УДК 621.8:681.5

М.Э. КУССУЛЬ, А.А. ГАЛИНСКАЯ

РАЗРЕШЕННЫЕ И ЗАПРЕЩЕННЫЕ АРХИТЕКТУРЫ МОДУЛЬНЫХ СЕТЕЙ

***Abstract*:** *Formal definitions of allowed and prohibited architectures of modular neural networks are proposed in this article. Using definitions and properties of cycles in modular networks, algorithms of determination of run's order for modular architectures containing cycles were suggested. Proposed method is intended for determination of rules for automatic construction of modules' run order, and also for control of architectures created by CAD users.*
Key words: *modular neural networks, graphs of neural networks, cycles, queues.*

***Анотація*:** *У даній статті запропоновані формальні визначення дозволених і заборонених архітектур модульних нейронних мереж. Завдяки використанню визначень і властивостей циклів у модульних мережах були запропоновані алгоритми визначення порядку перерахування для модульних архітектур, що містять цикли. Запропонована методика призначена для виводу правил автоматичної побудови послідовностей перерахунку модулів, а також для контролю архітектур, створених користувачем у САПР.*
Ключові слова: *модульні нейронні мережі, графи нейронних мереж, цикли, черги.*

***Аннотация*:** *В данной статье предложены формальные определения разрешенных и запрещенных архитектур модульных нейронных сетей. Благодаря использованию определений и свойств циклов в модульных сетях были предложены алгоритмы определения порядка пересчета для модульных архитектур, содержащих циклы. Предложенная методика предназначена для вывода правил автоматического построения последовательностей счета модулей, а также для контроля архитектур, созданных пользователем в САПР.*
Ключевые слова: *модульные нейронные сети, графы нейронных сетей, циклы, очереди.*

1. Введение

Модульный подход хорошо известен в различных областях науки и техники. В последнее время растет число задач, решаемых при помощи модульной организации нейронных сетей. Основными преимуществами использования модульных сетей являются: возможность разделения сложной задачи на подзадачи и выбор подходящих алгоритмов для решения подзадач; уменьшение сложности результирующей сети; снижение времени обучения сети и др. Преимущества модульных нейронных сетей обусловили как их широкое применение, так и разнообразие архитектур и способов их использования [1].

Возросший интерес к модульным сетям повлек за собой необходимость в создании различных средств их моделирования [2, 3], что требует решения некоторых общих вопросов применения модульных сетей. В системах автоматического проектирования модульных сетей, таких, как САПР МНС [3], где пользователю предоставляется возможность конструировать произвольную модульную архитектуру, необходимым является наличие средств проверки полученной сети на непротиворечивостью. Необходимы и средства автоматического определения последовательности счета модулей, для тех архитектур, где порядок может быть автоматически определен однозначно, а также интерактивные средства разрешения противоречий, где это сделать невозможно.

Для решения указанных задач требуются средства описания модульных сетей. В качестве одного из таких средств в работах [4, 5] было предложено использовать направленные графы. Предложенная модель позволяет описать модульные сети общего вида, в частности, сети, содержащие циклы. В работе [5] проанализированы свойства различных типов циклов в модульных сетях. Свойства циклов определяют возможности автоматического счета и обучения модульной сети без явного указания порядка работы модулей.

В данной статье на основе предложенной модели определяются разрешенные архитектуры модульных сетей, для которых возможно автоматическое определение счета сети, и запрещенные, для которых это сделать нельзя. Предлагается методика определения порядка счета. Второй раздел содержит определения графов модульных нейронных сетей и основных типов циклов. В третьем разделе рассматриваются разрешенные и запрещенные архитектуры.

2. Графы модульных сетей

Как показано в [4], модульная сеть может быть описана с помощью ориентированного графа $G(V,E)$, вершины $v \in V$ которого соответствуют модулям сети, а ребра $e \in E$ – связям между модулями. Путь и элементарный путь из вершины v_1 в вершину v_2 будем обозначать $w(v_1,v_2)$, при этом элементарный путь будем называть цепью.

