УДК 519.718

А.М. РОМАНКЕВИЧ, И.В. МАЙДАНЮК, В.А. РОМАНКЕВИЧ

Национальный технический университет Украины «КПИ», Киев

ЧАСТНЫЙ СЛУЧАЙ ГРАНИЧНЫХ ОЦЕНОК ПРИ ПОСТРОЕНИИ И ПРЕОБРАЗОВАНИИ GL-МОДЕЛИ

В работе рассматривается GL-модель отказоустойчивой многопроцессорной системы, которая теряет минимальное количество ребер при появлении вектора состояния системы, включающего 1 лишний отказ. Сделано обобщение утверждения об условиях одновременного пропадания той или иной пары ребер при появлении такого вектора. Представлены оценки сложности преобразования GL-модели базовой отказоустойчивой многопроцессорной системы путем введения дополнительных ребер. Они позволяют разработчику ОМС оценить необходимые ресурсы (вычислительные и временные) для расчета надежности статистическими методами с требуемой точностью на этапе ее проектирования.

Ключевые слова: *отказоустойчивые многомодульные системы, графо-логические модели, раскраска графа.*

Введение

Отказоустойчивые реконфигурированные многопроцессорные системы (ОМС) достаточно широко используются в сфере управления сложными объектами и объектами критического использования, где отказ системы управления может привести к значительным потерям. В связи с этим, требования к надежности подобных систем являются очень высокими.

Свойства ОМС и способы расчета их надежности очень интенсивно исследуются как в нашей стране [1] так и за рубежом [2].

При проектировании ОМС, а также расчете некоторых их параметров целесообразно провести анализ поведения системы в потоке отказов. Отслеживая реакцию ОМС на появление отказов разной кратности, можно, в частности, рассчитывать ее надежностные характеристики путем выполнения статистических экспериментов с соответствующими моделями.

Одной из таких моделей является GL-модель поведения ОМС в потоке отказов [3], которая формируется сравнительно просто.

На начальных этапах проектирования ОМС разработчику полезно иметь оценку сложности GL-модели, которая соответствует разрабатываемой системе, а также сложность отображения изменений на модели при трансформации системы.

Зная конечную сложность модели, можно оценить объем необходимых вычислительных ресурсов и время выполнения одного эксперимента с ней, от чего в конечном итоге зависит точность расчета надежности ОМС.

1. Постановка задачи

GL-модель представляет собой неориентированный циклический граф R, ребрам которого приписаны определенные булевы функции, аргументами которых являются двоичные индикаторные переменные x_i (i=1..n), отображающие работоспособность соответствующего компонента системы (“0” – отказ, “1” – исправен). Если функция, приписываемая тому или иному ребру, принимает нулевое значение, то это ребро исключается. Связность графа R соответствует работоспособности системы в целом. Последовательность значений (как реальных, так и смоделированных) переменных назовем вектором состояния системы.

ОМС, состоящую из n процессоров и сохраняющую работоспособность при отказе не более чем m ее компонентов, будем называть базовой и обозначать как K(m,n). Обозначим мощность множества ребер модели через r.

Определим $W=\{w_1,w_2,w_3,...\}$ как множество определенных векторов состояния системы с m+1 неисправным процессором, при появлении которых система сохраняет работоспособность. Такая ОМС уже не может считаться базовой, и мы будем называть ее небазовой.

При появлении вектора из множества W граф базовой модели теряет связность вследствие исключения ребер, приписанные функции которым равны 0 на этом векторе. Для того чтобы модель оставалась адекватной поведению системы, необходимо ее трансформировать. Понятно, что в небазовой модели потеря связности в указанных случаях должна быть блокирована. Достичь этого можно путем вве-

дения в граф модели дополнительных ребер со своими функциями, и в [4] решена задача оптимального проведения этих ребер. В [5] же рассматривались граничные оценки количества дополнительных ребер для циклической GL-модели общего вида, построенной по произвольному алгоритму.

В настоящей работе рассматривается конкретная модель, описанная в [3], особенностью которой является пропадание минимального числа ребер при появлении вектора состояния системы с m+1 нулем. Кроме того, эта модель имеет сравнительно простой алгоритм формирования реберных функций и относительно невысокую сложность. Детальный анализ поведения такой модели в потоке отказов, показывает, что при появлении вектора из W далеко не любая пара ребер может выпадать одновременно, в связи с чем возникает задача уточнения полученных в [5] оценок.

Более точно задача формулируется следующим образом. Определить границы числа дополнительных ребер, вводимых для обеспечения адекватности модели при ее преобразовании, если известна мощность множества W, и, с другой стороны, определить пределы, в которых может находиться величина мощности множества W, появление векторов из которого не приведет к потере связности графа модели, если ограничено число дополнительных ребер.

