УДК 681.518:681.326.7

Д.Е. ИВАНОВ, к.т.н., доц., с.н.с. отдела теории управляющих систем ИПММ НАН Украины, Донецк

ПРИМЕНЕНИЕ АЛГОРИТМОВ СИМУЛЯЦИИ ОТЖИГА В ЗАДАЧАХ ИДЕНТИФИКАЦИИ ЦИФРОВЫХ СХЕМ

В статье рассматривается опыт применения эволюционного алгоритма симуляции отжига к решению задач идентификации, возникающих при проектировании цифровых схем. Описывается общая структура алгоритма симуляции отжига, его компоненты решения задач построения идентифицирующих последовательностей и их оптимизации. Проводится сравнительный анализ эффективности с генетическими алгоритмами. Ил.: 2. Табл.: 2. Библиогр.: 20 назв.

Ключевые слова: эволюционные алгоритмы, идентификация, цифровая схема, симуляция отжига, генетический алгоритм.

Постановка проблемы и анализ литературы. Эволюционные алгоритмы находят широкое применение в задачах идентификации цифровых схем [1 – 3]. Наибольшее внимание авторы уделяют генетическим алгоритмам (ГА) построения входных идентифицирующих последовательностей: разработаны генераторы проверяющих тестов [4, 5], инициализирующих последовательностей [6, 7] и верифицирующих эквивалентность двух заданных схем [8]. Преимуществом указанных выше алгоритмов является то, что они позволяют обрабатывать большие последовательностные схемы и получать приемлемые для разработчика результаты в терминах полноты и времени работы. Это свойство объясняется тем, что алгоритмы данного рода не рассматривают внутреннюю структуру схемы и не строят деревьев решений, а используют итеративный анализ свойств потенциальных решений. Также известны работы по оптимизации тестовых последовательностей, в частности, с целью уменьшения рассеивания тепла при самотестировании [9]. Известно, что все указанные выше задачи являются NP-полными [10].

В настоящее время усиливается интерес к другим эволюционным подходам, среди которых можно выделить алгоритм симуляции отжига (СО) [11 – 12]. Однако большинство исследователей ограничиваются применением данного алгоритма к решению модельных задач: задача о рюкзаке, задача раскроя и т.д. Автор данной статьи в последнее время разработал ряд практических алгоритмов решения задач, которые возникают при проектировании цифровых схем [13 – 15].

Целью данной работы является обобщение имеющегося опыта применения алгоритмов СО к решению задач идентификации цифровых

схем, а также сравнение их поисковых свойств на данных задачах с ГА.

Общая схема алгоритма симуляции отжига. Алгоритм СО впервые был предложен в виде статистической модели в [11] для нахождения состояния группы атомов в остывающем слитке. Применение алгоритма СО как оптимизирующей стратегии было начато после публикации [12], в которой авторы на модельных задачах показали, что данный подход является оптимизирующей стратегией в широком смысле.

Общая структура алгоритма СО приведена на рис. 1.

