РОССИЙСКАЯ АКАДЕМИЯ НАУК

СИБИРСКОЕ ОТДЕЛЕНИЕ

А В Т О М Е Т Р И Я

2006, том 42, № 1

УДК 681.324 + 519.17

О. Г. Монахов

(Новосибирск)

ЭВОЛЮЦИОННЫЙ СИНТЕЗ АЛГОРИТМОВ НА ОСНОВЕ ШАБЛОНОВ

Описан новый подход к синтезу алгоритмов, основанный на эволюционных вычислениях, шаблонах (темплейтах, скелетонах) алгоритмов и заданного множества пар входных–выходных данных. Представлен алгоритм эволюционного синтеза, объединяющий преимущества генетических алгоритмов и генетического программирования и используемый при автоматизации открытия и синтеза некоторых вычислительных и комбинаторных алгоритмов.

Введение. В предлагаемой работе рассматривается проблема синтеза алгоритма A как задача поиска параметров и функций заданного шаблона T (темплейта) алгоритма с целью оптимизации заданной функции F, характеризующей качество алгоритма A. Шаблон (template, skeleton [1, 2], design pattern [3]) представляет собой управляющую параметризованную структуру алгоритма, которая описывает порядок сканирования структур данных алгоритма и определяет динамику вычислительного процесса в пространственно-временных координатах.

Пусть темплейт T алгоритма A содержит параметры $P = \{p_k\}, k \ge 0$, описывающие значения целочисленных и действительных коэффициентов и переменных, значения индексов, параметры структур данных, константы и некоторые примитивные операции алгоритма (величины инкрементов и декрементов, знаки переменных, логические операции и отношения, типы округления переменных). Пусть темплейт T содержит также множество функций (формул) $FM = \{f_n\}$, $n \ge 0$, алгоритма A. При задании значений параметров P и определении функций FM в данном темплейте T получим (синтезируем) алгоритм $A(T, P, FM)$.

Целевая функция F оценивает величину рассогласования между требуемыми Y_i и полученными выходными значениями алгоритма $Y_i' = A(T, P, FM, X_i)$ при заданных входных значениях X_i, $1 \le i \le N$.

Если заданы наборы значений входных параметров алгоритма X_i, $1 \le i \le N$, и выходных параметров Y_i, которые необходимо получить в результате исполнения алгоритма, то качество его работы будет характеризовать целевая функция F, оценивающая величину рассогласования между требуемыми Y_i и полученными выходными значениями алгоритма при заданных входных значениях $Y_i' = A(T, P, FM, X_i)$.

Кроме того, функция F должна оценивать не только точность, но и сложность получаемого алгоритма.

Таким образом, проблема синтеза алгоритма состоит в следующем: для данного темплейта T и заданных наборов значений входных–выходных параметров $\{X_i, Y_i\}$, $1 \le i \le N$, необходимо найти такие значения параметров P^* и функций FM^* темплейта T, определяющие алгоритм $A^*(T, P^*, FM^*)$, что

$$F(A^*(T, P^*, FM^*, X_i)) \le F(A(T, P, FM, X_i))$$

для всех $1 \le i \le N$ при любых других значениях параметров $P \in \mathrm{Dom}(P)$ и функций $FM \in \mathrm{Dom}(FM)$.

Для решения этой проблемы в данной работе предлагается новый подход и основанный на темплейтах эволюционный алгоритм для автоматизированного синтеза (открытия) алгоритмов, оптимизирующих заданный критерий качества F. Такой подход основан на интеграции генетических алгоритмов (ГА) [4] и генетического программирования (ГП) [5] и позволяет получить новые свойства, не имеющиеся у исходных компонентов (меньшее время исполнения и синтез алгоритмов с большей циклической сложностью и наличием рекурсивных функций по сравнению с ГП и синтез новых функций и предикатов, что невозможно при ГА).

