ВЕСТНИК ТОМСКОГО ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА

2021 Управление, вычислительная техника и информатика № 55

УДК 53.083.9
DOI: 10.17223/19988605/55/14

В.М. Соловьев, Д.В. Сперанский

АЛГОРИТМ МУРАВЬИНОЙ КОЛОНИИ СИНТЕЗА ТЕСТОВ ДЛЯ ЦИФРОВЫХ УСТРОЙСТВ

Описывается эволюционный алгоритм синтеза тестов для цифровых устройств (ЦУ), основанный на использовании алгоритма муравьиной колонии. Предлагаемый алгоритм ориентирован на диагностирование таких неисправностей ЦУ, которые обычно трудно обнаруживать и диагностировать другими широко практикуемыми методами.

Ключевые слова: цифровые устройства (ЦУ); контроль и диагностика ЦУ; алгоритм муравьиной колонии; синтез тестов.

Проблемы контроля и диагностирования цифровых устройств (ЦУ) относятся к числу актуальных, поскольку обеспечение высокой надежности функционирования ЦУ невозможно без их эффективного решения. Жесткие требования к надежности ЦУ диктуются постоянно возрастающей ответственностью функций, которые на них возлагаются. Это в полной мере относится, например, к ЦУ, используемым в системах управления опасными промышленными производствами (изготовление токсичных и взрывчатых веществ и т.п.), в системах обеспечения безопасности транспортных средств (авиация, железные дороги, морские суда) и т.д. Важным этапом обеспечения высокой надежности ЦУ является эффективная организация их тестового и функционального диагностирования [1].

Предлагаемая статья посвящена задаче синтеза тестов для ЦУ. Методы синтеза тестов можно условно разделить на две группы. К первой относятся сравнительно просто реализуемые методы, но вместе с тем получаемые ими тесты обнаруживают, как правило, сравнительно большую часть возможных неисправностей ЦУ, однако не все их множество. Примером может служить метод псевдослучайного синтеза тестов [1]. Ко второй группе относятся методы, более трудоемкие в реализации, но синтезируемые ими тесты способны обнаруживать не только «простые», но и «трудные» неисправности. Примером могут служить методы различающей функции и булевых производных [1]. Предлагаемый в статье метод относится именно к этой группе.

В последние два-три десятилетия получили широкое распространение так называемые эволюционные алгоритмы [2, 3], оказавшиеся эффективными для решения многих практических проблем. В статье описан метод синтеза тестов, базирующийся на идее одного из таких алгоритмов, алгоритме муравьиной колонии (АМК), разработанном М. Дориго [4, 5] и детально описанном в [6, 7].

1. Постановка задачи

Пусть задано ЦУ в виде структурной схемы и множество F его возможных неисправностей. Для простоты изложения далее предполагается, что ЦУ является комбинационным, а рассматриваемые неисправности – одиночными константными. Требуется разработать АМК для синтеза тестов, позволяющий обнаруживать все неисправности из множества F. Далее эту задачу будем именовать задачей синтеза тестов (ЗСТ).

Рассматриваемая задача наиболее близка к классической задаче коммивояжера (ЗК) [6], и аналогия с ней будет использована нами.

В ЗСТ моделью ЦУ является неориентированный граф $G = (V, E)$, где V – множество вершин, каждая из которых соответствует некоторому входному набору ЦУ. В множество V включаются

только те входные наборы ЦУ, которые обнаруживают неисправности из F. Если p – число входов, то $|F|$ может быть существенно меньше 2^p. Множество E образуют ребра между любой парой вершин графа G. Понятно, что граф $G = (V, E)$ является полным [8]. Заметим, что в ЗК используемый в качестве модели граф полным не является.

Каждому ребру ставится в соответствие число, называемое длиной (весом), о содержательном смысле которого будет сказано ниже. В терминах описанного графа решением ЗСТ является минимальный по длине путь (маршрут) в этом графе (сумма длин всех его ребер), которому соответствует тест, обнаруживающий все неисправности из F. Отметим, что если в ЗК ее решение есть путь, проходящий через все вершины графа , то в ЗСТ минимальный по длине «тестовый» путь может и не проходить через все вершины графа. При этом, как и в ЗК, каждая вершина «тестового» пути должна посещаться только один раз. В ЗСТ это требование объясняется тем, что повторение в тесте одного и того же входного набора не увеличивает числа обнаруживаемых неисправностей ЦУ.