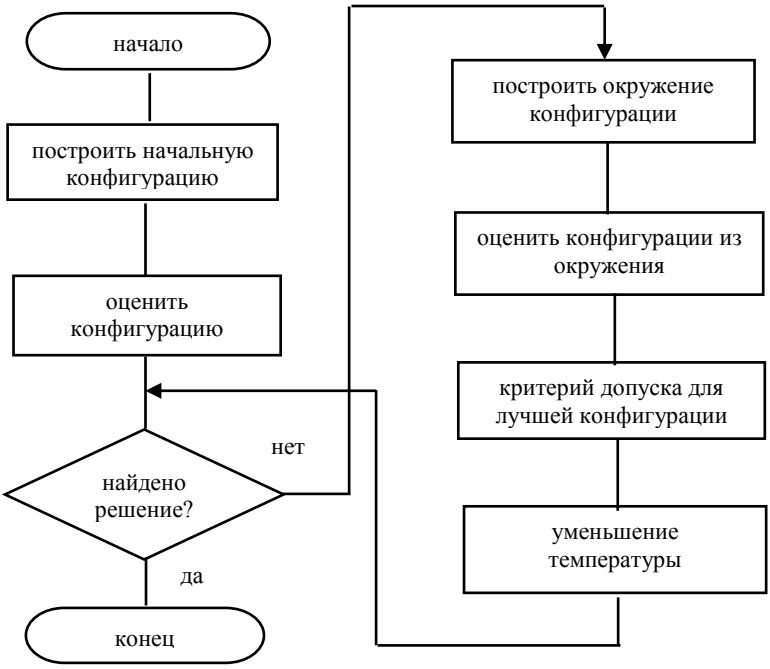


Рис. 1. Блок-схема алгоритма СО

Для задания алгоритма вводятся следующие понятия: конфигурация K_i шага i – потенциальное решение задачи; окружение $O(K_i)$ – соседние к данной конфигурации решения, которые строятся с помощью правил возмущения; оценка конфигурации $C(K_i)$, показывающая, насколько данная конфигурация хорошо решает поставленную задачу, обычно ищется решение с максимальной оценкой: $C(K_i) \to \max$;

расписание температур $\{T_j\}$ – убывающая последовательность. Понятие температуры играет особую роль в алгоритме. Её значение влияет на возможность принятия ухудшающих изменений: чем выше температура – тем чаще принимаются возмущения, которые ухудшают оценку конфигурации, и наоборот. Применение данного механизма позволяет алгоритму избегать сходимости к локальным экстремумам. В целом видно, что алгоритм представляет собой итеративный поиск, который может быть остановлен как при нахождении нужного решения, так и при достижении предельного числа итераций. Решить поставленную задачу с помощью алгоритма СО означает задать перечисленные компоненты и подобрать значения эвристических констант.

Алгоритмы СО построения входных идентифицирующих последовательностей. В зависимости от сложности решаемой задачи мы различаем одно- и двухуровневые схемы применения алгоритма СО. В первом случае происходит однократный вызов алгоритма СО для нахождения решения задачи. В случае двухуровневой схемы происходит итеративный процесс: определение локальной цели (верхний уровень) и вызов алгоритма СО для достижения локальной цели (нижний уровень). По одноуровневой схеме построены алгоритмы нахождения инициализирующих и верифицирующих эквивалентность последовательностей, по двухуровневой – генератор проверяющих тестов. Дадим более формальную постановку данных задач.

1) Построение инициализирующих последовательностей (ИП).

Пусть Q – множество всех состояний последовательностной схемы в трёхзначном алфавите моделирования $E_3 = \{0, 1, u\}$; пусть Z – множество всех возможных определённых состояний схемы, а Σ – множество всех возможных входных последовательностей s_i. Для выбранного трёхзначного моделирования $Z \subset Q$. Пусть $x \in Q$ начальное неопределённое состояние. Функция $F: Q \times \Sigma \to Q$ обозначает все состояния, достижимые схемой при поступлении на её вход последовательностей из Σ при использовании трёхзначного алфавита моделирования.

Определение 1. Последовательность $s \in \Sigma$ называется инициализирующей для заданной схемы, если в финальном состоянии $q = F(x, s)$ все состояния определены, т.е. $q \in Z$.

2) При построении последовательностей, проверяющих эквивалентность двух заданных схем, задача, по существу, сводится к доказательству несуществования входной последовательности, различающей две заданные схемы, т.е. ищется контрпример,

показывающий, что схемы различны.

Определение 2. Входная последовательность $s \in \Sigma$ называется различающей для цифровых схем A_0 и A_1, когда выходные реакции схем A_0 и A_1 различны: $A_0(s) \neq A_1(s)$.

3) Задача построения тестов. Пусть задано исправное устройство A_0 и конечное множество его неисправных модификаций $A = \{A_1, A_2, ..., A_n\}$, и $A_0 \neq A_i$ для $i = 1, ..., n$. Традиционно, мы используем множество одиночных константных неисправностей.

Определение 3. Тестом, проверяющим все неисправности модификаций A, называется такая входная последовательность, которая является проверяющей для всех $A_i \in A$, $i = 1, ..., n$.

Исходя из целей указанных задач – построение входных последовательностей – выбирается кодирование конфигураций и возмущающие операции (рис. 2). Отметим, что такое кодирование полностью соответствует кодированию особей в ГА, а возмущающие операции соответствуют операциям мутации в ГА.