Эти свойства рассматриваемого подхода основаны на предварительном знании прикладной области и выборе обобщенной вычислительной схемы алгоритма, задаваемой в виде темплейта. Традиционное генетическое программирование страдает от слабо ограниченного слепого поиска оптимальных значений в огромных пространствах параметров для реальных прикладных проблем и имеет большие временные затраты. В данной работе исследуется использование темплейтов алгоритмов для ограничения пространства поиска на допустимых моделях и структурах решений. Этот подход представляет гибкий и прямой способ включения знаний эксперта относительно параметров, функций и структуры решаемой проблемы в разрабатываемую вычислительную модель алгоритма. В предельных случаях, с одной стороны, если мы не знаем темплейт искомого алгоритма, данный подход дает нам традиционное генетическое программирование. С другой стороны, если мы знаем полную структуру алгоритма и не знаем только некоторые его параметры, такой подход дает нам традиционный генетический алгоритм для поиска оптимальных параметров. Примером такой параметрической оптимизации на основе заданного темплейта с помощью генетического алгоритма служит алгоритм, предназначенный для открытия аналитических описаний новых плотных семейств оптимальных регулярных сетей, представленный в [6, 7].

Практические подходы к применению гибридных алгоритмов ГА и ГП представлены в [8–10]. В этих работах темплейты не используются, алгоритм ГП осуществляет поиск не множества формул FM, а только одной необходимой формулы, и ГА ищет оптимальные значения констант, содержащихся в ней. Используются разные генетические операторы для манипуляции с формулами и константами, представленными в «хромосоме» в виде дерева и вектора соответственно. Другой интересный подход к синтезу алгоритмов и программ, основанный на эволюционных вычислениях, но без использования генетических операторов и темплейтов, приведен в [11]. Необходимо отметить одну из первых работ в данной области [12], где описана базовая идея

автоматического синтеза формул по заданному набору эмпирических данных с помощью алгоритма случайного поиска.

Алгоритм эволюционного синтеза основан на эволюционных вычислениях и моделировании процесса естественного отбора популяции особей, каждая из которых является точкой в пространстве решений задачи оптимизации. Особи представлены структурами данных Gen – хромосомами, включающими недоопределенные параметры p_k и функции (формулы) f_n темплейта T:

$$\text{Gen} = \{P, FM\} = \{p_1, p_2, \ldots, p_k;\ f_1, f_2, \ldots, f_n\}, \quad k, n \geq 0.$$

Эти параметры и функции определяют необходимый алгоритм $A(T, \text{Gen})$ на основе заданного темплейта T и хромосомы Gen. Каждая популяция является множеством структур данных Gen и определяет множество алгоритмов $A(T, \text{Gen})$, образующихся из данного темплейта T.

Основная идея алгоритма синтеза состоит в эволюционном преобразовании множества хромосом (параметров и формул темплейта) в процессе естественного отбора с целью выживания «сильнейшего». В нашем случае этими особями являются алгоритмы, имеющие наименьшее значение целевой функции. Алгоритм начинается с генерации начальной популяции. Все особи в этой популяции создаются случайно, затем отбираются наилучшие особи и запоминаются. Для создания популяции следующего поколения (следующей итерации) новые особи формируются с помощью генетических операций селекции (отбора), мутации, кроссовера и добавления новых элементов (для сохранения разнообразия популяции).

Примем, что целевая функция F (fitness function, функция качества, функция пригодности) вычисляет сумму квадратов отклонений выходных данных алгоритма $Y_i' = A(T, \text{Gen}, X_i)$ от заданных эталонных значений Y_i для определенных наборов множества входных данных X_i, $1 \leq i \leq N$:

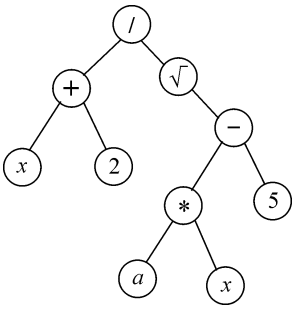
$$F = \sum_{i=1}^{N} (A(T, \text{Gen}, X_i) - Y_i)^2.$$

Среди алгоритмов с одинаковым значением функции F лучшими (для дальнейшего отбора) считаются алгоритмы с минимальным значением $F_2 = C(A(T, \text{Gen}))$ – показателем, характеризующим сложность алгоритма (время выполнения, число итераций, сложность формул). На практике было использовано число итераций для оценки сложности в случае синтеза итерационных алгоритмов и общая длина (сумма числа элементов) формул в других случаях.

Целью алгоритма является поиск минимума F.

