УДК 519.768

ОПТИМИЗАЦИЯ ПРОГРАММ И «ОТЛАДОЧНЫЕ» УРАВНЕНИЯ В МЕТРИЧЕСКИХ ПРОСТРАНСТВАХ

Бесланеев З.О., Кодзоков А.Х., Саншокова М.Л., Жабоев Ж.Ж.

ФГБОУ ВО «Кабардино-Балкарский государственный университет им. Х.М. Бербекова», Нальчик, e-mail: zalimbach@mail.ru

В работе ставятся и исследуются некоторые задачи программирования в метрических пространствах. Рассмотрены операторные уравнения, оптимизация программ по различным критериям. Для оценки априори качества программного обеспечения и среды программирования использованы метрические оценки и характеристики программ по Холстеду как длина, время отладки, структурная сложность и другие. Введены операции сложения программ и композиции программ. Операций сложения и композиции достаточно для описания структуры любой программы. Всякую программу можно представить через три базовые алгоритмические структуры: следование, ветвление, повторение. Данные структуры были описаны через введенные операции. Рассмотрена сходимость программ в пространстве программ. Из сходимости по схеме следует функциональная сходимость, обратное неверно. С точки зрения существования, единственности и построения оптимальной программы, достаточно рассмотреть операторы минимизации. Рассмотрены операторы минимизации по количеству операндов, по количеству типов операторов, по времени создания программы. В пространстве элементов определена метрика. Ошибка в каждом модуле может оказать воздействие на ошибку в другом модуле, на работу самого модуля (уменьшить или усилить влияние). Определена мера влияния; структура орграфом без петель и кратности ребер: вершины – ошибки (места их локализации), ребра – меры влияния. Рассмотрена задача нахождения такой подстановки, которая минимизирует функционал суммарного влияния для системы со структурой. Метрика определяема и как сумма всех весов ребер орграфов модулей, отличающихся при заданной структуре или отклоняющихся от оптимальной структуры; и как мера интеллектуальной работы программы, как разность энтропии до начала работы (статическое состояние) и после окончания работы (динамическое состояние). Исследованы операторные «отладочные» уравнения. Доказана теорема о единственности решения неоднородного уравнения для количества ошибок, обнаруженных в программной системе в момент времени при определенных условиях.

Ключевые слова: пространства программ, метрики Холстеда, операторные уравнения, оптимизация программ

OPTIMIZATION OF PROGRAMS AND THE «DEBUGGING» EQUATIONS IN METRIC SPACES

Beslaneev Z.O., Kodzokov A.Kh., Sanshokova M.L., Zhaboev Zh.Zh.

Kabardino-Balkarian State University named after Kh.M. Berbekov, Nalchik, e-mail: zalimbach@mail.ru

In work some tasks of programming in metric spaces are investigated. The equations in operator forms, optimization of programs for various criteria are considered. For an assessment a priori qualities of the software and a programming environment are used metric estimates and characteristics of programs for Holsted as length, debugging time, structural complexity and others. Addition operations of programs and composition of programs are entered. Addition operations and composition are enough for structure declaration of any program. Any program can be provided through three basic algorithmic structures: following, branching, repetition. These structures were described through the entered operations. Convergence of programs in space of programs is considered. The functional convergence follows from convergence according to the diagram, reverse is incorrect. From the point of view of existence, uniqueness and creation of the optimum program, it is enough to consider operators of minimization. Operators of minimization by quantity of operands, by quantity of types of operators, on time of creation of the program are considered. In space of elements the metrics is defined. The error in each module can make impact on an error in other module, on operation of the module (to reduce or strengthen influence). The influence measure is defined; structure the digraph without loops and a multiplicity of edges: peaks – errors (the place of their localization), edges – influence measures. The task of finding of such substitution which minimizes a functionality of summary influence for system with structure is considered. The metrics is defined and as the amount of all scales of edges of digraphs of the modules differing in case of the given structure or deviating optimum structure; and as a measure of an intellectual program runtime, as an entropy difference prior to operation (a static status) and after completion of work (a dynamic status). The operator «debug» equations are probed. The theorem of uniqueness of the solution of the non-uniform equation for quantity of the errors noticed in program system in timepoint under certain conditions is proved.

