УДК 004.2, 004.4

М.Б. Ильяшенко, канд. техн. наук, доцент,
Запорожский национальный технический университет, г. Запорожье, Украина
ilmatsoft@mail.ru

Решение задачи поиска граф-подграф изоморфизма для семантического анализа специализированных цифровых систем

Предлагается усовершенствованный алгоритм поиска изоморфизма графов и результаты исследования его эффективности. Объектом исследования является методика устранения противоречий, возникающих при анализе работы специализированных цифровых систем.

Ключевые слова: изоморфизм, взвешенные графы, семантический анализ, специализированные вычислители

Введение

Семантическая теория проектирования автоматов [1] предлагает эффективный метод решения проблем синтеза автоматов на основе общетеоретической интерпретации выражений входного языка в категориях выходного без фактического построения соответствующих конструкций на выходном языке. Такой подход является универсальным, так как позволяет проектировать оптимальные системы логического управления, например, с использованием интегральной, нейронной технологии. То есть способ реализации устройств управления принципиально не ограничен каким-либо базисом.

Синтаксическое эквивалентирование соответствует уровню знаний, при котором известна полная система аксиом. Синтаксическое эквивалентирование заключается в замене модели моделью для задачи синтеза на основании той или иной аксиомы или закона, полученного из системы аксиом, определенных в предметно-ориентированной области.

Актуальность. Такой подход позволяет укрупнить варианты решения и осуществить их сравнение не на уровне фактически полученных результатов, а на уровне исходной информации. При этом каждый сравниваемый вариант соответствует целому классу в некотором смысле изоморфных решений. Укрупнение перебора позволяет значительно сократить время решения задачи, к тому же интерпретация исходной информации осуществляется в терминах решения.

Постановка задачи. Одной из часто встречающихся подобного рода задач является устранение семиотических противоречий, возникающих при анализе результатов работы цифровых устройств.

Семиотическое противоречие является следствием неполного понимания принципов работы устройства из-за ограничений синтаксиса справочной документации. Для секвенциальных приборов выполняется семиотический анализ таблицы состояний. Иногда анализ целесообразно выполнять по отношению к обновленной таблице истинности комбинационных схем, имеющих трюковые входы включения / выключения специальных режимов работы.

Устранение противоречий равносильно обретению ясного представления о функциях микросхемы. Рассмотрим применение методики семиотического анализа [2] обновленной таблицы состояний микросхемы ID74163.

1 Пример семиотического анализа

Модель устранения противоречий формализуется в терминах теории графов. Необходимо с помощью компьютерного моделирования организовать обход всех противоречивых дуг графа GX и затем удалить недействительные дуги графа переходов прибора ID74163.

Шаг 1. Формирование графа противоречий GX.

На исходном графе переходов G0 выделяется подграф GX0, образованный противоречивыми переходами и смежными им узлами. Противоречивые переходы изображены пунктиром и помечены именами PUL1, PUL2, PUC, PUP1 и PUP2.

Шаг 2. Поиск маршрутов обхода противоречивых дуг.

Граф противоречий GX0 (рис. 1) ориентированный (орграф). Поэтому формирование маршрутов обхода определяется величинам полустепеней входа и выхода узлов.

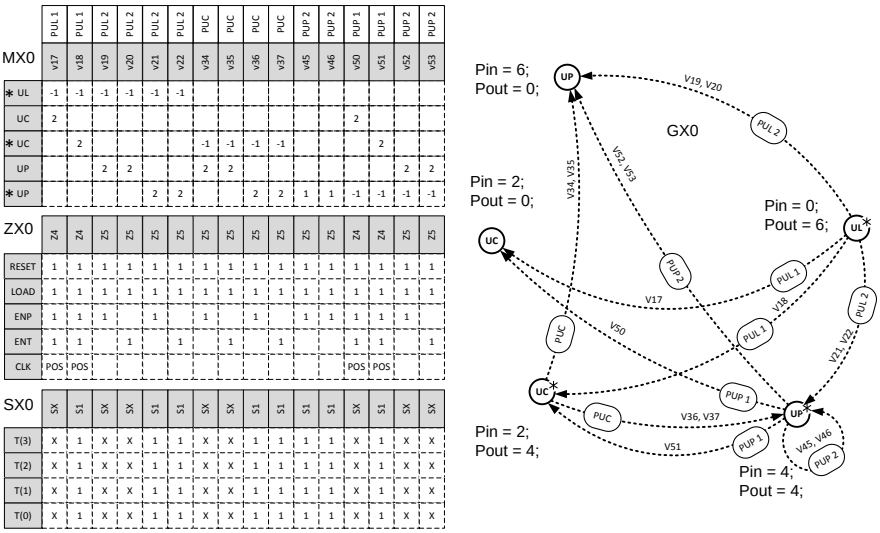

Рисунок 1 – Граф противоречивых переходов GX0

Полустепенью входа Pin узла графа называется число входящих дуг, смежных с каким-либо другим узлом графа. Полустепенью выхода Pout узла графа называется число исходящих дуг, смежных какому-либо другому узлу графа. Петли, образованные рефлексивными дугами графа, в расчет не принимаются.

На рис. 1 значения полустепеней показаны рядом с каждым узлом графа GX0

При особых значениях Pin и Pout, узлы ориентированного графа (орграфа) могут быть классифицированы:

1) если Pin ≠ 0, а Pout = 0, то узел орграфа называется финишным;
2) если Pin = 0, а Pout ≠ 0, то соответствующий узел орграфа считается тупиковым;
3) если Pin = 0 и Pout = 0, то узел орграфа мнимый.

Суть поиска маршрутов обхода заключается в выделении мнимых и тупиковых узлов, и, при положительном результате поиска, формировании рекомендаций для коррекции его топологии. Коррекция топологии, в данном случае, означает расширение множества дуг графа противоречий GX за счет добавления в него дуг переходов, имеющихся в исходном графе переходов G.

Идеальным способом коррекции графа противоречий GX является такое дополнение его мнимых и тупиковых узлов, что множество узлов графа GX не изменяется.

Это идеал, но на практике могут встречаться разные варианты. В табл. 1 перечислены приоритеты выбора топологической коррекции графа противоречий.

Таблица 1 – Приоритеты видов топологической коррекции

Приоритет	Вид коррекции	Последствия
Высокий	Петли, образованные рефлексивными дугами.	Изменяется множество дуг подграфа.
Средний	Входные дуги, состояния-источники которых принадлежат множеству узлов подграфа.	Изменяется множество дуг подграфа. Изменяется полустепени узлов подграфа.
Низкий	Входные дуги, состояния-источники которых принадлежат множеству узлов графа.	Изменяется множество дуг подграфа. Изменяется полустепени узлов подграфа. Изменяется множество узлов подграфа.