1.1. Общее описание алгоритма формирования модели

Существует несколько способов формирования реберных функций GL-модели. Как было сказано, мы остановимся на алгоритме, предложенном в [3]. Суть этого алгоритма заключается в последовательном разделении множества компонентов (индикаторных переменных) системы на два равных или почти равных непересекающихся подмножества и переборе всех вариантов распределения по ним текущего количества отказов, к которым система должна быть устойчива. Для удобства будем различать эти подмножества как первое и второе, причем (хотя не обязательно) в первое входят переменные с меньшими порядковыми номерами. Этот процесс продолжается до тех пор, пока количество отказов в текущем подмножестве либо станет равным мощности подмножества, либо будет равно единице. Далее записывается функция, соответствующая каждому такому распределению, и приписывается одному ребру графа модели. В результате получается, так называемая, каноническая GL-модель.

В [3] предложен алгоритм минимизации канонической GL-модели, основанный на операции склеивания. Полученная после минимизации модель теряет минимальное количество ребер при появлении вектора состояния системы с m+1 нулем. Следуя [3], обозначим через K'(m,n) множество реберных функций GL-модели K(m,n). Каждый элемент из K'(m,n) приписан только одному ребру.

В канонической модели дизъюнктивное выражение:

$$K'(m\text{-}j,n')\vee K'(j,n''),$$

где j=1..m, $n'+n''=n_1$, представляет некоторую совокупность реберных функций. В нем операцию дизъюнкции следует относить не к самим множествам, а к парам элементов (т.е. функций) из разных множеств вида K'. По алгоритму минимизации указанную совокупность в конечном итоге можно представить в виде одной реберной функции. Заметим, что на первой итерации n_1 равно n, и затем – либо $]n/2^i[$ (округление к большему целому), либо $[n/2^i]$ (целая часть), где i – этап (номер итерации) деления множества компонентов пополам.

В качестве примера построим GL-модель K(4,15) согласно алгоритму описанному в [3]. На первом этапе множество переменных $x_1,\ldots,x_{15}$ разбиваем на подмножества $x_1,\ldots,x_8$ и $x_9,\ldots,x_{15}$. Выполнив все шаги алгоритма, получим следующие функции модели:

$f_1 =(x_1x_2\vee x_3\vee x_4)(x_1\vee x_2\vee x_3x_4)((x_1\vee x_2)(x_1x_2\vee x_3x_4)(x_3\vee x_4)$
$\vee\ x_5x_6x_7x_8)(x_1x_2x_3x_4\vee(x_5\vee x_6)(x_5x_6\vee x_7x_8)(x_7\vee x_8))$
$(x_5x_6\vee x_7\vee x_8)(x_5\vee x_6\vee x_7x_8)\vee x_9x_{10}x_{11}x_{12}x_{13}x_{14}x_{15}$

$f_2 =(x_1\vee x_2)(x_1x_2\vee x_3x_4)(x_3\vee x_4)(x_1x_2x_3x_4\vee x_5x_6x_7x_8)(x_5\vee x_6)$
$(x_5x_6\vee x_7x_8)(x_7\vee x_8)\vee(x_9\vee x_{10})(x_9\vee x_{10}x_{11})(x_9x_{10}x_{11}\vee$
$\vee x_{12}x_{13}x_{14}x_{15})(x_{12}\vee x_{13})(x_{12}x_{13}\vee x_{14}x_{15})(x_{14}\vee x_{15})$

$f_3 =x_1x_2x_3x_4x_5x_6x_7x_8\vee(x_9\vee x_{10}\vee x_{11})((x_9\vee x_{10})(x_9\vee x_{10}x_{11})\vee$
$\vee\ x_{12}x_{13}x_{14}x_{15})(x_9x_{10}x_{11}\vee(x_{12}\vee x_{13})(x_{12}x_{13}\vee x_{14}x_{15})$
$(x_{14}\vee x_{15}))(x_{12}\vee x_{13}\vee x_{14}x_{15})(x_{12}x_{13}\vee x_{14}\vee x_{15})$

$f_4 =x_1\vee x_2\vee x_3\vee x_4$

$f_5 =(x_1x_2\vee x_3\vee x_4)(x_1\vee x_2\vee x_3x_4)\vee x_5x_6x_7x_8$

$f_6=(x_1\vee x_2)(x_1x_2\vee x_3x_4)(x_3\vee x_4)\vee(x_5\vee x_6)(x_5x_6\vee x_7x_8)(x_7\vee x_8)$

$f_7 =(x_5x_6\vee x_7\vee x_8)(x_5\vee x_6\vee x_7x_8)\vee x_1x_2x_3x_4$

$f_8 =x_5\vee x_6\vee x_7\vee x_8$

$f_9 =x_9\vee x_{10}\vee x_{11}\vee x_{12}$

$f_{10}=(x_9x_{10}\vee x_{11}\vee x_{12})(x_9\vee x_{10}\vee x_{11}x_{12})\vee x_{13}x_{14}x_{15}$

