Марон А.И.

МЕТОД ПОСТРОЕНИЯ АЛГОРИТМОВ ПОИСКА НЕИСПРАВНОСТЕЙ В СИСТЕМАХ ЖЕЛЕЗНОДОРОЖНОЙ АВТОМАТИКИ

Предложен метод построения оптимальных алгоритмов поиска неисправностей для систем, потери от отказов которых нарастают экспоненциально с увеличением времени восстановления. Такая ситуация характерна для систем железнодорожной автоматики.

Ключевые слова: *алгоритм поиска неисправностей, системы железнодорожной автоматики, потери от отказов.*

1. Введение

Техническое обслуживание систем железнодорожной автоматики включает в себя оперативное восстановление при отказах. Обязательным этапом процесса оперативного восстановления является поиск неисправности, длительность которого во многом определяет размер потерь от отказа. Время поиска и влияние «человеческого фактора» можно существенно снизить, применяя для важнейших систем заранее разработанные оптимальные алгоритмы поиска неисправностей, которые обычно оформляются в виде инструкций. Известные в настоящее время методы построения таких алгоритмов предполагают, что критерием оптимальности является среднее время поиска неисправности [1]. Вместе с тем, для многих ответственных систем автоматики потери при отказе не связаны линейно со временем их восстановления, а нарастают значительно быстрее. Так, при отказе системы автоблокировки на участке с интенсивным движением поездов суммарное время задержки поездов возрастает экспоненциально с увеличением времени восстановления [2]. В данной работе впервые предложен метод построения оптимального алгоритма поиска неисправностей для таких систем.

2. Постановка задачи и метод решения

Не уменьшая общности результатов, для простоты изложения сформулируем задачу следующим образом. Система, являющаяся объектом поиска неисправностей, представлена функциональной моделью из n блоков, с номерами от 1 до n. Один из блоков является выходным. Ему присвоен номер n. Проверка $П^k$ сигнала на выходе блока k (k = 1, 2,…,n) имеет положительный результат

$\pi^{к}_{1}$, тогда и только тогда, когда он исправен и исправны все блоки ему предшествующие. В противном случае проверка $П^{k}$ будет иметь отрицательный результат $\pi^{к}_{0}$, означающий, что либо блок k неисправен, либо неисправен один из блоков ему предшествующий. Проверка $П^{n}$ выполнена. Получен отрицательный результат, означающий отказ системы ввиду неисправности одного из блоков. Вероятность неисправности блока i равна p_i. Время проверки $П^{k}$ равно t_k. Поиск неисправности будет осуществляться по разработанному условному алгоритму А, в котором очередная проверка выбирается в зависимости от результата предыдущей. В этом алгоритме каждому блоку i соответствует своё время поиска его неисправности–T_i (A). Оно равно сумме времён проверок, которые придётся для этого выполнить в соответствии с алгоритмом А. Потери от простоя системы связаны экспоненциально со временем её восстановления. Будем считать, что время замены неисправного блока намного меньше времени поиска неисправности. Тогда можно записать, что потери при отказе блока i и принятом алгоритме поиска неисправностей А будут равны

$$L_i(A) = \exp(r*T_i(A)),$$

где r – числовой коэффициент, величина которого для систем автоблокировки зависит от интенсивности движения поездов на участке.

Средние потери от отказа системы – L(A) при заданном алгоритме А представляют собой сумму по i, от 1 до n, произведений вероятностей p_i на L_i(A).

Требуется построить алгоритм A^{*}, при котором средние потери от отказа системы **минимальны**.

Уже при n = 10-20 количество различных алгоритмов поиска неисправностей, которые можно построить даже для рассматриваемой простейшей топологии системы, огромно. Найти среди них оптимальный алгоритм методом полного перебора практически невозможно. Ниже предлагается метод, который позволяет преодолеть барьер размерности и реально построить искомый алгоритм.

Будем рассматривать поиск неисправностей как процесс управления движением системы. Её надо перевести из начального состояния, когда возможными являются все n неисправностей, в одно из конечных состояний, когда неисправный блок точно определён. Число таких конечных состояний n. Все прочие состояния промежуточные. Каждому состоянию соответствует некоторое подмножество блоков системы, с точностью до которых в ходе выполнения проверок локализован неисправный блок. Например, если для системы, представленной на рисунке 1, выполнить проверку $П^{1}$, то при её отрицательном результате система перейдёт в состояние S(2,3,4). Каждому состоянию соответствует определённое количество блоков системы ||S||, с точностью до которых в ходе выполнения проверок локализован неисправный блок. В дальнейшем эту величину будем называть уровнем локализации неисправности. Так для S(2,3,4) имеем ||S|| = 3. Каждому неконечному состоянию S соответствует определённое множество проверок П(S), которые можно выполнить для дальнейшей локализации неисправности. Так для S(2,3,4) это проверки $П^{2}$ и $П^{3}$. В результате выполнения в состоянии S проверки $П^{k}$, входящей в П(S), система перейдёт в состояние $S(\pi^{к}_{1})$, если будет получен результат $\pi^{к}_{1}$, или $S(\pi^{к}_{0})$, при результате $\pi^{к}_{0}$.

Автором доказано, что минимальные средние потери от отказа системы можно найти с помощью рекуррентного соотношения

$$L^{*}(S) = \min L(S,П^{k}) = \min\{\exp(r*t_k)*[L^{*}(S(\pi^{к}_{0})) + L^{*}(S(\pi^{к}_{1}))]\}. \quad (1)$$

Минимум в соотношении (1) берётся по всем проверкам $П^k$, входящей в П(S). Расчёты должны быть проведены последовательно для всех возможных состояний S, начиная с тех, у которых ||S|| = 2, и заканчивая начальным состоянием, где ||S|| = n. При этом для конечных состояний, у которых ||S||=1 следует считать, что $L^*(S) = p_i$.