Представление данных. Основными структурами данных в программной реализации эволюционного алгоритма являются хромосомы Gen. Для представления и реализации хромосомы был предложен новый подход, основанный на комбинации линейной структуры хромосомы для параметров p_k (как в генетических алгоритмах [4]) и древовидной структуры хромосомы для каждой из функций (формул) f_n (как в генетическом программировании [5]). Линейная структура хромосомы Gen применяется для представления таких параметров p_k темплейта T, как значения целочисленных и

Рис. 1. Представление формулы в виде дерева



действительных коэффициентов и переменных, значения индексов, величины инкрементов и декрементов, а также знаков переменных, логических операций и отношений, типов округления переменных. Древовидные структуры в хромосоме Gen применяются для представления каждой из функций (формул) f_n. Переменные и константы формулы f_n образуют листья (терминальные вершины) дерева TS, а операции и примитивные функции, используемые в формуле f_n, образуют остальные (нетерминальные) вершины дерева NS. Каждая операция (примитивная функция) и ее операнды (аргументы примитивной функции) представлены вершиной и ее сыновьями в дереве соответственно.

Пусть дана формула $f_n = (x+2)/\sqrt{ax-5}$, тогда имеем следующие терминальные $TS = \{x, a, 2, 5\}$ и нетерминальные $NS = \{+, -, *, /, \sqrt{\ }\}$ множества вершин. Пример соответствующего дерева представлен на рис. 1.

Для заданного темплейта T при создании хромосом Gen порождаются аналитические выражения для функций f_n и задаются значения параметров p_k, по которым можно производить оценивание и модификации алгоритма $A(T, \text{Gen})$.

Таким образом, для фиксированных значений функций f_n и параметров p_k мы можем вычислять выходные значения Y_i' алгоритмов $A(T, \text{Gen}, X_i)$, порожденных на основе заданных темплейтов T и полученных в ходе эволюции хромосом Gen, для требуемых наборов множества входных данных X_i, $1 \le i \le N$. После выполнения алгоритмов A мы получаем значения целевой функции F и выбираем лучшие алгоритмы в популяции.

Операторы алгоритма. Оператор *мутации* применяется к особям, случайно выбранным из текущей популяции с вероятностью $p_m \in [0,1]$. Мутация линейной части хромосомы Gen состоит в замене значения случайно выбранного параметра p_k другой, случайно выбранной величиной из множества допустимых значений. Мутация древовидной части хромосомы Gen состоит в замене значения случайно выбранной вершины в представлении функции f_n другой, случайно выбранной величиной из множества допустимых значений.

Оператор *кроссовера* (скрещивания) применяется к двум особям (родителям), случайно выбранным из текущей популяции с вероятностью $p_c \in [0,1]$. Кроссовер состоит в порождении двух новых особей путем обмена частями хромосом родителей (обмен подчастями линейных структур и обмен поддеревьями для древовидных структур хромосомы).

Оператор создания *нового элемента* (особи) состоит в генерации случайных значений параметров p_k и функций f_n, описывающих алгоритмы. Это позволяет увеличить степень разнообразия особей при создании популяции.

Оператор *селекции* (отбора) реализует принцип выживания наиболее приспособленных особей. Он выбирает наилучшие особи с минимальными значениями целевой функции.

Заметим, что только простейшие генетические операторы были использованы в алгоритме, но данный подход позволяет применять и более сложные операторы, разработанные для ГП и ГА.

Итерационный процесс. Для поиска оптимума заданной целевой функции F итерационный процесс вычислений в алгоритме эволюционного синтеза организован следующим образом.

Первая итерация: порождение начальной популяции. Все особи популяции создаются с помощью оператора *новый элемент* с проверкой и отсеиванием всех непригодных особей. После заполнения массива популяции лучшие особи отбираются и запоминаются в массиве best.

Промежуточная итерация: шаг от текущей к следующей популяции. Основной шаг алгоритма состоит в создании нового поколения особей на основе массива best и текущей популяции с использованием операций селекции, мутации, кроссовера и добавления новых элементов.

После оценки целевой функции для каждой особи в поколении проводится сравнение величин этих функций с величинами целевых функций тех особей, которые сохранены в массиве best. В том случае, если элемент из нового поколения лучше, чем элемент best$[i]$ для некоторого i, помещаем новый элемент на место i в массив best и сдвигаем в нем все остальные элементы на единицу вниз. Таким образом, лучшие элементы локализуются в верхней части массива best.

