**Метрика NSUB.** Кількість підсистем забезпечує розуміння наступних питань: розміщення ресурсів, планування, загальні затрати на інтеграцію. Кількісне значення ризику $\mathbf{P_{NSUB}}$ для відповідних значень метрики визначають згідно з графіком, наведеним на рисунку 5, *б*. Значення $\mathbf{P_{NSUB}}$ дозволяє визначити ранг ризику, пов'язаному зі складністю ПЗ.

Згідно зі статистичним способом підрахунку ймовірності та на основі вищевикладеного, ми маємо можливість визначити формули для отримання кількісних характеристик ризиків, пов'язаних зі складністю програмного продукту, що розробляється, та його функціональністю. Отже, маємо формулу для визначення $\mathbf{P_{СКЛ}}$:

$$P_{СКЛ} = \frac{P_{CS} + P_{NOO} + P_{NOA} + P_{SI} + P_{NKC} + P_{NSUB} + K_{NP}^{C} P_{NP}}{7}, \quad (2)$$

де $K_{NP}^{C}$ – коефіцієнт впливу метрики $P_{NP}$ на складність програмного продукту.

Формула для визначення $\mathbf{P_{ФУНК}}$ має вигляд:

$$P_{ФУНК} = \frac{P_{OC} + P_{OS} + K_{NP}^{Ф} P_{NP}}{3}, \quad (3)$$

де $K_{NP}^{Ф}$ – коефіцієнт впливу метрики $P_{NP}$ на функціональність програмного продукту. Коефіцієнти $K_{NP}^{C}$ та $K_{NP}^{Ф}$ пов'язані формулою

$$K_{NP}^{C} = 1 - K_{NP}^{Ф}. \quad (4)$$

**Висновки дослідження:**

- застосування метрик для оцінки об'єктно-орієнтованого програмного забезпечення дає можливість оцінити кількісні показники ризиків програмного проекту;
- метрики набору Лоренца-Кідда дають можливість визначити кількісні значення параметрів ризиків, пов'язаних зі складністю та функціональністю програмного забезпечення;
- застосування набору метрик Лоренца-Кідда спрощує ідентифікацію ризиків та дозволяє приймати управлінські рішення, ґрунтуючись на чітко визначених кількісних даних.

**Список використаних джерел**

1. Project Management Body of Knowledge (PMBOK), PMI Standard Committee.
2. Акименко А. М. Використання UML під час проектування складних програмних систем / А. М. Акименко // Вісник Чернігівського державного технологічного університету. Серія "Технічні науки": наук. зб. – 2011. – № 49. – С. 164-170.
3. Акименко А. М. Ідентифікація ризиків програмного проекту / А. М. Акименко // Вісник Чернігівського державного технологічного університету. Серія "Технічні науки": наук. зб. – 2011. – № 53. – С. 171-175.

УДК 004.02

**А.Н. Волокита**, канд. техн. наук
**Ву Дык Тхинь**, аспирант
**А.Ю. Якушев**, студент
Национальный технический университет Украины «КПИ», г. Киев, Украина

## ОБНАРУЖЕНИЕ ВТОРЖЕНИЙ В РАСПРЕДЕЛЕННЫЕ КОМПЬЮТЕРНЫЕ СИСТЕМЫ НА ОСНОВЕ ГЕНЕТИЧЕСКОГО ПРОГРАММИРОВАНИЯ

*В данной статье предложено использование генетического программирования (ГП) для решения задачи обнаружения вторжений в распределенные компьютерные системы. Сделано анализ существующих методов оптимизации выполнения ГП, проведено экспериментальное исследование эффективности ГП при поиске символьного выражения третьего порядка.*

**Постановка проблемы.** Распределенная компьютерная система (РКС) – группа объединенных в сеть компьютеров, представляющаяся их пользователям единой объединен-

ной системой, целью которой является решение различных задач (например, реализация облачного хранилища или Platform-As-A-Service хостинга). Обеспечение безопасности РКС должно быть организовано на отдельном сервере, который управляет агентами безопасности на узлах РКС. Каждый из узлов с определенной частотой посылает серверу сведения о состоянии системы в виде количественных характеристик безопасности. Задача сервера – определить по полученным характеристикам вероятность того, что узел сети скомпрометирован, и известить об этом администратора РКС.

