ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ И ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ

УДК 004.85

ВЫБОР ФУНКЦИИ ПРИСПОСОБЛЕННОСТИ ОСОБЕЙ ГЕНЕТИЧЕСКОГО АЛГОРИТМА С ПОМОЩЬЮ ОБУЧЕНИЯ С ПОДКРЕПЛЕНИЕМ

А.С. Афанасьева
Научный руководитель – ассистент М.В. Буздалов

Введение. В теории оптимизации известны задачи скалярной и многокритериальной оптимизации [1]. На практике возникает необходимость решать модифицированную скалярную задачу оптимизации, предполагающую наличие дополнительных критериев [2]. Она отличается от многокритериальной задачи тем, что целью ее решения является максимизация единственной целевой функции, а дополнительные критерии имеют лишь вспомогательное значение. В таких задачах целевая функция может некоторым образом зависеть от дополнительных критериев, поэтому в некоторых случаях вместо максимизации целевой функции оказывается выгодным оптимизировать дополнительные критерии.

Важным классом алгоритмов оптимизации являются генетические алгоритмы (ГА) [3], где в качестве критерия выступает целевая функция приспособленности (ФП). При наличии нескольких вспомогательных функций приспособленности выбор наиболее выгодной из них приходится производить вручную [2]. Подобный подход не вполне эффективен, так как предполагает многократный перезапуск ГА.

Целью описываемых **исследований** является разработка метода, позволяющего автоматически выбирать вспомогательную ФП, применение которой способствует ускорению выращивания особей с высокими значениями целевой ФП. Предлагаемый метод также должен осуществлять динамический выбор в случае, когда на разных этапах выполнения ГА выгодны разные вспомогательные ФП.

В литературе чаще всего встречаются разработки по автоматической настройке значений параметров ГА, таких как вероятность применения генетических операторов или число особей в поколении, а также по настройке некоторой фиксированной ФП [4]. Предлагаемая постановка задачи отличается новизной, так как предполагает выбор между качественно разными ФП.

Описание предлагаемого подхода. Для выбора вспомогательной ФП используются методы машинного обучения, а именно, обучение с подкреплением [5]. В алгоритмах этого типа обучение происходит одновременно с применением накопленного опыта, что позволяет выбрать и применить оптимальную ФП в течение одного запуска ГА.

Агент обучения применяет действия к среде, которая отвечает на каждое действие вознаграждением. Действие агента состоит в выборе ФП (целевой или одной из вспомогательных). Среде сопоставляется текущее поколение особей. Вознаграждение зависит от характера изменения значений целевой ФП: чем существеннее рост значений целевой ФП, тем выше вознаграждение. Благодаря тому, что агент выбирает оптимальную стратегию поведения, суммарная награда максимизируется, что приводит к ускорению роста значений целевой ФП.

Результаты. Разработанный метод применен к решению модельной задачи. Поставлен эксперимент, заключающийся в многократных запусках ГА с обучением и обычного ГА.

Результаты эксперимента подтверждают, что применение обучения ускоряет получение особей с более высокими значениями целевой ФП.

Реализованы два алгоритма обучения с подкреплением: Q-learning с ε-жадной стратегией поведения [5] и Delayed Q-learning [6]. Исследована применимость этих алгоритмов при разных значениях вероятности мутации особей. Результаты эксперимента позволяют судить о перспективности применения методов обучения с подкреплением к выбору вспомогательных оптимизируемых величин.

Литература

1. Лотов А.В., Поспелова И.И. Многокритериальные задачи принятия решений: Учебное пособие. – М.: МАКС Пресс, 2008. – 197 с.
2. Буздалов М.В. Генерация тестов для олимпиадных задач по теории графов с использованием эволюционных алгоритмов. Магистерская диссертация. – СПб: СПбГУ ИТМО. http://is.ifmo.ru/papers/2011-master-buzdalov/
3. Mitchell M. An Introduction to Genetic Algorithms. – MIT Press, Cambridge, MA, 1999. – 221 p.
4. Francisco Herrera and Manuel Lozano. Adaptation of Genetic Algorithm Parameters Based on Fuzzy Logic Controllers // Genetic Algorithms and Soft Computing. Physica-Verlag – P. 95–125.
5. Sutton R.S., Barto A.G. Reinforcement Learning: An Introduction. – MIT Press, Cambridge, MA, 1998. – 322 p.
6. Strehl A.L., Li L., Wiewora E., Langford J., Littman M.L. PAC model-free reinforcement learning // Proceedings of the 23rd international conference on Machine learning. ICML'06. – 2006. – P. 881–888.

УДК 004.4'242

ПОСТРОЕНИЕ АВТОМАТОВ УПРАВЛЕНИЯ СИСТЕМАМИ СО СЛОЖНЫМ ПОВЕДЕНИЕМ НА ОСНОВЕ ВЕРИФИКАЦИИ И СЦЕНАРИЕВ РАБОТЫ

К.В. Егоров, Ф.Н. Царев

Научный руководитель – д.т.н., профессор А.А. Шалыто

Автоматное программирование – парадигма программирования, в рамках которой программы предлагается проектировать в виде совокупности взаимодействующих автоматизированных объектов управления [1]. В автоматных программах выделяют три типа компонентов: поставщики событий, система управления и объекты управления. Система управления представляет собой конечный автомат или систему взаимодействующих конечных автоматов. Поставщики событий генерируют события, а система управления по каждому событию может совершать переход, считывая значения входных переменных у объектов управления для проверки охранного условия перехода.

Существуют различные способы построения автоматов управления со сложным поведением. Чаще всего такие системы строятся эвристически, но они могут содержать ошибки и требуют дополнительных проверок. В работе [2] был предложен способ построения автоматов с помощью генетического программирования на основе тестовых примеров. Однако такой вариант построения конечных автоматов требовал дополнительной валидации и верификации, а в случае обнаружения ошибки, необходимо было изменять тестовые примеры и заново строить систему. В работах [3, 4] было предложено строить систему совместно на основе обучающих примеров и темпоральных формул. Формулы записываются на языке логики линейного времени (*Linear Temporal Logic, LTL*) и позволяют утверждать, что построенная система соответствует заявленной спецификации.

В работе предлагается генетический алгоритм построения управляющих конечных автоматов на основе верификации, положительных и отрицательных тестовых примеров. Особь представляет собой «скелет» конечного автомата – с событием на переходах и с указанием числа выходных воздействий [2]. Положительный тестовый пример, это последовательность входных воздействий и соответствующая ей последовательность выходных воздействий. Такой пример должен выполняться в требуемом конечном автомате. Отрицательный тест представляет собой последовательность входных воздействий, которая в противоположность положительной последовательности, не должна выполняться в автомате управления. Как неоднократно замечалось в предыдущих работах [4, 5], нельзя утверждать о правильности построенной системы только на основе положительных тестов. Однако применение положительных и отрицательных тестов не может служить критерием правильности автомата управления, так как они не имеют такой же выразительности как временные утверждения.

Также в работе предлагается построение автоматов на основе сценариев работы, где сценарий работы представляет собой последовательность пар: входное воздействий и список выходных воздействий. В тестовых примерах не было однозначного соответствия между входными воздействиями и выходными, каждый тест записывался как входная последовательность и ожидаемая выходная последовательность воздействий. Теперь же предлагается перейти к интуитивно понятному представлению теста, когда мы заранее знаем список действий на каждое из событий. Это в некоторой степени сужение задачи, которое позволяет ускорить построение конечного автомата управления. Далее в работе все утверждения про тестовые примеры могут применяться и к сценариям работы, считая их подмножеством первых.

Отрицательные тестовые примеры используются при вычислении функции приспособленности, при скрещивании по тестам [2] и при мутации. Вклад в функцию приспособленности каждого отрицательного теста дискретен. Если существует последовательность событий в конечном автомате, на которой тест выполняется, значит, тест проходит, и его вклад – «-1», если последовательности не существует, значит, вклад теста равен нулю. Вклад всех отрицательных тестов считается как отношение суммы вкладов каждого из тестов к общему числу тестов. Конечная формула функции приспособленности принимает вид $FF = FF_{ptest} + FF_{ptest} \times FF_{LTL} - FF_{ptest} \times FF_{ntest}$, где FF_{ptest} – вклад положительных тестовых примеров; FF_{LTL} – вклад LTL-формул; FF_{ntest} – вклад отрицательных тестовых примеров. Каждый из вкладов находится в интервале [0, 1].