Из сказанного выше вытекает, что хотя между ЗК и ЗСТ, несомненно, имеется аналогия, вместе с тем существующие между ними различия требуют существенной модификации АМК решения ЗК для возможности адаптации его к решению ЗСТ.

2. Описание алгоритма

Далее предполагается знакомство читателя с терминологией и понятиями из области муравьиных алгоритмов хотя бы в рамках небольшой по объему публикации [6]. Процесс построения теста каждым k-м муравьем (их число выбирается заранее) является пошаговым. Он начинается с включения в тест некоторого входного набора с последующим добавлением к «текущему» тесту $T_{\text{тек}}$ нового набора на каждом очередном шаге. С помощью $T_{\text{тек}}$ обнаруживается некоторое множество неисправностей ЦУ, которое исключается из первоначально заданного множества неисправностей F, и в результате остается множество $F_k^{\text{н}}$ еще не обнаруженных неисправностей. Добавление к «текущему» тесту $T_{\text{тек}}$ нового набора продолжается до тех пор, пока множество $F_k^{\text{н}}$ не окажется пустым.

Перейдем теперь к содержательному описанию предлагаемого АМК, предварительно введя некоторые обозначения, сопровождаемые комментариями.

$F(w)$ – множество неисправностей из F, обнаруживаемых с использованием входного слова w.

$F_{k,t}^{\text{н}}T_{\text{тек}}$ – множество еще не обнаруженных тестом $T_{\text{тек}}$ k-м муравьем неисправностей из F на итерации t;

$d_{ij}(k,t) = |F_{k,t}^{\text{н}}(T_{\text{тек}}, i) \cap F_{k,t}^{\text{н}}(T_{\text{тек}}, j)|$ – число неисправностей, не обнаруживаемых k-м муравьем на итерации t как входным набором i, так и следующим за ним в текущем тесте входным набором j.

$$D_{ij} = \begin{cases} |F(i) \cap F(j)|, & \text{если } F(i) \neq F(j), \\ C - \text{большая константа}, & \text{если } F(i) = F(j), \end{cases}$$

D_{ij} – вес (длина) дуги (i, j) графа G).

Поясним содержательный смысл D_{ij}: чем больше множество «общих» неисправностей, обнаруживаемых наборами i и j, тем менее «привлекателен» для включения в тест после набора i входной набор j. Понятно, что вклад набора j в распознавание неисправностей из F в этом случае будет уменьшаться с ростом величины D_{ij}. Это хорошо согласуется с используемым в АМК для ЗК понятием видимости вершин графа [6]: чем больше значение D_{ij}, тем «дальше» вершина j от вершины i, т.е. тем хуже ее «видимость».

$\eta_{ij} = 1/D_{ij}$ – видимость, являющаяся локальной статической информацией, которая выражает естественное желание включить в качестве очередного элемента искомого теста входной набор с наибольшим вкладом в обнаружение неисправностей множества F.

Отметим, что значения элементов D_{ij} матрицы D находятся применением логического моделирования ЦУ при наличии в нем всех неисправностей множества F.

Самоорганизация в колонии муравьев [6] есть результат взаимодействия четырех компонентов – случайности, многократности, а также положительной и отрицательной обратных связей. Что касается многократности, в АМК для ЗСТ она реализуется за счет итерационного процесса поиска теста, осуществляемого не одним, а группой муравьев. При этом каждый муравей этой группы решает ЗСТ независимо от других. В процессе выполнения итерации каждый муравей группы решает задачу синтеза теста до конца.

Упомянутый выше принцип положительной обратной связи в АМК реализуется в виде имитации предпочтения муравьем при выборе очередного ребра (входного набора) графа при построении тестового пути. Муравей может отдать предпочтение такому из альтернативных ребер, на котором концентрация феромона максимальна. Большая привлекательность для муравья такого ребра в качестве очередного звена тестового маршрута интуитивно связана с тем, что этот след феромона «укреплен» за счет перемещения по ребру многих муравьев, задействованных для решения той же задачи.

Условимся далее концентрацию (количество) феромона на ребре (*i*, *j*) на итерации *t* обозначать как $\tau_{ij}(t)$. Откладываемый на ребрах феромон позволяет «хорошим» тестовым маршрутам сохраняться в глобальной памяти колонии муравьев. Последующие итерации АМК могут привести к улучшению таких тестовых путей. Заметим, что концентрация феромона на ребрах должна изменяться после каждой итерации АМК, что отражает приобретение муравьями опыта в поиске теста.