Общим в данных задачах является то, что необходимо искать входную последовательность, для которой качество решения задачи проверяется путём моделирования работы цифровой схемы на данной последовательности. Для задачи построения ИП функция оценки имеет вид:

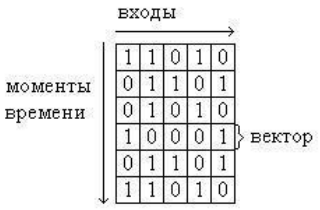
$$C(K_i) = C(s) = f(n_1, n_2, n_3) = (c_1 \times n_1 + c_2 \times n_2) \times c_3^{n_3},$$

где s – входная последовательность; n_1 – отношение числа инициированных триггеров к их общему числу; n_2 – активность схемы или число событий моделирования; n_3 – длина последовательности; c_1, c_2, c_3 – нормализующие константы.

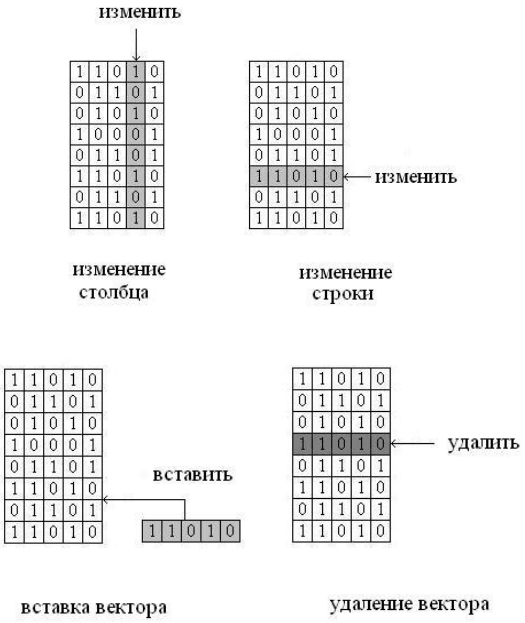
Для задачи верификации эквивалентности оценка вычисляется следующим образом:

$$C(K_i) = C(s) = f(n_1, n_2, n_3) = n_1 + c_1 \times n_2 + c_2 \times n_3,$$

где n_1, n_2, n_3 – число различных значений на внешних выходах, псевдовыходах, выходах логических элементов двух анализируемых схем соответственно.



а) конфигурация



б) возмущающие операции

Рис. 2. Представление решений и операции над ними

В задаче построения тестов проверяющая последовательность ищется для каждой из неисправностей целевого множества. Это заставляет применять в данном алгоритме двухуровневую схему. На верхнем уровне в множестве непроверенных неисправностей ищется такая неисправность $f_{\text{цел}}$, какую можно активировать с помощью псевдослучайной генерации, т.е. различие в поведении исправной и неисправной схемы распространено на внешние псевдовыходы схемы.

Далее на нижнем уровне для неисправности $f_{\text{цел}}$ с помощью алгоритма СО строится тест. Оценочная функция в этом случае имеет вид:

$$C(K_i, f_{\text{цел}}) = \sum_{i=1}^{i=\text{длина}} H^i \times h(v_i, f_{\text{цел}}),$$

где H^i – близкие к 1 константы, $h(v, f_{\text{цел}})$ определяет качество отдельного входного вектора v:

$$h(v, f_{\text{цел}}) = f_1(v, f) + c_1 \times f_2(v, f) = n_1 + c_1 \times n_2,$$

где c_1 – константа нормирования, равная отношению числа вентилей схемы к числу триггеров; фунции $f_1(v, f)$ и $f_2(v, f)$ определяют различие на множестве выходов логических элементов и псевдовыходов соответственно.

Алгоритмы СО оптимизации рассеивания тепла тестовых последовательностей. Алгоритмы СО могут быть применены не только к задачам построения идентифицирующих последовательностей, но и к их оптимизации. В [15] авторами разработана стратегия генерации энергоэффективных тестов. Она основана на понятии избыточного тестирования и состоит из трёх этапов.

На первом этапе производится генерация множества избыточных тестов $S = \{s_1, s_2, \ldots, s_l\}$. Тест называется избыточным с заданным фактором избыточности r, если каждая неисправность в целевом множестве проверяется не мене, чем r подпоследовательностями. Данный этап реализован с помощью двухуровневого алгоритма СО построения тестов путём добавления для каждой неисправности соответствующего счётчика.