Скрещивание по тестам описано в работе [2], его основная идея заключается в том, что выбирается несколько тестов, которые автомат проходит лучше всего, и переходы, задействованные в выбранных тестах, переходят в новую особь из двух родителей без изменений, а оставшиеся распределяются между потомками случайным образом. Для учета отрицательных тестов, предлагается запретить сохранение последнего перехода из теста в потомке без изменения, т.е. даже если переход участвует в положительном тесте, но позволяет пройти одному из отрицательных, то его нельзя сохранить в новой особи.

Мутация при использовании отрицательных тестов остается без изменений, только переход, позволяющий пройти отрицательному тесту, с большей вероятностью меняет свое входное воздействие или может быть удален.

Напомним, что особь генетического алгоритма представляет собой конечный автомат, где каждый переход содержит событие *E*, по которому он осуществляется, и фиксированное число выходных воздействий *N*. В предыдущих работах конкретные выходные воздействия определялись следующим образом: для различных тестовых примеров выбиралась та последовательность действий длины *N*, которая чаще всего встречалась. При построении на основе сценариев работы, алгоритм расстановки выходных воздействий изменился – число выходных воздействий каждого перехода стало переменным, и среди всех полностью или частично проходящих положительных тестовых примеров выбирается та выходная

последовательность для выбранного перехода, которая чаще всего встречается.

Были проведены экспериментальные исследования на примере автомата управления дверьми лифта из работы [4]. Было проведено 1000 построений для каждого из перечисленных выше методов. При использовании только тестовых примеров построенный автомат менее чем в 1% случаев соответствовал спецификации, при использовании совместно верификации и тестовых примеров среднее число вычислений функции приспособленности оказалось равным – 827857. При совместном использовании тестов и контрактов [5] – 710882. При использовании верификации совместно с положительными и отрицательными сценариями работы – 161592.

В работе исследована возможность применения верификации совместно с положительными и отрицательными тестовыми примерами и сценариями работы для автоматического построения автоматов управления систем со сложным поведением, предложен метод мутации и скрещивания на основе отрицательных тестов, проведено экспериментальное исследование метода на примере построения автомата управления дверьми лифта.

Литература

1. Поликарпова Н.И., Шалыто А.А. Автоматное программирование. – СПб: Питер, 2010.
2. Царев Ф.Н. Метод построения автоматов управления системами со сложным поведением на основе тестов с помощью генетического программирования /Материалы Международной научной конференции «Компьютерные науки и информационные технологии». – Саратов: СГУ, 2009. – С. 216–219.
3. Johnson C. Genetic Programming with Fitness based on Model Checking. Lecture Notes in Computer Science // Springer Berlin/Heidelberg, 2007. – V. 4445. – Pp. 114–124.
4. Егоров К.В., Царев Ф.Н., Шалыто А.А. Применение генетического программирования для построения автоматов управления системами со сложным поведением на основе обучающих примеров и спецификации // Научно-технический вестник СПбГУ ИТМО, 2010. – № 69. – С. 81–85.
5. Егоров К.В., Шалыто А.А. Применение генетического программирования для построения автоматов управления системами со сложным поведением на основе контрактов и тестовых примеров // Сборник научных трудов VI-ой Международной научно-практической конференции «Интегрированные модели и мягкие вычисления в искусственном интеллекте. – М.: Физмалит, 2011. – С. 610–615.

УДК 004.4'242

ГЕНЕРАЦИЯ КОНЕЧНЫХ АВТОМАТОВ С ПОМОЩЬЮ ГЕНЕТИЧЕСКИХ АЛГОРИТМОВ ДЛЯ РЕШЕНИЯ ЗАДАЧИ О ПОИСКЕ ЦЕЛИ СЕНСОРНЫМ АГЕНТОМ В ОБЛАСТИ С ПРЕПЯТСТВИЯМИ

А.А. Соколов

Научный руководитель – ассистент М.В. Буздалов

Введение. Задачи навигации возникают в различных отраслях современной науки, включая робототехнику. Так, на практике часто возникают задачи о перемещении робота из одной точки в другую в различном пространстве конфигураций – от простейшего случая передвижения робота на плоскости среди препятствий до поиска оптимальной стратегии перемещения в пространстве робота-манипулятора со множеством степеней свободы. При этом различные виды роботов в различных условиях могут хранить подробную карту местности, на которой ему предстоит работать, обладать лишь частичной информацией о местности или не знать о местности ничего, кроме положения конечной точки. Роботы также

могут обладать различными средствами получения информации о внешнем мире – от простейших контактных сенсоров до систем видеонаблюдения, в том числе и в инфракрасном диапазоне.

В работе рассматривается возможность применения генетического алгоритма для генерации конечных автоматов, которые можно использовать для решения простейшего варианта задачи навигации.

Цель работы. В работе рассматривается следующая задача навигации. Имеется некоторая бесконечная клетчатая двумерная область с препятствиями. Каждая из клеток либо свободна, либо является препятствием. Агент занимает одну клетку. Одна из сводных клеток является целью. В начальный момент времени агент находится в произвольной свободной клетке. Задача агента заключается в том, чтобы добраться до цели, если она достижима, или сообщить об этом, если она недостижима. Целью данной работы является создание генетического алгоритма, позволяющего генерировать автоматы для управления агентом.

Описание предлагаемого подхода. С помощью островного генетического алгоритма генерируются автоматы Мили, для которых входные воздействия – это информация, которой обладает агент по условию задачи, а выходные воздействия – это набор действий, которые агент может совершить. Переходы в автоматах Мили хранятся в виде деревьев решений. Для автоматов реализованы операторы скрещивания и мутации, которые применяются к особям в ходе работы генетического алгоритма.

Для оценки получаемых автоматов используется дополняемый набор тестов (случайно сгенерированных полей). Изначально есть небольшой набор тестов. Когда все текущие тесты пройдены, генерируется дополнительный набор тестов, такой что лучшая текущая особь не проходит новые тесты корректно. В том случае, если такой набор тестов сгенерировать не удается, работа генетического алгоритма завершается и лучший из полученных автоматов подходит для решения данной задачи навигации.

Рассмотрим запуск автомата на тестовом поле. Возможны несколько случаев: автомат может «врезаться» в препятствие, может начать бесконечно удаляться от цели, может войти в цикл и двигаться по замкнутой траектории. Так же возможны случаи выдачи неверного ответа: выдача сообщения о том, что цель недостижима для поля, с достижимой целью или для поля с недостижимой целью без достаточного исследования поля. Кроме того возможна выдача сообщения о достижении цели агентом, не достигшим цели. Для оценки пробега каждого из полей используется функция, которая учитывает результат пробега, его траекторию и корректность ответа для данного поля.

Результаты. В ходе выполнения работы было показано, что возможно получать алгоритмы для решения задач поиска пути с помощью генетического алгоритма. Получены автоматы, которые корректно работают для полей с достижимой целью и для полей с недостижимой целью. Таким образом, генетический алгоритм был успешно применен для данной задачи.

Литература

1. Lumelsky V.J., Stepanov A.A. Path planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape. Algorithmica, 2: 403–430, 1987.
2. Lumelsky V.J., Skewis T. Incorporating range sensing in the robot navigation function. IEEE Transactions on Systems, Man, and Cybernetics, 20(5):1058–1068, 1990.
3. Kamon I, Rimon E, Rivlin E. A New Range-Sensor Based Globally Convergent Navigation Algorithm for Mobile Robots. CIS – Center of Intelligent Systems 9517, Computer Science Dept., Technion, Israel, 1995.

4. Liu Y.H., Arimoto S. Path planning using a tangent graph for mobile robots among polygonal and curved obstacles. International Journal of Robotic Research, 11(4):376–382, 1992.
5. Поликарпова Н.И., Шалыто А.А. Автоматное программирование. – СПб: Питер, 2010.
6. Holland John H. Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor. 1975.
7. Hillis W. D. Co-evolving parasites improve simulated evolution as an optimization procedure. Phys. D. – V. 42. – Issue 1–3 (Jun. 1990). – Pp. 228–234.
8. Данилов В.Р. Технология генетического программирования для генерации автоматов управления со сложным поведением. – СПб: СПбГУ ИТМО, 2007. Бакалаврская работа. http://is.ifmo.ru/papers/danilov_bachelor

УДК 004.4’242

ПОСТРОЕНИЕ АВТОМАТНЫХ МОДЕЛЕЙ ВЕБ-ПРИЛОЖЕНИЙ

А.Ю. Законов
Научный руководитель – д.т.н., профессор А.А. Шалыто

В течение последних лет существенно выросла как сложность веб-приложении□, так и требования к их надежности. Зачастую по причине сжатых сроков разработки и часто меняющихся требовании□ заказчика проверка качества веб-приложении□ осуществляется вручную без формальных критериев. Отсутствие моделеи□ и формальноu□ спецификации делает процесс проверки качества трудоемким и не позволяет гарантировать отсутствие ошибок. В докладе предложен метод автоматического выделения явных состоянии□ и построения автоматноu□ модели для веб-приложении□ при помощи сочетания статического и динамического анализа. Наличие автоматноu□ модели позволит формализовать требования спецификации и использовать Model Checking для автоматизации процесса тестирования. Предложенныи□ в статье метод позволяет анализировать как приложения, состоящие из множества различных веб-страниц, связанных гиперссылками, так и приложения, реализованные в виде однои□ страницы с использованием технологии AJAX. Также разработан алгоритм поиска схожих состоянии□, благодаря которому, даже для сложных веб-приложении□, возможно автоматически строить модели с небольшим количеством состоянии□, которые будут понятны разработчику.