Однако применение только положительной обратной связи может привести к ситуации, когда все муравьи двигаются одним и тем же субоптимальным (близким к оптимальному, но не оптимальным) путем. По аналогии с АМК для ЗК в нашем алгоритме будет использоваться отрицательная обратная связь, вызываемая испарением феромона. Этот процесс приводит к тому, что группа муравьев, решающая ЗСТ, одновременно исследует разные точки пространства решений и передает свой опыт через изменение ячеек глобальной памяти колонии муравьев. Такая память представляет собой совокупность значений концентрации феромона на ребрах графа *G*, накопленных в результате итераций АМК.

Компонент случайности в нашем АМК реализуется при переходе от одного набора синтезируемого теста к следующему. По аналогии с ЗК вероятность перехода *k*-го муравья из вершины *i* в вершину *j* на *t*-й итерации имеет следующий вид:

$$\begin{cases} P_{ij,k}(t) = \dfrac{[\tau_{ij}]^{\alpha}[\eta]^{\beta}}{\sum\limits_{i \in J_{i,k}} [\tau_{il}(t)]^{\alpha}[\eta_{il}(t)]^{\beta}}, & \text{если} \quad j \in J_{i,k}, \\ P_{ij,k}(t) = 0, & \text{если} \quad j \notin J_{i,k}, \end{cases} \tag{1}$$

где $J_{i,k}$ – множество еще не использованных входных наборов ЦУ в синтезируемом тесте *k*-м муравьем, α и β – регулируемые параметры, задающие веса следа феромона и видимости при построении теста.

Понятно, что при $\alpha = 0$ концентрация феромона при выборе ребра не используется, т.е. выбор осуществляется только на основе видимости вершины *j*. Если же $\beta = 0$, то для выбора используется только концентрация феромона, что может привести к сваливанию тестовых путей к одному из субоптимальных.

Как и в АМК, для ЗК выбор очередного входного набора должен осуществляться по правилу рулетки. При этом каждый входной набор имеет в секторе рулетки площадь, пропорциональную вероятности (1).

По аналогии с ЗК формула корректировки концентрации феромона на ребре графа за счет его испарения (с параметром $\rho \in [0,1]$) имеет следующий вид:

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \Delta\tau_{ij}(t), \tag{2}$$

где $\Delta\tau_{ij}(t) = \sum\limits_{k=1}^{m} \Delta\tau_{ij,k}(t)$, *m* – количество муравьев в группе (колонии), синтезирующей тест (*m* является регулируемым параметром).

Итак, в предложенном АМК для ЗСТ самоорганизация в колонии муравьев базируется на взаимодействии компонентов случайности, многократности, положительной и отрицательной обратных связей.

Заключение

Предложенный алгоритм расширяет область приложений эволюционных вычислений на решение важного класса задач контроля и диагностирования ЦУ. Качество получаемых с применением алгоритма муравьиной колонии результатов во многих других проблемах дает основание рассчитывать на эффективные результаты и в рассмотренной нами области.

ЛИТЕРАТУРА

1. Сперанский Д.В., Скобцов Ю.А., Скобцов В.Ю. Моделирование, тестирование и диагностика цифровых устройств. 2-е изд. М. : Нац. открытый ун-т ИНТУИТ, 2016. 535 с.
2. Курейчик В.В., Курейчик В.М., Родзин С.И. Теория эволюционных вычислений. М. : Физматлит, 2012. 260 с.
3. Скобцов Ю.А., Сперанский Д.В. Эволюционные вычисления : учеб. пособие. М. : Нац. открытый ун-т ИНТУИТ, 2015. 326 с.
4. Dorigo M., Maniezzo V., Colorni A. The Ant System: Optimization by a colony of cooperating objects // IEEE Trans. on Systems, Man and Cybernetics. Part B. 1996. № 26 (1). P. 29–41.
5. Dorigo M., Gambardella M.A. Ant colony systems: a cooperative learning approach to the traveling salesman problem objects // IEEE Trans. on Evolutionary Computation. 1997. № 1 (1). P. 53–66.
6. Штовба С.Д. Муравьиные алгоритмы // Exponenta Pro. Математика в приложениях. 2003. № 4. С. 70–75.
7. Ермолаев С.Ю. Муравьиные алгоритмы оптимизации // Инфокоммуникационные технологии. 2008. Т. 6, № 1. С. 23–29.
8. Новиков Ф.А. Дискретная математика. 3-е изд. СПб. : Питер, 2017. 497 с.

Поступила в редакцию 3 сентября 2020 г.