Последняя итерация (критерий остановки): итерации заканчиваются либо после исполнения заданного числа шагов, либо после нахождения оптимального алгоритма $A(T, \mathrm{Gen})$ (с заданным значением целевой функции F). После выполнения данного количества шагов алгоритма синтеза получаем множество (популяцию) алгоритмов $A(T, \mathrm{Gen})$, содержащее в элементе best[0] алгоритм $A^{*}(T, \mathrm{Gen})$, имеющий минимальное значение целевой функции F.

Экспериментальные результаты. Алгоритм эволюционного синтеза на основе темплейтов был успешно применен при автоматизированном получении следующих алгоритмов: вычисления степени и факториала натурального числа, определения минимального (максимального) элемента массива, определения суммы квадратов элементов массива, вычисления скалярного произведения векторов, определения формул последовательностей Фибоначчи и Трибоначчи, вычисления суммы двух матриц, а также алгоритмов сортировки методами пузырька, слияния и Шелла, алгоритмов вычисления корней уравнений, параллельных алгоритмов балансировки нагрузки в многопроцессорных системах, алгоритмов определения минимального покрывающего дерева и кратчайших путей в графе. Данный подход был также применен при открытии аналитических описаний новых плотных семейств оптимальных регулярных сетей [6, 7] и синтеза нового алгоритма определения функции расстояния для циркулянтных графов степени 4, представленного далее.

Эволюционный синтез алгоритмов на основе шаблонов был реализован на языке программирования C, темплейты также задавались на этом языке. Число итераций и размер популяции выбирались экспериментальным путем, основываясь на параметрах из [4, 5].

Вычисление корней квадратного уравнения. Для примера рассмотрим процесс получения алгоритма вычисления корней квадратного уравнения

вида $f(x) = ax^2 + bx + c = 0$. Пусть задано 20 наборов пар вход–выход (будем рассматривать случай только действительных корней): $\{X_i, Y_i\}$, где $X_i = (a_i, b_i, c_i)$ – коэффициенты *i*-го уравнения; $Y_i = (r1_i, r2_i)$ – корни *i*-го уравнения, $1 \le i \le N$, $N = 20$. Выберем два темплейта. Первый T_1 имеет вид

$$r_{1,2} = \varphi_1(a, b, c) \pm \varphi_2(a, b, c)$$

(φ – неизвестные функции).

Второй темплейт T_2 задан в виде итерационного цикла:

1) $\{k = 0;\ x_k = x_{\text{beg}};$

2) do $\{x_{k+1} = \varphi_1(x_k, f(x_k), f'(x_k));\ k = k + 1\};$

3) while $((f'(x_k) = \varepsilon > 10^{-7}) \,\&\, (k < 500));$

4) return $x_k\}$

с предельным числом итераций $it < 500$, с заданной точностью $\varepsilon < 10^{-7}$, начальным значением x_{beg} и процедурой вычисления производной функции $f'(x)$. Терминальными символами для T_1 являются $TS_1 = \{a, b, c, Cr\}$, а для $T_2 - TS_2 = \{x_k, f(x_k), f'(x_k), Cr\}$, где Cr – множество случайных натуральных чисел. Набор операций, используемый для синтеза формул в обоих темплейтах, следующий: $NS = \{+, -, *, /, \sqrt{\ }, x^2\}$.

В результате выполнения алгоритм эволюционного синтеза в случае первого темплейта нашел известную формулу

$$r_{1,2} = -b \Big/ 2a \pm \sqrt{b^2 - 4ac} \Big/ 2a,$$

а в случае второго темплейта синтезировал выражение

$$x_{k+1} = x_k - f(x_k) \Big/ f'(x_k),$$

что является формулой метода Ньютона (метода касательных). Первый результат был получен за 10216 итераций за 70 с при величине популяции 200; второй результат был получен за 360 итераций за 10 с при величине популяции 200 (средние значения вычислены по результатам десяти экспериментов).

Определение формул последовательностей Фибоначчи и Трибоначчи. Покажем, как можно найти рекурсивную функцию для вычисления членов последовательности Фибоначчи на основе заданного темплейта и восьми первых членов с индексами $1 \le i \le 8$. Был использован темплейт

$$F(i) = \varphi_3(F(\varphi_1(i)),\ F(\varphi_2(i)))$$

с множеством операций $NS = \{+, -, *, /\}$ и множеством терминальных символов $TS = \{i, Cr\}$. Алгоритм эволюционного синтеза определил: $\varphi_1(i) = i - 1$, $\varphi_2(i) = i - 2$ и $\varphi_3(x_1, x_2) = x_1 + x_2$ после 411 итераций за время 12 с при популяции 3000 особей (средние значения получены по результатам десяти экспериментов).