Для представления в виде графов все входы модульной сети считаем модулем входов, а все выходы – модулем выходов сети. Модули входов и выходов виртуальны, поскольку не реализуют алгоритмов обработки данных. Входом или истоком (source) графа G будем называть вершину I, соответствующую виртуальному модулю входов сети. Вход сети определяется следующим условием: $s_{in}(I) = 0$, где $s_{in}(\cdot)$ – число входящих в вершину ребер. Выходом или стоком (sink) графа будем называть вершину O, соответствующую виртуальному модулю выходов. Для него справедливо утверждение $s_{out}(O) = 0$, где $s_{out}(\cdot)$ – число выходящих из вершины ребер.

Граф G модульной сети удовлетворяет следующим условиям [4]:

1) граф G – слабосвязанный, то есть неориентированный граф, соответствующий G – связному;

2) граф G не имеет кратных ребер;

3) в графе G существует единственная вершина-вход $I \in V$, и любая вершина достижима из входа, то есть $\forall v \in V : v \neq I \Rightarrow \exists\, w(I,v)$. Следовательно, $\forall v \in V : v \neq I \Rightarrow s_{in}(v) \geq 1$;

4) в графе G существует единственная вершина выход $O \in V$, и выход достижим из любой вершины, то есть $\forall v \in V : v \neq O \Rightarrow \exists\, w(v,O)$. Следовательно, $\forall v \in V : v \neq O \Rightarrow s_{out}(v) \geq 1$.

В отличие от принятого в теории графов определения путь w из вершины a в вершину b есть множество вершин и соединяющих их ребер, не включая конечную вершину. Аналогично, цепь – это элементарный путь (без самопересечений), не включающий конечную вершину.

Определение 1. Проекцией $P(b,a)$ (projection) вершины (модуля) $a \in V$ на вершину (модуль) $b \in V$ будем называть такой ориентированный подграф графа $G(V,E)$, который включает все возможные цепи $w(b,a)$ из b в a

$$P(b,a) = \bigcup w(b,a); \quad \forall w(b,a) = \{v_1, v_2, \ldots, v_n\} : v_1 = b,\ v_{n+1} = a; \quad v_i \neq v_j; \quad \forall i, j = \overline{1,n},$$

то есть в подграф проекции входят все вершины и соответствующие ребра всех цепей из b в a, кроме вершины a. Проекция произвольного модуля a на входы модульной сети I, записывается как $P(I,a)$.

Определение 2. Если каждой вершине $v \in V$ поставить в соответствие параметр $t(v)$ такой, что $t(v)=1$ в том случае, если выходы соответствующего модуля сети на данном шаге еще не вычислены, и равен нулю, если модуль уже посчитан, то неопределенностью (uncertainty) вершины a называется сумма

$$U(a) = \sum_{v \in P(I,a)} t(v),$$

при этом неопределенность модуля входов $U(I)$ всегда равна нулю.

Аксиома 1. Из двух модулей сети, при прочих равных условиях и если порядок счета модулей не определен специальными требованиями, первым должен быть посчитан модуль с меньшей неопределенностью.

Определение 3. Циклом с начальной вершиной a будем называть такой ориентированный подграф $C_a(V_a,E_a)$ графа модульной сети $G(V,E)$, который включает все возможные пути $w(a,a)$ из вершины a в саму себя:

$$C_a(V_a,E_a) = \bigcup w(a,a), \quad \forall w(a,a) = \{v_1,v_2,\ldots,v_n\} : v_1 = a, v_{n+1} = a.$$

В отличие от проекций, в определении циклов используются пути, а не цепи. Циклы, по определению 3, соответствуют сильносвязанным компонентам в орграфе [6].

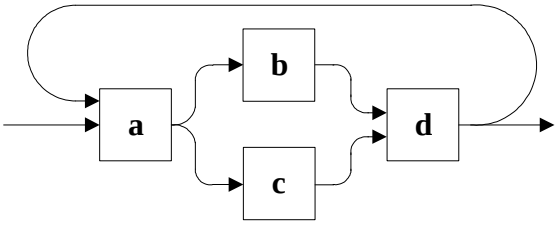


Рисунок 1. Схема простого цикла

Необходимость определения цикла как набора всех путей иллюстрируется архитектурой, представленной на рис. 2. Очевидно, что при пересчете модулей, входящих в цикл $C_a : V_a = \{a,b,c,d\}$, прежде чем считать модуль d, требуется посчитать оба модуля b и c. Такой же порядок счета требует и аксиома 1.