В данном случае, узлы UC* и UP*, имеющие ненулевые значения полустепеней входа и выхода, могут быть использованы для организации маршрутов обхода. Состояния UP и UC являются финишными, так как их Pout = 0. Топологию дуг, смежных этим четырем узлам графа противоречий, корректировать нет необходимости.

В коррекции нуждается узел UL*, значение полустепени выхода которого максимальное, – Pout = 6, а полустепень входа Pin = 0. В такой топологии узел UL* не входит ни в один из маршрутов обхода, так как отсутствует способ установления прибора ID74163 в состояние UL*.

Осуществляется коррекция. Тупиковый узел UL* графа противоречий GX0 дополняется рефлексивным относительно него переходом V14 – элементом множества дуг орграфа G0. В результате формируется окончательная топология графа противоречий GX1.

MX1

		PUL 1	PUL 1	PUL 2	PUL 2	PUL 2	PUL 2	PUC	PUC	PUC	PUC	PUP 2	PUP 2	PUP 1	PUP 1	PUP 2	PUP 2
	v14	v17	v18	v19	v20	v21	v22	v34	v35	v36	v37	v45	v46	v51	v50	v32	v33
* UL	1	-1	-1	-1	-1	-1	-1										
UC		2													2		
* UC			2					-1	-1	-1	-1			2			
UP				2	2			2	2							2	2
* UP						2	2			2	2	1	1	-1	-1	-1	-1

ZX1

	Z3	Z4	Z4	Z5	Z6	Z5	Z6	Z5	Z6	Z5	Z6	Z5	Z6	Z4	Z4	Z5	Z6
RESET	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
LOAD	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
ENP	0	1	1	1		1		1		1		1		1	1	1	
ENT	0	1	1		1		1		1		1		1	1	1		1
CLK	POS	POS	POS											POS	POS		

EX1

	E1	E0	E0	E0	E0	E0	E0	E0	E0	E0	E0	E0	E0	E0	E0	E0	E0
D(3)	1																
D(2)	1																
D(1)	1																
D(0)	1																

WX1

	W1	WX	W1	WX	WX	W1	W1	WX	WX	W1	W1	W1	W1	W1	WX	WX	WX
T(3)	1	X	1	X	X	1	1	X	X	1	1	1	1	1	X	X	X
T(2)	1	X	1	X	X	1	1	X	X	1	1	1	1	1	X	X	X
T(1)	1	X	1	X	X	1	1	X	X	1	1	1	1	1	X	X	X
T(0)	1	X	1	X	X	1	1	X	X	1	1	1	1	1	X	X	X
RCO	1	0	1	0	0	1	1	0	0	1	1	1	1	1	0	0	0

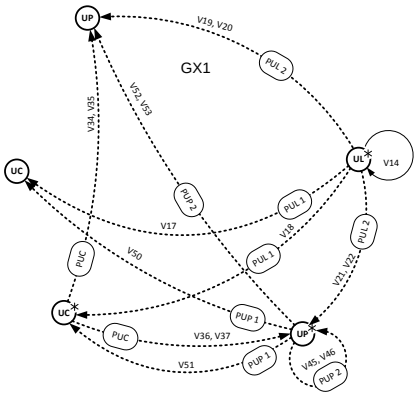

Рисунок 2 – Граф семиотических противоречий GX1

Автоматизация описанных выше и всех остальных возможных операций с графами обеспечивается табличной формой их представления.

На рис. 1 и 2 слева представлены таблицы, с помощью которых объект исследования формализуется одновременно в терминах теории орграфов и широкой модели микропрограммного взаимодействия:

– M – матрица инцидентности;
– Z – матрица термов;
– E – матрица входов;
– W – матрица выходов.

Например, для того, чтобы определить полустепень входа какого-либо узла, можно в матрице инцидентности M подсчитать число двоек в соответствующей строке. Полустепень выхода определяется подсчетом маркеров «-1» и т.д.

В дальнейшем, в зависимости от удобства, будут применяться оба способа описания графов: графический и табличный.

Шаг 3. Определение признаков устранения каждого противоречия и построение алгоритмов тестирования.

На графе GX1 выделено пять семиотических противоречий: PUL1, PUL2, PUC, PUP1 и PUP2. На основе информации, содержащейся в таблицах MX, ZX, EX и WX, можно составить пять отдельных алгоритмов, отражающих основные подходы к устранению противоречий.

Алгоритм устранения противоречия PUL1.

Если текущее состояние прибора UL* и на вход поступает терм Z4, то на графе GX1 возможны два варианта:

- переход V17 – в состояние UC, и в счетных триггерах T(3:0) формируется слово WX;

- переход V18 – в состояние UC*, и в счетных триггерах T(3:0) формируется слово W1.

Критерий устранения противоречия: двоичный код, хранящийся в T(3:0), определяет значение выходного сигнала RCO. Если RCO = '0', то осуществлен переход V17; если RCO = '1', то – переход V18.

(UL*) если ZX.Z4, то WX.RCO= '0' : V17 (UC); или WX.RCO= '1' : V18 (UC*);

Алгоритм устранения противоречия PUL2.

Если текущее состояние прибора UL* и на вход поступает терм Z5 или Z6, то на графе GX1 возможны два варианта:

- переходы V19, V20 – в состояние UP, и в счетных триггерах T(3:0) формируется слово WX;

- переходы V21, V22 – в состояние UP*, и в счетных триггерах T(3:0) формируется слово W1.

Критерий устранения противоречия: двоичный код, хранящийся в T(3:0), определяет значение выходного сигнала RCO. Если RCO = '0', то осуществлен переход V19 или V20; если RCO = '1', то – переход V21 или V22.

(UL*) если ZX.Z5, то WX.RCO= '0' : V19 (UP); или WX.RCO= '1' : V21 (UP*);

(UL*) если ZX.Z6, то WX.RCO= '0' : V20 (UP); или WX.RCO= '1' : V22 (UP*);

Алгоритм устранения противоречия PUC.

Если текущее состояние прибора UC* и на вход поступает терм Z5 или Z6, то на графе GX1 возможны два варианта:

- переходы V34, V35 – в состояние UP, и в счетных триггерах T(3:0) формируется слово WX;

- переходы V36, V37 – в состояние UP*, и в счетных триггерах T(3:0) формируется слово W1.

Критерий устранения противоречия: двоичный код, хранящийся в T(3:0), определяет значение выходного сигнала RCO. Если RCO = '0', то осуществлен переход V34 или V35; если RCO = '1', то – переход V36 или V37.

(UC*) если ZX.Z5, то WX.RCO= '0' : V34 (UP); или WX.RCO= '1' : V36 (UP*);

(UC*) если ZX.Z6, то WX.RCO= '0' : V35 (UP); или WX.RCO= '1' : V37 (UP*);

Алгоритм устранения противоречия PUP2.

