Изв. НАН Армении (сер. ТН), т. XLVII, № 3, 1994, с. 95—97.

ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА

УДК 681.3

В. Х. ШАХИН, В. В. БАГДАСАРЯН

К ПРОБЛЕМЕ ОПТИМАЛЬНОГО СИНТЕЗА ТЕСТОВ ДЛЯ ТЕСТИРОВАНИЯ ПРОГРАММНЫХ МОДУЛЕЙ

Рассматривается подход к решению проблемы оптимального синтеза тестов для тестирования программных модулей.
Библиогр.: 1 назв.

Աշխատանքում դիտարկվում է ծրագրային մոդուլների ստուգման համար նախատեսված տեստերի օպտիմալ համադրման խնդիրը։ Բերվում է տեստերի օպտիմալ համադրման ալգորիթմի նկարագրությունը։

Известно [1], что проблема оптимального проектирования комплексов программ тесно связана с решением задачи синтеза тестов для тестирования программных модулей. Это обусловливает необходимость получения оценок структурной сложности программ с учетом полного числа связей.

В данной работе рассматривается подход к решению проблемы оптимального синтеза тестов для тестирования программных модулей.

1. **Определения и обозначения.** Обозначим через $M=\{1, 2, ..., i, ..., j, ..., m\}$ модули комплекса программ. Пусть $i, j \in M$. Положим

$$\vec{a}_i = \begin{cases} 1, \text{ если i-й модуль имеет управляющие связи, которые вызывают его для исполнения j—м модулем.} \\ 0, \text{ в противном случае.} \end{cases}$$

$$\vec{a}_{ij} = \begin{cases} 1, \text{ если i-й модуль имеет управляющие связи, посредством которых он вызывает модуль j.} \\ 0, \text{ в противном случае.} \end{cases}$$

Тогда количество связей, посредством которых i-й модуль:

а) вызывается другими модулями, равно $\sum_{j=1}^{\vec{}} \vec{a}_{ij}$;

б) вызывает другие модули для исполнения, равно $\sum_{j=1}^{\vec{}} \vec{a}_{ij}$.

Рассмотрим оценки информационных связей между модулями [1].

Обозначим через $\vec{B}_{ik}$ количество имен переменных на входе, необходимых для нормального функционирования i-го модуля. Тогда общее количество имен будет равно $\vec{\Phi}_i = \sum_k \vec{B}_{ik}$.

Аналогично число результирующих переменных, подготавливаемых i-ым модулем, будет характеризовать степень информационной зависимости от данного модуля всех остальных в комплексе:

$$\vec{\Phi}_i = \sum_k \vec{B}_{ik}.$$

2. **Классификация программных модулей.** В соответствии с [1] можно произвести разбиение множества программных модулей M на следующие подмножества.

Отнесем к следующим типам множества программных модулей, соответственно удовлетворяющих условиям:

а) первому —

$$P_1=\left\{i \;\middle|\; \begin{array}{l} \beta \leqslant \sum_{j=1}^{n} \overset{*}{a}_{ij} \leqslant \alpha, \\ \sum_{j=1}^{n} \overset{*}{a}_{ij} \geqslant \gamma, \end{array} \quad \text{где } \beta=10,\ \alpha=100,\ \gamma=10 \right\};$$

б) второму —

$$P_2=\left\{i \;\middle|\; \sum_{j=1}^{n} \overset{*}{a}_{ij} \approx 3{,}1, \quad \sum_{j=1}^{n} \overset{*}{a}_{ij} \approx 3{,}8 \right\};$$

в) третьему —

$$P_3=\left\{ i \;\middle|\; \sum_{j=1}^{n} \overset{*}{a}_{ij} \approx \left\{ \begin{array}{ll} 0.8, & 75\% \\ 2, & 20\% \\ 3, & 5\% \end{array} \right\} \right\}.$$

При этом третий тип, в свою очередь, разобьем на два подмножества

$$P_{31}=\left\{i \in P_3 \;\middle|\; 20 \leqslant \sum_{j=1}^{n} \overset{*}{a}_{ij} \leqslant 30 \right\}, \quad P_{32}=\left\{ i \in P_3 \;\middle|\; 2 \leqslant \sum_{j=1}^{n} \overset{*}{a}_{ij} \leqslant 3 \right\}.$$

Особенностью программных модулей трех типов является:

а) первый тип — они практически не обрабатывают информацию и не используют глобальные переменные, однако имеют значительные связи по управлению; б) второй тип — они используют глобальные переменные: 10 ... 20 наименований; в) третий тип — они сравнительно редко вызывают другие программы. Обычно эти программы возвращают управление вызывавшим программам. Однако они могут также вызвать другие стандартные программы.

