АКАДЕМИЯ НАУК СССР
СИБИРСКОЕ ОТДЕЛЕНИЕ
АВТОМЕТРИЯ

№ 1 1972

УДК 681.2.08 + 621.3.019.3

Л. КУБАТ, Л. С. ТИМОНЕН, М. УЛЛРИХ
(Прага, Новосибирск)

СОСТАВЛЕНИЕ ОПТИМАЛЬНЫХ ПРОГРАММ ДИАГНОСТИКИ СОСТОЯНИЯ ТЕХНИЧЕСКИХ СИСТЕМ ПРИ ОГРАНИЧЕНИИ НА ВРЕМЯ ПРОВЕДЕНИЯ ПРОВЕРОК

Рассматривается задача диагностики состояния технических систем с целью обнаружения и локализации ее неисправностей при ограничении на время проведения проверок системы. Предлагается метод составления оптимальных программ диагностики, удовлетворяющих заданному ограничению. Метод основан на использовании рекуррентных соотношений динамического программирования.

Эффективные способы диагностики состояния сложных технических систем, осуществляемой с целью обнаружения и локализации неисправностей, основаны на использовании заранее составленных программ (стратегий) диагностики. Такая программа определяет как совокупность проверок (испытаний системы), проводимых в процессе диагностики, так и последовательность выполнения этих проверок. Методы составления оптимальных программ диагностики, обеспечивающих получение всей возможной информации о состоянии системы при минимальных средних или суммарных затратах времени на проведение проверок, описаны в [1, 2]. Однако в ряде случаев (и в частности, когда диагностика состояния системы производится непосредственно перед началом ее функционирования) на проведение проверок системы отводится ограниченное время, из-за чего не все неисправности могут быть обнаружены и локализованы с заданной степенью подробности. Поэтому возникает задача составления такой программы диагностики, которая при выполнении заданного ограничения на суммарное время проведения проверок сводит к минимуму потери, возникающие из-за неполной информации о состоянии системы. Ниже предлагается метод решения этой задачи, основанный на использовании принципов динамического программирования.

Рассмотрим систему, для которой определено конечное множество $S=\{s\}$ ее различных состояний и априори известно, что с вероятностью $P_s > 0$ она может находиться в состоянии S. Способы задания отдельных состояний могут быть различными. В частности, каждое состояние может быть задано некоторым перечнем отказавших блоков системы. В множество S может входить и такое состояние системы, когда все блоки исправны, т. е. система в целом работоспособна.

Известно множество $\Pi=\{1, 2, \ldots, n\}$ возможных проверок системы, каждая из которых имеет конечное число исходов. Для проведения

проверки $i=(1\leqslant i\leqslant n)$ требуется время τ_i, которое не зависит от выбранной последовательности проведения проверок.

Каждая проверка представляет собой некоторый эксперимент, исход которого заранее не известен и зависит от действительного состояния системы. Проверки системы можно проводить в любой последовательности независимо от получаемых исходов. В процессе выполнения проверок система не может перейти из одного состояния в другое. Повторное выполнение любой проверки нецелесообразно, так как если в результате проведения проверки i получен исход x_i, то при ее повторении будет получен этот же самый исход.

Задано конечное множество E_0, элементами которого являются все возможные наборы $(x_1 x_2 \ldots x_n)$ исходов, которые могут быть получены в результате проведения всех проверок из П. Множество E_0 не содержит одинаковых наборов и для любой проверки i в E_0 имеется по крайней мере два набора, в которых эта проверка имеет различные исходы. Предполагается, что в результате проведения всех проверок рассматриваемой системы нельзя получить какой-либо набор исходов, не входящий в E_0.

Для каждого $s\in S$ и каждого $(x_1 x_2 \ldots x_n)\in E_0$ известна вероятность $p(x_1 x_2 \ldots x_n|s)$ получения этого набора в результате проведения всех проверок из П при условии, что система находится в состоянии s. Возможно, что некоторые из этих вероятностей равны нулю, хотя для всех S

$$\sum_{(x_1 x_2 \ldots x_n)\in E_0} p(x_1 x_2 \ldots x_n|s)=1$$

и для любого $(x_1 x_2 \ldots x_n)\in E_0$ существует по крайней мере одно состояние s, для которого $p(x_1 x_2 \ldots x_n|s)>0$.