На втором этапе для каждой из построенных последовательностей оценивается рассеивание тепловой энергии:

$$E(s_{\text{вх}}) = 0{,}5 \times V^2 \times C \times f \times A(s_{\text{вх}}),$$

где: V – напряжение работы схемы; C – физическая ёмкость на выходе вентиля; f – частота работы схемы; $A(s_{\text{вх}})$ – число событий при моделировании на заданной входной последовательности.

Третий этап заключается в выборе такого подмножества последовательностей $S' \subset S$, чтобы полнота теста была не хуже начальной $P(S') = P(S)$, а рассеивание тепла минимальным

$E(S') \to \min$. Очевидно, что выбор различных подмножеств последовательностей будет влиять и на полноту теста и на параметр рассеивания тепла. Задача выбора такого подмножества является NP-полной и её решение строится на основании алгоритма СО. В отличие от предыдущих задач, в данной в качестве конфигурации K_i выступает множество номеров подпоследовательностей из S, которые фактически задают входную последовательность. Список номеров в разных конфигурациях изменяется в ходе эволюции, также как и число номеров в списке. Используются три классические операции возмущения для множеств: обмен элементами, удаление элемента и добавление случайного элемента.

Алгоритм делится на две фазы, в каждой из которых применяется свой вид функции оценки. В фазе 1 из множества S выбираются такие подмножества, для каждого из которых выполнено первое условие. Функция оценки при этом имеет вид

$$C(K_i) = P(S) - P(S').$$

При обнаружении набора подпоследовательностей, для которого условие выполнено, алгоритм СО переходит к фазе 2 оптимизации данного набора. Здесь развитие конфигурации идёт таким образом, чтобы уменьшить рассеивание тепла для подмножества последовательностей, и оценка имеет вид $C(K_i) = E_{\max} - E_i$, где $E_{\max} = E(S)$.

Эффективность алгоритма СО измеряется уменьшением рассеивания тепла для конфигураций на входе и выходе фазы 2.

Сравнение алгоритмов СО и ГА. Разработка алгоритмов, рассмотренных выше, даёт разработчикам возможность выбора между алгоритмами СО и ГА. Близость данных стратегий позволяет ставить вопрос о сравнительной эффективности их поисковых свойств. В работе [16] проводился практический сравнительный анализ ГА и СО, из которого следует, что метод СО на большинстве задач не проигрывает ГА, а на многих – выигрывает. Для сравнения практической эффективности алгоритмов СО и ГА изучались результаты численных экспериментов на схемах из контрольного каталога ISCAS-89 [17]. Аккуратное сравнение различных алгоритмов является сложной проблемой [18] и определить абсолютно лучший алгоритм не представляется возможным, поскольку сравнение производится сразу по нескольким критериям.

Результаты сравнительного анализа алгоритмов СО и ГА [7] построения ИП приведены в табл. 1. Видно, что оба подхода показывают

приблизительно равные результаты.

Таблица 1. Сравнение алгоритмов СО и ГА построения инициализирующих последовательностей.

Параметр сравнения	Результаты сравнения		
	лучше ГА	лучше СО	ничья
Число инициализированных триггеров	4	0	34
Длина инициализирующей последовательности	1	3	34
Рассмотрено число точек в пространстве поиска	14	18	6

При проведении экспериментов верификации эквивалентности применялась следующая стратегия. В исходную схему вносились миноритарные изменения (заменялся тип одного случайно выбранного логического вентиля) и для двух схем строилась различающая последовательность. Эксперимент с каждой схемой проводился 25 раз. В среднем алгоритм СО построил различающую последовательность в 96.71% экспериментов, алгоритм ГА [8] – в 94.7%, алгоритм VEGA – 88,35% [19], алгоритм AQUILA – 65,00% [20]. Последние два алгоритма являются структурными и для них не приводятся результаты работы с большими схемами.

При сравнении эффективности алгоритмов построения энергоэффективных тестов СО и ГА [9] для третьего этапа использовались одинаковые результаты, полученные на этапах 1 и 2. Результаты сравнения приведены в табл. 2.