Из определения цикла следует, что виртуальные модули (входы и выходы) никогда не входят ни в один цикл. Согласно условиям, накладываемым на граф модульной сети, любой цикл обязательно имеет как входящие, так и выходящие внешние ребра. То есть такие ребра, которые связывают вершины цикла с вершинами, не принадлежащими этому циклу.

Дадим формальное определение особых вершин, которые будут использоваться в дальнейшем. Для этого воспользуемся понятием длины пути $d(w)$, которое определяется как число ребер в этом пути.

Определение 4. Начальной вершиной f (first) цикла $C(V,E)$ является вершина, для которой $d_f = \min\limits_{v\in V}\ \min\limits_{w=w(I,v)} d(w)$. Конечной вершиной l (last) является вершина, для которой $d(w(l,f))=1$.

Данное определение не гарантирует единственность каждой из вершин, однако позволяет провести дальнейшую классификацию циклов. Приведем определения для двух типов циклов, принципиально важных с точки зрения счета модульных сетей [4].

Определение 5. Обычным или обыкновенным циклом (ordinary cycle) с начальной вершиной a будем называть такой цикл $Co_a(V_a,E_a)$, для которого не существует ни одной пары вершин, таких, что они принадлежат проекциям друг друга на вход, то есть $\forall\, v_1,v_2 \in V_a : v_1 \in P(I,v_2) \ \Rightarrow\ v_2 \notin P(I,v_1)$. Перекрестными циклами (crossed cycle) будем называть циклы $Cc_b(V_b,E_b)$, для которых данное условие не выполняется.

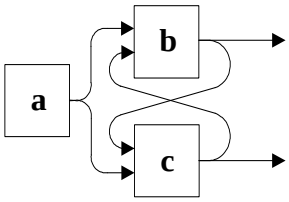


Рис. 2. Триггерная схема соединения модулей

Определение обыкновенных циклов необходимо для установления порядка пересчета модулей в сети и, в первую очередь, чтобы отделить все возможные варианты триггерных схем (рис. 2).

Обыкновенный цикл может быть представлен только единственным образом, в отличие от циклов общего вида, которые тождественны с точностью до выбора начальной вершины. Перекрестные циклы этим свойством не обладают.

Простым циклом мы называем такой обыкновенный цикл, для всех вершин которого любой замкнутый путь проходит через начальную вершину этого цикла (рис. 1). То есть цикл $Co_f(V_f,E_f)$ простой, если $\forall\, w(v,v) : v \in V_f \ \Rightarrow\ f \in w(v,v)$ (определение 6 в [4]).

Приведем несколько теорем, имеющих отношение к обсуждаемым вопросам. Подробно свойства циклов и доказательства приведенных теорем рассматриваются в статье [5].

Теорема 1. Для любых двух вершин a и b цикла $C_f(V_f,E_f)$ всегда существует проекция $P(a,b)$, и эта проекция целиком принадлежит циклу $\forall a,b \in V_f : \exists\, P(a,b) \subseteq C_f$.

Теорема 2. (Признак существования цикла). Для того чтобы граф модульной сети содержал цикл (циклы), достаточно, чтобы граф содержал хотя бы одну такую вершину, что для нее существует хотя бы одно входящее ребро, не принадлежащее проекции этой вершины на входы.

Теорема 3. (Теорема единственности обыкновенного цикла). Если обыкновенный цикл существует, то начальная вершина этого цикла может быть выбрана только единственным образом.

Следствие 3.1. Если обыкновенный цикл Co_f существует ($V_f \neq 0$), то на данном наборе вершин он определяется единственным образом.

Теорема 4. В обыкновенном цикле существует только одна вершина, связанная входящими ребрами с вершинами вне цикла, и эта вершина начальная.

Следствие 4.1. В обыкновенном цикле проекция начальной вершины на входы не содержит ни одной вершины цикла.

Следствие 4.2. Для любой вершины $v \neq f$ обыкновенного цикла C_f все цепи от входов содержат начальную вершину, то есть $P(I,v) = P(I,f) \cup P(f,v) \quad \forall v \in C_f$.