Рекурсивная функция для вычисления членов последовательности Трибоначчи была найдена на основе заданного темплейта:

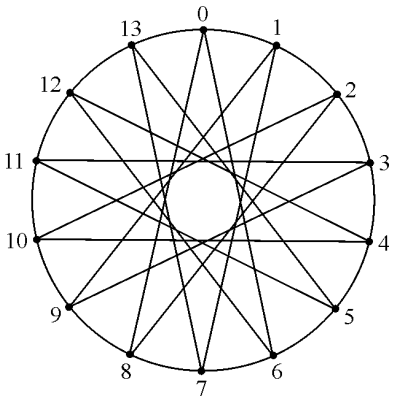


Рис. 2. Циркулянтный граф $C(14; 1, 6)$

$$F(i) = \varphi_4(F(\varphi_1(i)), F(\varphi_2(i)), F(\varphi_3(i))).$$

Алгоритм эволюционного синтеза определил после 460 итераций за время 209 с при популяции 30000 особей следующие формулы: $\varphi_1(i) = i-1$, $\varphi_2(i) = i-2$, $\varphi_3(i) = i-3$ и $\varphi_4(x_1, x_2, x_3) = x_1 + x_2 + x_3$ для заданных множества операций $NS = \{+, -\}$, множества терминальных символов $TS = \{i, Cr\}$ и первых десяти членов последовательности с индексами $1 \le i \le 10$.

Синтез алгоритма определения функции расстояния для циркулянтных графов степени 4. Циркулянтные графы [13–15] широко исследуются при проектировании и анализе вычислительных систем (ВС), в теории графов и дискретной математике, кроме того они реализованы как сети связи в некоторых многопроцессорных ВС. Дадим их определение.

Циркулянтный граф есть ненаправленный граф $G(N; s_1, s_2, ..., s_n)$ с N вершинами, помеченными числами $V = 0,1,2,..., N-1$, имеющий $i \pm s_1$, $i \pm s_2, ..., i \pm s_n (\mathrm{mod} N)$ вершины, смежные с каждой вершиной i. Отметки $S = (s_i)$ $(0 < s_1 < ... < s_n < N/2)$ есть образующие конечной абелевой группы автоморфизмов, связанной с графом. Циркулянтные графы вида $G(N; 1, s_2, ..., s_n)$ с единичной образующей известны еще как кольцевые сети [13]. Степень вершины в циркулянтном графе G равна $2n$, где n – размерность графа. Далее будут рассматриваться циркулянтные графы степени 4 с описанием $G(N; 1, s)$. Например, циркулянтный граф $C(14; 1, 6)$ степени 4 для $N = 14$, $s_1 = 1$, $s_2 = 6$ показан на рис. 2.

Диаметр графа определяется как

$$d(N; S) = \max_{u,\, \mathrm{v} \in V} D(u, \mathrm{v}),$$

где $D(u, \mathrm{v})$ – функция расстояния, или длина кратчайшего пути между двумя вершинами u и v в G. В силу симметрии циркулянтных графов достаточно рассматривать проблему нахождения кратчайшего пути между вершиной 0 и произвольной вершиной v в G.

Функция расстояния между вершинами 0 и v в циркулянтном графе $C(200; 1, s)$ с $N = 200$ для $0 < s < N/2$ и $0 < \mathrm{v} < N/2$ представлена на рис. 3.

Для определения функции расстояния $D(0, \mathrm{v})$ рассмотрим множество кратчайших путей циркулянтного графа G. Для любой вершины w определим $+s$ и $-s$ ребра в зависимости от того, используются они в пути к вершине $(w + s) \mathrm{mod} N$ (по часовой стрелке) или $(w - s) \mathrm{mod} N$ (против часовой стрелки). Аналогично определим +1 и –1 ребра. Заметим, что в циркулянтном графе кратчайший путь от 0 к v может использовать $(+s, +1)$, или $(+s, -1)$, или $(-s, +1)$, или $(-s, -1)$ ребра. Далее будем рассматривать только эти комбинации ребер.