Задача определения вторжения по косвенным характеристикам относится к разряду эвристических, так как не существует очевидной зависимости между значениями характеристик и вероятностью вторжения. Некоторые характеристики могут быть не важны, некоторые влиять только при определенных условиях (например, при определенных значениях других характеристик), и у каждой из них различные весовые коэффициенты.

Решение задач определения вторжения с помощью регрессионной модели представляет собой поиск такого выражения и его коэффициентов, чтобы входные параметры, подставленные в найденное выражение, выдавали соответствующие выходные значения. Символьная регрессия подразумевает топологический поиск решения параллельно с параметрическим, то есть изменение структуры решения.

Генетическое программирование (ГП) является одним из методов решения задач символьной регрессии с возможностью создавать структуру решения во время выполнения, и представлением решений в виде деревьев, что совпадает со структурной записью символьных выражений.

**Анализ исследований и публикаций.** В настоящее время зарегистрировано (и подкреплено соответствующими патентами) 36 случаев, когда решение, произведенное с помощью генетического программирования, превзошло, повторило либо улучшило существующее человеческое изобретение [1]. Решения относятся к следующим областям: автоматическое создание аналоговых электрических схем, метаболических цепей, синтез антенн.

Группе исследователей из Мадридского университета удалось успешно применить генетическое программирование в анализе хеш-функций на случайность [2]. Для того чтобы удостовериться в надежности хеш-функции на предмет наличия определенного вида зависимостей между выходным значением и входными данными, для функций была запущена сессия ГП с большим максимальным поколением. Функции, для которых не было найдено ни одного похожего индивида в пределах допустимого сходства, считались безопасными.

**Цель статьи.** Целью статьи является разработка алгоритма на основе символьной регрессии для вероятностной оценки возможности атаки на распределенную компьютерную систему в режиме реального времени, применяя метод генетического программирования.

На рис. 1 показана РКС, состоящая из M узлов с агентами безопасности, каждый из которых посылает серверу значения N различных характеристик.

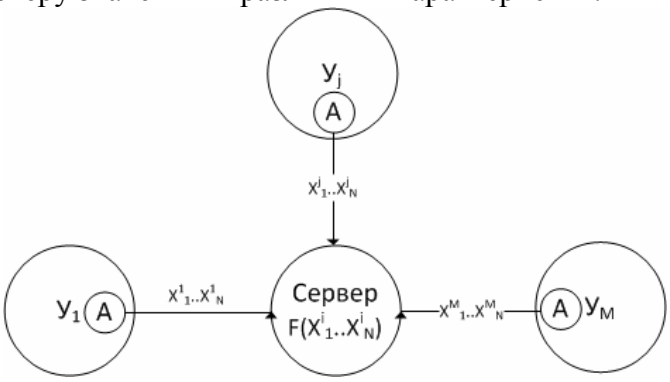


*Рис. 1. Схема обеспечения обнаружения атак на РВС. А – агент безопасности, У1..УМ – узлы РКС, Xi1..XiN – значения характеристик i-го узла*

Сервер безопасности получает от каждого узла набор характеристик узла $X^i_1..X^i_N$, далее для этих характеристик применяется функция F, которая возвращает вероятность того, что данный узел является скомпрометированным (взломанным злоумышленником). Нахождение такой функции – трудоемкий процесс, который невозможно выполнить аналитическими методами, поэтому в данной работе предлагается использование символьной регрессии в качестве средства нахождения функции. При этом для выполнения символьной регрессии будет использоваться алгоритм генетического программирования.

**Изложение основного материала.** Генетическое программирование и эволюционное вычисление зарекомендовали себя как хороший способ для постройки топологии системы и ее настройки. Генетическое программирование показывает прекрасные результаты в нетривиальных сложных задачах, так как использует нетривиальные методы и подходы, до которых бы с трудом додумался человек, специализирующийся в этой области. Таким образом, у генетического программирования и эволюционных вычислений зафиксировано немало случаев удачного применения, а также огромный нераскрытый потенциал для использования в других, еще не освоенных, областях.