$f_{11}=(x_9\vee x_{10})(x_9x_{10}\vee x_{11}x_{12})$
$(x_{11}\vee x_{12})\vee(x_{13}\vee x_{14})(x_{13}\vee x_{14}x_{15})$

$f_{12}=x_5x_6x_7x_8\vee x_{13}\vee x_{14}\vee x_{15}$

2. Дерево иерархии реберных функций модели

Мы будем использовать понятие дерева иерархии реберных функций (далее просто дерево), описанное в [6], где рассматривалась модель и ее дерево для случая K(3,n). Здесь мы обобщим это понятие для любой степени отказоустойчивости (m>2). В дереве иерархии каждой вершине соответствует

определенное ребро GL-модели, точнее его реберная функция.

Условно на дереве можно выделять уровни иерархии (этапы разделения множества переменных пополам). Далее, говоря о ниже или выше лежащем уровне будем иметь ввиду уровень с большим или, соответственно, меньшим порядковым номером, т.е.дальше или ближе к корню.

Корень дерева не имеет на GL-модели соответствующего ребра и введен лишь для упорядочения. Реберные функции модели упорядочиваются на дереве по уровням (по вертикали) в соответствии с мощностью множества переменных, от которых эти функции зависят. Если путь к корню дерева от вершины А проходит через вершину В, то функция, приписанная вершине А, зависит от подмножества переменных функции вершины В. На рис. 1 приведен пример такого дерева для модели К(4,15) и отмечены множества переменных, от которых зависят функции этой модели.

Назовем группой упорядоченную совокупность вершин, лежащих на одном уровне и которые имеют общую смежную вершину.

Вершины в группе упорядочиваются в соответствии с количеством отказов, попавших в первое подмножество компонентов, от которых зависят приписанные вершинам функции.

Соседними (соседями) будем называть вершины одной группы, функции которых отличаются на единицу по количеству отказов, попавших в первое подмножество переменных (на дереве они расположены рядом).

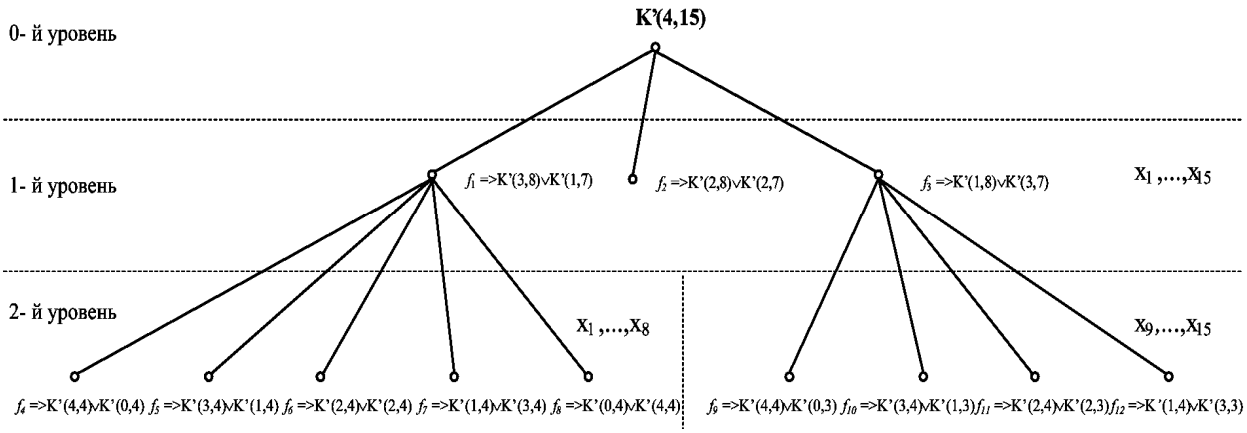

Рис. 1. Пример дерева для модели К(4,15)

В [6] доказано утверждение о двух ребрах минимизированной модели типа К(3,n), которые могут пропадать одновременно. Обобщим без доказательства это утверждения на случай К(m,n).

***Утверждение* 1.**

При появлении любого вектора состояния системы имеющего m+1 нулевую компоненту из графа GL-модели базовой ОМС К(m,n), построенной по [3], *принимают нулевые значения только такие две функции, которым соответствуют вершины на дереве иерархии со следующими свойствами: а) либо они расположены на одной ветви дерева, б) либо являются соседними.*

3. Формализации задачи

Как и ранее, рассматриваем случай появления вектора $w_i \in W$. Как показано в [3], при этом из графа модели исключаются два ребра. Множество пар ребер, исключение которых при преобразовании базовой модели в небазовую не должно приводить к потере связности графа, обозначим через L, а его мощность – через l. Отметим, что к исчезновению одной и той же пары ребер могут приводить несколько различных векторов, однако нас будут интересовать границы для l, так как именно пары выпадающих ребер определяют усложнение GL-модели.