Если текущее состояние прибора UP* и на вход поступает терм Z5 или Z6, то на графе GX1 возможны два варианта:

- переходы V52, V53 – в состояние UP, и в счетных триггерах T(3:0) формируется слово WX;

- переходы V45, V46 – в состояние UP*, и в счетных триггерах T(3:0) формируется слово W1.

Критерий устранения противоречия: двоичный код, хранящийся в T(3:0), определяет значение выходного сигнала RCO. Если RCO = '0', то осуществлен переход V52 или V53; если RCO = '1', то – переход V45 или V46.

(UP*) если ZX.Z5, то WX.RCO= '0' : V52 (UP); или WX.RCO= '1' : V45 (UP*);

(UP*) если ZX.Z6, то WX.RCO= '0' : V53 (UP); или WX.RCO= '1' : V46 (UP*);

Алгоритм устранения противоречия PUP1.

Если текущее состояние прибора UP* и на вход поступает терм Z4, то на графе GX1 возможны два варианта:

- переход V50 – в состояние UC, и в счетных триггерах T(3:0) формируется слово WX;

- переход V51 – в состояние UC*, и в счетных триггерах T(3:0) формируется слово W1.

Критерий устранения противоречия: двоичный код, хранящийся в T(3:0), определяет значение выходного сигнала RCO. Если RCO = '0', то осуществлен переход V50; если RCO = '1', то – переход V51.

(UP*) если ZX.Z4, то WX.RCO= '0' : V50 (UC); или WX.RCO= '1' : V51 (UC*);

В результате формируются восемь кратких правил устранения противоречий в работе микросхемы.

1) (UL*) если ZX.Z4, то WX.RCO= '0' : V17 (UC); или WX.RCO= '1' : V18 (UC*);
2) (UL*) если ZX.Z5, то WX.RCO= '0' : V19 (UP); или WX.RCO= '1' : V21 (UP*);
3) (UL*) если ZX.Z6, то WX.RCO= '0' : V20 (UP); или WX.RCO= '1' : V22 (UP*);
4) (UC*) если ZX.Z5, то WX.RCO= '0' : V34 (UP); или WX.RCO= '1' : V36 (UP*);
5) (UC*) если ZX.Z6, то WX.RCO= '0' : V35 (UP); или WX.RCO= '1' : V37 (UP*);
6) (UP*) если ZX.Z5, то WX.RCO= '0' : V52 (UP); или WX.RCO= '1' : V45 (UP*);
7) (UP*) если ZX.Z6, то WX.RCO= '0' : V53 (UP); или WX.RCO= '1' : V46 (UP*);
8) (UP*) если ZX.Z4, то WX.RCO= '0' : V50 (UC); или WX.RCO= '1' : V51 (UC*);

На их основе можно синтезировать восемь тестовых последовательностей, предназначенных для ввода в поле редактирования лицевой панели генератора слов, размещенного на электрической схеме модели.

Шаг 4. Формирование тестовых последовательностей для устранения противоречий. В качестве примера рассмотрим устранение противоречий PUL1 и PUL2.

Устранять противоречия следует не все сразу, а по очереди, постепенно «расчищая» граф противоречий GX от недействительных переходов. За счет этого, на каждой итерации, увеличивается доля действительных переходов на графе, следовательно, возрастает степень доверия результатам компьютерного моделирования.

Из всех узлов на графе GX1 только состояние UL* имеет однозначно воспринимаемый вход – петлю V14. Поэтому, на первом этапе, «расчищаются» противоречия, определенные относительно узла UL*. Краткие правила устранения противоречий PUL1 и PUL2:

1) (UL*) если ZX.Z4, то WX.RCO= '0' : V17 (UC); или WX.RCO= '1' : V18 (UC*);
2) (UL*) если ZX.Z5, то WX.RCO= '0' : V19 (UP); или WX.RCO= '1' : V21 (UP*);
3) (UL*) если ZX.Z6, то WX.RCO= '0' : V20 (UP); или WX.RCO= '1' : V22 (UP*);

На рис. 3 схематично изображен принцип формирования тестовой последовательности разрешения противоречий PUL1 и PUL2.

Первый тестовый вектор каждой последовательности, – 008Fh, соответствует переходу V14 и предназначен для установления исходного состояния прибора UL*. С помощью второго вектора осуществляется проверка неопределенностей.

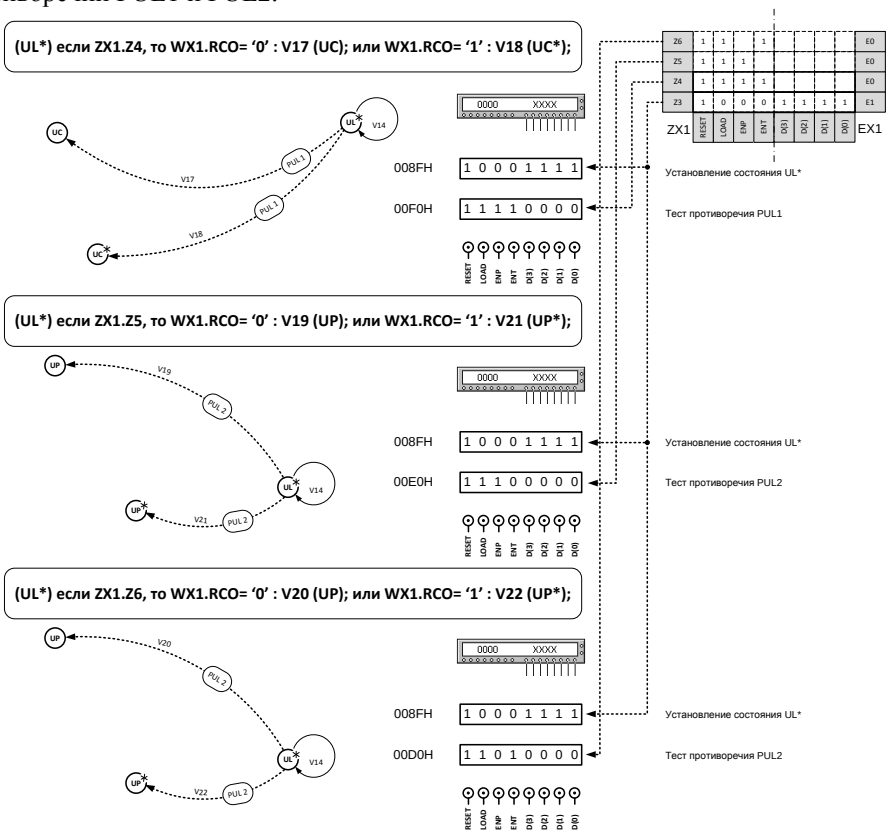

Рисунок 3 – Синтез теста для устранения противоречий PUL1 и PUL2

Шаг 5. Компьютерное моделирование и оценка результатов на противоречивых переходах.