При решении задач с помощью генетического программирования следует особое внимание уделять сходимости алгоритма. Сходимость алгоритма – способность алгоритма приводить к результату за конечное число шагов. Скорость сходимости алгоритма – один из важных показателей качества математических моделей, предназначенных для решения задач на ЭВМ. Обычно скорость оценивается количеством итераций, необходимых для получения искомого решения. Для итерационных алгоритмов скорость сходимости является критерием эффективности, по которому зачастую и выбирается алгоритм для решения конкретной задачи.

Так же, как и остальные недетерминированные эвристические и метаэвристические алгоритмы многокритериального поиска, генетическое программирование не гарантирует сходимости и получения точного результата для конкретного запуска алгоритма. Существует ненулевая вероятность того, что алгоритм остановится в точке локального максимума на время достаточно большое, чтобы считать запуск алгоритма неудачным. В таком случае популяции или дается шанс выбраться из локального максимума (что для такой популяции означает ожидание удачной случайной мутации), или алгоритм запускается заново. Скорость сходимости генетического программирования также нельзя определить аналитически, она сильно зависит от специфики конкретной задачи и выбранных стратегий работы самого алгоритма.

Для генетического программирования и генетических алгоритмов в целом существует проблема так называемой преждевременной сходимости. Преждевременная сходимость – это ситуация, когда фитнес-доминантные индивиды (представители локального максимума) вытесняют из популяции особей, которые несут полезные гены, но обладают меньшим показателем фитнеса [3].

Следуя из вышесказанного, при использовании генетического программирования необходимо решить две противоположные задачи: максимизировать скорость сходимости и предотвратить преждевременную сходимость алгоритма. Для решения этих задач существуют теоретически обоснованные и экспериментально испытанные методики и принципы конфигурации систем генетического программирования.

Выбор таких характеристик как размер популяции, процент особей, участвующих в репродукции, частота мутаций не оказывает особого влияния на скорость сходимости генетического программирования и сходимость в целом [4]. Тем не менее, размер популяции должен быть достаточным для того, чтобы:

- обеспечивать разнообразие в начальной популяции, так как подавляющие большинство свойств будущие поколения получат именно из начальной популяции;

- сохранять индивидуумов, значение фитнеса которых ниже, чем у представителей локального максимума в популяции, но которые обладают нужными генами.

Дэвидом Фоджелом была выведена формула минимального размера популяции необходимого для работоспособности генетического алгоритма [3]:

$$1+\log_2 \frac{1}{1-P^{\frac{1}{L}}},$$

где $P$ – требуемая вероятность того, что случайный набор хромосом будет содержать все необходимые гены для каждого локуса; $L$ – длина хромосомы. Поскольку в задачах генетического программирования поиск хромосомы осуществляется не линейно, а в пространстве (представленном в виде бинарного дерева), то вместо длины хромосомы в формуле нужно использовать $2^L$.

В теории генетических алгоритмов существуют множество методов отбора особей для репродукции. Каждый из методов может быть более или менее эффективным для конкретной задачи, поэтому правильный выбор стратегии отбора повлечет за собой уменьшения времени сходимости. Существуют следующие основные стратегии отбора индивидов [5]:

- отбор, пропорциональный фитнесу (или рулетка) – сначала высчитывается значение фитнеса для каждой особи, нормализуется, затем популяция сортируется по возрастанию нормализованного значения фитнеса, после этого каждому индивиду присваивается аккумулированное значение приспособленности (его собственное значение фитнеса плюс суммарное значение фитнеса всех индивидов до него). Затем выбирается случайное число R от 0 до 1, и выбирается первый индивид, который обладает большим аккумулированным значением приспособленности, чем R;
- стохастический сэмплинг – модификация рулетки, в которой с помощью одного случайного числа через равные промежутки из популяции выбираются все необходимые индивиды. Этот метод лишен предвзятости и имеет минимальный разброс, что связано с уменьшением роли случая в конечном отборе;
- турнирный метод – по очереди из популяции случайным образом выбираются особи, между которыми проводится «турнир». Победитель каждого турнира выбирается для скрещивания. Преимущество этого метода в том, что изменение размера турнира позволяет настроить строгость селекции. Чем больше размер турнира, тем меньше шансов у слабого индивида быть отобранным.