Keywords: spaces of programs, Holsted's metrics, operator equations, optimization of programs

Метрическая теория программ Холстеда исходит из статического выражения алгоритма на конкретном языке программирования, поскольку лишь программы в машинном коде представляют собой «динамическое» исключение – они могут быть непосредственно выполнены. Все же остальные выражения, или реализации, алгоритмов должны исследоваться в первую очередь по их текстуальным представлениям. Исходя из простых и естественных комбинаторных соображений, а также учитывая структурные особенности исследуемых языков программирования и текстов написанных на них программ, Холстед получает

пригодные для практического использования соотношения между основными метрическими характеристиками.

Метрические оценки и характеристики программ по Холстеду такие как, например, длина, время отладки, структурная сложность и другие позволяют оценить априори качество программного обеспечения и среды программирования. Динамическая характеристика сложности – важная оценка не только при выбранном подходе эквивалентности по структуре, но и по функции.

Каждая алгебраическая модель программ строится над выбранным конечным базисом операторных символов и логических переменных, замениших собой в моделируемых программах операторы и логические условия. При фиксированном базисе все алгебраические модели имеют общее множество своих объектов, и одна отличается от другой отношением эквивалентности объектов. Проблема эквивалентности в отдельной модели состоит в поиске алгоритма, который, получив на свой вход две схемы программ из этой модели, распознаёт, эквивалентны они в ней или нет. Если такой алгоритм найден, то он именуется разрешающим эквивалентность в этой модели.

В теории алгебраических моделей программ установлено существование моделей с неразрешимой проблемой эквивалентности, т.е. таких, для которых нет разрешающих в них эквивалентность алгоритмов. Это обстоятельство и выдвигает задачу разработки методик распознавания эквивалентности в алгебраических моделях программ.

Алгебраическими моделями программ описываются такие модели, как дискретные преобразователи, а именно: для каждого дискретного преобразователя можно построить равносильную ему схему программы. На этом основании факты по разрешению эквивалентности дискретных преобразователей переносятся в теорию алгебраических моделей программ.

Математически, программой x назовем упорядоченную последовательность команд, обеспечивающих отображение x: $X \to Y$, где X – входной вектор, Y – выходной вектор, $D(X)$ – множество входных, $D(Y)$ – множество выходных векторов. Множество $D = D(X) \bigcup D(Y)$ – множество состояний параметров программы.

Программы x, y эквивалентны функционально [3] (по результату) на X, если $x(D(X)) = y(D(X))$. Пусть $d(x)$ – количество переменных в программе x или введенных в программу переменных (без входных–выходных), $f(x)$ – количество типов операторов и отношений в программе x («if-then-else», «;», «<> = », « = », «for-to-do» и т.д.).

Введем операцию *сложения программ*:

сумма программ x, y с входным вектором D и выходными векторами $D1$, $D2$ – это программа z с входным D и результатом $D1 U D2$: $x(D) + y(D) = z(D)$. Сложение программ – транзитивно, коммутативно, ассоциативно.

Введем *композицию* программ:

композиция $y*x$ двух программ x, y, где $x(D1) = D2, y(D3) = D4$ – программа $z = y*x$ такая, что $z(D1 \cap D3) = y(x(D1) \cap D3)$. $D1$ принадлежит входному множеству x, $D2$ и $D3$ принадлежат подмножеству входного множества y и подмножеству выходного множества x, $D1$ принадлежит выходному множеству y.

Операций сложения и композиции достаточно для описания структуры любой программы. Как известно, по теореме Бёма – Якопини [5], всякую программу можно представить через три базовые алгоритмические структуры: следование, ветвление, повторение. Опишем их через введенные операции.

Следование $X = A \to B$ – это (по определению) композиция $X = A*B$.

Неполному ветвлению («без иначе») с условием a сопоставим программу $A(\{M, a\})$ с результатом – множеством M, если a – истинно и пустым множеством, если a – ложно. Формула полного ветвления: $X = B*A + C*!A$, где $!A(\{M, a\}) = A(\{M, \text{не } a\})$.

