ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА И ПРИКЛАДНАЯ МАТЕМАТИКА

УДК 519.816

И.А. Цветков

УСЛОВИЯ ЧЕРНОВА ДЛЯ ФУНКЦИЙ ОДИНОЧНО-ПУСТОГО ВЫБОРА

Определены все конфигурации условий Чернова для функций одиночно-пустого выбора на множествах вариантов с двумя и более элементами. Найдено число функций одиночно-пустого выбора с этими конфигурациями для множеств вариантов с двумя, тремя и четырьмя элементами.

Ключевые слова: *функция выбора, условие наследования, условие константности, условие согласия, условие независимости.*

Введение. В теории выбора вариантов [1, 3] изучаются *функции выбора*. Для конечного n-элементного (где $n \geq 2$) множества вариантов функция выбора C ставит в соответствие каждому аргументу X — *непустому* подмножеству множества вариантов — значение $C(X)$, являющееся подмножеством (возможно, пустым) X.

Функция выбора может, в частности, рассматриваться как модель выбора объектов. Например, если всего есть два объекта: a, b и из них выбирают b, при наличии только a не выбирают ничего, при наличии только b выбирают b, то множество вариантов — это $\{a,b\}$, значения функции выбора C таковы: $C(\{a,b\})=\{b\}$, $C(\{a\})=\varnothing$, $C(\{b\})=\{b\}$.

Для функций выбора изучают условия *наследования*, *константности*, *согласия*, *независимости* от отбрасывания отвергнутых вариантов (кратко — независимости) [1]. Назовем эти четыре условия *условиями Чернова* (их предложил *Herman Chernoff* в работе [4]).

В теории выбора вариантов исследованы условия Чернова для функций *непустого* выбора (т.е. функций, все значения которых — непустые подмножества) [1] и функций *одиночного* выбора (т.е. функций, все значения которых — 1-элементные подмножества) [1].

Но в известных публикациях не исследовались условия Чернова для таких функций выбора, значения которых — *пустые или 1-элементные* подмножества, причем хотя бы одно значение — пустое подмножество и хотя бы одно значение — 1-элементное подмножество. Эти функции выбора предложены в работе автора [2] и названы в ней функциями *одиночно-пустого* выбора.

Цели работы. Целями статьи являются:

1) нахождение всех конфигураций условий Чернова для функций одиночно-пустого выбора;

2) получение соотношения для числа всех функций одиночно-пустого выбора на n-элементном множестве вариантов ($n \geq 2$);

3) определение числа функций одиночно-пустого выбора с каждой из возможных конфигураций условий Чернова для множеств вариантов с двумя, тремя и четырьмя элементами.

Для достижения первой цели в статье доказываются леммы о возможных и невозможных конфигурациях условий Чернова для функций одиночно-пустого выбора на множествах вариантов с тремя и более элементами. На основании этих лемм доказывается теорема о *пяти* возможных конфигурациях условий Чернова для таких функций. Определены все *три* возможные конфигурации условий Чернова для функций одиночно-пустого выбора на множестве вариантов с двумя элементами.

Для достижения второй цели в статье доказывается теорема о числе всех функций одиночно-пустого выбора на n-элементном множестве вариантов ($n \geq 2$).

Для достижения третьей цели выполнены

экспериментальные исследования на ЭВМ. В статье приведены их результаты — число всех функций одиночно-пустого выбора с каждой из возможных конфигураций условий Чернова для множеств вариантов с двумя, тремя и четырьмя элементами.

Теоретическая часть. *Функция выбора* C на n-элементном (где $n \geq 2$) множестве вариантов A задана, если для каждого *непустого* $X \subseteq A$ определено значение функции $C(X) \subseteq X$ [1]. (Символы $\subseteq$ и $\subset$ здесь и далее обозначают нестрогое и строгое включения соответственно.)

Для функции выбора C на множестве вариантов A (кратко — для функции C на A) выполняются условия Чернова: 1) условие *наследования*, если и только если для любых непустых $X \subseteq A$ и $Y \subset A$ при $Y \subset X$ имеет место $Y \cap C(X) \subseteq C(Y)$; 2) условие *константности*, если и только если для любых непустых $X \subseteq A$ и $Y \subset A$ при $Y \subset X$ из $C(X) = \varnothing$ следует $C(Y) = \varnothing$, а из $Y \cap C(X) \neq \varnothing$ следует $Y \cap C(X) = C(Y)$; 3) условие *согласия*, если и только если для любых непустых $X \subset A$ и $Y \subset A$ имеет место $C(X) \cap C(Y) \subseteq C(X \cup Y)$; 4) условие *независимости* от отбрасывания отвергнутых вариантов (кратко — условие независимости), если и только если для любых непустых $X \subseteq A$ и $Y \subset A$ при $C(X) \subseteq Y \subset X$ имеет место $C(X) = C(Y)$ [1].