На рис. 4 приведены результаты компьютерного моделирования. Следует учитывать, что реакция объекта исследования, выраженная значениями выходных сигналов на графике логического анализатора, «опаздывает» на один период синхронизации.

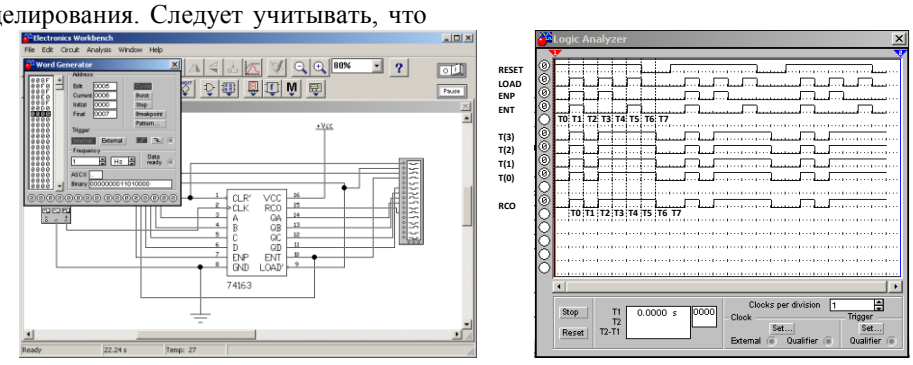

Рисунок 4 – Результат компьютерного моделирования

Таблица 2 – Результаты теста на устранение противоречий PUL1 и PUL2

Имя	T0	T1	T2	T3	T4	T5	T6	T7
RESET	1	1	1	1	1	1	0	0
LOAD	0	1	0	1	0	1	0	0
ENP	0	1	0	1	0	0	0	0
ENT	0	1	0	0	0	1	0	0
RCO	1	0	1	1	1	1	0	0

Таким образом, переходы V18, V19 и V20 являются недействительными. Противоречия PUL1 и PUL2 разрешены.

Шаг 6. Коррекция графа противоречий.

После удаления недействительных дуг получается новый граф противоречий GX2 (рис. 5).

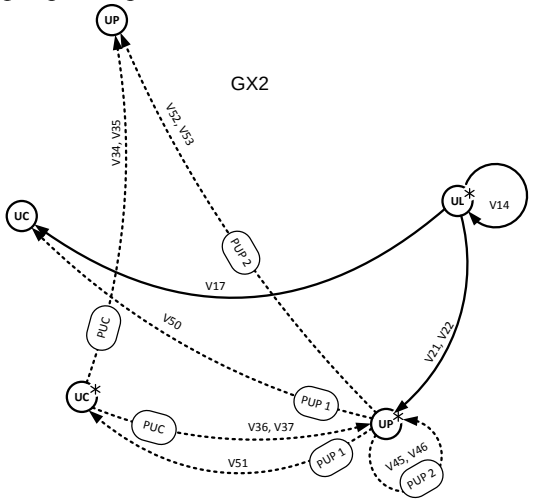

Рисунок 5 – Граф противоречий GX2

Последовательно выполняя шаги 4, 5, 6, осуществляется семиотически корректное формирование тестовых последовательностей для устранения противоречий PUC, PUP1 и PUP2. После удаления недействительных дуг граф противоречий GX не содержит ни одной противоречивой дуги. Можно приступать к моделированию работы микросхемы ID74163.

2 Алгоритм установления граф-подграф изоморфизма для графов помеченных по вершинам и ребрам

Основой предложенного в работе метода устранения семантических противоречий, возникающих при анализе результатов работы цифровых устройств, является алгоритм нахождения граф-подграф изоморфизма для графов помеченных по вершинам и ребрам.

Постановка задачи установления граф-подграф изоморфизма для помеченных графов. Пусть даны графы $G_1 = (V_1, E_1, L_1, R_1)$ и $G_2 = (V_2, E_2, L_2, R_2)$, где V – множество вершин графа, E – множество ребер графа, L – метки, приписанные вершинам графов и R – метки, приписанные ребрам графов. Граф G_1 изоморфен подграфу графа G_2 (обозначается, как $G_1 \cong S_2 \subseteq G_2$), если существует подстановка $\varphi : V_2 \to V_1$, такая, что для каждой пары вершин $v_i, v_j \in V_2$, если $(v_i, v_j) \in E_2$, то $(\varphi(v_i), \varphi(v_j)) \in E_1$ и для всех $l_i \in L_2$ выполняется $l_i \in L_2 = \varphi(l_i) \in L_1$ и для всех $r_i \in R_2$ выполняется $r_i \in R_2 = \varphi(r_i) \in R_1$.

Алгоритм установления граф-подграф изоморфизма для помеченных графов является развитием и продолжением алгоритма установления изоморфности [3].

Алгоритм установления изоморфизма удобно описывать в терминах поиска в пространстве состояний. Каждое состояние s процесса совмещения вершин соответствует частичной подстановке $\varphi(s)$, которая содержит лишь часть вершин полной подстановки. Каждому состоянию так же соответствуют подграфы $G_1(s)$ и $G_2(s)$, полученные из вершин графов G_1 и G_2, вошедших в частичную подстановку $\varphi(s)$, и ребер, соединяющих эти вершины. В дальнейшем обозначим через $\varphi_1(s)$ и $\varphi_2(s)$ проекции подстановки $\varphi(s)$ на V_1 и V_2.

Алгоритм состоит из предварительной и основной части. В предварительной части выполняются операции упорядочивания вершин графов и выполнения однократных, по ходу алгоритма, операций, призванных сократить область поиска основной, переборной части алгоритма.

3 Предварительная часть алгоритма

Основные действия, выполняемые в предварительной части алгоритма – сортировка вершин графов и формирование матрицы возможных совмещений.

Матрица возможных совмещений $M_{i,j}$ - это бинарная таблица размером $|V_1| \times |V_2|$. Каждому элементы таблицы соответствует пара вершин исходных графов $V_{1,i}$ и $V_{2,j}$. Значения матрицы формируются следующим образом:

$M_{i,j} = 0$, если на основании предварительных проверок вершины $V_{1,i}$ и $V_{2,j}$ совместить нельзя;

$M_{i,j} = 1$, в противном случае.

Смысл матрицы возможных совмещений в том, чтобы выполнить однократно в рамках предварительной части алгоритма все проверки, не основанные на информации, полученной в процессе совмещения вершин, тем самым, ускорить обработку соответствующих ограничений, сведя ее к одной операции сравнения.