Располагая множествами П и E_0, найдем совокупность $E=\{e\}$, состоящую из всех возможных для данной системы различных наборов исходов, которые могут быть получены в результате проведения любого количества проверок из П. Для этого образуем множество $W=\{\omega\}$, элементами которого являются все непустые подмножества проверок из П и само П. Возьмем некоторые $\omega\in W$ и из каждого набора $(x_1 x_2 \ldots x_n)\in E_0$ удалим исходы тех проверок, которые не вошли в ω. Полученное множество наборов обозначим через $B(\omega)$. Из наборов, принадлежащих $B(\omega)$, образуем множество $F_0(\omega)$, которое отличается от $B(\omega)$ только тем, что в нем нет одинаковых наборов. Очевидно, что $E_0(\Pi)=E_0$. Искомая совокупность представляет собой объединение множеств $E_0(\omega)$, образованных для всех $\omega\in W$, т. е.

$$E=\bigcup_{\omega\in W} E_0(\omega).$$

В дальнейшем будем считать, что пустой набор исходов, обозначаемый через $\varnothing$, также входит в E.

Следует отметить, что каждый непустой набор $e=(x_{i_1} x_{i_2} \ldots x_{in})$ из E можно рассматривать как некоторое неупорядоченное множество, элементами которого являются исходы проверок, причем все элементы считаются различными. Такая возможность является следствием того, что под x_i подразумевается обозначение как некоторого исхода x, так и проверки i, имеющей этот исход. Отметим также, что x_i и x_j могут обозначать как различные, так и одинаковые исходы проверок i и j. Например, если каждая из проверок i и j может иметь исходы 0 и 1, то возможны следующие обозначения: $0_i 0_j$; $0_i 1_j$; $1_i 0_j$; $1_i 1_j$.

В рассматриваемой постановке задачи информация о состоянии системы, получаемая в результате проведения некоторой последовательности проверок, представляется в виде апостериорного распределения

вероятностей, заданного на множестве S. Для каждого непустого набора $e\,(x_{i_1}x_{i_2}\ldots x_{i_n})$ исходов, полученного в результате проведения проверок $i_1, i_2, \ldots, i_n$, это распределение находится следующим образом.

Будем говорить, что набор $(x_1\ x_2\ \ldots\ x_n)\in E_0$ содержит в себе не пустой набор $(x_{i_1}x_{i_2}\ldots x_{i_k})$, если в этих наборах проверки $i_1, i_2, \ldots, i_k$ имеют одинаковые исходы, т. е. $(x_{i_1}x_{i_2}\ldots, x_{i_k})\subseteq(x_1x_2\ldots x_n)$.

В множестве E_0 найдем все наборы, которые содержат в себе набор e, и полученное множество наборов обозначим через $A(e)$. Заметим, что из способа образования совокупности E и из того, что $e\in E$, следует, что $A(e)$ содержат по крайней мере один набор. Поскольку вероятность получения набора e при условии, что система находится в состоянии s, равна

$$p(e|s)=\sum_{(x_1x_2\ldots x_n)\in A(e)} p(x_1x_2\ldots x_n|s),$$

то апостериорная вероятность пребывания системы в состоянии $s\in S$ в случае получения набора e находится из следующего соотношения:

$$p(s|e)=\frac{p_s\,p(e/s)}{\sum\limits_{s\in S} p_s\,p(e|s)}.$$

Возможно, что некоторые вероятности $p(s|e)$ окажутся равными нулю. Это означает, что действительное состояние системы принадлежит подмножеству тех состояний из S, для которых соответствующие апостериорные вероятности больше нуля. Естественно, что для пустого набора $e=\varnothing$ имеем $p(s|e)=p(s)$ и поэтому можно считать, что для него $A(\varnothing)=E_0$.

Предполагается, что на проведение проверок из П отведено ограниченное время T такое, что $\tau_i<T$ хотя бы для одного $i=1, 2, \ldots, n$, однако $\sum\limits_{i=1}^{n}\tau_i>T$. Так как время, затрачиваемое на получение набора $e=(x_{i_1}x_{i_2}\ldots x_{i_k})$, равно $\tau(e)=\sum\limits_{i=1}^{k}\tau_{i_j}$, то это ограничение равносильно невозможности получения тех наборов из E, для которых $\tau(e)>T$.

Исходя из множеств П, E_0 и совокупности E, можно составить целый ряд различных программ диагностики, удовлетворяющих заданному ограничению на время выполнения проверок. Составление какой-либо конкретной программы производится в два этапа.