Цикл с постусловием, аналогично, имеет формулу вида $X = B + A*X$. Цикл с предусловием – формулу $X = A*(B + X)$.

Рассмотрим сходимость программ в пространстве программ. Программы $x_1, x_2, x_3, \ldots, x_n, \ldots$ сходятся по схеме к программе y, если $\lim\limits_{n\to\infty} p(x_n, y) \to 0$. Программы сходятся функционально к программе y, если $\lim\limits_{n\to\infty} x_n \to y$.

Из сходимости по схеме следует функциональная сходимость, обратное неверно.

Пусть M – пространство программ над алфавитами X и Y, $M(P_i)$, $i = 1,2,\ldots$ множество программ которые реализуют одну задачу P_i. Назовем оператором над программой некую функцию A, такую, что $A(x) = y$.

Операторы A_n сходятся к оператору A функционально (по схеме), если для любой программы z: $A_n(z)$ сходятся к $A(z)$ функционально (по схеме).

С точки зрения существования, единственности и построения оптимальной программы, достаточно рассмотреть операторы минимизации:

1) Ad – минимизация по количеству операндов:

$$Ad(x) = y, \ \forall z \in P : d(y) \le d(z),$$

где $d(y)$ – количество операндов в y;

2) *Af* – минимизация по количеству типов операторов:

$$Af(x) = y,\ \forall z \in P : f(y) \le f(z),$$

где $f(y)$ – количество типов операторов в y;

3) *At* – минимизация по времени создания программы:

$$Ad(x) = y,\ \forall z \in P : t(y) \le t(z),$$

где $t(y)$ – время создания программы y, оцениваемая по Холстеду:

$$t(y) = (Nf(y)\log_2 d(y)\log_2(d(y) + f(y)))/(2S),$$

где N – длина программы,

$$N = f(y)\log_2 f(y) + d(y)\log_2 d(y),$$

S – число Страуда.

«Число Страуда» введено психологом Джоном Страудом в работе «Тонкая структура психологического времени». Дж. Страуд определил «момент» как время, требуемое человеческому мозгу для выполнения наиболее элементарного различения. Он обнаружил, что в течение всего времени бодрствования человек воспринимает эти «моменты» со скоростью «от пяти до двадцати или чуть меньшего числа раз» в секунду. Следует отметить, что, хотя Страуд исследовал лишь скорость мысленной обработки, отличную от скорости ввода-вывода, в диапазон приведенных им цифр попадает число кадров в секунду, превращающее кинофильм из последовательности отдельных снимков в непрерывное изображение. Обозначая через S число страудовских «моментов» в секунду, мы можем записать: $5 <= S <= 20$ в секунду.

Естественно, что любой человек, занимающийся реализацией алгоритма, способен в зависимости от степени своей сосредоточенности отвлечь какую-то часть мысленных различений на посторонние предметы. Пользуясь терминологией вычислительной техники, можно сказать, что, если он находится «в режиме разделения времени», S представляет собой лишь верхнюю границу. С другой стороны, если программист выполняет эквивалент машинной операции «запретить все прерывания» и сосредоточивает внимание на программировании, то применимо действительное значение S.

В пространстве X элементов $x_1, x_2, x_n, \ldots$ определим метрику

$$\rho(x, y) = \sum_{i,j=1}^{n} r_{ij},$$

где n – количество всех модулей в x, y, r_{ij} – сумма весов (длин) дуг орграфа, вершинами которого являются модули x, y, а ребрами – действия над ними.

Ошибка в каждом модуле может оказать воздействие на ошибку в другом модуле, на работу самого модуля (уменьшить или усилить влияние). Меру влияния определим как r_{ij}, $R = \{r_{ij} : i = 1,\ldots, n-1;\ j = 2,\ldots,n\}$. Определим в структуру орграфом без петель и кратности ребер: вершины – ошибки (места их локализации), ребра – меры влияния.