В программе реализованы следующие предварительные проверки:

1. $M_{i,j} = 0$, если $|V_{1,i}| < |V_{2,j}|$, где $|V_{X,Y}|$ - степень вершины Y графа X;

2. $M_{i,j} = 0$, если $|V_{1,i}^{in}| < |V_{2,j}^{in}|$, где $|V_{X,Y}^{in}|$ - число входящих ребер вершины Y графа X;

3. $M_{i,j} = 0$, если $|V_{1,i}^{out}| < |V_{2,j}^{out}|$, где $|V_{X,Y}^{out}|$ - число исходящих ребер вершины Y графа X;

4. $M_{i,j} = 0$, если $W_{1,i}^{vertex} < W_{2,j}^{vertex}$, где $W_{X,Y}^{vertex}$ - число вершин в волновом разложении подграфа окружения вершины Y графа X;

5. $M_{i,j} = 0$, если $\sum_{l=1}^{k} W_{1,i,l}^{vertex} < \sum_{l=1}^{k} W_{2,j,l}^{vertex}, k = 1..|W_{2,j}^{vertex}|, l = 1..4$,

где $W_{X,Y,l}^{vertex}$ - число вершин в l-ой волне волнового разложения графа X, начиная с вершины Y;

6. $M_{i,j} = 0$, если $W_{1,i}^{ribes} < W_{2,j}^{ribes}$, где $W_{X,Y}^{ribes}$ - число ребер в волновом разложении подграфа окружения вершины Y графа X;

7. $M_{i,j} = 0$, если $\sum_{l=1}^{k} W_{1,i,l}^{ribes} < \sum_{l=1}^{k} W_{2,j,l}^{ribes}, k = 1..|W_{2,j}^{ribes}|, l = 1..4$,

где $W_{X,Y,l}^{ribes}$ - число ребер в l-ой волне волнового разложения графа X, начиная с вершины Y.

8. $M_{i,j} = 0$, если $L_{1,i} \neq L_{2,j}$, где $L_{X,Y}$ - метка вершины Y графа X.

Возможно использование и других критериев для оценки возможности совмещения вершин графов. Метод разработки таких критериев основан на волновом разложении графов, начиная с заданной вершины [4]. По мере распространения волны получаются подграфы окружения вершин. Сравнивая параметры соответствующих подграфов окружения вершин графов, которые предполагается совмещать, делается вывод о потенциальной возможности или принципиальной невозможности такого совмещения. В приведенных критериях для этого использовались сумма вершин и ребер в подграфах окружения сравниваемых вершин для всех этапов распространения волны.

Сортировка вершин графов производится с целью ускорения нахождения изоморфной подстановки, в случае, если такая подстановка существует. В переборной части алгоритма переставляются только вершины большего графа, в то время, как порядок вершин меньшего графа не меняется. Порядок следования вершин меньшего графа определяется в предварительной части алгоритма.

Пусть $T_{2,i}$ - количество ребер инцидентных вершинам с меньшими номерами и

$P_{2,i} = \sum_{j=1}^{|V_1|} M_{j,i}$ - суммарное количество вариантов совмещения вершины i графа G_2 с вершинами графа G_1. Тогда порядок сортировки вершин графа G_2 следующий:

$$V_{2,i} = V_{2,k}, \text{ где } T_{2,k} = \min_{j=i+1}^{|V_2|}(T_{2,j}).$$

Если $T_{2,i} = T_{2,j}$, то $V_{2,i} = V_{2,k}$, где $P_{2,k} = \min(P_{2,i}, P_{2,j})$.

Т.е. вершины графа G_2 сортируются в порядке убывания количества связей с вершинами имеющими меньшие номера, или в порядке убывания количества вариантов совмещения вершин, если количество связей одинаково. Такой порядок следования вершин обусловлен тем, что чем больше связей, с уже совмещенными имеет вершина, тем жестче будет ограничивающее условие, включающее эту вершину, и, соответственно, меньше общее количество совмещений, которые необходимо перебрать.

4 Основная часть алгоритма

Основная часть алгоритма представляет собой последовательное наложение вершин с возвратом, описывать которое удобно в терминах метода поиска в пространстве состояний.

Вершины графа G_2 остаются нетронутыми и каждой из них ставиться в соответствие одна из вершин графа G_1. При этом проверяется допустимость такого совмещения. Если удается найти соответствие всем вершинам графа G_2, при этом выполнено условие изоморфизма, то найденное состояние возвращается как искомая подстановка.

Пусть $T_{1,i}$ - количество связей вершины i графа G_1 с вершинами $V_{1,j} \in \varphi_1(s)$, а $T_{2,i}$ - количество связей вершины i графа G_2 с вершинами $V_{2,j} \in \varphi_2(s)$.

Начальному состоянию $\varphi(s)_0 = 0$ соответствует состояние, при котором не совмещено еще ни одной пары вершин.

Для получения i-го состояния для вершины $V_{2,i}$ ищется соответствие среди вершин $V_{1,j}$, таких что:

1. $M_{i,j} = 1$, т.е. вершины совместимы на основании предварительных проверок.

2. $T_{1,i} \geq T_{2,j}$

3. Для $k = 1..i$, если $(v_i, v_k) \in E_1$, то $(\varphi(v_i), \varphi(v_k)) \in E_2$.

4. Для $k = 1..i$, $(r_{1,i}, r_{1,k}) \in R_1 = (\phi(r_{2,i}), \phi(r_{2,k})) \in R_2$

Если выполнены все условия, из которых третье является прямым следствие определения граф-подграф изоморфизма, то соответствующая пара вершин входит в частичную подстановку и формируется новое состояние $\varphi(s)_i$.

Перебор состояний производится методом поиска в глубину.

5 Анализ производительности алгоритма для задачи декомпозиции семейства алгоритмов на элементарные

Для тестирования производительности алгоритма формировались случайные графы с использованием следующих параметров:

nv – Число вершин в большем из пары генерируемых графов;

ns – Число вершин в меньшем из генерируемых графов (подграфе);

p – Плотность реберного заполнения графов;

l – Количество различных меток, используемых для маркировки вершин графов;

e – Количество различных меток, используемых для маркировки ребер графов.

Граф семейства алгоритмов формировался как связный граф с числом вершин, определяемым параметром nv, и плотностью реберного заполнения p. Всем вершинам и ребрам графа ставились случайные метки из диапазона $[0, l)$ для меток вершин и $[0, e)$ для меток ребер соответственно.

Подграф формировался путем удаления части вершин из графа семейства алгоритмов. Начиная со случайной вершины, пускалась волна, проходящая по ребрам графа, для формирования связного подграфа. Как только на очередном шаге размер подграфа вошедшего в волновое разложение превышал параметр ns, все вершины, которые накрыла волна, выделялись в виде подграфа.

Далее приводятся результаты численного исследования производительности алгоритма на основании набора сгенерированных графов, описанного выше. Все вычисления производились на компьютере Athlon XP 1700+, 1 Gb RAM. Каждое значение формировалось как суммарное время поиска подграфа для 100 пар графов.