На первом этапе находится совокупность E_T тех наборов из E, для которых $\tau(e)\leqslant T$. Для каждого набора $e\in E_T$ определяется подмножество проверок $\Pi_T(e)$, которые можно выполнять после получения набора e без нарушения заданного ограничения на время выполнения проверок. Очевидно, что в $\Pi_T(e)$ целесообразно включать только те проверки из П, проведение которых после получения набора e может дать не менее двух исходов.

Определение подмножества $\Pi_T(e)$ для каждого непустого набора $e\in E_T$ (если $e=\varnothing$, то $\Pi_T(e)=\Pi$) осуществляется следующим образом. Находится множество $A(e)$ тех наборов из E_0, которые содержат в себе e и для каждой проверки $i\in\Pi$ находится множество $X_i(e)$ всех ее различных исходов, встречающихся в наборах из $A(e)$. Если теперь удалить из П те проверки i, для которых соответствующее $X_i(e)$ содержит только один исход, и те, для которых $\tau(e)+\tau_i>T$, то оставшиеся проверки будут составлять искомое подмножество $\Pi_T(e)$. При этом может оказаться, что $\Pi_T(e)$ не содержит ни одной проверки ($\Pi_T(e)=\varnothing$).

Это означает, что после получения набора e следует прекратить процесс проведения проверок.

Далее для каждого $e \in E_T$ выбирается проверка $i \in \Pi_T(e)$, которая должна проводиться после получения набора e, и формируется пара $[e; i]$. В том случае, когда подмножество $\Pi_T(e)$ не содержит ни одной проверки, формируется пара $[e; i^*]$, в которой i^* обозначает прекращение процесса проведения проверок.

На втором этапе среди пар $[e; i]$, сформированных для всех наборов из E_T, находятся те, которые образуют составляемую программу. Для этого берется пара $[(\varnothing); i_1]$, сформированная для пустого набора $e=\varnothing$, которая всегда входит в составляемую программу. С проведения проверки, указанной в этой паре, начинается реализация программы. Затем для проверки i_1 путем просмотра всех наборов из E_0 находится множество $x_{i_1}(\varnothing)$ ее различных исходов. Так как каждый исход x_{i_1} этой проверки является набором из E_T, то находятся все пары вида $[(x_{i_1}); i_2]$, которые тоже входят в программу. Аналогично определяются все остальные пары, входящие в программу, т. е. если пара $[(x_{i_1} x_{i_2} \ldots x_{i_{k-1}}); i_k]$ входит в программу и $i_k \neq i^*$, то каждая пара вида $[(x_{i_1} x_{i_2} \ldots x_{i_{k-1}} x_{i_k}); i_{k+1}]$, где $x_k \in X_{i_k}(x_{i_1} x_{i_2} \ldots x_{i_{k-1}})$, тоже входит в программу. Следует отметить, что составленная таким образом программа диагностики задается некоторой совокупностью пар

$$H = \{[(\varnothing); i_1], [(x_{i_1}); i_2], \ldots, [(x_{i_1} x_{i_2} \ldots x_{i_{k-1}}); i_k], \ldots\}.$$

Другим способом задания программы диагностики является изображение ее в виде графа-дерева. При этом проверке i_1 из пары $[(\varnothing); i_1]$ ставится в соответствие корневая вершина, а каждому исходу $x_{i_1} \in X_{i_1}(\varnothing)$ — дуга, исходящая из этой вершины. Затем проверке i_2 из пары $[(x_{i_1}); i_2]$ ставится в соответствие вершина, в которую заходит дуга x_{i_1}, а каждому исходу из $x_{i_2}(x_{i_1})$ ставится в соответствие дуга, исходящая из этой вершины. Если $i_2 = i^*$, то такой проверке ставится в соответствие висячая вершина графа, которая не имеет исходящих из нее дуг. Аналогичные соответствия устанавливаются для всех остальных пар, входящих в программу. Каждой висячей вершине графа приписывается обозначение информации о состоянии системы, получаемой в случае реализации тех исходов проверок, которые поставлены в соответствие дугам, встречающимся на пути от корневой вершины графа к данной висячей вершине.