Далее с целью сокращения выкладок введение дополнительных ребер для сохранения связности графа при исчезновении пар ребер из множества L будем называть блокированием множества L, или просто блокированием.

В [4] решена задача оптимального блокирования и приводится алгоритм, согласно которому количество и размещение дополнительных ребер определяется разбиением множества ребер модели на S-подмножества: в одном S-подмножестве присутствуют только такие ребра, выборка по два которых не совпадает ни с одним элементом из множества L. Определим через p количество S-подмножеств. Согласно [4] минимальное количество дополнительных ребер e=]p/2[, где]x[- округление к ближайшему большему целому.

Сказанное выше позволяет записать поставленную в п. 1 задачу в приведенных выше обозна-

чениях (еще раз напомним, что речь идет о минимизированной модели, построенной по [3]):

- определить границы p, исходя из заданных n, m и l;

- определить границы для l, при заданных p и n, m.

Определения:

Верхняя граница $p_{max}(l)$ – это такое количество S-подмножеств, которое способно блокировать любой набор элементов множества L, мощностью в l, выполняя условие минимальности количества S-подмножеств.

Нижняя граница $p_{min}(l)$ – это минимальное количество S-подмножеств, обеспечивающее возможность блокирования хотя бы одного набора из l пар ребер.

Нижняя граница $l_{min}(p)$ – такое минимальное значение l, что для блокирования элементов множества L требуется оптимальное разбиение множества ребер графа на p S-подмножеств, т.е. это такая комбинация из l пар, что удаление хотя бы одного элемента из множества L потребовало бы менее чем p S-подмножеств (с учетом требования минимальности к их количеству).

Верхняя граница $l_{max}(p)$ – это максимальное количество пар ребер, для блокирования которых достаточно разбиения графа на p S–подмножеств.

В [5] доказано, что функции $p_{min}(l)$ и $l_{max}(p)$, а также $l_{min}(p)$ и $p_{max}(l)$ являются взаимно обратными. Это позволяет, найдя одну функцию из пары взаимно-обратных, определить (если будет возможно) другую.

Вслед за [4] используем понятие V-графа: неограф, в котором вершинам ставятся в соответствие ребра исходного графа GL-модели, а ребра проводятся в соответствии с парами из множества L. Правильная раскраска вершин V-графа (под правильной подразумеваем раскраску, требующую минимальное количество цветов) определяет разбиение графа R модели на S-подмножества, причем ребра, имеющие один цвет, входят в одно S-подмножество. Отметим, что здесь и далее при построении графа V для упорядочивания его вершин мы будем использовать дерево иерархии, другими словами, порождающий граф V (не имеющий ребер) можно получить из дерева путем удаления всех ребер и вершины, соответствующей корню дерева.

4. Определение границ

Пусть r – количество ребер модели, а значит и количество вершин V-графа. Отметим, что согласно [5] r=n-m+1.

Обозначим через i - номер уровня (как на дереве иерархии так и на графе V), причем на дереве нумерация идет с нуля, а в V-графе с единицы, и k = max(i).

Пусть X(i) – общее количество вершин i-го уровня (мощность уровня). Понятно, что каждый из уровней (кроме, может быть, последнего) содержит вполне определенное количество вершин. Другими словами, все уровни, кроме (возможно) k-го, – полные.

Каждая группа такого уровня содержит (m-1) вершину, а количество таких групп на уровне определяется его номером и равно 2^{i-1}. Следовательно, $X(i) = 2^{i-1}(m-1)$ для $i<k$. Тогда мощность последнего уровня:

$$X(k)=r-\sum_{i=1}^{k-1} 2^{i-1}(m-1) = n-2^{k-1}(m-1).$$

Понятно, что $0<X(k)$. С другой стороны, поскольку максимальное количество ребер в группе k-го уровня равно m+1, то $X(k)\le 2^{k-1}(m+1)$. После простых преобразований получим $n/m\le 2^k$, и, следовательно, $k=]\log_2(n/m)[$.

4.1. $l_{min}(p)$

Пусть известно p – хроматическое число графа V, соответствующего GL-модели K(m,n), построенной по [3]. Поскольку l равно количеству ребер V-графа, то для определения $l_{min}(p)$ достаточно определить минимальное количество ребер, формирующих такой V-граф, который потребует p красок для правильной раскраски своих вершин.

Здесь и далее следует учитывать ограничения утверждения 1 на пары выпадающих ребер модели, и, соответственно, на проводимые ребра графа V.