Используя свойства циклов, можно предложить формальные определения разрешенных и запрещенных архитектур, а также способы разрешения противоречий в случае триггерных циклов.

3. Разрешенные и запрещенные архитектуры модульных сетей

Обработка данных с помощью модульной нейронной сети подразумевает, что последовательность счета модулей известна. Приведенные выше определения и формулировки предназначены для вывода правил автоматического построения последовательностей счета модулей, а также для контроля архитектур, созданных пользователем в САПР. Соответственно, необходимо определить, какие архитектуры модульных сетей являются разрешенными без введения дополнительных условий.

Определение 6. Очередью $Q(G)$ будем называть такую нумерацию графа $G(V,E)$, то есть отображение $Q: G \to [1...N]$, где N – число вершин графа (модулей сети), что

$$\forall a,b \in V : \exists\ w(a,b)\ \&\ Q(a) < Q(b) \quad \Rightarrow \quad U(a) \leq U(b). \qquad (1)$$

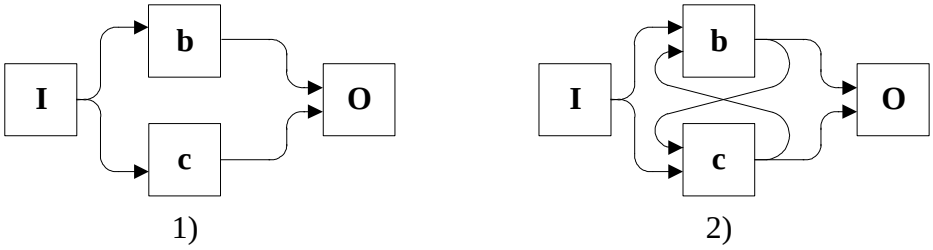
Очередь определяет возможный порядок счета модулей сети в соответствии с аксиомой 1. Условие (1) будет гарантированно выполняться, если использовать следующую тривиальную процедуру построения очереди:

1) поставить в очередь модуль с нулевой неопределенностью;

2) рассматривая поставленный в очередь модуль как уже посчитанный, вычислить неопределенность для модулей еще не поставленных в очередь;

3) если есть модули с нулевой неопределенностью, вернуться к п.1.

Определение 7. Разрешенной будем называть такую очередь, которую можно построить с помощью тривиальной процедуры, а разрешенными архитектурами модульных сетей будем называть такие архитектуры, для которых можно построить разрешенную очередь.



1) 2)

Рисунок 3. Пример модулей с одинаковой неопределенностью

Рассмотрим пример, представленный на рис. 3.1 и 3.2. На обоих рисунках модули a и b имеют одинаковые степени неопределенности (0 и 1 соответственно), но для первого рисунка очереди $Q_1 = \{a,b\}$ и $Q_2 = \{b,a\}$ являются эквивалентными, тогда как для второго нет, то есть

результат вычислений будет разным. Поскольку для сети 3.2 невозможно построить разрешенную очередь, такая архитектура не будет разрешенной, согласно определению 7.

Определим условия, при которых архитектура сети является разрешенной. Сначала рассмотрим модульную сеть без циклов.

Утверждение 1. Любая архитектура сети без циклов является разрешенной.

Доказательство. В качестве доказательства приведем алгоритм построения разрешенной очереди и подсчитаем количество эквивалентных очередей.

В сети без циклов вершины v можно естественным образом распределить по «уровням», используя длину d_v наиболее длинного пути из входной вершины I в v. Вершины, имеющие одинаковое значение d_v, образуют один уровень сети. Для вершин первого уровня $U(v^1)=0$, так как они связаны напрямую с входом сети $(d_v=1)$. Поскольку любая вершина достижима из входов и граф не содержит циклов, то всегда существует хотя бы одна вершина на каждом уровне.

Пусть сеть содержит K уровней и первый уровень содержит M_1 вершин $\{v_1^1, v_2^1, \ldots, v_{M_1}^1 / d_v = 1\}$, второй – M_2 вершин $\{v_1^2, v_2^2, \ldots, v_{M_2}^2 / d_v = 2\}$ и т.д.

