УДК 681.518

Ю.А. Скобцов, Д.Е. Иванов, В.Ю. Скобцов
Институт прикладной математики и механики НАН Украины, г. Донецк

Генетические алгоритмы в диагностике и проектировании цифровых схем

Разработка САПР для цифровых устройств является одной из наиболее важных задач на текущий момент. Для улучшения характеристик и эффективности таких систем применяются различные методы искусственного интеллекта. В последнее десятилетие методы, основанные на применении генетических алгоритмов (ГА), эффективно используются для решения этих задач. В данной работе обсуждаются преимущества и недостатки различных техник решения задачи построения тестов для цифровых последовательностных схем, основанных на ГА, и описывается метод, предложенный авторами и реализованный в системе АСМИД–Е. Также описывается применение различных подходов, основанных на ГА, на различных этапах проектирования СБИС. В работе показывается, что методы эволюционных вычислений могут успешно конкурировать с более традиционными подходами или быть интегрированными с ними.

Введение

Для повышения уровня «интеллекта» современных САПР компьютерных систем, расширения возможностей и повышения эффективности применяются различные методы искусственного интеллекта. Одним из самых перспективных направлений является использование эволюционных вычислений.

Особенности идей теории эволюции и самоорганизации заключаются в том, что они находят свое подтверждение не только для биологических систем, успешно развивающихся многие миллиарды лет. В настоящее время бурно развивается новое направление в теории и практике искусственного интеллекта – эволюционные вычисления (термин, обычно используемый для общего описания алгоритмов поиска, оптимизации или обучения, основанных на некоторых формализованных принципах естественного эволюционного отбора). Эволюционные вычисления используют различные модели эволюционного процесса. Среди них можно выделить следующие основные парадигмы:

1. Генетические алгоритмы.
2. Эволюционные стратегии.
3. Эволюционное программирование.
4. Генетическое программирование.

Отличаются они, в основном, способом представления искомых решений и различным набором используемых в процессе моделирования эволюции операторов.

Генетические алгоритмы

Генетические алгоритмы (ГА), являясь одной из парадигм эволюционных вычислений, представляют собой алгоритмы поиска, построенные на принципах, сходных с принципами естественного отбора. Эти принципы основаны на следующих механизмах эволюции.

1. Первый принцип ГА основан на концепции выживания сильнейших и естественного отбора по Дарвину, которая была сформулирована им в 1859 г. в книге «Происхождение видов путем естественного отбора». Согласно Дарвину, особи, которые лучше способны решать задачи, в своей среде выживают и больше размножаются (репродуцируют). В ГА каждая особь представляет собой некоторое решение данной проблемы (например, поиска экстремума функции). По аналогии с этим принципом потомки с лучшими значениями целевой (фитнесс) функции (ЦФ) имеют большие шансы выжить и репродуцировать.

2. Второй принцип ГА обусловлен тем фактом, что хромосома потомка состоит из частей, получаемых из хромосом родителей. Этот важнейший принцип был открыт в 1865 г. Менделем.

3. Третий принцип, используемый ГА, основан на концепции мутации, открытой в 1900 г. de Vries. Первоначально этот термин использовался для описания существенных (резких) изменений свойств потомков и приобретения ими свойств, отсутствующих у их родителей. По аналогии с этим принципом ГА используют подобный механизм для изменения свойств потомков, тем самым повышая разнообразие (изменчивость) особей в популяции (множестве решений).

Эти три принципа составляют ядро ГА. В соответствии с ними простой ГА использует три основных оператора: репродукция, кроссинговер, мутация. При репродукции хромосомы копируются согласно их значениям ЦФ. Копирование лучших хромосом с большими значениями ЦФ определяет большую вероятность их попадания в следующую генерацию. Оператор репродукции реализует принцип «выживания сильнейших» по Дарвину.