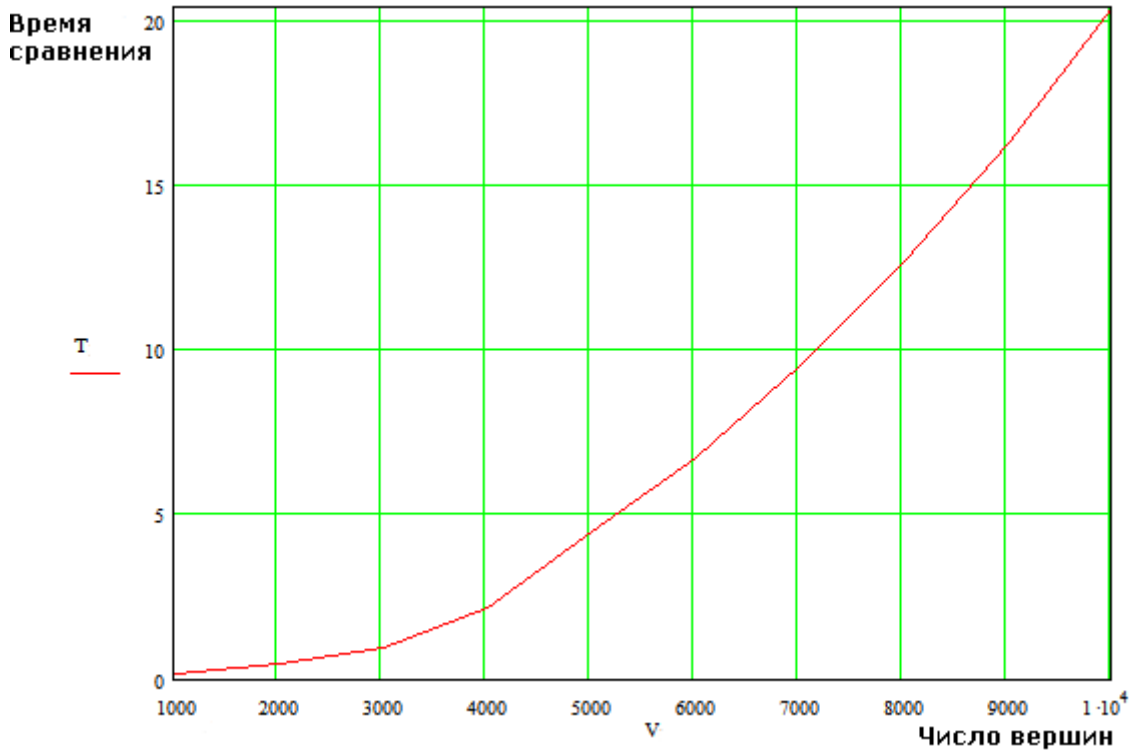

Рисунок 6 – Зависимость времени поиска подграфа от числа вершин в графе семейства алгоритмов (параметры nv=1000…10000, ns=200, p=0.1, l=100, e=5)

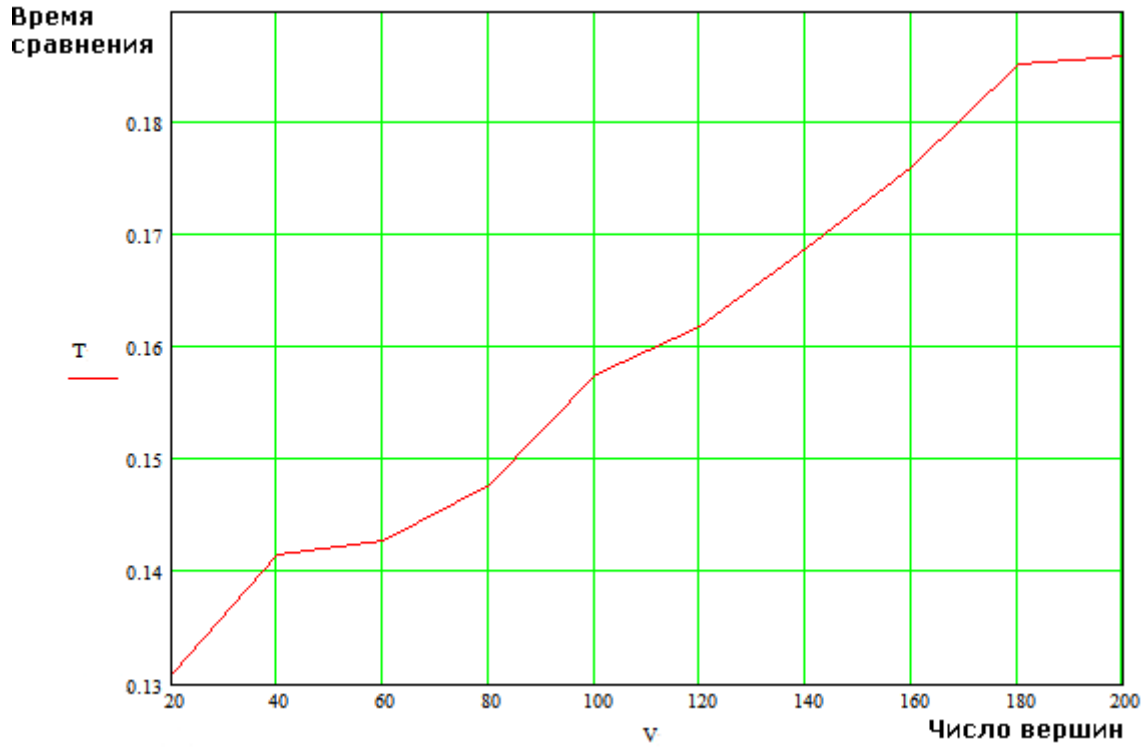

Рисунок 7 - Зависимость времени поиска подграфа от числа вершин в подграфе элементарного алгоритма (параметры nv=1000, ns=20…200, p=0.1, l=20, e=5)