Построим очередь $Q = \{v_1^1, \ldots, v_{M_1}^1, v_1^2, \ldots, v_{M_2}^2, \ldots, v_{M_K}^K\}$. Очевидно, что такая очередь может быть построена с помощью предложенной тривиальной процедуры и будет удовлетворять условию (1), так как при переходе на следующий уровень сети все входы для вершин данного уровня определены и, следовательно, $U(v^k) < U(v^n)\ \forall n > k$.

Для сетей без циклов очередь по данному алгоритму может быть построена $\prod_{j=1}^{K} M_j!$ способами, так как результат пересчета сети не зависит от порядка постановки в очередь модулей одного уровня.

Предложенный алгоритм аналогичен алгоритму поиска в ширину для графов [7]. Порядок пересчета вершин (модулей) совпадает с порядком обнаружения вершин при поиске в ширину, начинающегося от входа и примененного для графа модульной сети, из которого удалены ребра, пересекающие несколько уровней.

По аналогии с алгоритмами поиска в теории графов можно предложить алгоритм пересчета сети «в глубину». Как и для предыдущего алгоритма, пересчет начинается с модуля первого уровня, имеющего входящее ребро только от входа сети.

1. Выбираем вершину первого уровня.

2. Просматриваем список вершин (модулей), связанных входящими ребрами с только что посчитанным.

3. Если в списке находится вершина v, связанная только с уже пройденными вершинами (то есть $U(v)=0$), то считаем соответствующий ей модуль. Если нет, то переходим к следующей вершине.

По построению алгоритма очевидно, что такая очередь будет разрешенной.

3.1. Очереди для сетей с циклами

Рассмотрим модульную сеть, содержащую циклы. Как показано в [3], модульная сеть может содержать обыкновенные и перекрестные циклы, при этом ни одна вершина одного цикла не принадлежит никакому другому циклу. Тогда любой цикл можем заменить одной метавершиной. Будем обозначать метавершины как Y, а граф, полученный путем замены всех циклов метавершинами, – G'.

Метавершина Y, замещающая цикл $C(V,E)$, будет связана входящими ребрами со всеми вершинами, которые имеют связи с вершинами цикла:

$$\exists\, e(p_j,\mathrm{Y}) \in G' \Leftrightarrow \exists\, e(p_j,v),\ v \in V \quad j = \overline{1, s_{in}(\mathrm{Y})};$$
$$s_{in}(\mathrm{Y}) = |\Omega| \quad \Omega = \{e(p,v) : v \in V \ \&\ p \notin V\}.$$

Аналогично, метавершина будет связана со всеми вершинами, к которым идут выходящие ребра от вершин цикла:

$$\exists\, e(\mathrm{Y}, p_j) \in G' \Leftrightarrow \exists\, e(v, p_j),\ v \in V \quad j = \overline{1, s_{out}(\mathrm{Y})};$$
$$s_{out}(\mathrm{Y}) = |\Omega| \quad \Omega = \{e(v,p) : v \in V \ \&\ p \notin V\}.$$

Граф G' уже не содержит циклов и к нему может быть применен один из алгоритмов построения очереди, описанных выше. С точки зрения построения очередей, метавершине соответствует не единственный номер, а подочередь. Для сети, содержащей один цикл, построенная очередь будет иметь вид $Q = \{v_1,...,v_{j-1},Q(Y),v_{j+n},...,v_N\}$, где n – количество модулей, входящих в цикл. Очередность пересчета вершин внутри цикла $Q(Y)$ определяется в зависимости от типа цикла.

Представление циклов в виде метавершин является очень удобным средством для разработки систем автоматического проектирования модульных нейронных сетей. В первую очередь, выделение в сети метавершин как отдельных элементов позволяет ввести параметры, относящиеся к циклу в целом. Например, для итерационных циклов таким параметром будет число итераций.

Согласно определению 7, архитектура модульной сети, содержащей циклы, будет разрешенной, если все подочереди (подграфы) циклов этой сети являются разрешенными.

Любой цикл представляет собой подграф, полностью удовлетворяющий требованиям к графу модульной сети G, если все внешние ребра, входящие в вершины этого подграфа считать выходящими из одной виртуальной вершины I' – «входы», а все внешние выходящие ребра вершин подграфа считать входящими в одну виртуальную вершину O' – «выходы». При этом подграф цикла будем обозначать так же, как и метавершину, – Y.