Оператор кроссинговера (ОК) обычно выполняется в три этапа:

1. Два стринга (хромосомы, особи) $A = a_1 a_2 ... a_n$ и $B = b_1 b_2 ... b_n$ выбираются случайно из текущей популяции после репродукции.
2. Выбирается также случайно точка кроссинговера k ($1 \le k < n$).
3. Хромосомы A и B обмениваются частями после k-й позиции и производят два новых стринга $A' = a_1 a_2 ... a_k b_{k+1} ... b_n$ и $B' = b_1 b_2 ... b_k a_{k+1} ... a_n$.

Оператор мутации (ОМ) случайным образом (с небольшой вероятностью) изменяет ген (элемент) стринга.

Используя эти три основные оператора, популяция (множество решений данной проблемы) эволюционирует от поколения к поколению Эволюция такой искусственной популяции представлена на рис.1.

Таким образом генетические алгоритмы используют случайный направленный поиск для построения (суб)оптимального решения данной проблемы. При этом направленный случайный поиск моделирует процесс естественной эволюции. Каждое решение задачи (особь) представляется хромосомой – стрингом элементов (генов). Классический «простой» генетический алгоритм [1] использует двоичные стринги – строки из двоичных

элементов 0,1 (например, 0011101), что делает его привлекательным для решения задач генерации проверяющих тестов логических схем, где решение задачи представляется в виде двоичных наборов или их последовательностей. На множестве решений определяется целевая (fitness) функция, которая позволяет оценить близость каждой особи к оптимальному решению.

Таким образом, чтобы задать генетический алгоритм, необходимо определить понятия особи, популяции, операции скрещивания и мутации, задать оценочную функцию. Очевидно также, что эффективность генетического алгоритма зависит от целого ряда параметров: размера популяции, метода выбора особей из предыдущей популяции, скрещивания и мутации, а также вида оценочной функции. Далее мы рассмотрим различные варианты реализации генетических алгоритмов при построении тестов в различных системах генерации тестов.