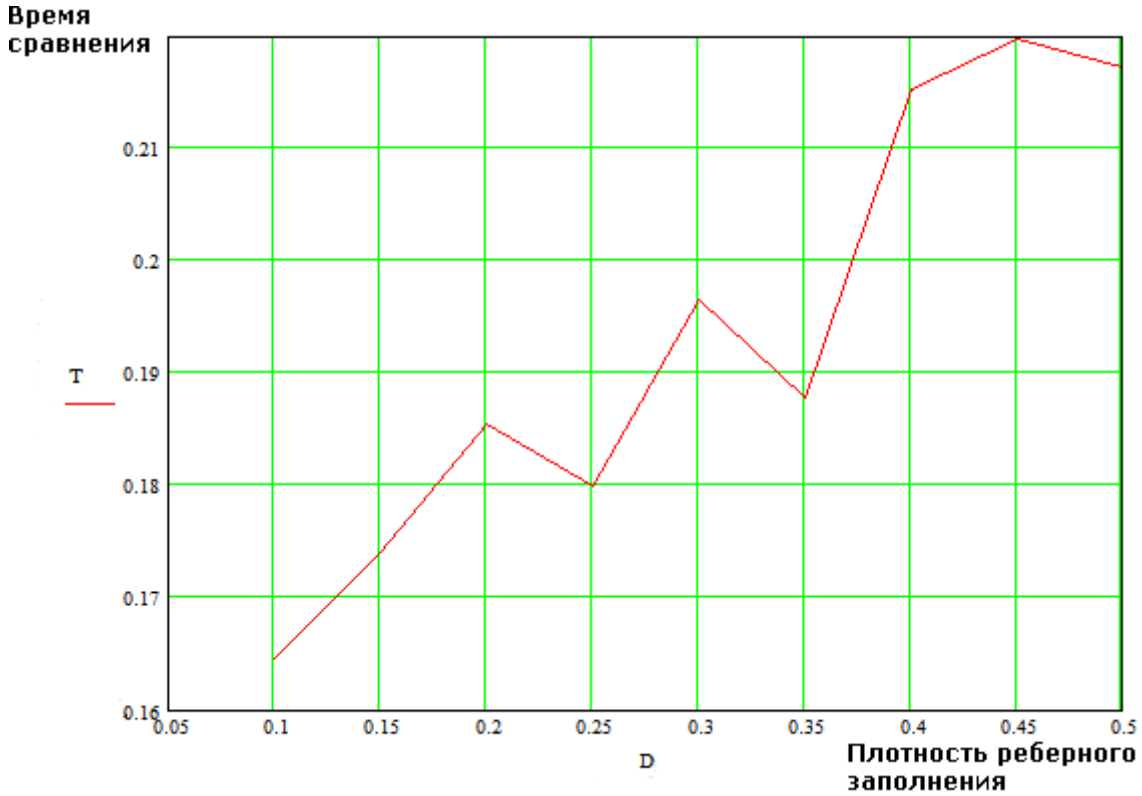

Рисунок 8 - Зависимость времени поиска подграфа от плотности реберного заполнения (параметры nv=1000, ns=200, p=0.1..0.5, l=20, e=5)

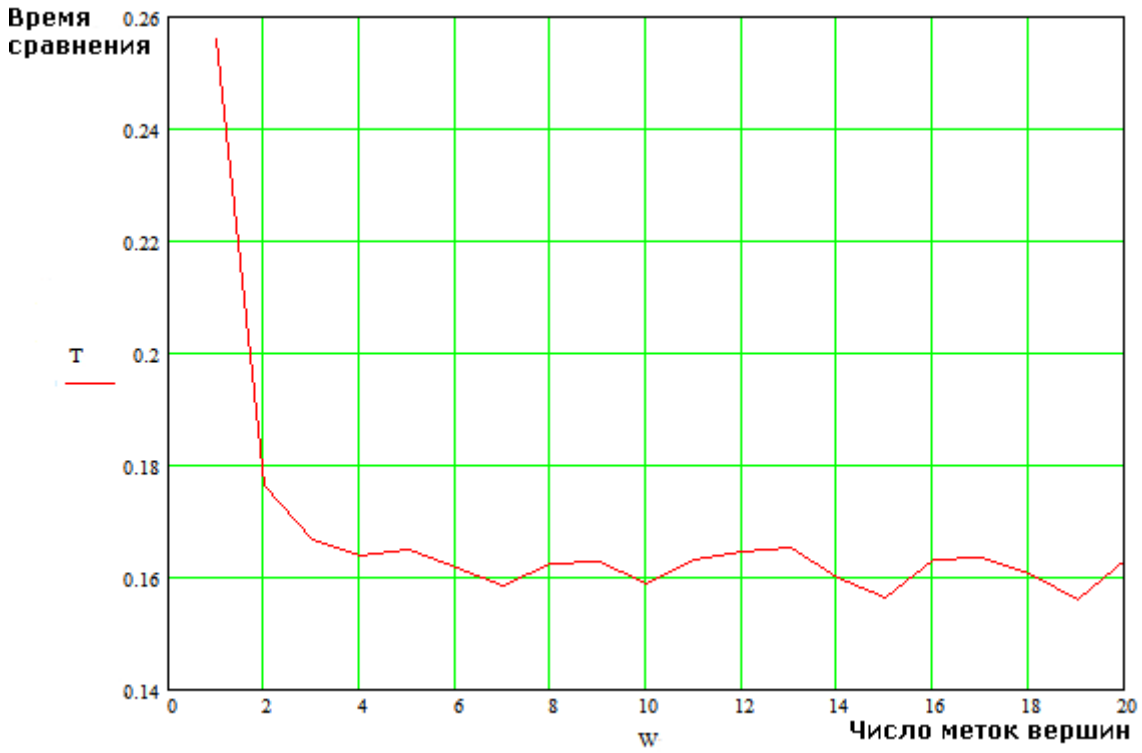

Рисунок 9 - Зависимость времени поиска подграфа от числа различных меток приписываемых вершинам графов (параметры nv=1000, ns=100, p=0.2, l=1..20, e=5)