Solovyev V.M., Speranskiy D.V. (2021) THE ANT COLONY ALGORITHM FOR TEST SYNTESIS FOR DIGITAL DEVICES. *Vestnik Tomskogo gosudarstvennogo universiteta. Upravlenie, vychislitelnaja tehnika i informatika* [Tomsk State University Journal of Control and Computer Science]. 55. pp. 122–126

DOI: 10.17223/19988605/55/14

The problem of test synthesis for digital devices (DD) is studied. Test synthesis methods can be divided into two groups. The first includes methods that are relatively easy to implement, but the tests they synthesize are detecting significant part of all possible faults. An example is the pseudo-random test synthesis method. The second group includes methods that are more laborious in implementation, but are capable of detecting faults that are difficult to diagnose. These include methods, for example, of the distinguishing function and Boolean derivatives.

The proposed method belongs to this group. Nowadays, so-called evolutionary algorithms are widespread. These algorithms are very effective for many practical applications. The article proposes a method for test synthesis based on the ideas of evolutionary ant colony algorithms.

We briefly describe the problem under study. Let DD be determined in the form of a structural scheme and the set F of its possible faults. For simplicity, it is assumed that the DD is a combinational device, and the set *F* consists of single constant faults. We consider the problem of test synthesis (PTS) for DD, detecting out of faults from the set *F*.

In the PTS the mathematical model of the DD is the undirected graph $G = (V,E)$, where V is the set of vertices. Each of its vertices is some binary input of DD. Only the inputs that detect faults are included in the set V. The set E forms edges between any pair of vertices of the graph G. Each edge (i,j) is assigned to a some number (weight). Its meaning is to assess the feasibility of including the input j in the step-by-step process synthesis of test after the input i.

Considered PTS is the most close to the classic traveling salesman problem (TSP).The basis of the proposed ant colony algorithm for the PTS is the analogy with the ant colony algorithm for solving the TSP. The article presents formulas similar to those used in the ant colony algorithm for solving TSP to calculate the probabilities of transitions from one input symbol of the test to the next, correction of the pheromone concentration on the edges of the graph G etc. Note that in the proposed ant colony algorithm for solving an TSP the basic principle of self-organization is based on the interaction of components of randomness, multiplicity, positive and negative feedbacks.

Keywords: digital devices; control and diagnostics of DD; ant colony algorithm; test synthesis.

SOLOVYEV Vladimir Mikhailovich (Candidate of Technical Sciences, Associate Professor, Head of the Volga Regional Center for New Information Technologies of Saratov State University, Saratov, Russian Federation).
E-mail: ign1122@mail.ru

SPERANSKIY Dmitriy Vasilyevich (Professor, Doctor of Technical Sciences, Russian University of Transport (MIIT), Moscow, Russian Federation).
E-mail: speranskiy.dv@gmail.com

REFERENCES

1. Speranskiy, D.V., Skobtsov, Yu.A. & Skobtsov, V.Yu. (2016) *Modelirovanie, testirovanie i diagnostika tsifrovykh ustroystv* [Modeling, testing and diagnostics of digital devices]. 2nd ed. Moscow: INTUIT.
2. Kureychik, V.V., Kureychik, V.M. & Rodzin, S.I. (2012) *Teoriya evolyutsionnykh vychisleniy* [Theory of Evolutionary Computations]. Moscow: Fizmatlit.
3. Skobtsov, Yu.A. & Speranskiy, D.V. (2015) *Evolyutsionnye vychisleniya* [Evolutionary Computations]. Moscow: INTUIT.
4. Dorigo, M., Maniezzo, V. & Colorni, A. (1996) The Ant System: Optimization by a colony of cooperating objects. *IEEE Trans. On Systems, Man and Cybernetics*. Part B. 26(1). pp. 29–41.
5. Dorigo, M. & Gambardella, M.A. (1997) Ant colony systems: a cooperative learning approach to the traveling salesman problem objects. *IEEE Trans. on evolutionary computation*. 1(1). pp. 53–66.
6. Shtovba, S.D. (2003) Ant algorithms. *Exponenta Pro*. *Mathematics in Applications*. 4. pp. 70–75.
7. Ermolaev, S.Yu. (2008) Ant algorithms of optimization. *Infokommunikatsionnye tekhnologii*. 6(1). pp. 23–29.
8. Novikov, F.A. (2017) *Diskretnaya matematika* [Discrete mathematics]. 3rd. ed. St. Petersburg: Piter.