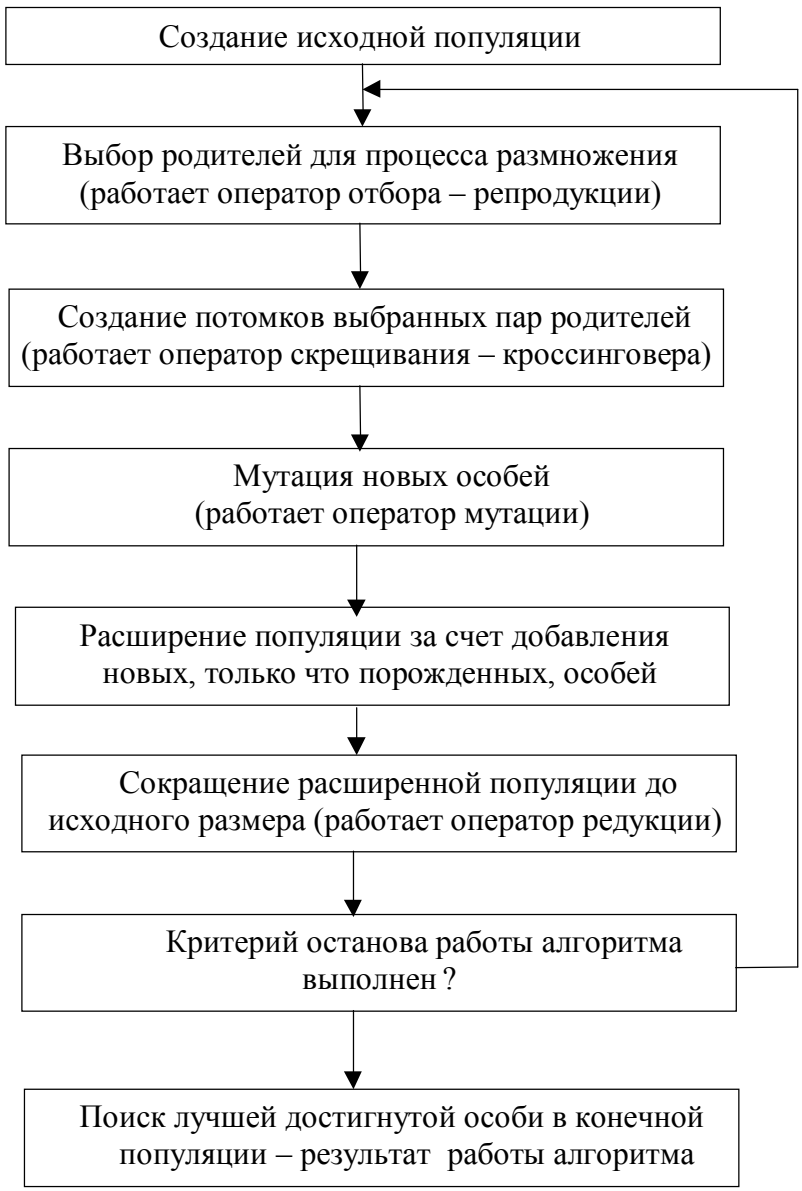

Рис. 1

Применение ГА при построении проверяющих тестов

Использование ГА при генерации проверяющих тестов является естественным развитием псевдослучайных методов генерации тестов. При применении ГА для решения этой задачи в простейшем случае в качестве особей рассматриваются отдельные тестовые двоичные наборы. Популяцией является тестовая последовательность. Такой подход, как правило, эффективен только на начальном этапе генерации тестов. Далее, особенно для последовательностных схем, целесообразно в качестве особи брать всю тестовую последовательность, длина которой заранее не известна и не ограничена. Поскольку целью генерации тестов является построение последовательности, на которой максимально отличаются значения сигналов в исправной и неисправной схемах, то качество тестовой последовательности (fitness-функция) оценивается как мера отличия значений сигналов в исправной и неисправной схемах. В программных системах генерации тестов используются различные fitness-функции, зависящие от следующих основных параметров, которые приведены в табл.1.

Таблица 1

N	Число узлов в схеме
N_d	Число узлов, имеющих различные значения сигналов в исправной и неисправной схемах
T	Число триггеров в схеме
T_d	Число триггеров, изменивших состояние
E	Число событий в исправной и неисправной схеме
L	Длина тестовой последовательности
F	Число неисправностей в схеме
F_d	Число проверенных неисправностей
F_{dt}	Число неисправностей, активизированных до триггеров
D	Обнаруживаемость неисправности
W	Мощность последовательности
O	Наблюдаемость триггеров
E_f	Число событий в неисправной схеме
T_s	Число трудно устанавливаемых триггеров

Помимо них характеристики алгоритмов генерации тестов существенно зависят от стандартных параметров ГА, таких, как мощность популяции, вероятности репродукции, кроссинговера и мутации, числа поколений и т.д.

В системе CRIS [2] используется техника иерархического моделирования, сокращающая затраты памяти и позволяющая обрабатывать схемы большой размерности. Здесь применяется традиционный ГА, в котором популяции эволюционируют посредством операторов мутации и кроссинговера. В качестве

особи берется последовательность входных наборов. CRIS основывается прежде всего на непрерывной мутации данной входной тестовой последовательности и анализе мутировавших наборов путем моделирования в основном исправных схем с целью определения тестового множества. Данная система показала хорошие результаты при построении тестов с высоким уровнем покрытия неисправностей в комбинационных и последовательностных схемах большой размерности. Однако данный подход имеет существенный недостаток – «ручную» подстройку параметров ГА для каждой схемы.

Система GATEST [3] ориентирована на последовательностные логические схемы и основана на двухуровневом ГА: на нижнем уровне с помощью ГА строятся входные наборы, а далее ГА применяется для генерации входных последовательностей на основе полученных наборов. Соответственно на нижнем уровне особью является входной вектор, а на верхнем – входная последовательность. В ГА применяются различные виды операторов кроссинговера: одноточечный, двухточечный, однородный [1]. Первый уровень в свою очередь подразделяется на три фазы. В целом, метод образует четыре фазы, которые определяют соответствующие различные оценочные функции:

– в фазе 1 целью алгоритма является инициализация триггеров, поэтому оценочная функция определяется как $h_1 = T + \frac{T_d}{T}$. При этом оценка выполняется с помощью только исправного моделирования;
– в фазе 2 предполагается, что все триггеры установлены, и целью является увеличение покрытия неисправностей полученной последовательностью путем построения новых векторов, обнаруживающих новые неисправности. Оценочная функция для этой фазы имеет вид $h_2 = F_d + \frac{T_d}{FT}$; когда генерируется набор, не обнаруживающий новых неисправностей, алгоритм построения теста переходит в фазу 3;
– в фазе 3 подсчитывается количество новых сгенерированных наборов. Для стимуляции эволюции входных наборов, тестирующих новые неисправности, в оценочной функции этой фазы учитываются события в исправной и неисправной схемах $h_3 = F_d + \frac{F_{dt}}{FT} + \frac{E}{NF}$. Если найден набор, обнаруживающий новые неисправности, то алгоритм возвращается к фазе 2. Иначе при переполнении счетчика неиспользуемых наборов процесс построения теста переходит к фазе 4;
– в фазе 4 выполняется построение тестовых последовательностей на основе полученного множества входных наборов с помощью ГА, оценочная функция определяется как

$$h_4 = F_d + \frac{F_d}{F\,TL}.$$

В фазах 2–4 оценка особей выполняется с помощью моделирования неисправностей, что несколько замедляет работу GATEST. Система успешно применялась к последовательностным устройствам достаточно большой размерности и строила тесты в тех случаях, где детерминированный метод

HITEC [4] не мог построить тесты. К недостаткам можно также отнести то, что разработчики вручную подбирают многие параметры ГА, включая размерность алфавита, размер популяции, уровень мутации.

Интересным является подход, представленный в работе [5] системой DIGATE. Он состоит из двух фаз:

– в фазе 1 происходит выбор неисправности и ее активизация (то есть распространения до триггеров) с помощью ГА;
– в фазе 2 ищется последовательность, которая сделала бы целевую неисправность наблюдаемой на внешних выходах схемы. Поиск выполняется с помощью ГА путем эволюционирования различающей последовательности, полученной заранее. Техника, применяемая во второй фазе, является ключевой для данного метода. Объединение двух полученных последовательностей образует тест.

В качестве особей, очевидно, выбраны входные последовательности. Оценка выполняется моделированием неисправностей, при этом оценочные функции представляют собой взвешенные суммы. Соответственно для фазы 1 и 2 оценочные функции имеют следующий вид:

$$h_5 = 0.2D + 0.7(W + O) + 0.1(E_f + T_s + T_d),$$

$$h_6 = 0.8D + 0.1(W + O) + 0.1(E_f + T_s + T_d).$$

Весовые коэффициенты подобраны разработчиками эвристически для каждой фазы, однако они универсальны для произвольной схемы. Этим рассмотренный метод выгодно отличается от систем, рассмотренных ранее.

В другой системе GATTO в качестве особей также выбраны входные последовательности [6]. Основное внимание авторы алгоритма уделяют определению эффективной оценочной функции как меры удаленности особи от искомого оптимума. Особи оцениваются с помощью моделирования неисправностей из соображений увеличения активности неисправности в схеме (чем больше количество линий, на которых значения в исправной и неисправной схемах различны, тем больше вероятность обнаружения неисправности). Поэтому оценочная функция определяется тремя эвристическими параметрами: взвешенное количество вентилей с разными значениями в исправном и неисправном устройствах; взвешенное количество триггеров с разными значениями в исправном и неисправном устройствах; длина последовательности-особи, параметр используется для получения компактной тестовой последовательности. В качестве весов первых двух параметров используются соответствующие наблюдаемости. Таким образом, оценочная функция есть $h_7 = \max_s L^i * h(v_i))$, где s – особь-последовательность, v_i – i-й вектор последовательности, $h(v_i)$ – сумма нормализованных первых двух параметров.

В этом подходе, также, как и в предыдущем, отсутствует ручной подбор параметров для отдельной схемы.