Задача – найти такую подстановку, которая минимизирует функционал суммарного влияния для системы со структурой S:

$$F = \sum_{i=1}^{n-1}\sum_{j=2}^{n} r_{ij} = \min.$$

В качестве r_{ij} можно взять меру, учитывающую время начала воздействия точек локализации, предшествующих данной x_j и интенсивность их влияния. Если можно определить безусловные вероятности наступления событий S_i, то такие оценки можно взять байесовыми. Если для событий S_i невозможно указать безусловные вероятности их наступления, то применим следующую процедуру.

1. Для каждого события S_i указать события $\{s_{ij}\}$, влияющие на S_i. В результате получаем дерево событий. Его нижний уровень – события S_{io}, для которых можно экспертно указать безусловные вероятности их наступления.

2. Каждый эксперт указывает безусловную вероятность наступления событий S_{i0}, $i = 1, 2,\ldots, n_0$, а также веса p_{i0} событий. Методом Дельфи (можно средневзвешенным осреднением, учитывая веса p_{i0}) событию данного уровня ставим в соответствие $P_{i0}(t)$ – вероятность наступления S_{i0} в момент времени t. Практически функция $P_{i0}(t)$ задается таблицами$P_{i0}(\tau_1)$, $P_{i0}(\tau_2)$ и т.д. Так как на нижнем уровне S_0 состоит в одновременном осуществлении независимых событий S_{i0}, $i = 1, 2,\ldots, n_0$, то по теореме умножения вероятностей вычисляем вероятность события S_0 в момент времени t:

$$P_0(t) = \prod_{i=1}^{n_0} P_{i0}(t).$$

3. Спустя время τ после осуществления S_0, методом Дельфи, например, дают оценку условной вероятности события S_1. Усредняя (с учетом весов), получаем функцию $f(\tau)$. Далее найдем безусловную вероятность наступления S_1 к моменту t:

$$P_1(t) = \int_0^t f(t - z)\, dP_0(z).$$

4. Последовательность 1–3 (перехода от верхнего уровня иерархии к нижнему) повторяем для всех уровней, в результате находим функцию $P(t)$ – наступления события S.

Активности модулей взаимодействуют (прямо или косвенно), например, с помощью соотношений [1]:

$$\begin{cases} \dfrac{ds(t)}{dt} = \sum\limits_{i=1}^{n} \phi_i(s, s_i), \\ \dfrac{ds_i(t)}{dt} = \psi_i(s_i, s) + Q_i(t). \end{cases}$$

Здесь $s(t)$ – структурная (логическая) активность программ, $Q_i(t)$ – функционал меры чувствительности отклонений x_i от x_{iopt}. Например, $Q_i(t) = k||x_i - x_{iopt}||$, $k > 0$.

На функции $\varphi_i(t) = \varphi_i(s(t), s_i(t))$, $\psi_i(t) = \psi_i(s(t), s_i(t))$ накладываются определенные ограничения, в частности периодичность, затухание, наличие равновесного состояния и др.

Метрика $\rho(x, y)$ определяема и как сумма всех весов ребер орграфов модулей, отличающихся при заданной структуре или отклоняющихся от оптимальной структуры:

$$\rho(x, y) = \sum_{i,j=1, i \neq j}^{n} r_{ij}(x, y).$$

Можно метрику задавать и как меру интеллектуальной работы программы, как разность энтропии до начала работы (статическое состояние) и после окончания работы (динамическое состояние).

Макс Планк часто подчеркивал различие между двумя типами изменений, встречающихся в природе. Природа, писал Планк, по-видимому, отдает «предпочтение» определенным состояниям. Необратимое увеличение энтропии описывает приближение системы к состоянию, неодолимо «притягивающему» ее, предпочитаемому ей перед другими, – состоянию, из которого система не выйдет по «доброй воле».

«Согласно этому способу выражения, в природе невозможны те процессы, при которых природа дает меньшее предпочтение конечному состоянию, чем начальному. Предельный случай представляет обратимые процессы; в них природа испытывает одинаковое предпочтение как к начальному, так и к конечному состоянию, и поэтому переход из одного состояния в другое может происходить в обоих направлениях».

