semenkin E., Bukhtoyarov V. Michigan and Pittsburgh Methods Combining for Fuzzy Classifier Generating with Coevolutionary Algorithm for Strategy Adaptation. In: 2011 IEEE Congress on Evolutionary Computation (CEC'2011), June 5-8, 2011, New Orleans, LA.
24. Syswerda G. Uniform crossover in genetic algorithms. In J. Schaffer, editor, Proceedings of the Third International Conference on Genetic Algorithms. Morgan Kaufmann, 1989.
25. Yao X. Evolving Artificial Neural Network. In: Proceedings of the IEEE, vol. 87, No 9:1423-1447, September 1999.

Yu J.J.Q., Lam A.Y.S., Li V.O.K. Evolutionary Artificial Neural Network Based on Chemical Reaction Optimization In: 2011 IEEE Congress on Evolutionary Computation (CEC'2011), New Orleans, LA., 2011.

Семенкина Мария Евгеньевна. Старший преподаватель Сибирского государственного аэрокосмического университета им. академика М.Ф. Решетнева. Окончила Сибирский государственный аэрокосмический университет им. академика М.Ф. Решетнева в 2012 году. Автор 30 печатных работ. Область научных интересов: эволюционные алгоритмы, интеллектуальный анализ данных, поддержка принятия решений. E-mail: semenkina88@mail.ru