В системе АСМИД-Е [7], разработанной авторами, для повышения быстродействия алгоритма построения тестов в программную реализацию интегрированы две программы параллельного моделирования с неисправностями. Первая программа, реализующая параллельный по неисправностям метод

моделирования, используется в фазе 1 для проверки активизации произвольной неисправности, случайно генерированной последовательностью.

Для вычисления оценочной функции используется вторая программа моделирования с неисправностями, реализующая алгоритм «параллельного моделирования по тестовым наборам» и написанная специально для работы с генетическим алгоритмом построения тестов. После моделирования очередного тестового вектора происходит вычисление оценочной функции текущего вектора по формуле: $h(v) = c_1 N_d(v) + c_2 T_d(v)$, где v – текущий входной набор. В качестве весов используются меры наблюдаемости схемы, вычисляемые на этапе предварительной обработки схемы.

После того как для последовательности, моделируемой в определённом разряде, достигнута эффективная длина или произведено моделирование на последнем наборе, вычисляется оценочная функция всей последовательности:

$$H(s, f) = \sum_{i=1}^{i=длина} LH^i * h(v_i, f),$$

где s – анализируемая последовательность; v_i – вектор из рассматриваемой последовательности, i – позиция вектора в последовательности, f – заданная неисправность, LH – предварительно заданная константа в диапазоне $0 < LH <= 1$, благодаря которой предпочтение отдаётся более коротким последовательностям.

Применение данного алгоритма существенно повышает скорость вычисления оценочных функций особей в популяции, а следовательно, и скорость работы алгоритма в целом.

В рассмотренных методах процесс построения теста рассматривается как процесс поиска оптимума в некотором пространстве поиска с помощью ГА. Однако для преодоления ограничений, налагаемых ГА, был разработан ряд гибридных методов, объединяющих в себе преимущества ГА-методов и детерминированных алгоритмов генерации тестов. Кратко рассмотрим два основных направления в данных исследованиях.

Была предложена система Hybrid CRIS [8], в которой объединены структурный алгоритм построения тестов и метод CRIS, рассмотренный выше. Основная идея состоит в том, что выполнение алгоритма CRIS должно быть приостановлено при достижении локального экстремума эффективности. Это может быть, например, когда для заданного числа поколений процент проверяемых неисправностей не возрастает. В этом случае активизируется структурный алгоритм генерации тестов, который может проверить нетестируемые неисправности или получить тестовые последовательности для новых неисправностей. Последние добавляются в популяцию, и после преодоления критической точки работа CRIS возобновляется в режиме ГА.

Другим примером гибридного подхода является метод GA-HITEC [9] – результат объединения GATEST и HITEC. В данном методе детерминированный алгоритм HITEC применяется на этапе активизации неисправности и распространения ее до внешних выходов. GATEST применяется на этапе построения установочной последовательности для состояния схемы, полученного в результате активизации и распространения влияния неисправности.

Экспериментальные результаты показывают, что метод GA-HITEC превосходит HITEC по скорости построения тестов и уровню покрытия неисправностей.

Применение ГА на этапе конструкторско-технического проектирования

На данном этапе ГА применяются при решении различных задач комбинаторной оптимизации, таких, как размещение, упаковка, компоновка элементов, трассировка и т.п. [10]. Базовой задачей проблем комбинаторной оптимизации является классическая NP-полная задача коммивояжера. Она используется при проектировании разводки коммутаций, разработке архитектуры вычислительных сетей и т.п. Ее постановка следующая. Коммивояжеру необходимо посетить N городов, не заезжая в один и тот же город дважды, и вернуться в исходный пункт по маршруту с минимальной стоимостью. Дан граф G(X, E), где X=1…N – множество вершин (городов), $E \in X \times X$ – множество дуг (пути между городами). Дана матрица стоимостей путей A(i, j), где $i < j \in 1...N$. Требуется найти перестановку φ на множестве X такую, что целевая функция

$$H(\varphi) = A(\varphi(1), \varphi(N)) + \sum \{A(\varphi(i), A()\varphi(i+1))\} \longrightarrow \min .$$

Известно, что двоичное представление (кодирование) тура (маршрута) для этой задачи неэффективно. Здесь применяется два основных генетических представления возможного решения: 1) кодирование решения в виде списка вершин (последовательность посещаемых городов), 2) кодирование решения в виде списка ребер, образующих тур. Разработано множество генетических операторов ОК для указанных способов кодирования [10]. В качестве примера приведем «жадный» кроссинговер, который определяется так.