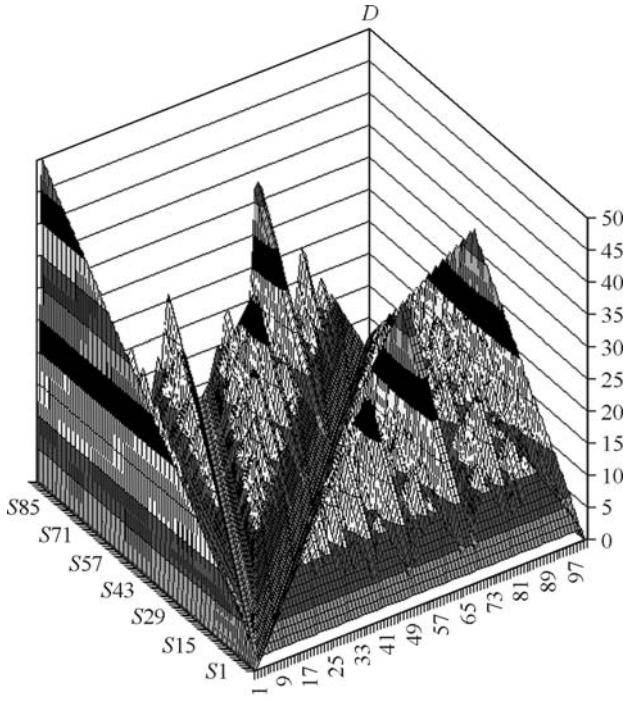


Рис. 3. Функция расстояния циркулянтного графа $C(200; 1, s)$

Пусть $(+s, +1)$-путь есть путь от 0 к v, использующий только $+s$ и $+1$ ребра. Для других комбинаций ребер будем использовать аналогичную нотацию. Далее $x/s = \lfloor x/s \rfloor$, $x \,\%\, s = x \bmod s$.

Для того чтобы пройти к вершине v от 0 посредством четырех возможных путей, мы должны использовать:

1) $(+s, +1)$-путь (v/s ребер типа $+s$ и $\mathrm{v} \,\%\, s$ ребер типа $+1$);

2) $(+s, -1)$-путь ($\mathrm{v}/s + 1$ ребер типа $+s$ и $s - \mathrm{v} \,\%\, s$ ребер типа -1);

3) $(-s, -1)$-путь ($(N - \mathrm{v})/s$ ребер типа $-s$ и $(N - \mathrm{v}) \,\%\, s$ ребер типа -1);

4) $(-s, +1)$-путь ($(N - \mathrm{v})/s + 1$ ребер типа $-s$ и $s - (N - \mathrm{v}) \,\%\, s$ ребер типа $+1$).

Это соответствует одному перемещению по кольцу по часовой стрелке и одному перемещению по кольцу против часовой стрелки ($t = 0$). Обобщая этот процесс для $t \geq 0$, получаем для вершины v:

1) все $(+s, +1)$-пути $((\mathrm{v} + tN)/s$ ребер типа $+s$ и $(\mathrm{v} + tN) \,\%\, s$ ребер типа $+1$);

2) все $(+s, -1)$-пути $((\mathrm{v} + tN)/s + 1$ ребер типа $+s$ и $s - (\mathrm{v} + tN) \,\%\, s$ ребер типа -1);

3) все $(-s, -1)$-пути $(((t+1)N - \mathrm{v})/s$ ребер типа $-s$ и $((t+1)N - \mathrm{v}) \,\%\, s$ ребер типа -1);

4) все $(-s, +1)$-пути $(((t+1)N - \mathrm{v})/s + 1$ ребер типа $-s$ и $s - ((t+1)N - \mathrm{v}) \,\%\, s$ ребер типа $+1$).

Заметим, что в силу симметрии циркулянтных графов $(-s,+1)$- и $(-s,-1)$-путей от 0 к вершине $\mathrm{v}+tN$ могут быть заменены $(+s,-1)$- и $(+s,+1)$-путями соответственно от 0 к вершине $(t+1)N-\mathrm{v},\ t\geq 0$.

Необходимо найти кратчайшие пути всех четырех типов, и кратчайший из них даст глобальный кратчайший путь между 0 и v. Число циклов $t<s$, потому что $\mathrm{v}\ \%\ s<s$ для любых $0\leq \mathrm{v}<N$.