Условно разделим все множество ребер, которые можно провести в V-графе, на два типа: ребра, расположенные между вершинами разных уровней (ребра уровней), и ребра между вершинами, входящими в одну группу (ребра группы). Заметим, что ребра группы можно проводить только между соседними вершинами, которые упорядочивались, как указано выше.

Для начала покажем, что хроматическое число V-графа не может превышать величину k+1. Для доказательства рассмотрим худший случай, а именно, проведем все возможные ребра в графе V. Определим количество красок, необходимых для раскраски полученного графа. Вершины, входящие в любую цепь на дереве иерархии, будут образовывать клику в графе V, и, следовательно, мощность максимальной клики (состоящей из ребер уровней) равна длине максимальной цепи, т.е. количеству уровней V-графа. С другой стороны, если на нижнем уровне найдется хотя бы одна группа, куда входит более 1-й вершины, то понадобится еще одна

краска. Поскольку такой V-граф имеет древовидную структуру, легко увидеть, что, действительно, для раскраски всего графа понадобится не более, чем k+1 краска.

Теперь становится понятным, что для определения минимального количества σ ребер в V-графе, раскраска которого потребовала бы p красок, достаточно выбрать подграф, где эти σ ребер образуют клику, состоящую из p вершин. А поскольку количество ребер в полном графе из p вершин определяется как $\frac{p(p-1)}{2}$, то искомая граница примет вид:

$$l_{min}(p)= \frac{p(p-1)}{2}, \text{ где } p \le k+1 \qquad (1)$$

4.2. $p_{max}(l)$

Поскольку $l_{min}(p)$ – монотонная функция, можно, выразив величину p из (1), как функцию от l, и решив квадратное уравнение, получить $p_{max}(l)$:

$$p_{max}(l)= \frac{1+\sqrt{1+8l}}{2} \text{ и } p_{max}(l) \le k+1 \qquad (2)$$

Действительно, пусть в соответствии с рассуждениями из 4.1, выбран граф V, содержащий клику из p вершин. Поскольку любой другой V-граф с тем же количеством ребер и вершин потребует для своей раскраски не большего количества цветов, то приведенные рассуждения дают максимальную оценку числа красок, т.е. верхнюю границу $p_{max}(l)$.

4.3. $l_{max}(p)$

Пусть имеется порождающий V-граф (без ребер), каждая вершина которого имеет свой цвет. Напомним, что вершины V-графа упорядочены на основе дерева иерархии и при проведении любых ребер в графе V учитываются ограничения утверждения 1.

Проведем все возможные ребра между вершинами, имеющими различный цвет. Теперь можно сказать, что поставленная в заголовке задача сводится к поиску алгоритма раскраски, на основе которой можно провести максимальное число ребер.

Вначале сделаем некоторые замечания.

Говоря о рёбрах, которые можно провести из некоего уровня, мы будем иметь в виду только рёбра, которые соединяют вершины рассматриваемого уровня с вершинами выше лежащих уровней графа V.

Будем различать узловые вершины, т.е. вершины, которые имеют нижележащие смежные вершины на дереве иерархии, и лепестки – которые таких потомков не имеют.

Положим, что $p \ge 4$, $m \ge 4$, количество уровней $k>p$ и $n = 2^k m$. При этом значении n $X(k)=2^{k-1}(m+1)$, т.е. все уровни дерева полные.

Ниже покажем процесс постепенной оптимизации искомого алгоритма, осуществляемый в три этапа. Раскраску, соответствующую каждому из этапов улучшения алгоритма, будем иллюстрировать на примере GL-модели K(4,128), дерево иерархии реберных функций которой представлено на рис. 2.а.

1-й этап.

Для простоты вначале будем окрашивать все вершины, принадлежащие одному уровню, в один цвет. Таким образом мы исключаем проведение ребер групп и сосредоточим внимание на ребрах уровня. Количество ребер, проводимых из данного уровня, определяется легко: $X(i) \cdot z(i)$, где z(i) - количество вышележащих уровней с отличной краской. Следовательно, общее количество ребер уровней будет $\sum_{i=1}^{k} X(i)z(i)$.

Интуитивно понятно, что количество ребер будет максимальным, если p-1 нижних уровней раскрасить каждый в отдельную краску, а оставшиеся уровни, скажем, в 1-ю и называть их уровнями 1-й краски (пример раскрашивания приведен на рис. 1.б). Но все же докажем это.

Напомним, что $X(i)=2X(i-1)$ при $i<k$. Пусть на данном этапе это справедливо и для i=k, тогда

$$\sum_{j=1}^{i-1} X(j) = X(i)-1.$$

Раскрашивать уровни будем последовательно, начиная с нижнего. Раскрасим k-й уровень в p-й цвет. Далее, для (k-1)-го уровня можно использовать либо уже использованную (т.е. p-ю) краску, либо другую, скажем, (p-1)-ю. Рассмотрим оба варианта. Для первого варианта z(k) получится меньше в лучшем случае на единицу в сравнении со вторым вариантом, но при этом не исключено, что ко всем z(i), для $i< k$ можно прибавить 1.