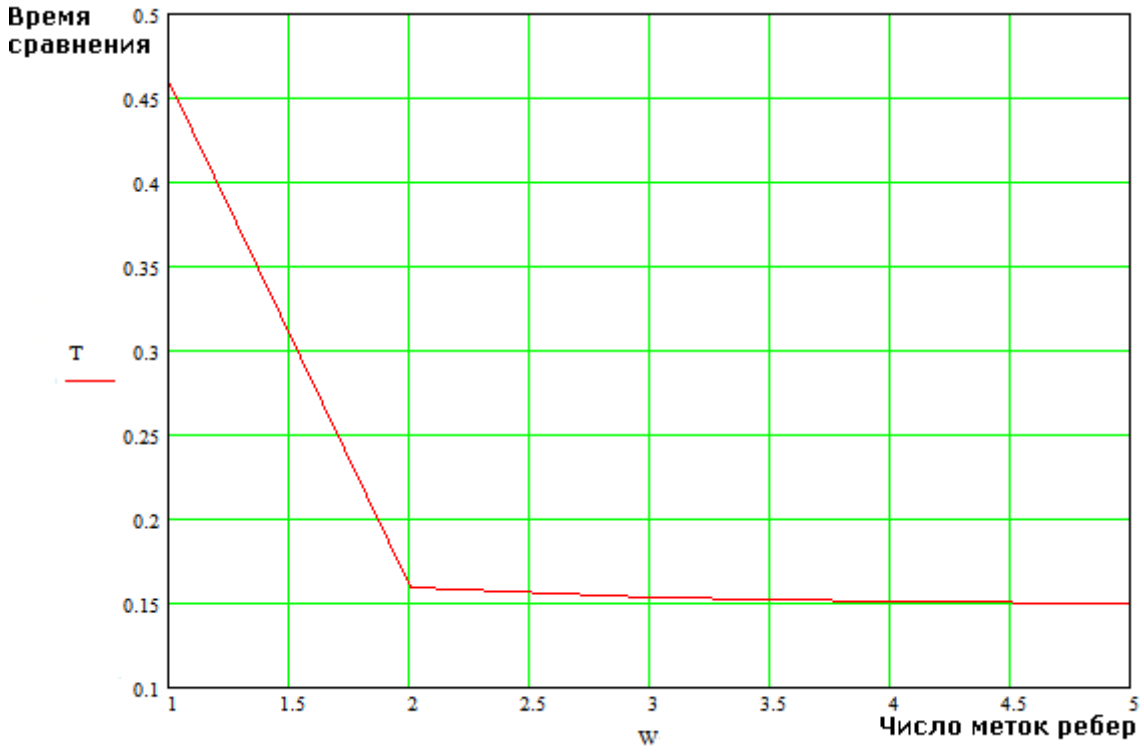

Рисунок 10 - Зависимость времени поиска подграфа от числа различных меток приписываемых ребрам графов (параметры nv=1000, ns=100, p=0.2, l=5, e=1..5)

Как следует из графика (рис. 6), для графов с числом вершин до 10000 включительно и размере графа элементарного алгоритма порядка 200 вершин, алгоритм, в среднем, тратит на поиск изоморфного подграфа не более 21 секунды машинного времени, что достаточно производительно для решения задачи на реальных данных, используемых при декомпозиции семейства алгоритмов. Графы такого размера характеризую современные пиковые требования к системам декомпозиции алгоритмов в реальных прикладных областях применения этого метода. Размер графа семейства алгоритма является наиболее важным параметром, влияющим на производительность алгоритма. При изменении его размера в 10 раз (с 1000 до 10000 вершин) время вычислений изменялось в 136 раз (с 0.15 до 20.44 секунды).

Как следует из графика (рис. 7), влияние размера искомого подграфа на общую производительность алгоритма является следующим параметром после размера графа семейства алгоритмов, влияющим на производительность алгоритма. При изменении размера подграфа в 10 раз (с 20 до 200 вершин) время вычислений изменилось в 1.4 раза (с 0.13 до 0.18 секунды). С учетом того, что большинство элементарных алгоритмов имеют размер, не превышающий 200 вершин, фактор размера подграфа не является решающим для производительности алгоритма.

Из результатов приведенных на рис. 8 следует, что при изменении плотности реберного заполнения графов с 10% до 50% время вычислений изменилось в 1.3 раза (с 0.16 до 0.21 секунды). Следовательно, как и фактор размера искомого подграфа, плотность реберного заполнения не столь значительно влияет на производительность алгоритма, однако с ростом плотности реберного заполнения производительность алгоритма все же незначительно падает.

Из приведенного графика (рис. 9) следует, что фактор изменения числа различных меток приписываемых вершинам графов (т.е. числа различных вычислительных узлов, применяемых при формировании конечного автомата) значительно влияет на производительность алгоритма в целом лишь при незначительном числе различных меток (1-4), а при большем числе различных меток приписываемых вершинам графов более важными для ограничения глубины рекурсивного просмотра дерева решения, и, следовательно, повышения производительности алгоритма, оказываются условия, основанные на структуре графов. Для исследуемого набора графов, с ростом числа различных меток, приписываемых вершинам графов, время вычислений

оставалось в достаточно узких пределах и колебалось около 0.16 секунды для одной пары графов.

Как и в случае с числом различных меток, приписываемых вершинам графов, с ростом числа различных меток, приписываемых ребрам графов, общая производительность алгоритма повышается, но с числа меток, влияние этого параметра на время работы алгоритма резко убывает (рис. 10). Так при введении 2-х меток ребер время работы изменилось с 0.45 до 0.16 секунды, а с увеличением числа меток до 5 время работы уменьшилось незначительно до 0.15 секунды. Из всех параметров, число меток приписываемых ребрам графов, является наименее влиятельным на общую производительность алгоритма.

Заключение

В работе представлен подход к решению задачи оптимизации аппаратных затрат на этапе проектировании конечных автоматов с программируемой процедурой, основанный на алгоритме поиска граф-подграф изоморфизма для помеченных графов. Приводится детальное описание алгоритма и результатов исследования его производительности для графов характерных для конечных автоматов с программируемой процедурой. Исследовано влияние различных параметров графов конечных автоматов на производительность алгоритма поиска подграфа.

Результаты исследования показали, что разработанный алгоритм обладает достаточной производительностью для применения его в реальных системах разработки и оптимизации конечных автоматов, как на современном этапе, так и в будущем, с ростом размерности решаемых задач.

Список литературы

1. Горбатов В.А. Фундаментальные основы дискретной математики. Информационная математика. – М.: Наука. Физматлит, 2000. – 544 с.
2. Квазигомоморфное преобразование гиперграфов в автоматизации проектирования устройств управления / А.А. Голдобин // Радиоэлектроника. Информатика. Управление. – 2006. – №1. – С. 41-48.
3. Ильяшенко М.Б. Разработка и исследование параллельного алгоритма проверки граф-подграф изоморфизма // Радиоэлектроника. Информатика. Управление. – 2006. – № 1. – С. 63-69.
4. Пинчук В.П. Основанная на волновом разложении система инвариантов для простых графов и алгоритм распознавания изоморфности / В.П. Пинчук. – К., 1995. – [Деп. в ГНТБ Украины 10.05.95]. – № 1002. – Ук95.

Надійшла до редакції 31.10.2012

М.Б. ІЛЬЯШЕНКО
Запорізький національний технічний університет

Вирішення задачі пошуку граф-підграф ізоморфізму для семантичного аналізу спеціалізованих цифрових систем

Запропоновано вдосконалений алгоритм пошуку граф-підграф ізоморфізму графів та результати дослідження його ефективності. Об'єктом дослідження є методика усунення протиріч, що існують при аналізі роботи спеціалізованих цифрових систем

Ключові слова: ізоморфізм, зважені графи, семантичний аналіз, спеціалізовані обчислювачі

M.B. ILYASHENKO
Zaporizhzhya National Technical University

Graph-Subgraph Isomorphism Problem Solving for Semantic Analysis of Specialized Digital Systems

We propose improved graph-subgraph isomorphism search algorithm and provide the results of its efficiency estimation. We studied the methods of solving conflicts in the analysis of specialized digital computers performance.

Keywords: texturizing, tiles, level of detail, texture computation