Метод построения оптимального алгоритма будет состоять из следующих этапов.

1) Для всех возможных состояний S, начиная с тех, у которых уровень локализации неисправности равен двум, и заканчивая конечным состоянием, вычислить с помощью соотношения (1) величину $L^*(S)$. Значение, найденное для конечного состояния, представляет собой минимальные потери от отказа системы. Запомнить эти значения и номера проверок k^*, при которых они получены.

2) Последовательно, начиная с начального состояния, выполнить следующие действия. Проверку, зафиксированную на первом этапе, принять в качестве первой проверки искомого оптимального алгоритма поиска неисправностей. Определить состояния, возникающие при её отрицательном и положительном результатах. Проверки, найденные на первом этапе для этих состояний, принять в качестве следующих при соответствующем результате предыдущей проверки. Продолжать процесс до тех пор, пока не будут достигнуты все конечные состояния. В результате будет получен искомый алгоритм A^*.

Для иллюстрации приведём простой пример.

На рисунке 1 приведена система из четырёх блоков. Вероятности неисправностей записаны над блоками, а длительности проверок в минутах указаны над соответствующими им выходами блоков. Коэффициент r = 0,25. В таблице 1 приведены возможные состояния, соответствующие им значения $L^*(S)$ и номера проверок k^*. Поясним вычислительную схему первого этапа на примере состояния S(2,3,4). После выполнения проверки $П^2$ система перейдёт или в состояние S(3,4) или в конечное состояние S(2). Соответственно $L(S,П^2) = \exp(0{,}25*8)*(0{,}25 + 1{,}12) = 10{,}13$.

Для проверки $П^3$ получим $L(S,П^3) = 13{,}69$. Поскольку $L(S,П^2) < L(S,П^3)$, то в соответствии с (1) принимаем $L^*(S) = 13{,}69$ и выбираем проверку $П^2$. Запоминаем эти значения в таблице 1. Поясним процедуру второго этапа. Рассмотрим начальное состояние. Соответствующую ему проверку $П^3$ принимаем в качестве первой проверки. Проверки, найденные на первом этапе для состояний S(1,3) и S(2,4), принимаем в качестве следующих за ней при отрицательном и положительном результатах. Оптимальный алгоритм построен и приведен на рисунке 1 в виде двоичного дерева. Вершины соответствуют проверкам, дуги их результатам, а листья неисправностям. Ему соответствуют минимальные средние потери $L(A^*) = 46{,}01$. Заметим, что заменив в (1) минимум на максимум, можно найти максимальные потери и соответствующий им наихудший алгоритм. Для данного примера $\max L(A) = 172{,}92$(!).

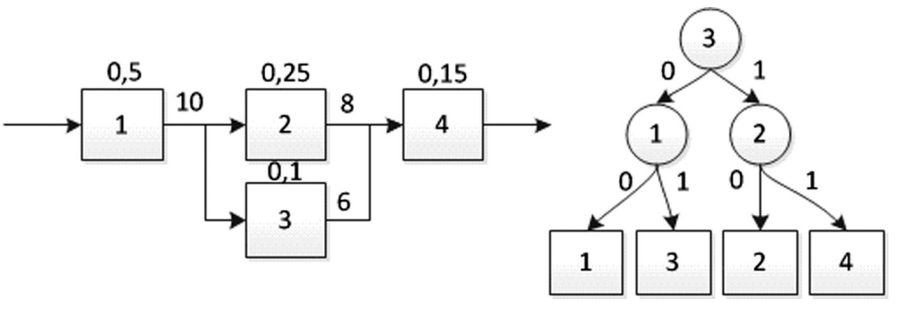

Рис. 1. Пример: система и алгоритм поиска неисправностей

Таблица 1. Пример: расчёты

‖S‖	S	k	$S(\pi^{\kappa}_0)$	$S(\pi^{\kappa}_1)$	$L(S,\Pi^k)$	L*(S)	k*
2	1,2	1	1	2	9,14	9,14	1
	1,3	1	1	3	7,31	7,31	**1**
	2,4	2	2	4	2,96	2,96	**2**
	3,4	3	3	4	***1,12***	**1,12**	3
3	2,3,4	2	2	3,4	10,13	10,13	2
		3	3	2,4	13,69		
4	1,2,3,4	1	1	2,3,4	129,45	**46,01**	**3**
		2	1,2	3,4	75,79		
		3	1,3	2,4	46,01		

3. Заключение

Впервые предложен метод построения оптимальных алгоритмов поиска неисправностей для систем, при отказах которых потери нарастают экспоненциально с увеличением времени восстановления. Такая ситуация характерна для систем железнодорожной автоматики. Для реализации метода разработано специальное программное обеспечение. Результат может найти применение и для диагностики ответственных проектов [3].

Литература

1. **Сапожников Вл.В., Сапожников В.В.** Основы технической диагностики – М.: Маршрут, 2004.

2. **Перникис Б.Д., Ягудин Р.Ш.** Предупреждение и устранение неисправностей в устройствах СЦБ.- М.: Транспорт,1994.

3. **Марон А.И., Марон М.А.** Информационный подход к организации контроля проектов // Бизнес – информатика, – 2012.-№4. – с.54 – 60.