Теперь, зная, что $\sum_{j=1}^{k-1} X(j) = X(k)-1$, нетрудно увидеть, что второй вариант в самом худшем случае дает на 1 ребро больше.

Применяя последовательно подобную схему окрашивания для остальных уровней, получим первоначальный вариант алгоритма.

2-й этап.

Теперь снимем ограничение предыдущего этапа и предоставим возможность красить вершины одного уровня в различные цвета.

Предложенный на первом этапе алгоритм можно улучшить, если сохранить рассмотренную выше раскраску только для узлов, а лепесткам присваивать краску из множества цветов, в которые окрашены узлы ниже лежащих уровней. В случае предпоследнего уровня можно для раскраски лепестков использовать p-ю и (p-1)-ю краски. Для нечетного m один из узлов группы (k-1)-го уровня покрасить в p-й цвет, а соответствующую группу k-го уровня перекрасить в (p-1)-й (пример дан на рис. 2.в).

Теперь мы можем дополнительно (если сравнивать с предыдущим этапом) провести все ребра групп, кроме ребер групп на последнем уровне, и провести ребра уровней для лепестков уровней первой краски, не уменьшая при этом количество ребер, проведенное на предыдущем этапе.

3-й этап.

Поскольку понятно, что для уровней первой краски нельзя ничего улучшить без более ощутимых потерь, попытаемся подобрать более выгодную расстановку p-1 красок для p-1 нижних уровней.

Поскольку V-граф обладает горизонтальной симметричностью, достаточно рассмотреть любое поддерево дерева иерархии из (p-1)-го нижних уровней и соответствующий ему подграф графа V.

Согласно 1-му этапу корень данного подграфа имеет цвет 2. Согласно 2-му этапу в данном подграфе проведены все возможные ребра уровней и все ребра групп за исключением $2^{p-3}m$ ребер групп нижнего (p-го) уровня. Отличительные положения алгоритма 3-го этапа следующие. Узлу 1-го уровня (подграфа) оставить краску 2, а узлам i-го уровня - i-ю (i=2..p-2). Лепесткам присваивать краски нижележащих уровней. На (p-1)-й уровень остается две краски (p-1)-я и p-я. Другими словами, для (p-2)-х нижних уровней может быть использована (p-1)-на краска, что, согласно 4.1, достаточно для проведения всех возможных ребер.