Пусть на наборах из E задана неотрицательная функция $f(e)$, которая характеризует потери, возникающие от неполной информации о действительном состоянии системы. Значение этой функции на наборе e зависит от апостериорного распределения $p(s|e)$ и равно нулю только в том случае, если для некоторого $s \in S$ $p(s|e) = 1$. Предполагается, что $f(e)$ обладает тем свойством, что для каждого набора $e \in E_T$, для которого $\Pi_T(e) \neq \varnothing$, и любого исхода $x_i \in X_i(e)$ проверки $i \in \Pi_T(e)$ выполняется соотношение* $f(e) \geqslant f(ex_i)$. Например, такую функцию можно задать следующим образом. Пусть множество S содержит состояния $s_1, s_2, \ldots, s_n$. Обозначим через $Z = \{z_r\}$, $r = 1, 2, \ldots, m$ множество возможных решений, принимаемых по результатам выполнения проверок, где z_r означает решение «система находится в состоянии s_r». Задана квадратная матрица $V = \|v_{jr}\|$, $j, r = 1, 2, \ldots, m$, в которой $v_{jr} = 0$, если $j = r$, и $v_{jr} > 0$, если $j \neq r$. В этой матрице элемент v_{jr} представляет

* В дальнейшем набор, отличающийся от набора e наличием только одного исхода x_i проверки i, исходы которой не содержатся в e, будем обозначать через ex_i.

$$1 \leq r \leq m \left(\sum_{j=1} \right)$$

Другим примером функции потерь может служить остаточная энтропия

$$f(e) = \sum_{s \in S} p(s|e) \log_2 p(s|e),$$

характеризующая неопределенность состояния системы после получения набора e.

Рассмотрим некоторую программу диагностики, которая представлена совокупностью пар

$$H = \{[(\varnothing); i_1], [(x_{i_1}); i_2], \ldots, [(x_{i_1} x_{i_2} \ldots x_{i_{k-1}}); i_k], \ldots\}.$$

Обозначим через $E_H (E_H \subset E_T)$ совокупность всех наборов, содержащихся в тех парах $[e; i] \in H$, в которых $i = i^*$. Тогда средние потери, возникающие при использовании программы H, находятся из следующего соотношения

$$f_H = \sum_{e \in H} p(e) f(e),$$

где $p(e) = \sum_{s \in S} p(e|s)$ — вероятность получения набора исходов e. Максимальные потери, возникающие при использовании программы H, равны

$$f_H^0 = \max_{e \in E_H} \{f(e)\}.$$

Перейдем теперь к составлению оптимальной программы, для которой при заданном ограничении на время проведения проверок достигается минимум либо средних, либо максимальных потерь. Нетрудно убедиться в том, что составление оптимальной программы, для которой достигается минимум средних потерь, можно осуществить при помощи следующей функции $F(e)$, вычисляемой для всех наборов $e \in E_T$ и представляющей собой рекуррентное соотношение динамического программирования: если подмножество $\Pi_T(e)$ пусто, то $F(e) = f(e)$, а если $\Pi_T(e)$ содержит хотя бы одну проверку, то

$$F(e) = \min_{i \in \Pi_T(e)} \left\{ \sum_{x_i \in X_i(e)} \frac{p(ex_i)}{p(e)} F(ex_i) \right\},$$

где $p(ex_i)$ — вероятность получения набора ex_i. Заметим, что отношение $p(ex_i)/p(e)$ представляет собой вероятность получения набора ex_i при условии, что набор e уже получен.

Составление оптимальной программы, для которой достигается минимум максимальных потерь, можно осуществить при помощи следующей функции $F^0(e)$: если подмножество $\Pi_T(e)$ пусто, то $F^0(e) = f(e)$; в противном случае

$$F^0(e) = \min_{i \in \Pi_T(e)} \left\{ \max_{x_i \in X_i(e)} F^0(ex_i) \right\}.$$

Функция $F(e)$ или $F^0(e)$ используется на первом этапе составления программы при определении проверок i для всех пар $[e; i]$. Чтобы составить оптимальную программу, необходимо в качестве проверки i, входящей в пару $[e; i]$, выбрать ту проверку из $\Pi_T(e)$, для которой до-

стигается минимальное значение соответствующей функции. Кроме того, для всех наборов e, для которых $\Pi_T(e)$ пусто, следует сформировать пары $[e; i^*]$. Легко проверить, что значение функции $F(e)$ при $e=\varnothing$ представляет собой средние потери для оптимальной программы, а значение $F^0(e)$ при $e=\varnothing$ представляет собой максимальные потери для этой программы.