Таким образом, для построения очередей любая метавершина может быть рассмотрена независимо как отдельная модульная сеть. Но, кроме того, чтобы повторно не рассматривать сам цикл, образующий метавершину, необходимо «разорвать» все обратные ребра цикла $e_k(l_k, f)$. То есть такие ребра, которые соединяют конечные вершины данного цикла с его начальной

вершиной. Каждое такое ребро заменяется ребром, идущим от конечной вершины цикла l_k к выходам O': $\forall\, e(l_k, f) \in G \;\Rightarrow\; \exists\, e(l_k, O') \in Y$.

Для вновь полученной модульной сети можно повторить анализ на наличие циклов и определить, возможна ли разрешенная очередь для этой сети. Такое последовательное раскрытие циклов продолжается до тех пор, пока вновь полученная модульная сеть не окажется ациклической.

Вообще говоря, можно предположить, что найдутся проблемы, для решения которых архитектуры модульных сетей будут содержать такие циклы, для которых последовательность счета модулей невозможно определить однозначно без прямого указания пользователя. Рассмотрим типы циклов, для которых можно предложить автоматическое построение очередей счета.

3.2. Разрешенные, условно разрешенные и запрещенные циклы

Сначала отметим, что для цикла, в котором невозможно выделить единственную начальную вершину, удовлетворяющую определению 4, разрешенной очереди не существует и архитектура модульной сети, содержащая такой цикл, не является разрешенной по определению 7.

Этот вывод достаточно очевиден, поскольку при «триггерном» соединении результат счета будет зависеть от того, который из модулей был посчитан раньше или оба модуля считались одновременно. Следовательно, для триггерного соединения модулей невозможно автоматически определить последовательность счета.

С другой стороны, метод замещения циклов метавершинами и предложенная выше процедура построения очереди позволяет получить очередь для перекрестных циклов, у которых начальная вершина определяется однозначно.

Покажем, что *разрешенными* будут такие архитектуры модульных сетей, которые либо не содержат циклов, либо содержат только обыкновенные циклы. Для таких архитектур можно предложить «естественную» последовательность счета модулей. Архитектуры, содержащие триггерные циклы, очевидно, следует считать *запрещенными*. Соответственно, *условно разрешенными* будем называть все остальные перекрестные циклы.

Рассмотрим модульные сети, содержащие обыкновенные циклы.

Теорема 5. (Теорема о разрешенных архитектурах). Для того, чтобы архитектура модульной сети была разрешенной, необходимо и достаточно, чтобы граф сети содержал только обыкновенные циклы.

Доказательство. Согласно теореме 4, только начальная вершина f обыкновенного цикла $Co_f(V,E)$ связана входящими ребрами с вершинами вне цикла, и, по следствию теоремы, ее проекция на входы не содержит ни одной вершины цикла. Отсюда следует, что любая вершина обыкновенного цикла имеет большую неопределенность, чем начальная, и, используя предложенную процедуру построения очереди, начальная вершина обыкновенного цикла обязательно будет поставлена в очередь.

Обозначим обыкновенный цикл метавершиной и преобразуем его в граф, замещающий эту метавершину $Y_f(V,E')$, с новыми виртуальными модулями входов I' и выходов O'. Для этого,

как предложено выше, исключим из рассмотрения все обратные ребра цикла. Из доказательства теоремы 4 и по способу построения графа метавершины очевидно, что для любой вершины в Y_f ее проекция на входы содержит вершину f : $\forall\, v \in V \quad \Rightarrow \quad f \in P(I', v)$.

Следовательно, если исходный цикл не содержал ни одной пары вершин, входящих в проекции на входы друг друга (по определению обыкновенного цикла), то и граф Y не содержит таких вершин. Это значит, что если граф Y содержит циклы, то эти циклы являются обыкновенными.

Согласно теореме 3, начальные вершины обыкновенных циклов определяются однозначно. Тогда при повторении процедуры замещения циклов метавершинами мы всегда будем получать графы сетей, содержащих только обыкновенные циклы. Процедура повторяется до тех пор, пока встретившийся цикл не оказывается простым.