На основе приведенных свойств множества кратчайших путей циркулянтного графа $C(N;1,s)$ для определения функции расстояния $\mathrm{dist}=D(0,\mathrm{v})$ был использован следующий темплейт:

1) int $t,k,k2,r,r2,d,d1,d2,\mathrm{dist}=N;$

2) for $(t=0;\ t<s;\ t=t+1);$

3) $\{k=(\mathrm{v}+tN)/s;\ r=(\mathrm{v}+tN)\ \%\ s;$

4) $k2=((t+1)N-\mathrm{v})/s;\ r2=((t+1)N-\mathrm{v})\ \%\ s;$

5) $d1=\varphi_1(k,r,s);\ \ d2=\varphi_1(k2,r2,s);$

6) $d=\min(d1,d2);$ if $(\mathrm{dist}>d)\ \mathrm{dist}=d;\}$

7) return dist

В строке 1 темплейта определены необходимые локальные переменные. В строке 2 имеем оператор for с ограничением на число циклов $t<s$. В строках 3 и 4 переменные k и r определяют число $+s$ и $+1$ ребер соответственно для пути от 0 к v по часовой стрелке и аналогично переменные $k2$ и $r2$ определяют число $-s$ и $+1$ ребер соответственно для пути от 0 к v против часовой стрелки. В строке 5 неопределенная функция $\varphi_1(k,r,s)$ для текущего цикла t должна вычислить длину $d1$ кратчайшего пути от 0 к v по часовой стрелке и аналогично функция $\varphi_1(k2,r2,s)$ должна вычислить длину $d2$ кратчайшего пути от 0 к v против часовой стрелки. В строке 6 вычисляется длина d (dist) кратчайшего пути от 0 к v для текущего цикла t (для всех циклов $t'<t$). В строке 7 получаем результат: $\mathrm{dist}=D(0,\mathrm{v})$.

Терминальными символами для неопределенной функции $\varphi_1(x_1,x_2,x_3)$ являются локальные переменные $\{k,k2,r,r2\}$, глобальная переменная $\{s\}$ и множество случайных натуральных чисел $\{Cr\}$. Набор операций, используемый для синтеза формулы φ_1, следующий: $NS=\{+,-,\min,\lfloor x\rfloor\}$.

Для данного темплейта алгоритм эволюционного синтеза нашел выражение

$$\varphi_1(x_1,x_2,x_3)=\min((x_1+x_2),(x_1+1+x_3-x_2)).$$

Функция $\varphi_1(k,r,s)$ вычисляет длину кратчайшего пути от 0 к v по часовой стрелке как минимум длин $(k+r)$ и $((k+1)+(s-r))$ для $(+s,+1)$- и $(+s,-1)$-путей соответственно, и аналогично функция $\varphi_1(k2,r2,s)$ вычисляет длину кратчайшего пути от 0 к v против часовой стрелки (для текущего цикла t).

Этот результат был получен для заданных 99 пар вход–выход $\{\mathrm{v},D(0,\mathrm{v})\}$, $1\leq \mathrm{v}\leq 99$, графа $C(200;1,s)$, $1\leq s\leq 99$, после 127 итераций за время 270 с при популяции 500 особей (средние значения получены по результатам десяти экспериментов).

Корректность вычисления функции расстояния $D(u,\mathrm{v})$ циркулянтного графа $C(N;1,s)$ с помощью полученного алгоритма на основе заданного темплейта была проверена экспериментально и обоснована теоретически.

Верхняя граница s на число циклов $t < s$ в строке 2 данного темплейта может быть уменьшена.

Лемма 1. Число циклов в алгоритме вычисления функции расстояния (указанное в строке 2 заданного темплейта) не превышает величины $\lfloor (s/2+1)/\lfloor N/s \rfloor \rfloor$.

Как результат имеем следующую лемму.

Лемма 2. Вычисление функции расстояния $D(0, \mathrm{v})$, $0 \le \mathrm{v} < N$, для циркулянтного графа $C(N; 1, s)$ с помощью полученного алгоритма на основе заданного темплейта T и с числом циклов, определенным леммой 1, является корректным.

Алгоритм вычисления функции расстояния может быть использован для решения таких проблем в циркулянтных графах $C(N; 1, s)$ степени 4, как маршрутизация или определение диаметра графа. Для решения первой проблемы, например, достаточно запомнить число шагов и знаки двух образующих в кратчайшем пути, определяемом в строке 6 данного темплейта.

В работах [16–18] приведены алгоритмы поиска кратчайшего пути между любыми парами вершин в циркулянтных графах $C(N; 1, s)$ степени 4. Алгоритм, полученный с помощью алгоритма эволюционного синтеза, отличается от всех известных, и его оценки не хуже.