1. Для каждой пары родителей взять случайный город в качестве точки старта и взять его в качестве текущего города.
2. Сравнить два ребра (пути), ведущих из текущего города, и выбрать из них кратчайший.
3. Если выбранный таким образом город уже посещался, то взять случайный город из множества еще непосещенных. Присвоить полученный таким образом город текущему.
4. Повторять пункты 2 и 3, пока все города не будут посещены.

Оператор ОМ можно, например, определить как случайную перестановку двух городов в туре.

При решении задачи трассировки часто применяют алгоритмы построения кратчайших связывающих деревьев Прима и Штейнера [10]. При решении этих задач применяются методы такого же типа, как и при решении задачи коммивояжера.

Задачу размещения элементов СБИС можно определить как задачу упаковки максимального числа элементов различного размера в определенное число линеек одинаковой длины. Это также задача комбинаторной оптимизации, связанная с разделением множества объектов на подмножества.

Заключительным этапом конструкторского проектирования является верификация. Одной из важнейших задач верификации является установление взаимно-однозначного соответствия между структурной и электрической схемой СБИС и ее топологической реализацией. Эта задача сводится к установлению изоморфизма между графом электрической схемы и графом ее топологии.

Литература

1. Goldberg, David E. Genetic Algorithms in Search, Optimization & Machine Learning. – Addison-WesleyPublishing Company; Inc., 1989.
2. Saab D.G., Saab Y.G., Abraham J. CRIS: A Test Cultivation Program for Sequential VLSI Circuits // Proc. Int. Conf. on Computer Aided Disign. – 1992. – P. 216-219.
3. Rudnick E.M., Patel J.H., Greenstein G.S., Niermann T.M. Sequential Circuit Test Generation in a Genetic Algorithm Framework // Proc. Design Automation Conf. – 1994. – P. 40-45.
4. Niermann T., Patel J.H. HITEC: A Test Generator Package for Sequential Circuits // Proc. European Design Automation Conf. – 1991. – P. 214-218.
5. Hsiao M.S., Rudnick E.M., Patel J.H. Automatic Test Generation Using Genetically-Engineered Distinguishing Sequences // Proc. IEEE Test Symp. – 1996. – P. 216-223.
6. Prinetto P., Rebaudengo M., Sonza Reorda M. An Automatic Test Pattern Generator for Large Sequential Circuits based on Genetic Algorithms // Proc. Int. Test Conf. – 1994. – P. 240-249.
7. Иванов Д.Е., Скобцов Ю.А. Ускорение работы генетических алгоритмов при построении тестов // Искусственный интеллект. – 2001. – № 1. – С. 52-60.
8. Saab D.G., Saab Y.G., Abraham J. Iterative [Simulation-Based+Deterministic Tec hniques]=Complete ATPG // Proc. Int. Conf. on Computer Aided Design. – 1994. – P. 40-43.
9. Rudnick E.M., Patel J.H. Combining Deterministic and Genetic Approaches for Sequential Circuit Test Generation // Proc. Design Automation Conf. – 1995. – P. 183-188.
10. Курейчик В.М. Генетические алгоритмы. – Таганрог: Изд-во ТРТУ, 1998.

The development of CAD systems for digital devices is one of the most important problems now. For improving performance and effectiveness of such systems different methods of artificial intelligence are applied. In the past decade, genetic algorithm–based (GA–based) methods are effectively used for decision this problems. This paper discusses the advantages and disadvantages of different GA–based techniques for solving test generation problem for digital sequential circuits and describes GA–based method, suggested by authors and implemented in ASMID–E system. Also describes application of GA–based approaches at different phases of VLSI circuits design. The paper shows that evolutionary computation methods can successfully compete with more traditional approaches or be integrated with them.

Материал поступил в редакцию 12.04.02.