Существуют также другие стратегии отбора индивидуумов для скрещивания. Например, часто используемым является элитарный метод, при котором самый сильный индивид в популяции гарантировано без изменений переходит в следующее поколение.

Существует еще один метод улучшения сходимости ГП. Это автоматически определяемые функции (*англ.* automatically defined functions, ADF) – эксклюзивный для генетического программирования (из всех эволюционных алгоритмов) метод, позволяющий контролировать сложность индивидов в популяции, а следовательно упростить поиск и ускорить сходимость. Автоматически определенные функции позволяют индивиду повторно использовать повторяющиеся шаблоны и формы хромосом вместо того, чтобы эволюционно получать их заново в каждом необходимом локусе.

ADF (одна или несколько) развиваются вместе с индивидом, могут в процессе эволюции появляться, исчезать, менять свою арность (количество аргументов), передаваться через скрещивания целиком или частично другим индивидам, изменять свою структуру с помощью генов, полученных от других индивидов. Иными словами, ADF – полноправная часть особи.

Следует отметить, что использование ADF для задач, где повторения генов маловероятны, может только снизить скорость сходимости. Для простых задач следует избегать использования ADF, так как одиночно встречающиеся двух- или трехкратные повторения генов не оправдывают увеличения индивидуумов почти в два раза.

**Эксперименты.** Целью эксперимента является попытка с помощью генетического программирования найти неизвестную функцию, которая прямым образом задает функцию фитнеса для генетического эксперимента.

Функция представляет собой бинарное дерево, в вершинах веток которого могут находиться терминальные операции из заданного списка (в нашем случае, это операции сложения, вычитания, умножения и защищенного деления), а в конечных узлах дерева – терминальные символы (в нашем случае, переменная x и случайные действительные числа). Если принять, что количество терминальных символов в поставленной задаче – s, количество терминальных операций – p, а максимальная глубина бинарного дерева функции – n, то сложность алгоритма полного перебора такой функции:

$$O(n) = (sp)^{2^{n-1}}.$$

Исходя из этого, и приняв значения переменных p = 4, s = 5, n = 8 (половина стандартной максимальной глубины дерева решения по [4]), количество элементов полного перебора равно $10^{165}$.

В качестве языка для решения задачи был выбран язык Clojure, как язык из подсемейства Lisp-языков. Причиной такого выбора стала необходимость использования в программе символьной алгебры и требования к динамичности для быстрого написания прототипа программы. Алгоритм, который использует данная программа, состоит из следующих этапов:

1. Сгенерировать начальную популяцию (в данном случае – набор случайных бинарных деревьев).
2. Проверить, нет ли в популяции особи, полностью отвечающей искомой функции. Если такая особь существует или превышено граничное время, завершить работу, предъявив найденную особь.
3. Каждая особь в популяции сравнивается с функцией фитнеса путём их интегрирования на определенном участке и сравнения дельты площади.
4. Особи в популяции сортируются в порядке убывания схожести с фитнес-функцией.
5. Происходят процессы мутации, кросс-овера и репликации. Чем больше особь похожа на функцию фитнеса, тем выше у нее шанс стать участником кросс-овера или реплицироваться в следующей популяции.
6. Повторяется шаг 2.

Программа является демонстрацией принципиальной возможности рассматриваемого подхода для решения задачи обнаружения вторжений, но не является полноценным решением нахождения арбитрарной функции в заданном условием контексте. В программе присутствуют такие недостатки, как недостаточно сложный механизм нормализации полученной функции и низкая скорость выполнения базовых математических операций из-за динамичности языка.

Для проверки была случайно выбрана функция $y(x) = 3x^4 - 3x + 1$ или в виде бинарного дерева:

$$\mathrm{y} = \mathrm{add}(\mathrm{sub}(\mathrm{mul}(\mathrm{mul}(\mathrm{mul}(\mathrm{x},\mathrm{x}),\mathrm{mul}(\mathrm{x},\mathrm{x})),3),\mathrm{mul}(3,\mathrm{x})),1).$$

Стоит отметить, что выбранная для эксперимента функция полностью случайна, и ее форма и значения коэффициентов никаким образом не свидетельствуют об ограниченности генетического программирования или предрасположенности к решению именно таких функций.