3. **Алгоритм синтеза программ из системы программных модулей.** Последовательность программных модулей $\{M_i\}_{i=1, \ldots, k}$ назовем функциональной программой, если для любых M_i, M_{i+1} имеет место следующее: M_{i+1} вызывается M_i и либо использует глобальные переменные на своем входе, либо использует обменные переменные модуля M_i.

Определим матрицу связности $\|C_{ij}\|_{\substack{i=1,\ n \\ j=1,\ n}}$ программных модулей следующим образом:

$$C_{ij}=\begin{cases} (1, -1), & \text{если } \overset{*}{a}_{ij}=\overset{*}{a}_{ji}=1, \\ (1, \ \ 0), & \text{если } \overset{*}{a}_{ij}=1, \ \overset{*}{a}_{ji}=0, \\ (0, -1), & \text{если } \overset{*}{a}_{ij}=0, \ \overset{*}{a}_{ji}=1, \\ (0, \ \ 0), & \text{если } \overset{*}{a}_{ij}=\overset{*}{a}_{ji}=0. \end{cases}$$

Пусть программные модули удовлетворяют следующим условиям. Для любой пары программных модулей (i, j) имеет место одно из следующих условий:

а) $i \in P_1$ следует $j \in P_1 \cup P_2 \cup P_{21} \cup P_{22}$.

в) $i \in P_2$ следует $j \in P_3 \cup P_2$,

б) $i \in P_3$ следует $j \in P_1 \cup P_3$.

Тогда алгоритм синтеза программ из программных модулей можно будет определить следующим образом.

Определим матрицу $\overline{C}$, равную C, как рабочую.

Алгоритм 1.

Шаг 1. Выбираем $i_1 \in P_1$.

Шаг 2. Определим: а) $j_1 = \min j$ $C_{i_1 j} = (1, 0)$ или б) $j_1 = \min j$ $C_{i_1 j} = (1, -1)$.

Шаг 3. Положим $\overline{c}_{i_1 j_1} = 0$. В случае «а» переходим к шагу 4. В случае «б» переходим к шагу 6.

Шаг 4. Определим программу $i_1 j_1$. В строке j_1 определим: а) $j_2 = \min j$ $C_{j_1 j_2} = (1, 0)$ или б) $j_2 = \min j$ $C_{j_1 j_2} = (1, -1)$.

Шаг 5. Положим $C_{j_1 j_2} = 0$. В случае «а» повторяем шаг 4. В случае «б» переходим к шагу 6.

Шаг 6. Определим программу $i_2 j_2$. Переходим к шагу 2, если $j_1 < |P_1| + |P_2| + |P_3|$, в противном случае — к шагу 1 и рассмотрим $i_2 = P_1 \setminus \{i_1\}$.

Продолжаем этот процесс до тех пор, пока $P_1 \setminus \{i_1\}, \ldots, \{i_k\}\} = \Phi$. Переходим к алгоритму 2.

Алгоритм 2.

Шаг 1. Выбираем $i_1 \in P_2$.

Шаг 2. Определим: а) $j_1 = \min j$ $C_{i_1 j} = (1, 0)$ либо б) $j_1 = \min j$ $C_{i_1 j} = (0, -1)$. В случае "а": $j_1 \in P_2$, положим $C_{i_1 j_1} = 0$ и переходим к шагу 3. В случае "б": $j_1 \in P_1$ переходим к алгоритму 1.

Шаг 3. Определим: а) $j_2 = \min j$ $C_{j_1 j} = (1, 0)$ либо б) $j_2 = \min j$ $C_{j_1 j} = (0, -1)$.

Если $j_2 < |P_1| + |P_2| + |P_3|$, то приходим к шагу 2. В противном случае выбираем $i_2 \in P_2 \setminus \{i_1\}$.

Если $P_2 \setminus \{i_1\} \neq \Phi$, то переходим к шагу 1, заменив i_1 на i_2. Продолжаем процесс до тех пор, пока $P_2 \setminus \{\{i_1\}, \ldots, \{i_k\}\} = \Phi$.

Свойство алгоритма. Алгоритм не допускает циклов, так как на каждом шаге соответствующий элемент матрицы приравнивается к нулю.

ЛИТЕРАТУРА

1. Липаев В. В. Качество программного обеспечения.— М.: Финансы и статистика, 1983.—263 с.

ГИУА

25 IX 1993