Постановка задачи. Под состоянием веб-приложения будем понимать исходныи□ код веб-страницы, на которои□ находится пользователь. События – действия пользователя, которые включают в себя нажатия на кнопки и прочие управляющие элементы, ввод текста и заполнение форм. Каждое из деи□ствии□ пользователя может привести к новому состоянию веб-приложения: изменению текущеи□ веб-страницы (JavaScript обработчики событии□, AJAX-запросы и их обработка) или к переходу на новую страницу.

Для автоматического построения модели веб-приложения необходимо обнаружить максимальное количество возможных состоянии□ приложения. В работе предложен алгоритм обнаружения новых состоянии□, основанныи□ на случаи□нои□ генерации деи□ствии□ пользователя.

Цель исследования: предложить метод автоматического построения автоматноu□ модели веб-приложения, метод применения Model Checking и автоматизации тестирования для повышения качества веб-приложении□.

В работе решаются следующие задачи:

1. разработать метод автоматического выделения явных состоянии□ и переходов между ними для веб-приложении□;

2. оценить применимость методов Model Checking для автоматных моделей веб-приложении и предложить соответствующие инструменты для верификации построенных моделеи;
3. предложить метод автоматическои генерации тестов по построеннои модели.

Практические результаты. Для опробации предложенного подхода разработан набор инструментов на языке Python 2.7. Для работы с веб-страницами и эмуляции деиствии пользователя используется фреимворк Selenium. Для укладки графа и представления автоматнои модели в виде PNG-изображения используется фреимворк GraphViz. Разработанныи в рамках данного исследования инструмент для заданного веб-приложения автоматически создает модель в виде конечного автомата, которыи описывает обнаруженные состояния приложения и возможные переходы между ними. Модель сохраняется в форматах XML и PNG. PNG-изображение модели позволит разработчику лучше понять принцип работы приложения и записать формальные требования к явно выделенным состояниям на языке LTL. Автоматная модель веб-приложения преобразуется в язык Promela и подается на вход верификатору Spin, которыи автоматически проверяет записанные требования спецификации.

УДК 004.021

ПРИМЕНЕНИЕ ДЕРЕВЬЕВ ДЛЯ РЕАЛИЗАЦИИ МАССОВЫХ ОПЕРАЦИЙ НА МНОГОМЕРНЫХ МАССИВАХ ДАННЫХ

А.Г. Банных
Научный руководитель – к.т.н., доцент А.С. Станкевич

Введение. Для эффективной работы с информацией были разработаны разнообразные структуры данных. Одна из самых распространенных из них – дерево. Широко известны квадродеревья, кд-деревья, обобщения дерева отрезков и дерева Фенвика. Существующие структуры данных для работы с многомерными массивами данных позволяют эффективно получать статистическую информацию и изменять *отдельные* элементы.

Цель работы заключается в том, чтобы исследовать возможность выполнения массовых обновлений – единообразного изменения *областей* данных. Ни одна из вышеперечисленных структур данных в чистом виде не предоставляет подобную функциональность.

Постановка задачи. Рассмотрим d-мерное пространство $\mathbf{Z}^d$. Введем частичный порядок на элементах этого пространства – скажем, что $\mathbf{x} \le \mathbf{y} \Leftrightarrow \forall k \in \{1,2..d\}\, \mathbf{x}_k \le \mathbf{y}_k$. Кроме того, будем считать, что $\mathbf{1}_d = (1,1,\ldots,1) \in \mathbf{Z}^d$.

Для любых $\mathbf{x},\mathbf{y} \in \mathbf{Z}^d, \mathbf{x} \le \mathbf{y}$ *областью* будем называть множество $P_{\mathbf{x},\mathbf{y}} = [\mathbf{x}_1..\mathbf{y}_1]\times[\mathbf{x}_2..\mathbf{y}_2]\times\ldots\times[\mathbf{x}_d..\mathbf{y}_d]$. Будем опускать нижние индексы в тех случаях, когда это не приводит к неоднозначностям. Будем считать, что мы работаем с областью $W = P_{\mathbf{1},\mathbf{n}}$, где $\mathbf{n} \in \mathbf{Z}^d, \mathbf{1} \le \mathbf{n}$.

Рассмотрим множество G и отображение $A : W \to G$. Можно рассматривать A, как многомерный массив с множеством индексов W и множеством допустимых значений ячеек G.

Рассмотрим бинарные операции $+$ и $\circ$ такие что $< G,+ >$ – коммутативная полугруппа, $< G,\circ >$ – полугруппа, и выполнен дистрибутивный закон умножения относительно сложения.

Массовый запрос определим следующим образом: $\mathrm{get}(A,P)=\sum_{\mathbf{x}\in P}A(\mathbf{x})$.

Массовое обновление: $\mathrm{update}(A,P,v)=A':W\to G$, где

$$A'(\mathbf{x})=\begin{cases}A(\mathbf{x})\circ v & \mathbf{x}\in P\\ A(\mathbf{x}) & \mathbf{x}\notin P\end{cases}.$$

Задача состоит в том, чтобы реализовать структуру данных, которая позволила бы эффективно выполнять эти массовые операции.

Основной результат. В работе был выбран частный случай $+\equiv\circ$, $<G,+>$ – *абелева* группа. Для него был разработан общий метод сведения задачи с массовыми обновлениями к хорошо изученной задаче с единичными обновлениями. Разработанный метод можно использовать с любыми существующими структурами данных. Это свойство позволяет делать выбор оптимальной структуры данных для каждой задачи отдельно.

При фиксированной размерности пространства d асимптотическая оценка на выполнение массовых операций составляет $O(\log^d|\mathbf{n}|)$. Потребление памяти зависит от используемой структуры данных. В случае сильно разреженных данных потребление памяти может возрасти в 4^d раз по сравнению с решением аналогичной задачи без массовых обновлений.

При малой размерности пространства можно говорить об эффективности и универсальности полученного метода. Однако вопрос об обобщении подхода на более широкие классы задач остается открытым.

УДК 004.85

ГЕНЕРАЦИЯ СЛОЕВ НЕЙРОННЫХ СЕТЕЙ КАСКАДНОЙ КОРРЕЛЯЦИИ ФАЛЬМАНА С ИСПОЛЬЗОВАНИЕМ ГЕНЕТИЧЕСКИХ АЛГОРИТМОВ

Ю.Ю. Коноплев
Научный руководитель – д.т.н., профессор И.П. Гуров

Краткое вступление, постановка проблемы. При описании процессов или явлений реального мира программист создает математическую модель, которую впоследствии программирует. Если выбранная модель совсем плоха, то ищется другая; в большинстве же случаев подбираются параметры модели для лучшего соответствия описанию исходной задачи (решается задача оптимизации).

При решении реальных проблем, приходиться решать составные задачи, алгоритмы решение разных подчастей которых сильно отличны друг от друга, и их в буквально случае нужно брать из разных разделов наук, перебирать или угадывать основываясь на опыте; т.е. нет обратной связи для алгоритмов, которая, например, есть в алгоритмах обучения нейронных сетей (НС).

В работе решается проблема создания подобной обратной связи с помощью НС такой, чтобы результирующая НС подбирала модели, в которых решается задача, в некотором широком классе моделей, а не только решала задачу оптимизации для узкого класса однажды формализованных моделей каким-либо образом соединенных между собой.

Цель работы. Разработать и исследовать составные алгоритмы с общей обратной связью. В качестве таких алгоритмов были выбраны алгоритмы нейронных сетей каскадной корреляции Фальмана (НСКРФ) ввиду их адаптивности; обратная связь реализуется с помощью соединения отдельных НСКРФ в граф с помощью генетических алгоритмов.