Пусть ds/dt – изменение энтропии отлаживаемого программного комплекса, ds_1/dt – изменение энтропии за счет структурных изменений, потоков комплекса (открытой системы), ds_2/dt – изменение энтропии за счет отладочных усилий. Справедливо уравнение Пригожина [4]: $ds/dt = ds_1/dt + ds_2/dt$.

При исследовании программ важно рассматривать пространства векторов $x = (x_1, x_2, \ldots, x_n)$, где x_i – характеристика ошибок в программе или структурная связность процедур, u_i – количество ошибок в i-ом модуле комплекса $P(u) = P(u_1, u_2, \ldots, u_n)$.

Можно исследовать операторные «отладочные» уравнения. Пусть $u(x, t)$ – количество ошибок, обнаруженных в программной системе в момент времени t, а x – мера (уровень) ошибок. Рассмотрим уравнение (см. [2]): $Lu + Tu = f$,

T – оператор, определяющий уровень начальных ошибок в программе, L – некоторый линейный ограниченный оператор отладки, $L: U \to V$, U, V – линейные нормированные пространства $D(L) \subseteq U$, $R(L) \subseteq V$.

Теорема. Если $R(L) = V, \forall u \in D(L)$, $\exists c = const$: $Lu \geq cu$, $T < c$, то это уравнение имеет единственное решение $u \in U$.

Доказательство. Условия утверждения гарантируют существование обратного оператора L^{-1} и его непрерывность, причем $L^{-1} < 1/c$. Тогда $u = L^{-1}(f - Tu)$. Для однородного уравнения

$$(1 - L^{-1}T)u < 0.$$

Отсюда следует, что $u = 0$. Неоднородное уравнение имеет единственное решение.

Пример. Пусть $u_{\max}$ – максимальный уровень синтаксических ошибок в программе P, $u(t)$ – оставшееся их количество к моменту времени t. Исходя из модели $du/dt + cu_{max} = 0$, $u(0) = u_0$ можно заключить, что уровень ошибок убывает при $ca \neq -1$ $(0 < a < T)$ по закону: $u(t) = u_0(1 + c(a - t))/(1 + ca)$.

Если дополнительно задать $u(b) = B$, то закон изменения ошибок находится по дополнительному значению $a = -u_0 b / (cB - cu_0) - 1/c$.

Выводы

Поставлены и исследованы некоторые задачи программирования в метрических пространствах.

Рассмотрены операторные уравнения, оптимизация программ по различным критериям.

Использованы метрические оценки и характеристики программ по Холстеду для оценки априори качества программного обеспечения и среды программирования.

Введены операции сложения программ и композиции программ.

Рассмотрена сходимость программ в пространстве программ.

Рассмотрены операторы минимизации по количеству операндов, по количеству типов операторов, по времени создания программы.

В пространстве элементов определена метрика.

Определена мера влияния.

Рассмотрена задача нахождения такой подстановки, которая минимизирует функционал суммарного влияния для системы со структурой.

Исследованы операторные «отладочные» уравнения.

Доказана теорема о единственности решения неоднородного уравнения для количества ошибок, обнаруженных в программной системе в момент времени при определенных условиях.

Список литературы

1. Казиев В.М. Введение в анализ, синтез и моделирование систем. – М.: Бином. Лаборатория знаний – Интуит, 2007. – 244 с.

2. Казиев В.М. Исследование некоторых задач в алгебрах и пространствах программ // Вестник КБГУ, сер. Физ.-мат. науки. – 2002. – С. 45–48.

3. Подловченко Р.И. Об одной методике распознавания эквивалентности в алгебраических моделях программ // Программирование. – 2011. – № 6. – С. 33–43.

4. Пригожин И., Стингер И. Порядок из хаоса: Новый диалог человека с природой. – М.: Едиториал УРСС, 2014. – 304 с.

5. Bohm, Corrado, Giuseppe Jacopini. Flow Diagrams, Turing Machines and Only Two Formation Rules. Communications of the ACM 9 (5): P. 366–371.