Терминальными операциями являлись операции сложения, вычитания, умножения и защищенного деления; терминальными символами – переменная x (с вероятностью выпадения 50 %) и случайные действительные числа. Размер популяции был равен 20 особям. За одну смену поколения в популяции происходило 10 репликаций, 2 мутации и 10 кросс-оверов (каждый из них дающий по 2 потомка). Количество поколений, являющееся граничным (если идеальный индивид так и не был найден) равно 1000.

После прохождения 100 тестов среднее значение приведенного фитнеса было равно 0,782 (где 1 – идеальный экземпляр), или абсолютное значение фитнеса 0,278 на промежутке -10..+10. Минимальное значение фитнеса после 100 тестов равно 0,734, что лежит в пределах 10 % допустимого отклонения для признания сходимости алгоритма.

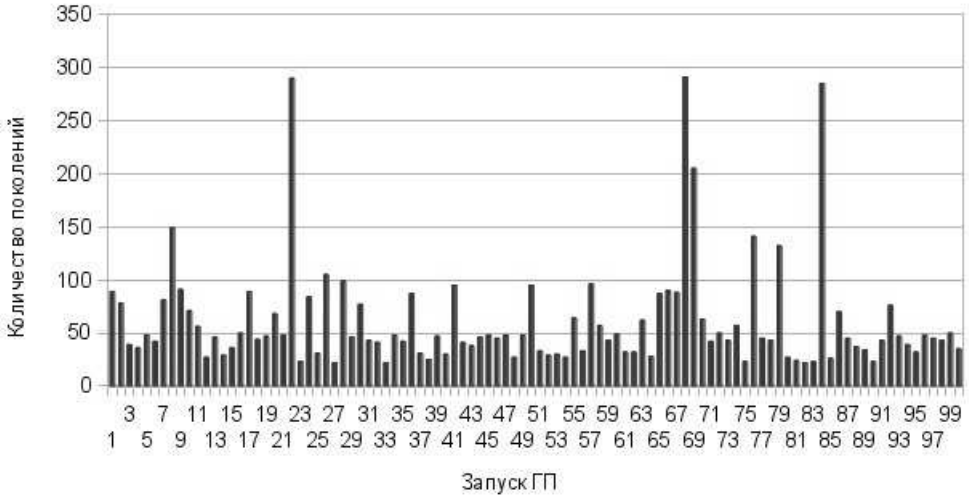


*Рис. 2. Количество использованных поколений для каждого запуска*

Как видно из рис. 2, в 65 случаях алгоритм ГП нашел искомое символьное выражение за менее чем 50 поколений (1000 особей). В 27 случаях алгоритму потребовалось до 100 поколений, и в 8 случаях алгоритм затратил менее 300 поколений (6000 особей) для определения выражения. Каждый из 8 запусков был проанализирован отдельно, и выявлено, что причиной столь затянувшегося поиска идеального решения было отсутствие полноценного механизма нормализации функций, о котором было сказано выше. Тем не менее, каждый из 100 запусков привел к конечному результату, что говорит об эффективности используемых методов обеспечения сходимости.

**Выводы.** Экспериментальное исследование алгоритма генетического программирования показало, что данный алгоритм можно использовать в решении задачи обнаружения вторжения в распределенных компьютерных системах. Было продемонстрировано, что генетическое программирование значительно эффективнее прямого перебора для этого класса задач. Сравнение с другими регрессионными методами не было выполнено из-за их ограниченности только параметрическим поиском, без возможности менять топологию результата. Также необходимо учесть, что алгоритм генетического программирования можно выполнять параллельно – значение фитнеса для каждого индивида может вычисляться на отдельном процессоре.

**Список использованных источников**

1. J. R. Koza, L. W. Jones, M. A. Keane. Toward automated design of industrial strength analog circuits by means of genetic programming / Boston, MA: Kluwer Academic Publishers, 2004. – 22 p.
2. C. Estebanez, J. C. Hernandez-Castro, A. Ribagorda. Finding State-of-the-Art Non-cryptographic Hashes with Genetic Programming / Universidad Carlos III de Madrid, 2000. – 10 p.
3. D. B. Fogel. Evolutionary Computation: Towards a New Philosophy of Machine Intelligence / New York: IEEE Press, 2000. – 384 p.