Базовые положения исследования. Нейронные сети каскадной корреляции Фальмана – это специализированная многослойная нейронная конструкция, в которой подбор структуры сети происходит параллельно с ее обучением путем добавления на каждом этапе обучения одного скрытого нейрона. Стимулом выбрать именно их явилось то, что в ней изменяет топология, а функции активации добавляемых нейронов могут различаться.

Генетический алгоритм это – эвристический алгоритм построения и оптимизации модели путем комбинирования и вариации параметров модели, напоминающий биологическую эволюцию.

Промежуточные результаты. Сформулированы тезисы описывающие конечную структуру, получаемую в результате работы генетических алгоритмов и обучения НС.

- Иногда проще конструировать несколько различных НС и выбирать более правдоподобную, чем использовать более сложные алгоритмы для конструирования одной единственной НС.
- Вычислительно проще конструировать несколько маленьких НС, чем одну большую.
- НС должны быть предсказуемы относительно изменения топологии.
- В НС должна быть возможность изменять как весовые коэффициенты связей так и функции активации.
- К конечному обученному состоянию НС имеет возможность подойти инкрементально.
- Алгоритмы подобные методу обратного распространения ошибки должны иметь действия на все элементы результирующей сети.
- ГА должны иметь возможность менять связи между НС, но не должны менять сами НС.
- Сети генерируемые ГА служат для декомпозиции входных сигналов по характерным признакам.
- Совместная работа всех частей алгоритмов должна иметь возможность: 1) изменять производимые вычисления внутри сети (computational nets); 2) изменять логику обработки входных сигналов (logic nets); 3) классифицировать результирующие данные (classification nets).

Проведен информационно-аналитический обзор различных нейронных сетей, которые будут использоваться в качестве элементов сети генерируемой ГА.

Основной результат. В результате работы были реализованы алгоритмы на языках Clojure и Mathematica, позволяющие исследовать поведение и структуру всей сети и отдельных ее составляющих и использовать сеть в качестве классификатора входных сигналов.

УДК 004.4’242:004.272.43

ПОРТИРОВАНИЕ ГЕНЕТИЧЕСКИХ АЛГОРИТМОВ НА ПЛАТФОРМУ OPENCL НА ПРИМЕРЕ ГЕНЕРАЦИИ АВТОМАТА В ЗАДАЧЕ «УМНЫЙ МУРАВЕЙ»

А.И. Бочкарев
Научный руководитель – аспирант, ассистент М.В. Буздалов

Введение. OpenCL – программное обеспечение для написания компьютерных программ, связанных с параллельными вычислениями на различных графических (англ. GPU) и центральных (англ. CPU) процессорах. Разработчику предоставляется возможность из своего приложения вызвать некоторый программный код, называемый «ядром» (англ. Kernel). Код ядра базируется на подмножестве языка C с некоторыми расширениями [1].

Код ядра запускается одновременно в несколько потоков, различающихся только своим

номером, который можно узнать при выполнении. Таким образом, параллельно можно запускать только одинаковые ядра, при этом отличия в их поведении могут базироваться лишь на номерах. Такая модель вычислений хорошо подходит для выполнения расчетов над множеством однотипных объектов, например, расчетом нового поколения или вычислением функции приспособленности в генетическом алгоритме. Работа посвящена реализации такого алгоритма и сравнению скорости его выполнения со стандартной реализацией на центральном процессоре.

Цель работы. Задача работы состоит в изучении способов выращивания управляющих автоматов с использованием технологии OpenCL, и экспериментальном сравнении скорости ее работы со стандартными технологиями многопоточности на центральном процессоре. В качестве примера для исследования взята генерация автомата в известной задачи об умном муравье [2].

Описание предлагаемого подхода. Для решения данной задачи необходимо разработать алгоритмы для распараллеливания генерации и тестирования особей, которые, в свою очередь, отличаются друг от друга при использовании различных технологий. Также требуется произвести оптимизации указанных алгоритмов для наиболее эффективного использования возможностей доступных вычислительных платформ [3].

Результаты. Разработанные методы применены к решению задачи об умном муравье. Полученные результаты показывают, что сравнение скорости вычислений на графическом и центральном процессорах является очень сложной задачей. Даже при исследовании всего одной задачи наблюдается преимущество то одной, то другой платформы. Несколько небольших оптимизаций могут кардинально изменить имеющуюся картину результатов. Во многом это связано с тем, что технология OpenCL генерирует эффективный код для центрального процессора, который может выполняться намного быстрее, чем написанный с использованием стандартных средств. Так, например, при выборе оптимальных параметров код на GPU работает в 6 раз быстрее, чем реализованный вручную на языке C++, а при выполнении кода, сгенерированного OpenCL для CPU, ускорение составило до 17 раз.

Таким образом, технология OpenCL показала себя перспективной для выращивания автоматов с помощью генетических алгоритмов, и ее использование оправдывает себя не только при работе с графическими процессорами.

Литература

1. The OpenCL Specification, Version 1.1, 2011. http://www.khronos.org/registry/cl/specs/opencl-1.1.pdf
2. Бедный Ю.Д., Шалыто А.А. Применение генетических алгоритмов для построения автоматов в задаче «Умный муравей» // Научно-технический вестник СПбГУ ИТМО, 2008. – Вып. 53. Автоматное программирование. – С. 88–99. http://is.ifmo.ru/works/2008/Vestnik/53/06-genetic-automata-robocode.pdf
3. AMD Accelerated Parallel Processing OpenCL™ Programming Guide (v1.3f), 2011. http://developer.amd.com/sdks/AMDAPPSDK/assets/AMD_Accelerated_Parallel_Processing_OpenCL_Programming_Guide.pdf

УДК 004.92

СТАБИЛИЗАЦИЯ ВИДЕОПОСЛЕДОВАТЕЛЬНОСТЕЙ В РЕЖИМЕ РЕАЛЬНОГО ВРЕМЕНИ

А.В. Буслаев, А.П. Минюк
Научный руководитель – магистрант Д.Е. Родиков

Введение. С развитием робототехники проблемы компьютерного зрения и обработки фото и видеоинформации становятся все более актуальными, причем основной поток информации, поступающей из внешнего мира, приходится на визуальные данные. Большое значение имеет производительность алгоритмов, так как информация должна быть актуальной для выполнения текущих задач с высокой точностью. Важную роль в компьютерном зрении играет обработка видео. Стабилизацией видеопоследовательностей называется удаление нежелательного беспорядочного движения камеры. Стабилизация делится на два вида: аппаратная и программная. В работе рассматривается проблема программной стабилизации.

Постановка задачи. Эффект от аппаратного стабилизатора в камере достаточно велик, тем не менее в ряде ситуаций полезно совмещать аппаратную и программную стабилизации. К примеру, при съемках с большим приближением, аппаратная стабилизация не улавливает слабых колебаний. Более того, при больших колебаниях камеры аппаратный стабилизатор в силу конструктивных ограничений не обеспечивает требуемого качества. Помимо этого существуют задачи компьютерного зрения, такие как поиск движения или ориентирование в пространстве, которые используют основные алгоритмы задачи стабилизации. Таким образом, требуется решение проблемы программной стабилизации.

В большинстве известных подходов к данной проблеме речь идет о нахождении смещения по координатным осям или матрицы аффинного преобразования. В данной работе представлен алгоритм нахождения проективного преобразования, что позволяет решать задачу стабилизации в условиях сложного движения камеры и ряд других задач.

Практические результаты. Изучены основные методы программной стабилизации изображений и видеопоследовательностей, произведен поверхностный анализ их возможностей и производительности. Было принято решение использовать алгоритм Лукаса-Канаде (Lucas–Kanade method) для поиска оптического потока в ряде точек изображения и нахождения проективного преобразования. После вычисления матрицы проективного преобразования применяется экспоненциальное сглаживание для получения стабилизированной картинки. Подобный подход позволил разработать эффективный алгоритм стабилизации видеопоследовательностей в режиме реального времени на CPU.