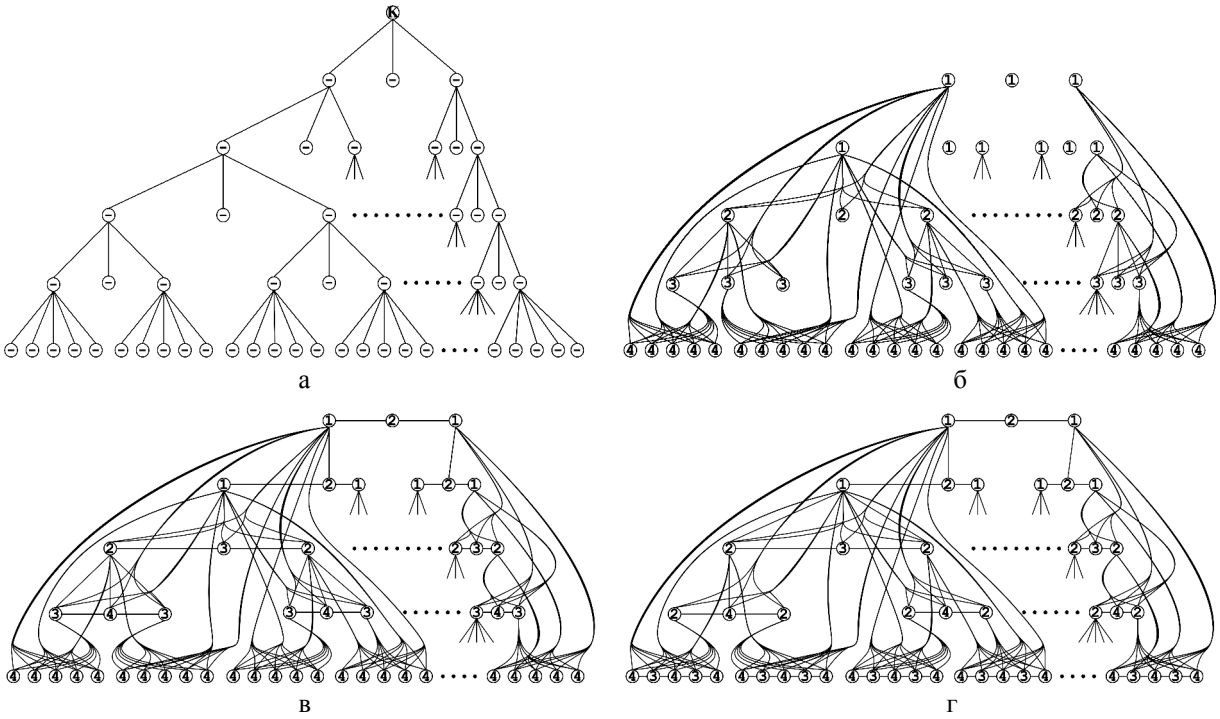

Рис. 2. Деревья иерархий

При такой расстановке красок можно провести все ребра групп и все ребра уровней, за исключением 2-х ребер между узлами 1-го и 2-го уровня (см. пример на рис 2, г). Как видим данная раскраска дает большее количество ребер по сравнению со вторым этапом.

Таким образом, алгоритм раскраски порождающего V-графа примет следующий вид.

1) Всем узлам уровня с 1-го по k-p+1 присваиваем краску 1. Для лепестков этих уровней определяем 2-ю и 3-ю краску.

2) Узлам k-p+2 уровня ставим 2-ю, лепесткам 3-ю и 4-ю.

3) Для узлов уровня с k-p+3 до (k-1)-го используем краски с номерами от 2 по (p-2), Для лепестков этих уровней можно использовать краски узлов (лепестков для предпоследнего) нижележащих уровней.

4) Для лепестков k-го уровня используем p-ю и (p-1)-ю краску.

Определим общее количество ребер. Для этого сначала приведем ряд простых формул расчета количества ребер без предварительного вывода.

Общее количество ребер уровня, которые могут быть проведены из вершин с 1-го по (k-1)-й уровень: $\sum_{i=1}^{k-1} 2^{i-1}(m-1)(i-1)$

Количество ребер уровня, которые могут быть проведены из вершин k-го уровня: $2^{k-1}(m+1)(k-1)$.

Общее количество узловых ребер уровня, которые могут быть проведены из вершин с 1-го по k-p+1 уровень: $\sum_{i=1}^{k-p+1} 2^{i-1}2(i-1)$.

Количество ребер уровня между узлами, имеющими 2-й цвет: 2^{k-p+3}.

Общее количество ребер групп с 1-го по (k-1)-й уровень: $\sum_{i=1}^{k-1} 2^{i-1}(m-2)$.

Количество ребер групп на k-м уровне: $2^{k-1}m$. Суммируя полученные выражения:

$$l_{max}(k,p) = \sum_{i=1}^{k-1} 2^{i-1}(m-1)(i-1) + 2^{k-1}(m+1)(k+1) -$$

$$- \sum_{i=1}^{k-p+1} 2^{i-1}2(i-1) - 2^{k-p+3} + \sum_{i=1}^{k-1} 2^{i-1}(m-2) + 2^{k-1}m =$$

$$2^k m(k-1) - 2^{k-p+2}(k-p+1) + m - 4,$$

получаем окончательно

$$l_{max}(k,p) = 2^k m(k-1) - 2^{k-p+2}(k-p+1) + m - 4 \quad (3)$$

Для случая $p \geq 4$, $m \geq 4$, но k=p либо k+1=p и $n=2^k m$ используем небольшую модификацию представленного алгоритма: узлы (k-p+2)-го уровня красятся не во 2-й цвет, а в 1-й.

Для этой модификации справедливо следующее соотношение:

$$l_{max}(k,p) = 2^k m(k-1) + m - 2^{k-p+3}(k-p) - 4 \quad (4)$$

Случай m=2 не рассматриваем, ввиду того, что для подобных ОМС, чаще всего используется другая модель [7].

Построение дерева для m=3, имеет свои особенности. Не вдаваясь в детали, приведем конечный результат:

$$l_{max}(k,p) = 3 \cdot 2^k (k-1) - 2^{k-p+2}(k-p+3) \quad (5)$$

В случаях произвольных значений n (т.е. когда $n \neq 2^k m$) можно получить грубую (упрощенную) оценку $l_{max}(p)$ простой заменой в (3)-(5) величины k=]$\log_2(n/m)$[на величину $k=\log_2(n/m)$.

4.4. $p_{min}(l)$

Поскольку выражения (3)-(5), не имеют простых решений для получения обратных функций, для поиска приближенных значений $p_{min}(l)$ можно воспользоваться графическим представлением упомянутых грубых оценок.

Заключение

Результаты, полученные в работе, дают возможность более точно оценить сложность преобразования GL-модели, построенной по практически важному и наиболее часто используемому алгоритму. Особенность этой модели – потеря всегда 2-х ребер при появлении вектора состояния ОМС, содержащего m+1 нулевую компоненту.