При составлении оптимальной программы может оказаться, что для некоторых наборов e выполняется соотношение $F(e)=f(e)$. Это означает, что после получения набора e дальнейшее выполнение проверок не приводит к уменьшению значения потерь. Поэтому с целью уменьшения времени реализации оптимальной программы необходимо на первом этапе ее составления проверять выполнение соотношения $F(e)=f(e)$ для каждого непустого набора e, и если оно выполняется, то формировать пару вида $[e; i^*]$.

Иногда в множестве $\Pi_T(e)$ может быть несколько проверок, для которых соответствующая функция $E_0(e)$ имеет одно и то же минимальное значение. Объединим такие проверки в множество $\Pi_F(e)$ ($\Pi_F(e) \subseteq \Pi_T(e)$). В этом случае для выбора проверки при формировании набора $[e; i]$ можно использовать следующую дополнительную функцию $t(e)$, позволяющую минимизировать средние затраты времени на реализацию оптимальной (в смысле потерь) программы: если подмножество $\Pi_T(e)$ пусто, то для набора $e = (x_{i_1} x_{i_2} \ldots x_{i_k})$

$$t(e) = \sum_{j=1}^{k} \tau_{i_j};$$

если множество $\Pi_F(e)$ пусто и минимальное значение соответствующей функции $F(e)$ или $F^0(e)$ достигается для проверки $i \in X_i(e)$, то

$$t(e) = \sum_{x_i \in X_i(e)} \frac{p(ex_i)}{p(e)} t(ex_i),$$

а если множество $\Pi_F(e)$ содержит две или более проверок, то

$$t(e) = \min_{i \in \Pi_F(e)} \left\{ \sum_{x_i \in X_i(e)} \frac{p(ex_i)}{p(e)} t(ex_i) \right\}.$$

Для образования пары $[e; i]$ в случае, когда множество $\Pi_F(e)$ не является пустым, выбирается та проверка $i \in \Pi_F(e)$, для которой достигается минимальное значение функции $t(e)$.

В заключение следует отметить, что предложенный выше метод и позволяет составить оптимальную программу диагностики без необходимости перебора всех возможных программ, однако из-за его основных недостатков (большой объем вычислений и необходимость запоминания большого числа промежуточных результатов), свойственных всем задачам динамического программирования, он может быть практически использован только для составления сравнительно небольших программ.

Для иллюстрации предложенного метода составления оптимальной программы рассмотрим следующий пример. Пусть система состоит из четырех элементов (ее схема показана на рис. 1). Возможные состояния системы заданы множеством $S=\{s_0, s_1, s_2, s_3, s_4\}$, где в состоянии s_0 все элементы работоспособны, а в состоянии s_i — i-й элемент отказал ($i=1, 2, 3, 4$). Задано множество всех проверок $\Pi=\{\pi_1, \pi_2, \pi_3, \pi_4\}$, где π_i ($i=1, 2, 3, 4$) означает контроль сигнала на выходе i-го элемента. Связь между исходами x_i для проверки π_i ($i=1,2,3,4$) и состояниями из S задана табл. 1, где $x_i=1$, если сигнал на выходе i-го

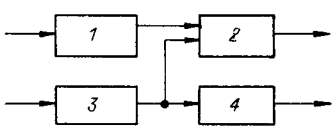

Рис. 1.

элемента удовлетворяет заданным требованиям, и $x_i = 0$ в противном случае.

Пусть, далее, для проведения проверки $i=1, 2, 3, 4$ требуется время τ_i и T — заданное максимальное суммарное время для проведения проверок. Предположим, что

$$\tau_i < T \quad (i=1,2,3,4);$$
$$\tau_1+\tau_3+\tau_4<T; \quad \tau_1+\tau_2+\tau_3+\tau_4>T;$$
$$\tau_1+\tau_2>T; \quad \tau_2+\tau_3>T; \quad \tau_2+\tau_4>T.$$