УДК 004.415.532.3:004.832.23

ПРИМЕНЕНИЕ ЭВОЛЮЦИОННЫХ АЛГОРИТМОВ ДЛЯ ПОКРЫТИЯ КОДА ТЕСТАМИ

М.В. Буздалов
Научный руководитель – д.т.н., профессор А.А. Шалыто

Введение. Тестирование программного обеспечения (ПО) занимает до 50% времени и более 50% стоимости производства ПО [1]. По этой причине, автоматизация тестирования ПО, за счет уменьшения затрачиваемого времени и снижения влияния человеческого фактора, приводит к сокращению периода выхода программного продукта на рынок, снижению его себестоимости и уменьшению числа программных ошибок в нем. Такие методологии производства ПО, как экстремальное программирование [2], уделяют

значительное внимание написанию так называемых модульных тестов, а также автоматизации их регулярного запуска, однако, написание модульных тестов выполняется программистами, а, следовательно, этот процесс нуждается в усовершенствовании. В частности, актуальной задачей является автоматическая генерация тестов.

Одним из показателей качества набора тестов является то, насколько полно, в смысле возможных путей выполнения и возможных случаев, встречающихся в коде, тестируемый код покрыт тестами. Существует множество критериев покрытия кода тестами, как общих, так и специализированных. Примером общего критерия является число (доля) покрытых строк кода (т.е., число таких строк, которые хотя бы один раз исполнялись при тестировании кода на рассматриваемом наборе тестов). Примером специализированного критерия для тестирования баз данных SQL является доля покрытых тестами операторов SELECT.

Цель работы. В работе рассматривается автоматическая генерация набора тестов, покрывающего код согласно заданному критерию, с применением эволюционных алгоритмов [3, 4]. Рассматриваются как широко распространенные критерии покрытия кода, так и новые критерии, ориентированные на покрытие значений, в особенности – граничных значений.

Описание предлагаемого подхода. Предлагаемый подход к генерации покрывающих тестов принадлежит методологии «тестирования белого ящика», т.е., требует доступа к коду тестируемой программы (исходных кодов или исполняемого кода). Он состоит из двух этапов. На первом этапе, в соответствии с конечной целью генерации набора тестов, выбирается критерий покрытия. Может быть выбран как один из традиционных критериев, такой как доля покрытых инструкций кода или доля покрытых ветвей кода, так и какой-либо другой критерий, который лучше подходит для поставленной задачи. Среди таких критериев в работе выделены критерии, ориентированные на покрытие различного рода граничных значений, принимаемых переменными и параметрами процедур. Мотивируется это тем, что при тестировании алгоритмов дискретной математики полное покрытие, согласно традиционным критериям, может быть достигнуто на небольшом числе случайных тестов, при этом возможные случаи, возникающие в работе алгоритма, могут быть рассмотрены не полностью. Если число различных значений некоторой величины невелико, то целью может быть покрытие всех возможных значений этой величины. Примерами могут служить значения перечислимых типов (enum) небольшого размера, а также значения правого аргумента операции побитового сдвига. В последнем случае рассматриваются значащие (для этой операции) биты указанной величины, а также то, установлены ли какие-либо другие биты.

Второй этап состоит в генерации тестов. Для этого вначале строится несколько случайных тестов, тесты добавляются в итоговый набор, и анализируется полученное ими покрытие. Затем для некоторого непокрытого фрагмента программы, согласно выбранному критерию, строится оптимизируемая функция (или, в терминах эволюционных алгоритмов, функция приспособленности). Функция вычисляется при выполнении тестируемой программы путем модификаций ее исходного кода. В данной работе она имеет сложный вид, зависящий от последовательности выполненных операций и значений, принимаемых переменными в процессе выполнения. Чем «ближе» траектория выполнения программы на тесте подходит к выбранному фрагменту, тем «лучше» значение функции приспособленности.

Для оптимизации указанной функции производится запуск эволюционного алгоритма. Особью алгоритма является тест. Когда фрагмент оказывается покрытым некоторым тестом, этот тест добавляется в итоговый набор, затем вновь анализируется покрытие программы текущим итоговым набором, и описанная процедура повторяется, пока покрытие не будет признано удовлетворяющим поставленной задаче.

Литература
1. Myers G. The Art of Software Testing, Second Edition. – John Wiley & Sons, 2004.
2. Бек К. Экстремальное программирование. – Питер, 2002.
3. Mitchell M. An Introduction to Genetic Algorithms. – MIT Press, 1996.
4. Luke S. Essentials of Metaheuristics. – Lulu, 2009.

УДК 004.4'242

ПРИМЕНЕНИЕ МУРАВЬИНЫХ АЛГОРИТМОВ ДЛЯ ПОСТРОЕНИЯ КОНЕЧНЫХ АВТОМАТОВ

Д.С. Чивилихин, В.И. Ульянцев
Научный руководитель – аспирант, ассистент Ф.Н. Царев

Постановка задачи. В рамках парадигмы автоматного программирования ключевыми компонентами программ и программных комплексов являются конечные автоматы. В некоторых случаях автоматы могут быть построены вручную, однако для большинства реальных задач эвристическое построение автоматов является весьма трудоемким. Для автоматической генерации конечных автоматов обычно применяются различные эволюционные алгоритмы.

Управляющий конечный автомат задается множеством состояний, начальным состоянием, множеством входных событий и множеством выходных воздействий, а также функцией перехода, сопоставляющей каждому состоянию автомата и каждому входному событию другое состояние и выходное воздействие.

Для каждой задачи, функция приспособленности вещественнозначная функция, заданная на множестве всех конечных автоматов с заданными параметрами. В общем случае, задача генерации конечного автомата по заданной функции приспособленности может быть сформулирована следующим образом: по заданному числу состояний, множеству входных событий и множеству выходных воздействий найти конечный автомат с целевым значением заданной функции приспособленности.

Муравьиные алгоритмы семейство методов решения задач оптимизации, которые могут быть сведены к поиску по графу, таких как, например, задача о коммивояжере. В рамках муравьиных алгоритмов решения строятся набором агентов-муравьев, использующих при выборе пути в графе некоторую стохастическую стратегию. Решения могут быть представлены как путями в графе, так и отдельными его вершинами. В данной работе исследуется возможность применения муравьиных алгоритмов для построения конечных автоматов по заданной функции приспособленности путем сведения этой задачи к поиску целевой вершины в графе.

Целью работы является разработка метода генерации конечных автоматов для заданной функции приспособленности на основе муравьиных алгоритмов и оценка его эффективности.

Описание предлагаемого подхода. В качестве представления пространства поиска, т.е. множества всех конечных автоматов с заданными параметрами, рассматривается граф, вершины которого соответствуют решениям (конечным автоматам), а ребра мутациям конечных автоматов. Под мутацией конечного автомата понимается небольшое изменение его структуры, например, изменение выходного воздействия на переходе или изменение состояния, в которое ведет переход.

Каждому ребру графа ставится в соответствие некоторое вещественное число значение феромона. Значение феромона на ребре может быть увеличено, если по этому ребру

прошел муравей. Также, на каждой итерации алгоритма происходит испарение уменьшения значений феромона на всех ребрах графа в одинаковое число раз. Алгоритм генерации автомата может быть разделен на несколько этапов:

1. Построение решений муравьями-агентами.

Муравей, находясь в определенной вершине графа, выбирает следующую вершину руководствуясь одним из двух правил:

- построить новые ребра графа, производя мутации автомата, соответствующего текущей вершине и выбрать лучшее из новых решений;
- выбрать следующую вершину исходя из значений феромона на ребрах, ведущих из текущей вершины по классической формуле выбора пути в муравьиных алгоритмах.

2. Обновление значений феромона на ребрах.
3. Проверка условий останова.

Шаги 1–3 повторяются до тех пор, пока не будет найдено решение, удовлетворяющее условиям задачи, или не выполнится другое условие останова, например, истечет заданное максимальное число итераций алгоритма.

Результаты. Был разработан и реализован метод генерации конечных автоматов по заданной функции приспособленности, основанный на муравьином алгоритме. Эффективность разработанного метода была оценена путем сравнения с генетическими алгоритмами для задачи об «Умном муравье» и задаче о генерации автоматов на основе тестовых примеров. Эксперименты показали, что среднее время работы предлагаемого алгоритма для построения целевых автоматов в этих задачах в несколько раз меньше времени работы генетического алгоритма.

СИНТЕЗ РЕГУЛЯТОРОВ С ИСПОЛЬЗОВАНИЕМ ГЕНЕТИЧЕСКИХ АЛГОРИТМОВ

С.А. Кононенко
Научный руководитель – к.т.н., доцент В.И. Бойков

При разработке современных систем управления возникает проблема отсутствия явного аналитического решения поставленной задачи вследствие большого числа противоречивых требований к системе управления. Многие из них, такие как ограничение коэффициентов передачи, учитываются только на стадии моделирования аналитически полученного результата. Для одновременного учета многих требований требуется комплексный алгоритм поиска оптимального результата.