4. J. R. Koza. Genetic Programming IV: Routine Human-Competitive Machine Intelligence / New York: Springer, 2005. – 606 p.

5. J. R. Koza. Genetic Programming II: Automatic Discovery of Reusable Programs / Boston: MIT Press, 1994. – 768 p.

УДК 681.518.3:528.92

**В.І. Зацерковний,** канд. техн. наук
Чернігівський державний інститут економіки і управління, м. Чернігів, Україна

## РОЗРОБКА МОДЕЛІ ПРИЙНЯТТЯ РІШЕНЬ ЗА ДОПОМОГОЮ РЕГІОНАЛЬНОЇ ГЕОІНФОРМАЦІЙНОЇ СИСТЕМИ

*Виконано аналіз нагальних потреб управління територіями на рівні регіону (області), які вимагають методів та управлінських процедур, здатних функціонувати в умовах певної невизначеності. Сформульовано особливості моделей прийняття рішень у складних системах, якими є регіональні геоінформаційні системи. Здійснено оцінку рівня невизначеності об'єкта дослідження, на підставі якої зроблений висновок, що моделі прийняття рішень повинні будуватись на підставі використання методів декомпозиції і координації.*

**Постановка проблеми.** Основним чинником, який забезпечує прийняття ефективних управлінських рішень у сучасних ринкових умовах, як на рівні керівника підприємства, так і на рівні управління регіону (області) і держави, є своєчасна та достовірна інформація. Оскільки практично вся інформація (80-85 %) про ресурси певного регіону має просторову прив'язку [1], то цілком очевидно, що базовою інформаційною технологією повинна виступати регіональна геоінформаційна система (ГІС).

Впровадження геоінформаційних технологій (ГІТ) у процес управління регіоном дозволяє не тільки значно спростити ведення інформаційних баз і знизити ймовірність виникнення помилок, але й впровадити нові методи підтримки прийняття управлінських рішень на основі аналізу даних дозволяє підняти його на якісно новий рівень і, врешті-решт, підняти їх ефективність.

Проблеми прийняття рішень в організаційному управлінні переважно унікальні й нестандартні, але вони у своїй ситуаційній основі мають такі загальні риси: неповторність ситуації вибору; складний для оцінювання характер досліджуваних альтернатив; недостатня визначеність наслідків дій (невизначеність післядій); наявність сукупності різнорідних чинників, які необхідно враховувати під час прийняття рішень; наявність особи або групи осіб, які несуть відповідальність за прийняття управлінських рішень [2].

**Аналіз останніх досліджень і публікацій.** Розвитку і становленню ГІС суспільство завдячує багатьом зарубіжним, радянським, пострадянським і вітчизняним ученим, серед яких: Дж. Данджермонд, Елті Дж., Калкінз Х., де Мерс, Томплінсон З.Ф., Берлянт О.М., Бугаєвський Л.М., Бурачек В.Г. [3], Бурштинська Х.В., Виноградов Б.В., Дорожинський О.Л., Железняк О.О., Іщук О.О., Кошкарьов А.В., Капралов Є.Г., Корольов Ю.К., Тікунов В.С., Морозов В.В., Серединін Є.С., Могильний С.Г., Карпинський Ю.О., Лященко А.А., Світличний О.О., Суховірський Б.І., Швебс Г.І., Шипулін В.Д., Цвєтков В.Я. [4] та багато інших.

В їх працях підкреслена особлива роль ГІС у вирішенні нагальних проблем управління територіями. Завдяки дослідженням цих учених значно прискорилися процеси формування окремих геоінформаційних проектів, інфраструктури просторових даних, розробки теоретичних і практичних аспектів створення ГІС.

Розглядаючи регіональну ГІС, як мегапроект управління територіальним розвитком, доцільно відзначити, що спроби щодо розробок з теорії управління мегапроектами, де інструментарієм та інформаційною базою управління були б геоінформаційні системи, в науковій літературі дуже мало [4].