Таблица 1

	π_1	π_2	π_3	π_4
s_0	1	1	1	1
s_1	0	0	0	0
s_2	1	0	1	1
s_3	1	0	0	0
s_4	1	1	1	0

В данном примере множество всех наборов E равно

$E=\{(0_1), (1_1), (0_2), (1_2), (0_3), (1_3), (0_4), (1_4), (0_1 0_2), (1_1 0_2), (1_1 1_2), (0_1 1_3), (1_1 0_3), (1_1 1_3), (0_1 1_4), (1_1 0_4), (1_1 1_4), (0_2 0_3), (0_2 1_3), (1_2 1_3), (0_2 0_4), (0_2 1_4), (1_2 0_4), (1_2 1_4), (0_3 0_4), (1_3 0_4), (1_3 1_4), (0_1 0_2 1_3), (1_1 0_2 0_3), (1_1 0_2 0_3), (1_1 0_2 1_3), (1_1 1_2 1_3), (0_1 0_2 1_4), (1_1 0_2 0_4), (1_1 0_2 1_4), (1_1 0_2 1_4), (1_1 1_2 1_4), (0_1 1_3 1_4), (1_1 0_3 0_4), (1_1 1_3 0_4), (1_1 1_3 0_4), (0_2 0_3 0_4), (0_2 1_3 1_4), (1_2 1_3 0_4), (1_2 1_3 1_4), (1_1 1_2 1_3 1_4), (0_1 0_2 1_3 1_4), (1_1 0_2 1_3 1_4), (1_1 0_2 0_3 0_4), (1_1 1_2 0_3 0_4)\}.$

Поэтому

$$E_T = \{(0_1), (1_1), (0_2), (1_2), (0_3), (1_3), (0_4), (1_4), (0_1 1_3), (1_1 0_3), (1_1 1_3), (0_1 1_4), (1_1 0_4), (1_1 1_4), (0_3 0_4), (1_3 0_4), (1_3 1_4), (0_1 1_3 1_4), (1_1 0_3 0_4), (1_1 1_3 0_4), (1_1 1_3 1_4)\}.$$

В табл. 2 приведены все множества $\Pi_T(e)$ для отдельных наборов e.

Все возможные программы диагностики, удовлетворяющие заданному ограничению T, можно записать в следующем виде:

$H_1=\{[\varnothing; 1]; [(0_1); i^*]; [(1_1); 3]; [(1_1 0_3); i^*]; [(1_1 1_3); 4]; [(1_1 1_3 0_4); i^*]; [(1_1 1_3 1_4); i^*]\};$

$H_2=\{[\varnothing; 1]; [(0_1); i^*]; [(1_1); 4]; [(1_1 0_4); 3]; [(1_1 0_4 0_3); i^*]; [(1_1 0_4 1_3); i^*]; [(1_1 1_4); i^*]\};$

$H_3=\{[\varnothing; 2]; [(0_2); i^*]; [(1_2); i^*]\};$

$H_4=\{[\varnothing; 3]; [(0_3); i^*]; [(1_3); 1]; [(1_3 0_1); i^*]; [(1_3 1_1); 4]; [(1_3 1_1 0_4); i^*]; [(1_3 1_1 1_4); i^*]\};$

$H_5=\{[\varnothing; 3]; [(0_3); i^*]; [(1_3); 4]; [(1_3 0_4); i^*]; [(1_3 1_4); 1]; [(1_3 1_4 0_1); i^*]; [1_3 1_4 1_1); i^*]\};$

$H_6=\{[\varnothing; 4]; [(0_4); 3]; [(0_4 0_3); i^*]; [(0_4 1_3); i^*]; [(1_4); 1]; [(1_4 0_1); i^*]; [(1_4 1_1); i^*]\}.$

Графы-деревья этих программ изображены на рис. 2.

$$E_{H_1} = \{(0_1), (1_1 0_3), (1_1 1_3 0_4), (1_1 1_3 1_4)\};$$
$$E_{H_2} = \{(0_1), (1_1 0_4 0_3), (1_1 0_4 1_3), (1_1 1_4)\};$$
$$E_{H_3} = \{(0_2), (1_2)\};$$
$$E_{H_4} = \{(0_3), (1_3 0_1), (1_3 1_1 0_4), (1_3 1_1 1_4)\};$$
$$E_{H_5} = \{(0_3), (1_3 0_4), (1_3 1_4 0_1), (1_3 1_4 1_1)\};$$
$$E_{H_6} = \{(0_4 0_3), (0_4 1_3), (1_4 0_1), (1_4 1_1)\}.$$

Если функция f, заданная на множестве всех наборов $e \in E$, такая, что $f(e)$ есть число неразличимых состояний, которые соответствуют набору e, то