Заключение. Представленный подход к синтезу алгоритмов, основанный на эволюционных вычислениях, шаблонах алгоритмов и заданного множества пар входных–выходных данных, был успешно применен при автоматизации открытия алгоритмов и синтеза математических формул. Используемые шаблоны могут иметь сложную структуру, включая итерации, рекурсии, циклы, описывать сканирование различных структур данных (матриц, массивов, деревьев, графов, сетей) и содержать неизвестные параметры и формулы. Рассматриваемый подход к эволюционному синтезу на основе темплейтов может быть использован для синтеза новых алгоритмов, функций, математических моделей, которые затем могут быть теоретически обоснованы и экспериментально исследованы.

СПИСОК ЛИТЕРАТУРЫ

1. **Cole M.** Algorithmic Skeletons: Structured Management of Parallel Computation. Cambridge: The MIT Press, 1989.
2. **Mirenkov N., Mirenkova T.** Multimedia skeletons and filmification of methods // Proc. of the First Intern. Conf. on Visual Information Systems. Melbourne, Australia: Victoria University, 1996. P. 58.
3. **Gamma E., Helm R., Johnson R., Vlissides J.** Design Patterns: Elements of Reusable Object-Oriented Software. Reading, MA: Addison-Wesley, 1994.
4. **Goldberg D. E.** Genetic Algorithms in Search, Optimization and Machine Learning. Reading, MA: Addison-Wesley, 1989.
5. **Koza J.** Genetic Programming. Cambridge: The MIT Press,1992.
6. **Монахов О. Г., Монахова Э. А.** Синтез новых семейств оптимальных регулярных сетей на основе эволюционных вычислений и шаблонов функций // Автометрия. 2004. **40**, № 4. С. 106.
7. **Monakhov O., Monakhova E.** An algorithm for discovery of new families of optimal regular networks // Proc. of 6th Inter. Conf. on Discovery Science (DS 2003). Berlin, Heidelberg: Springer-Verlag, 2003. P. 244.

8. **Andre D.** Automatically defined features: The simultaneous evolution of 2-dimensional feature detectors and algorithm for using them // Advances in Genetic Programming /Ed. K. Kinnear. Cambridge: MIT Press/Bradford Books. 1994. P. 477.
9. **Nguyen T., Huang T.** Evolvable 3D modeling for model-based object recognition systems // Advances in Genetic Programming /Ed. K. Kinnear. Cambridge: MIT Press/Bradford Books. 1994. P. 459.
10. **Lee W.-P., Hallam J., Lund H. H.** A hybrid GP/GA approach for co-evolving controllers and robot bodies to achieve fitness – specified tasks // Proc. of IEEE Intern. Conf. on Evolutionary Computation. Piscataway, NJ: IEEE Press. 1996. P. 114.
11. **Olsson J. R.** Population management for automatic design of algorithms through evolution // Proc. of IEEE Intern. Conference on Evolutionary Computation. Piscataway, NJ: IEEE Press. 1998. P. 64.
12. **Бонгард М. М.** Проблема узнавания. М.: Наука, 1967.
13. **Bermond J.-C., Comellas F., Hsu D. F.** Distributed loop computer networks: a survey // Journ. Parallel Distributed Comput. 1995. **24**. P. 2.
14. **Монахов О. Г., Монахова Э. А.** Параллельные системы с распределенной памятью: структуры и организация взаимодействий. Новосибирск: Изд-во СО РАН, 2000.
15. **Hwang F. K.** A survey on multi-loop networks // Theor. Comput. Sci. 2003. **299**. P. 107.
16. **Mukhopadhyaya K., Sinha B. P.** Fault-tolerant routing in distributed loop networks // IEEE Trans. Comput. 1995. **C-44**. P. 1452.
17. **Narayanan L., Opatrny J.** Compact routing on Chordal Rings of Degree Four /Eds. D. Krizanc, P. Widmayer. Toronto: Carleton Scientific, 1997. P. 125.
18. **Robic B., Zerovnik J.** Minimum 2-terminal routing in 2-jump circulant graphs // Comput. and Artificial Intelligence. 2000. **19**. P. 37.

Институт вычислительной математики
и математической геофизики СО РАН,
E-mail: monakhov@rav.sscc.ru

Поступила в редакцию
3 августа 2005 г.