Таблица 2. Сравнение алгоритмов СО и ГА построения энергоэффективных тестов

	Результат ГА, в среднем	Результат СО, в среднем	Лучше ГА, раз	Лучше СО, раз	Ничья, раз
Уменьшение рассеивания тепла	85.59%	86.34%	0	22	2
Уменьшение длины результирующей последовательности	12.04 раз	12.89 раз	3	15	6
Время работы фазы 3	-	-	4	5	15

Видно, алгоритм СО в подавляющем числе экспериментов показал лучшие результаты как в терминах уменьшения рассеивания тепла, так и

уменьшения длинны тестовой последовательности.

Выводы. В статье рассмотрены основные подходы применения стратегии СО к решению задач построения идентифицирующих последовательностей и их оптимизации. Из анализа результатов машинных экспериментов видно, что алгоритмы, построенные на основе данной стратегии, показывают очень хорошие эксплуатационные характеристики. Более того, несмотря на упрощение эволюции – одно решение в алгоритмах СО против популяции решений в ГА – данные алгоритмы часто находят лучшие решения. На основании этого можно сделать вывод о целесообразности применения данной стратегии к решению других задач в цикле проектирования цифровых схем.

Список литературы: **1.** *Goldberg D.E.* Genetic Algorithm in Search, Optimization, and Machine Learning / *D.E. Goldberg*. – Addison-Wesley, 1989. – 432 p. **2.** *Скобцов Ю.А.* Основы эволюционных вычислений / *Ю.А. Скобцов.* – Донецк: ДонНТУ, 2008. – 326 с. **3.** Неітеративні, еволюційні та мультиагентні методи синтезу нечітко логічних і нейромережних моделей: Монографія / *Під. заг. ред. С.О. Суботіна*. – Запоріжжя: ЗНТУ, 2009. – 375 с. **4.** *Corno F.* GATTO: a Genetic Algorithm for Automatic Test Pattern Generation for Large Synchronous Sequential Circuits / *F. Corno, P. Prinetto, M. Rebaudengo, M. Sonza Reorda* // IEEE Transactions on Computer-Aided Design, August 1996. – Vol. 15. – № 8. – P. 943-951. **5.** *Skobtsov A.* Genetic algorithms in test generation for digital circuits / *Y.A. Skobtsov, D.E. Ivanov, V.Y. Skobtsov* // Proceedings of the 8th Biennial Baltic Electronics Conference. – Tallinn Technical University, 2002. – P. 291-294. **6.** *Corno F.* A Genetic Algorithm for the Computation of Initialization Sequences for Synchronous Sequential Circuits / *F. Corno, P. Prinetto, M. Rebaudengo, M. Sonza Reorda, G. Squillero* // ATS97: The Sixth IEEE Asian Test Symposium, Akita (JP). – 1997. – P. 56–61. **7.** *Иванов Д.Е.* Построение инициализирующих последовательностей синхронных цифровых схем с помощью генетических алгоритмов / *Д.Е. Иванов, Ю.А. Скобцов, А.И. Эль-Хатиб* // Проблеми інформаційних технологій. – Херсон: ХНТУ. – 2007. – № 1. – С. 158–164. **8.** *Иванов Д.Е.* Генетический подход проверки эквивалентности последовательностных схем / *Д.Е. Иванов* // "Радіоелектроніка. Інформатика. Управління". – Запоріжжя: ЗНТУ. – 2009. – № 1 (20). – С. 118–123. **9.** *Иванов Д.Е.* Генетический алгоритм оптимизации рассеивания тепловой энергии входных тестовых последовательностей / *Д.Е. Иванов* // Наукові праці Донецького національного технічного університету. Серія: "Обчислювальна техніка та автоматизація". – Випуск 18 (169). – Донецьк: ДонНТУ, 2010. – С. 206–215. **10.** *Krishnamurthy B.* On the Complexity of Estimating the Size of a Test Set / *B. Krishnamurthy, S. B. Akers* // IEEE Trans. on Computers. – August 1984. – P. 750–753. **11.** *Metropolis N.* Equation of State Calculation by Fast Computing Mashines / *N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, E. Teller* // Journal of Chem.Phys. – 1953. – Vol. 21. – № .6. – P. 1087–1092. **12.** *Kirkpatrick S.* Optimization by simulating annealing / *S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi* // Science. – 1983. – Vol. 220. – P. 671–680. **13.** *Иванов Д.Е.* Алгоритм построения инициализирующих последовательностей цифровых схем, основанный на стратегии симуляции отжига / *Д.Е. Иванов, Р. Зуауи* // Искусственный интеллект, 2009. – № 4. – С. 415–424. **14.** *Иванов Д.Е.* Верификация эквивалентности цифровых схем с использованием стратегии симуляции отжига / *Д.Е. Иванов, Р. Зуауи* // "Науковий вісник Чернівецького університету". – Вип. 479. – "Комп'ютерні системи та компоненті", 2009. – С. 33–41. **15.** *Иванов Д.Е* Алгоритм симуляции отжига оптимизации рассеивания тепла диагностических тестов / *Д.Е. Иванов, Р. Зуауи* // "Радіоелектронні і комп'ютерні системи", 2010. – № 7 (48). – С. 170–175. **16.** *Inberg L.* Simulated Annealing: Practice versus Theory / *L. Inberg* // Journal of