Полученные оценки призваны помочь разработчику в определении ресурсов (вычислительных и временных), необходимых для расчета надежности ОМС с требуемой точностью на этапе ее проектирования.

Литература

1. Харченко В.С. Многоверсионные системы, технологии, проекты / В.С. Харченко, В.Я. Жихарев, В.М. Илюшко, Н.В. Нечипорук. - Харьков; Нац. аэрокосм. ун-т "Харьк.авиац. ин-т", 2003. – 486 с.

2. Kuo Way. Optimal Reliability Modeling / Way Kuo, Ming J. Zuo. – John Willey & Sons, 2002. – 560 p.

3. Романкевич В.А. GL-модель поведения отказоустойчивых многопроцессорных систем с минимальным числом теряемых рёбер / В.А. Романкевич, Е.Р. Потапова, Бахтари Хедаятоллах, В. В. Назаренко // Вісник НТУУ "КПІ" Інформатика, управління та обчислювальна техніка. - 2006. - № 45. - С. 93-100.

4. Романкевич А.М. Анализ отказоустойчивых многомодульных систем со сложным распределением отказов на основе циклических GL-моделей / А.М. Романкевич, В.В. Иванов, В.А Романкевич // Электронное моделирование. - 2004. - Т. 26, № 5. - С. 67-81.

5. Романкевич А.М. Граничные оценки числа ребер GL-моделей поведения отказоустойчивых многопроцессорных систем в потоке отказов / А.М. Романкевич, В.А. Романкевич, И.В. Майданюк // Электронное моделирование. – 2008. – Т. 30, № 1. - С. 59-70.

6. Романкевич В.А. Условие существования попарных реберных циклов в GL-моделях K(3,n) / В.А. Романкевич, А.А. Кононова, Бахтари Хедаятоллах // Вісник НТУУ "КПІ" Інформатика, управління та обчислювальна техніка. - 2007. - № 46. - С. 54-61.

7. Романкевич А.М. Особенности трансформации GL-моделей базовых двухустойчивых ОМС к небазовым / А.М. Романкевич, А.А. Кононова // Радіоелектронні і комп'ютерні системи. – 2006. - № 5. - С. 48-53.

Поступила в редакцию 28.01.2010

Рецензент: д-р техн. наук, проф., проф. кафедры Вычислительной техники, В.П. Широчин, Национальный технический университет Украины «КПИ», Киев.

ЧАСТКОВИЙ ВИПАДОК ГРАНИЧНИХ ОЦІНОК ПРИ ПОБУДОВІ І ПЕРЕТВОРЕННІ GL-МОДЕЛІ

О.М. Романкевич, І.В. Майданюк, В.О. Романкевич

У роботі розглядається *GL*-модель відмовостійкої багатопроцесорної системи, яка втрачає мінімальну кількість ребер при появі вектора стану системи, що включає одну зайву відмову. Зроблено узагальнення твердження про умови одночасного зникнення тієї чи іншої пари ребер при появі такого вектора. Представлені оцінки складності перетворення *GL*-моделі базової відмовостійкої багатопроцесорної системи шляхом введення додаткових ребер. Вони дозволяють розробнику ОМС оцінити необхідні ресурси (обчислювальні й часові) для розрахунку надійності статистичними методами з необхідною точністю на етапі її проектування.

Ключові слова: відмовостійкі багатомодульні системи, графо-логічні моделі, розфарбування графа.

PRIVATE CASE, THE BOUNDARY OF ESTIMATES IN THE FORMATION AND TRANSFORMATION GL-MODEL

O.M. Romankevich, I.V. Maidanyuk, V.O. Romankevich

In this work the GL-model of fault-tolerant multiprocessor system, that loses the minimum number of edges in the appearance of the state vector systems, including one extra refusal is discussed. Made generalized statements about the conditions of simultaneous disappearance of a pair of edges with the appearance of such a vector. Presents estimates of the transformation GL-model basic fault-tolerant multiprocessor system by introducing additional edges. They allow the developer of FMS to assess the necessary resources (computing and time) to calculate the reliability of statistical methods with the required accuracy in its inception

Key words: fault-tolerant multimodule systems, graph-logical models, graph coloration.

Майданюк Иван Викторович – аспирант кафедры специализированных компьютерных систем Национального технического университета Украины «Киевский политехнический институт», Киев, Украина, e-mail: maidanyuk_vanya@ukr.net.

Романкевич Алексей Михайлович – д.т.н., проф., проф. кафедры специализированных компьютерных систем Национального технического университета Украины «Киевский политехнический институт», Киев, Украина, e-mail: romankev@scs.ntu-kpi.kiev.ua.

Романкевич Виталий Алексеевич – к.т.н, доц., доц. кафедры специализированных компьютерных систем Национального технического университета Украины «Киевский политехнический институт», Украина, e-mail: romankev@scs.ntu-kpi.kiev.ua.