В связи со сложностью аналитического решения поставленной задачи для поиска результата можно применить стохастический подход, используя, например, генетический алгоритм (ГА).

Для проверки возможности использования генетического алгоритма для синтеза систем автоматического управления необходимо провести машинное моделирование поиска решения и проанализировать полученные результаты. Для проведения моделирования требуется спроектировать и разработать специальное программное обеспечение (ПО). Решение задачи рассмотрено на примере синтеза пропорционального регулятора для системы стабилизации.

Определены основные понятия генетического алгоритма применительно к поставленной задаче:

- особь;
- популяция;
- скрещивание;
- мутация.

Разработано программное обеспечение позволяющее оценить качество переходного

процесса динамической системы, что позволило реализовать синтез регулятора на базе ГА. На примере системы второго порядка проверена работоспособность разработанного ПО.

Поиск оптимального решения заключался в поиске коэффициентов передачи пропорционального регулятора обеспечивающих минимальное время переходного процесса при условии ограничения их абсолютных значений.

Произведен анализ параметров, определяющих работоспособность генетического алгоритма:

- время расчета переходного процесса;
- временной шаг расчета;
- число особей в популяции;
- минимальная оценка приспособленности;
- коэффициент мутации;
- время поиска.

В результате, разработанный подход позволит синтезировать регуляторы автоматизированных систем, при значительном количестве требований предъявленных к системе.

УДК 004.4'23 004.432.42

ПОЛИМОРФНЫЕ ПО ЧИСЛУ АРГУМЕНТОВ ФУНКЦИИ

Я.М. Малаховски
Научный руководитель – к.т.н., доцент Г.А. Корнеев

Одной из важных областей применения функционального программирования [1] является создание, так называемых, встроенных предметно-ориентированных языков (embedded Domain Specific Language, eDSL) [2]. При этом, использование MixFix парсеров [3] позволяет достичь практически произвольного синтаксиса разрабатываемого eDSL, а использование функций высших порядков, монад и зависимых систем типов – производить многие виды предметно-специфичных проверок [4].

Другим важным направлением в области функциональных языков является конкатенативное программирование. Данная методология является логическим продолжением pointfree [5] стиля и предлагает использовать композицию функций вместо аппликации везде, где это возможно, однако, в отличие от pointfree, использование композиции распространяется не только на чистые функции, но и на функции, выполняющие ввод-вывод. При этом «природа» композиции (двухаргументность и ассоциативность) обеспечивает линейность зависящих друг от друга вычислений и параллельность независимых.

Одной из проблем, возникающих при создании eDSL, является лифтинг [6] функций базового языка в функции предметно-ориентированного языка. Обычно такие функции пишутся программистами вручную или генерируются при помощи механизмов типа Template Haskell [7] или C++11 Templates [8] (язык шаблонов языка C++ является чистым функциональным языком программирования). При этом использование различных расширений системы типов языка (например, классов типов [9]), как правило, вызывает трудности для обобщенного выражения таких функций.

Заметной проблемой конкатенативного программирования является отсутствие общего механизма комбинирования функций с различными числами аргументов [10], что приводит к появлению различного рода ухищрений и трюков [10, 11].

В настоящей работе предлагается метод выражения функций, полиморфных по числу аргументов, на функциональных языках программирования с зависимыми системами типов. Данный метод позволяет полностью решить проблему лифинга функций eDSL и частично

решает проблему комбинирования функций нескольких аргументов.

Для апробации и практического применения предложенного подхода был выбран язык программирования Agda [12]. В качестве примера eDSL выбран язык из работы [4].

Литература

1. Bird R. Introduction to Functional Programming Using Haskell, 2nd Edition. Prentice-Hall, 2nd edition, 1998.
2. Haskell eDSL Tutorial – Shared expenses. [Электронный ресурс]. Режим доступа http://lpenz.github.com/articles/hedsl-sharedexpenses/index.html свободный. Яз. англ. (дата обращения 29.02.12).
3. Danielsson N., Norell U. Parsing MixFix Operators, 2009.
4. Малаховски Я.М. Применение систем типов для валидации и верификации автоматных программ. Магистерская диссертация. – СПб: СПбГУ ИТМО, 2011.
5. Pointfree. [Электронный ресурс]. Режим доступа http://www.haskell.org/haskellwiki/Pointfree свободный. Яз. англ. (дата обращения 29.02.12).
6. Lifting. [Электронный ресурс]. Режим доступа http://www.haskell.org/haskellwiki/Lifting свободный. Яз. англ. (дата обращения 29.02.12).
7. Sheard T., Peyton Jones S. Template metaprogramming for Haskell. In proceedings of Haskell Workshop 2002, Pittsburgh. – P. 1–16.
8. C++ Standards Committee Papers. [Электронный ресурс]. Режим доступа http://www.open-std.org/jtc1/sc22/wg21/docs/papers/ свободный. Яз. англ. (дата обращения 29.02.12).
9. Hudak P., Peterson J., Fasel J.A Gentle Introduction to Haskell, Version 98. 2000. [Электронный ресурс]. Режим доступа http://www.haskell.org/tutorial/ свободный. Яз. англ. (дата обращения 29.02.12).
10. Purdy J. Why Concatenative Programming Matters. [Электронный ресурс]. Режим доступа http://evincarofautumn.blogspot.com/2012/02/why-concatenative-programming-matters.html свободный. Яз. англ. (дата обращения 29.02.12).
11. Okasaki C. Techniques for Embedding Postfix Languages in Haskell. Haskell Workshop, October 2002. – P. 105–113.
12. Norrel U. Dependently Typed Programming in Agda. Advanced Functional Programming, 2008.

УДК 004.4’242

ПРИМЕНЕНИЕ МЕТОДОВ РЕШЕНИЯ ЗАДАЧИ УДОВЛЕТВОРЕНИЯ ОГРАНИЧЕНИЯМ ДЛЯ ПОСТРОЕНИЯ УПРАВЛЯЮЩИХ КОНЕЧНЫХ АВТОМАТОВ ПО СЦЕНАРИЯМ РАБОТЫ

В.И. Ульянцев
Научный руководитель – аспирант, ассистент Ф.Н. Царев

Введение. При применении парадигмы автоматного программирования для реализации сущности со сложным поведением выделяется система управления и объект управления. На начальном этапе проектирования программы выделяются события, входные переменные и выходные воздействия. После этого проектирование программы может идти разными путями. Один из них состоит в написании сценария работы программы, по которому далее эвристически строится автомат. Пример построения автомата таким способом приведен в работе [1].

К автоматной программе, как правило, предъявляются два требования:

– непротиворечивость – не должно быть двух переходов, исходящих из одного состояния управляющего автомата и одновременно выполнимых при некоторой комбинации события и входных переменных;
– полнота – любой комбинации события и входных переменных должен соответствовать переход в каждом состоянии.

Ранее авторами был предложен метод построения автоматных программ, удовлетворяющих требованию непротиворечивости, но не удовлетворяющих требованию полноты. В настоящей работе предлагается метод, не обладающий данным недостатком.

Цель работы. В указанной работе сначала был составлен неформальный сценарий в виде текста, далее он был формализован – был получен текст с указанием обозначений входных событий и выходных воздействий, соответствующих стадиям работы программы. После этого выполнялось построение автомата. Все три указанных этапа выполнялись вручную.

Первые два этапа, скорее всего, не могут быть автоматизированы и должны выполняться человеком. Ранее авторами был разработан метод автоматизированного построения управляющих конечных автоматов по сценариям работы [2]. **Целью настоящей работы** является разработка метода построения управляющих автоматов с гарантией не только непротиворечивости системы переходов, но и ее полноты.

Базовые положения исследования. Сценарием работы программы будем называть последовательность троек $<e, f, A>$, где e – входное событие, f – булева формула от входных переменных, задающая охранное условие, A – последовательность выходных воздействий. На вход разрабатываемому алгоритму подается набор сценариев работы. Построение управляющего автомата осуществляется в пять этапов.

1. Построение дерева сценариев.
2. Построение графа совместимости вершин дерева сценариев.
3. Построение ограничений на целочисленные переменные, задающей требования к раскраске построенного графа и выражающей непротиворечивость и полноту системы переходов результирующего автомата.
4. Запуск сторонней программы, решающей задачу удовлетворения построенным ограничениям (constraint satisfaction problem, CSP).
5. Построение автомата по найденной выполняющей подстановке.