Mathematical Computer Modelling. – 1993. – Vol. 18. – № 1. – P. 29–57. **17.** *Brgles F.* Combinational profiles of sequential benchmark circuits / *F. Brgles, D. Bryan, K. Kozminski* // International symposium of circuits and systems, ISCAS-89. – 1989. – P. 1929–1934. **18.** *Corno F.* Comparing topological, symbolic and GA-based ATPGs: an experimental approach / *F. Corno, P. Prinetto, M. Rebaudengo, M. Sonza Reorda* // Proceedings of the IEEE International Test Conference on Test and Design Validity, Washington (USA). – 1996. – P. 39–47. **19.** *Corno F.* VEGA: A Verification Tool Based on Genetic Algorithms / *F. Corno, M. Sonza Reorda, G. Squillero* // ICCD98, International Conference on Circuit Design, Austin, Texas (USA). – 1998. – P. 321–326. **20.** *Cheng K.-T.* AQUILA: An Equivalence Checking System for Large Sequential Designs / *K.-T. Cheng, S.-Yu. Huang, K.-Ch. Chen, F. Brewer, C.-Y. Huang* // IEEE Transactions on Computers. – 2000. – Vol. 49. – № 5. – P. 443–464.

Статья представлена д.т.н., проф., зав. каф. АСУ ДонНТУ Скобцовым Ю.А.

УДК 681.518:681.326.7

Застосування алгоритмів симуляції відпалювання в задачах ідентифікації цифрових схем / Іванов Д.Є. // Вісник НТУ "ХПІ". Тематичний випуск: Інформатика і моделювання. – Харків: НТУ "ХПІ". – 2011. – № 17. – С. 60 – 69.

У статті розглянуто досвід застосування еволюційного алгоритму симуляції відпалювання до вирішення задач ідентифікації, що виникають при проектуванні цифрових схем. Описано загальну структуру алгоритму симуляції відпалювання, його компоненти рішення задач побудови ідентифікуючих послідовностей та їх оптимізації. Проведено порівняльний аналіз ефективності з генетичними алгоритмами. Іл.: 2. Табл.: 2. Бібліогр.: 20 назв.

Ключові слова: еволюційні алгоритми, ідентифікація, цифрова схема, симуляція відпалювання, генетичний алгоритм.

UDC 681.518:681.326.7

The use of simulation annealing algorithms in problems of identification of digital circuits / Ivanov D.E. / Herald of the National Technical University "KhPI". Subject issue: Information Science and Modelling. – Kharkov: NTU "KhPI". – 2011. – №. 17. – P. 60 – 69.

The article describes the experience of use of evolutionary algorithm of the simulated annealing to solve the identification problems in the design of digital circuits. It is described the overall structure of the simulated annealing algorithm and its components for algorithm of constructing of the identifying sequences and to optimize them. A comparative analysis of the effectiveness with genetic algorithms is performed. Figs.: 2. Tabl.: 2. Refs.: 20 titles.

Key words: evolutionary algorithm, identification, digital circuit, simulated annealing, genetic algorithm.

Поступила в редакцию 11.02.2011