По определению, для любой вершины простого цикла путь из самой в себя проходит через начальную вершину. Следовательно, после удаления обратных ребер замещающий метавершину граф представляет собой сеть без циклов и, согласно утверждению 1, допускает построение разрешенной очереди. Для сети, содержащей простой цикл, построенная очередь будет иметь вид: $Q(Co_f) = \{ v_1, ..., v_{j-1}, Q^1(Y), \ldots, Q^k(Y), v_{j+n}, ..., v_N \}$, где n – количество вершин цикла, а k – количество итераций, если цикл итерационный. Поскольку существует однозначная процедура построения разрешенной очереди для модульной сети, содержащей обыкновенные циклы, следовательно, достаточность условия доказана.

Необходимое условие следует из определений обыкновенного и перекрестного циклов, а также разрешенной очереди. Поскольку в перекрестном цикле существуют вершины, содержащие друг друга в проекциях на входы, то, по крайней мере, эти вершины ни на каком шаге тривиальной процедуры построения очереди не будут иметь нулевую неопределенность. То есть для тривиальной процедуры всегда

$$\forall\, a,b: \quad a \in P(I,b) \,\&\, b \in P(I,a) \quad \Rightarrow \quad 0 < U(a) \,\&\, 0 < U(b),$$

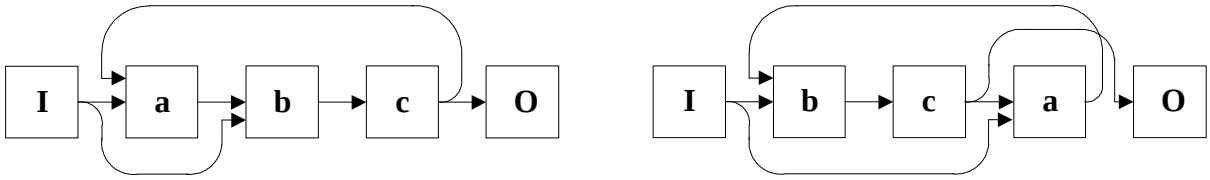
поскольку за один шаг в очередь может быть добавлена только одна вершина и только с нулевой неопределенностью.

Следовательно, для того чтобы архитектура модульной сети была разрешенной, необходимо, чтобы в ней отсутствовали перекрестные циклы. Чтобы архитектура была разрешенной для сетей с циклами, необходимо, чтобы все циклы сети были обыкновенными.

Рассмотрим перекрестные циклы. В общем случае, сети, содержащие перекрестные циклы, требуют построения очереди счета пользователем. Тем не менее для наглядности и автоматизации построения очередей в системах автоматического проектирования желательно иметь эвристику, позволяющую непосредственно по заданной архитектуре построить очередь счета модулей. В сетях с перекрестными циклами такие эвристики носят искусственный характер, однако оказываются полезны для работы в интерактивных средах, включающих САПР.

На рис. 4 показаны два варианта схемы, представляющие один и тот же перекрестный цикл. Одной из возможных эвристик постановки модулей в очередь счета является выбор в качестве начальной такой вершины, которая после автоматической нумерации модулей имеет

наименьший номер среди вершин, удовлетворяющих определению 4. Обычно в САПР принят принцип автоматической нумерации модулей «сверху - вниз» и «слева - направо». Используя такую эвристику для схем, представленных на рис. 4, получим следующие очереди счета: 1) $Q=\{a,b,c\}$ и 2) $Q=\{b,c,a\}$.



1) 2)

Рисунок 4. Варианты графического представления сети с триггерным циклом

Недостатком приведенной эвристики является то, что при построении архитектур в САПР разработчики руководствуются принципами, которые обычно не связаны с последовательностью работы алгоритмов. Например, минимизация длины связей, эстетика или наглядность получаемой схемы и т.п.

Для сохранения наглядности представления и обеспечения произвола расположения модулей на схеме можно использовать альтернативные эвристики, позволяющие автоматически строить очереди и для перекрестных циклов. Одной из таких эвристик является предположение о том, что если начальная вершина определяется однозначно, то она ставится в очередь первой, и далее применяется метод замещения цикла метавершиной с последующим раскрытием циклов.