Основной результат. Разработан метод автоматизированного построения управляющих конечных автоматов по сценариям работы. Этот метод основан на сведении указанной задачи к задаче выполнимости ограничений. Работоспособность метода проверена на задаче построения автомата управления часами с будильником. На этих задачах соответствующие управляющие автоматы были построены корректно, а время работы алгоритма составляло меньше секунды на персональном компьютере с процессором Intel Core 2 Quad Q9400 (2,67 ГГц), что позволяет говорить о достаточно высокой производительности разработанного метода.

Литература

1. Мазин М.А., Парфенов В.Г., Шалыто А.А. Разработка интерактивных приложений Macromedia Flash на базе автоматной технологии. http://is.ifmo.ru/download/flash.pdf
2. Ulyantsev V., Tsarev F. Extended Finite-State Machine Induction using SAT-Solver / Proceedings of the Tenth International Conference on Machine Learning and Applications, ICMLA 2011, Honolulu, HI, USA // IEEE Computer Society, 2011. – V. 2. – P. 346–349.

УДК 004.421.2:519.677

О МЕТОДЕ ВОССТАНОВЛЕНИЯ ФУНКЦИЙ ДВУХ ПЕРЕМЕННЫХ, ЗАДАННЫХ ТАБЛИЧНО, ПОСРЕДСТВОМ ЛИНЕЙНЫХ КОМБИНАЦИЙ СДВИГОВ И СЖАТИЙ ОДНОЙ ФУНКЦИИ

А.В. Масальских
Научный руководитель – д.т.н., профессор В.Г. Парфенов

Пусть функция двух переменных задана таблично. Требуется восстановить значения функции на всей области определения. Задача восстановления функции является классической и описана во многих научных работах. К классическим способам восстановления функции можно отнести построение функции путем расчета промежуточных значений на основе линейной интерполяции или восстановление посредством многочленов. В первом случае, несмотря на достаточную точность, и использование всей исходной информации, полученная функция не будет являться гладкой, что не всегда применимо, во втором случае, приходится рассчитывать многочлены высокой степени, что трудоемко в смысле машинного времени. Широко применяются для решения задачи кубические сплайны: область задания функции делят на части, на каждой из которых строят аппроксимационный многочлен невысокой степени, при этом следят за обеспечением необходимой гладкости результирующего восстанавливающего агрегата.

В работах [1, 2] построен математический аппарат приближения функций. На основе указанного математического аппарата был реализован метод восстановления функций двух переменных, заданных таблично, состоящий из двух этапов. Сначала по табличным данным строится простая аппроксимирующая функция, затем к полученной функции, применяется сглаживающий интегральный оператор. Можно подобрать интегральный оператор таким образом, чтобы обеспечить нужную гладкость восстанавливаемой функции. Особое прикладное значение для восстановления функций на ограниченной области имеет использование финитных ядер в интегральном операторе. Требуемым свойствам отвечает интегральный оператор с ядрами Стеклова различных порядков. Финитность ядра оператора дает нам ряд преимуществ: возможность построения восстанавливающего агрегата для функций, заданных на ограниченной области, возможность параллельно производить расчеты для большого объема данных и некоторые другие полезные в прикладном плане свойства.

Для ядер Стеклова различных порядков, получены конечные вычислительные формулы, на основе которых реализован вычислительный алгоритм, позволяющий восстанавливать по известным табличным значениям функцию двух переменных, обладающую необходимой степенью гладкости.

Рассматриваемый метод, благодаря ряду полезных в прикладном плане свойств, может найти применение при решении широкого спектра задач в областях компьютерной графики, обработки изображений, вычислительных алгоритмах.

Литература

1. Додонов Н.Ю., Жук В.В. О равномерном приближении непериодических функций, заданных на всей оси // Проблемы математического анализа, 2004. – 29. – С. 25–35.
2. Додонов Н.Ю. О приближении функций в пространствах $L_p(R)$ посредством сдвигов и сжатий одной функции // Проблемы математического анализа, 2005. – 30. С. 47–59.

УДК 004.421

ИЕРАРХИЧЕСКОЕ ВОССТАНОВЛЕНИЕ РАЗРЕЖЕННОЙ СТРУКТУРЫ ПРОСТРАНСТВА И ТОЧЕК СЪЕМКИ ПО НАБОРУ ФОТОГРАФИЙ

Д.Е. Родиков
Научный руководитель – д.т.н., профессор А.А. Шалыто

Введение. Проблемы компьютерного зрения становятся все более и более актуальными по мере развития техники, вычислительных мощностей компьютеров и эффективности алгоритмов. Одним из важных направлений является 3D реконструкция структуры пространства по фото и видео данным. 3D реконструкция позволяет строить ортофотопланы, измерять объемы, роботам успешно ориентироваться в пространстве.

Постановка задачи. В задаче восстановления 3D одной из важных подзадач является задача восстановления разреженной структуры пространства и точек съемки (SFM – Structure from motion) по фотографиям. Это первый шаг в реконструкции 3D, от его точности зависит качество получающихся моделей. При обработке аэрофотоснимков объем обрабатываемых данных регулярно бывает большим, поэтому важно быстродействие применяющихся алгоритмов. Классическим подходом является последовательное добавление фотографий к группе фотографий, для которых уже вычислены точки съемки. После каждого подобного добавления применяется нелинейная оптимизация (bundle adjustment) положений точек съемки. Так как увеличением числа фотографий затраты на нелинейную оптимизацию быстро растут и линейно растет число применений оптимизации, то время работы всего алгоритма сильно увеличивается при увеличении числа фотографий (асимптотика порядка $O(n^4)$ при достаточно малой константе). Возникает задача увеличения быстродействия алгоритма для больших значений n.

Практические результаты. В работе предлагается иерархический подход к проблеме восстановления точек съемки. Исходное множество фотографий разбивается на группы по 16–64 фотографии и в каждой группе выполняется восстановление положений камер стандартным алгоритмом. Затем смежные группы объединяются в более крупные группы. Объединение производится по наборам соответствующих трехмерных точек и после каждого объединения происходит шаг оптимизации. Выбор фотографий для объединения в группы, а также групп для объединения осуществляется построением максимального остовного дерева. Вершинами графа являются фотографии (или их группы), а веса – число общих 3D точек между фотографиями (группами фотографий).

Подобный подход позволил сократить время работы алгоритма SFM в лучшем случае до $O(n^3)$ и значительно увеличить быстродействие в среднем (3–5 раз и более). При этом точность алгоритма осталась прежней. Представленный метод был успешно реализован в коммерческом продукте PhotoScan компании Agisoft.

УДК 004.021

МЕТОД СБОРКИ ГЕНОМА С ИСПОЛЬЗОВАНИЕМ ТЕХНОЛОГИИ MAPREDUCE

А.В. Александров, С.В. Казаков, С.В. Мельников, А.А. Сергушичев, П.В. Федотов, Ф.Н. Царев
Научный руководитель – д.т.н., профессор А.А. Шалыто

Многие современные задачи биологии и медицины требуют знания геномов живых организмов, который состоит из нескольких нуклеотидных последовательностей молекул дезоксирибонуклеиновой кислоты (ДНК). В связи с этим возникает необходимость в дешевом и быстром методе секвенирования, т.е. определения последовательности нуклеотидов в образце ДНК.

Изучение генома человека и других живых существ имеет важное прикладное значение. На основании результатов сборки генома конкретного человека возможна реализация персонифицированной медицины – определения предрасположенности человека к различным болезням, создание индивидуальных лекарств и т.д. Кроме этого, на основе результатов исследования геномов растений и животных с использованием методов биоинженерии могут быть выведены новые их виды, обладающие определенными свойствами.

Задача разработки методов сборки геномных последовательностей является, в определенном смысле, центральной среди всех задач биоинформатики. Это объясняется тем, что без ее решения нельзя приступить к детальному изучению генома живого существа и его анализу с применением других алгоритмов биоинформатики.

В середине первого десятилетия XXI века широкое распространение получили так называемые технологии next generation sequencing (технологии секвенирования нового поколения). По оценкам экспертов [1] эти технологии в настоящее время развиваются существенно быстрее, чем компьютерные технологии и алгоритмы сборки геномных последовательностей.

Сборка генома из набора фрагментов, полученных на секвенаторе, – алгоритмически и вычислительно сложная задача, невозможная без использования суперкомпьютерных кластеров. Например, каждая сборка генома печеночного сосальщика, основанная на данных нескольких запусков секвенатора, требует до недели работы кластера из двух десятков узлов по 8 ядер и 8 Гб оперативной памяти в каждом, объединенных по интерфейсу MPI [2]. Однако одного запуска почти всегда недостаточно — таких сборок может быть несколько из-за необходимости подбора оптимальных параметров алгоритма и добавления новых экспериментальных данных.