Таблица 2

e	$\Pi_T(e)$	$F^0(e)$	e	$\Pi_T(e)$	$F^0(e)$	e	$\Pi_T(e)$	$F^0(e)$
(0_1)	$\varnothing$	1	(1_4)	$\{\pi_1\}$	2	$(0_3\ 0_4)$	$\varnothing$	1
(1_1)	$\{\pi_3, \pi_4\}$	2	$(0_1\ 1_3)$	$\varnothing$	1	$(1_3\ 0_4)$	$\varnothing$	1
(0_2)	$\varnothing$	3	$(1_1\ 0_3)$	$\varnothing$	1	$(1_3\ 1_4)$	$\{\pi_1\}$	2
(1_2)	$\varnothing$	2	$(1_1\ 1_3)$	$\{\pi_4\}$	2	$(0_1\ 1_3\ 1_1)$	$\varnothing$	1
(0_3)	$\varnothing$	1	$(0_1\ 1_4)$	$\varnothing$	1	$(1_1\ 0_3\ 0_4)$	$\varnothing$	1
(1_3)	$\{\pi_1, \pi_4\}$	2	$(1_1\ 0_4)$	$\{\pi_3\}$	1	$(1_1\ 1_3\ 0_4)$	$\varnothing$	1
(0_4)	$\{\pi_3\}$	1	$(1_1\ 1_4)$	$\varnothing$	2	$(1_1\ 1_3\ 1_4)$	$\varnothing$	2

$f^0_{H_1} = f^0_{H_2} = f^0_{H_4} = f^0_{H_5} = f^0_{H_6} = 2$, но $f^0_{H_3} = 3$, т. е. любая из программ H_1, H_2, H_4, H_5, H_6 является оптимальной.

Построим оптимальную программу диагностики, для которой достигается минимум максимального числа неразличимых состояний. При использовании такой функции имеем:

$$F^0(e) = f^0(e),$$

если $\pi_T(e)$ пусто, и

$$F^0(e) = \min_{\pi_i \in \Pi_T(e)} \{ \max_{x_i \in X_i(e)} F^0(ex_i) \},$$

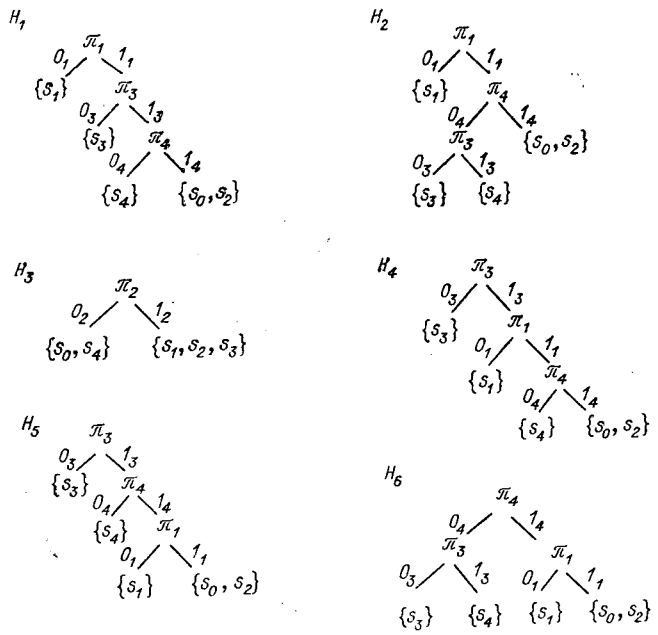

Рис. 2.

если $\pi_T(e)$ содержит хотя бы одну проверку. Значения функции $F^0(e)$ для всех $e \in E_T$ приведены в табл. 2. Найдем

$$F^0(\varnothing) = \min\{\max[F^0(0_1), F^0(1_1)]; \max[F^0(0_2), F^0(1_2)], \max[F^0(0_3), F^0(1_3)]; \max[F^0(0_4), F^0(1_4)]\} = \min\{\max[1, 2]; \max[3, 2]; \max[1, 2]; \max[1, 2]\} = \min\{2; 3; 2; 2\} = 2,$$

т. е. оптимальной программой является любая программа, которая не начинается с проверки π_2.

ЛИТЕРАТУРА

1. Г. Ф. Верзаков, Н. В. Киншт, В. И. Рабинович, Л. С. Тимонен. Введение в техническую диагностику. М., «Энергия», 1968.
2. M. Ullrich, L. Kubát. A. Generalized Approach to Fault — Finding Procedures.— Kybernetika, 1966, № 1.

Поступила в редакцию
10 мая 1971 г.