Предложенные выше методы применимы для этой группы перекрестных циклов, и такие циклы можно назвать условно разрешенными. Они разрешены условно, поскольку при создании автоматической системы проектирования модульных сетей обязательно должен быть предусмотрен механизм прямого определения пользователем порядка счета модулей в цикле.

Такая эвристика не работает для циклов, в которых невозможно выделить одну начальную вершину по определению 4. То есть для запрещенных циклов мы не имеем средств автоматического построения очереди. Однако можно предложить «естественный» и наглядный способ явного указания единственной начальной вершины цикла.

Пусть САПР содержит модуль, не реализующий никакого алгоритма обработки информации, а предназначенный исключительно для наглядности схематического изображения архитектуры и определения порядка счета модулей. Будем обозначать такой модуль как Dl (delay). Тогда постановка требуемого количества таких модулей в проекции на входы позволяет точно и наглядно задать начальную вершину цикла.

На рис. 5 показан исходный триггерный цикл (рис. 4) и тот же цикл, где графически указан порядок счета модулей. Вершина Dl не реализует алгоритма обработки информации, а служит исключительно для указания последовательности счета. Очевидно, что в таком случае первым будет применен алгоритм, реализованный в модуле a.

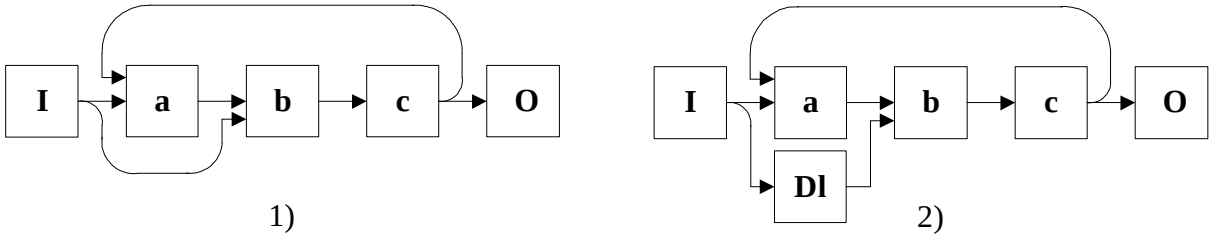


Рисунок 5. Графическое задание порядка счета модулей

4. Заключение

В данной статье предложены формальные определения разрешенных и запрещенных архитектур модульных нейронных сетей. Благодаря использованию предложенных определений и свойств циклов, рассмотренных в работах [4, 5], разработаны алгоритмы построения очередей для модульных сетей с циклами. Предложена методика замены всех циклов сети метавершинами, что позволяет единообразно работать с ацикличными модульными архитектурами и сетями, содержащими циклы разных типов.

Представление циклов в виде метавершин является очень удобным средством для разработки САПР систем при проектировании модульных сетей. Выделение в сети метавершин как отдельных элементов дает удобный способ введения параметров, относящихся к циклу в целом. Также введение метавершин позволило предложить эвристику для построения очередей для условно запрещенных архитектур и способ разрешения противоречий для запрещенных архитектур.

В последующих публикациях будут приведены алгоритмы анализа архитектур модульных сетей, основанные на свойствах предложенных графов.

СПИСОК ЛИТЕРАТУРЫ

1. Галинская А.А. Модульные нейронные сети: обзор современного состояния разработок // Математические машины и системы. – 2003. – № 3, 4. – С. 87–102.
2. Neural Works Professional II/Plus, http://www.mathworks.com/.
3. Резник А.М., Куссуль М.Э., Сычев А.С., Садовая Е.Г., Калина Е.А. Система проектирования модульных нейронных сетей САПР МНС // Математические машины и системы. – 2002. – № 3. – С. 28–37.
4. Куссуль М.Э. Графы модульных нейронных сетей // Математические машины и системы. – 2005. – № 1. – С. 26–38.
5. Куссуль М.Э., Галинская А.А. О свойствах циклов в модульных нейронных сетях // Математические машины и системы. – 2005. – № 2. – С. 54–62.
6. Дистель Р. Теория графов. – Новосибирск: Изд-во Ин-та математики, 2002. – 336 с.
7. Bang-Jensen J., Gutin G. Digraphs: Theory, Algorithms and Applications. Springer Monographs in Mathematics. – Springer-Verlag, London, 2000.