На сегодня процедура работы секвенатора и сборки генома на суперкомпьютере отличаются по времени в зависимости от конкретного оборудования и используемых алгоритмов, но в целом – это величины одного порядка. Выше было отмечено, что в ближайшие годы темпы роста производительности секвенаторов ожидаются более высокими, по сравнению с ростом производительности суперкомпьютеров, а значит, «узким» местом в получении геномной последовательности будет становиться именно процедура восстановления генома, выполняемая после получения результатов работы секвенатора.

Использование существующих в настоящее время алгоритмов на персональных компьютерах приведет к тому, что сборка одного генома займет месяцы, а может растянуться и на год. Для успешного решения этой задачи нужно переходить на новые алгоритмы, использовать «быстрые» языки программирования, переходить от персональных компьютеров даже не к кластерам, а к вычислительным системам петафлопсного и экзафлопсного уровней производительности.

В настоящей работе предлагается сверхмасшитабируемый параллельный метод сборки геномных последовательностей, который основан на использовании технологии MapReduce.

Сборку генома предлагается осуществлять в три этапа – исправление ошибок в чтениях, сборка квазиконитгов, сборка контигов. Алгоритмы для каждого из этапов отличаются от своих «последовательных» версий, предложенных авторами в предыдущих работах, – они построены на основе распараллеливания по данным.

Основной идеей масштабирования алгоритма исправления ошибок в данных секвенирования (наборе чтений геномной последовательности) состоит в разбиении исходных данных на достаточно большое число частей так, чтобы эти части можно было обрабатывать независимо друг от друга.

Идея распараллеливания алгоритма сборки квазиконтигов также состоит в распараллеливании по данным – чтения геномной последовательности с внесенными в них исправлениями разбиваются на группы, каждая из которых обрабатывается отдельно, на отдельном вычислительном узле. Для осуществления такого разбиения был разработан сверхмасштабируемый алгоритм кластеризации чтений геномной последовательности.

Для сборки контигов из квазиконтигов применятся алгоритм, основанный на подходе Overlap-Layout-Consensus, модифицированный для работы на одном узле с большим количеством ядер.

До начала сборки предполагается, что все квазиконтиги как-то распределены по используемым узлам. Подготовительным этапом сборки контигов является этап кластеризации, который описан выше. Благодаря ему квазиконтиги так распределяются по вычислительным узлам, что обработку на каждом узле можно вести независимо.

Такой процесс повторяется, при этом на каждом следующем этапе контиги, построенные на предыдущем этапе, считаются входными данными.

Литература

1. Зубов В.В. Приборы для чтения ДНК // Химия и жизнь, 2010. – № 7. – С. 4–7; www.dubna-oez.ru/images/data/gallery/10_2948_.pps
2. Прохорчук Е.Б. Код жизни: прочесть не значит понять. http://biomolecula.ru/content/778/

УДК 004.021

МЕТОД СБОРКИ КОНТИГОВ ГЕНОМНЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ НА ОСНОВЕ СОВМЕСТНОГО ПРИМЕНЕНИЯ ГРАФОВ ДЕ БРЮИНА И ГРАФОВ ПЕРЕКРЫТИЙ

А.В. Александров, С.В. Казаков, С.В. Мельников, А.А. Сергушичев, Ф.Н. Царев
Научный руководитель – д.т.н., профессор А.А. Шалыто

Многие современные задачи биологии и медицины требуют знания геномов живых организмов, который состоит из нескольких нуклеотидных последовательностей молекул дезоксирибонуклеиновой кислоты (ДНК). В связи с этим возникает необходимость в дешевом и быстром методе секвенирования, т.е. определения последовательности нуклеотидов в образце ДНК.

Существующие секвенаторы – устройства для чтения ДНК – не позволяют считать за один раз всю молекулу ДНК. Вместо этого они позволяют читать фрагменты генома небольшой длины. Длина фрагмента может быть разной, она является важным параметром секвенирования – от нее напрямую зависит стоимость секвенирования и время, затрачиваемое на чтение одного фрагмента: чем больше длина считываемого фрагмента, тем выше стоимость чтения и тем дольше это чтение происходит. В связи с этим сейчас получил распространение следующий дешевый и эффективный подход: сначала вычленяется случайно расположенный в геноме фрагмент длиной около 500 нуклеотидов, а затем считываются его префикс и суффикс (длиной порядка 80–120 нуклеотидов каждый). Эти

префикс и суффикс называются парными чтениями. Описанный процесс повторяется такое число раз, чтобы обеспечить достаточно большое покрытие генома чтениями. Указанным образом работают, например, секвенаторы компании Illumina.

Отметим, что описанные выше префикс и суффикс читаются с разных нитей ДНК: один – с прямой, другой – с обратно-комплементарной, причем неизвестно, который откуда. Поэтому удобно рассматривать геном и чтения, дополненные своими обратно-комплементарными копиями.

Задачей сборки генома является восстановление последовательности ДНК (ее длина составляет от миллионов до миллиардов нуклеотидов у разных живых существ) на основании информации, полученной в результате секвенирования. Этот процесс делится, как правило, на следующие этапы:

1. исправление ошибок в данных секвенирования;
2. сборка контигов – максимальных непрерывных последовательностей нуклеотидов, которые удалось восстановить;
3. построение скэффолдов – последовательностей контигов, разделенных промежутками, для длины которых найдены верхние и нижние оценки.

Одной из наиболее часто используемых при сборке генома математических моделей является так называемый граф де Брюина. На его использовании основаны следующие программные средства: Velvet, Allpaths, AbySS, SOAPdenovo, EULER.

Одним из недостатков, которым обладают перечисленные программные средства, является большой объем оперативной памяти, необходимый им для сборки генома, сходного по размерам с геномом человека (2–3 миллиарда нуклеотидов). Так, например, SOAPdenovo необходимо порядка 140 ГБ оперативной памяти, а ABySS – 21 компьютер с 16 ГБ каждый (всего – 336 ГБ). Такие затраты памяти обусловлены наличием ошибок секвенирования в исходных данных (такие ошибки ведут к увеличению размера графа де Брюина), а также неоптимальным методом хранения этого графа. Другим недостатком существующих методов является отсутствие внутреннего контроля качества сборки.

В работе предлагается метод, лишенный указанных недостатков. Построение скэффолдов в настоящей работе не рассматривается.

Сборка контигов в предлагаемом методе выполняется в два этапа.

1. Сборка квазиконтигов из чтений геномной последовательности. Квазиконтигами называются последовательности, по длине бо́льшие чтений, но не являющихся контигами в смысле невозможности наращивания вправо и влево. Этот этап осуществляется с использование графа де Брюина.
2. Сборка контигов из квазиконтигов. Осуществляется с использованием графа перекрытий и метода Overlap-Layout-Consensus.

Экспериментальные исследования разработанного метода проводились в рамках проекта Assemblathon 2, организованного университетом Калифорнии Санта-Круз. Одним из наборов данных, который был подготовлен организаторами, – набор чтений рыбы Maylandia zebra. Размер генома этой рыбы оценивается примерно в один миллиард нуклеотидов.

Для сборки контигов использовался набор чтений со средним размером фрагмента 180 и 60-кратным покрытием. Общий объем исходных данных составлял 140 ГБ (в сжатом виде), из них авторами были использованы только 44 ГБ.

Алгоритмы сборки генома были реализованы на языке программирования Java. Для запуска программ использовался компьютер с 32 ГБ оперативной памяти и двумя 4-ядерными процессорами. Суммарное время работы всех трех этапов – исправления ошибок, сборки квазиконтигов и сборки контигов – составило пять суток. Опишем подробнее результаты каждого из этапов.

Перед исправлением ошибок чтения были обрезаны, чтобы вероятность отдельной ошибки в каждом нуклеотиде не превышала 10%. После этого длина всех чтений в среднем уменьшилась на 20%. Исправление ошибок работало в течение 42 часов. В результате было

найдено 150 миллионов исправлений. Всего чтений было 600 миллионов, поэтому было исправлено в среднем каждое четвертое чтение. Сборка квазиконтигов заняла 38 часов. Квазиконтиги были получены из 60% чтений. Сборка контигов выполнилась за 26 часов. В результате было получено 734165 контигов, суммарный размер которых составляет $680 \cdot 10^6$ нуклеотидов. Длина максимального составляет 23514 нуклеотидов, средняя длина – 927, значение метрики N50 – 1799.