Назовем *конфигурацией условий* Чернова (кратко — конфигурацией условий) строку из четырех символов, каждый из которых равен 0 или 1 и соответствует невыполнению (символ 0) и выполнению (символ 1) условий в порядке слева направо: наследования, константности, согласия, независимости. Например, конфигурация условий 1010 означает, что условия наследования и согласия выполняются, условия константности и независимости не выполняются.

Обозначим $K = \{0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111\}$ множество конфигураций условий.

Функцию выбора C на множестве вариантов A называем, следуя [2], *функцией одиночно-пустого выбора*, если $|C(X)| \in \{0,1\}$ для любого непустого $X \subseteq A$ и существуют такие непустые $Y \subseteq A$ и $Z \subseteq A$, что $|C(Y)| = 0$ и $|C(Z)| = 1$.

Обозначим E_n множество всех функций одиночно-пустого выбора на n-элементном (где $n \geq 2$) множестве вариантов. Обозначим $E_n^{(k)}$ множество всех имеющих конфигурацию условий $k \in K$ функций одиночно-пустого выбора на n-элементном (где $n \geq 2$) множестве вариантов. Например, $E_3^{(1010)}$ — это множество всех имеющих конфигурацию условий 1010 функций одиночно-пустого выбора на 3-элементном множестве вариантов.

Установим, для каких $k \in K$ при любых натуральных $n \geq 3$ множества $E_n^{(k)}$ непустые.

Лемма 1. Для любого натурального $n \geq 3$ множество $E_n^{(0000)}$ непустое.

Доказательство. Пусть множество вариантов A такое, что $\{a, b, c\} \subseteq A$, $|A| = n$, $n \geq 3$. Обозначим $P = \{\{a, b\}, \{a, c\}\}$ семейство его подмножеств.

Определим функцию выбора C так: $C(Z) = \{a\}$ при $Z \in P$; $C(Z) = \varnothing$ при $Z \subseteq A$, $Z \notin P$, $Z \neq \varnothing$. Тогда $C \in E_n$.

Покажем, что для функции C на A не выполняется условие наследования.

Пусть $X = \{a, b\}$, $Y = \{a\}$. По определению функции C имеем $C(X) = \{a\}$ (так как $X \in P$) и $C(Y) = \varnothing$ (так как $Y \subset A$, $Y \notin P$, $Y \neq \varnothing$). Тогда $Y \cap C(X) = \{a\}$. При $Y \subset X$ не имеет места $Y \cap C(X) \subseteq C(Y)$. Следовательно, для функции C на A не выполняется условие наследования.

Покажем, что для функции C на A не выполняется условие согласия.

Пусть $X = \{a, b\}$, $Y = \{a, c\}$. По определению функции C имеем $C(X) = C(Y) = \{a\}$ (так как $X \in P$, $Y \in P$). Тогда $C(X) \cap C(Y) = \{a\}$, $C(X \cup Y) = C(\{a, b, c\}) = \varnothing$ (так как $\{a, b, c\} \subseteq A$, $\{a, b, c\} \notin P$). При $X \subset A$ и $Y \subset A$ не имеет места $C(X) \cap C(Y) \subseteq C(X \cup Y)$. Следовательно, для функции C на A не выполняется условие согласия.

Покажем, что для функции C на A не выполняется условие независимости.

Пусть $X = \{a, b\}$, $Y = \{a\}$. По определению функции C имеем $C(X) = \{a\}$ и $C(Y) = \varnothing$. При $C(X) \subseteq Y \subset X$ не имеет места $C(X) = C(Y)$. Следовательно, для функции C на A не выполняется условие независимости.

Так как для функции C на A не выпол-

няется условие независимости, согласно [1, с. 34] для функции C на A не выполняется условие константности.

Итак, для рассматриваемой функции выбора $C \in E_n$ на таком множестве вариантов A, что $|A| = n$ (при любом натуральном $n \geq 3$), не выполняются условия наследования, константности, согласия, независимости. Следовательно, $C \in E_n^{(0000)}$, т.е. множество $E_n^{(0000)}$ непустое для любого натурального $n \geq 3$. ♦

Символ ♦ здесь и далее показывает окончание доказательства.

Лемма 2. Для любого натурального $n \geq 3$ множество $E_n^{(1000)}$ непустое.

Доказательство. Пусть множество вариантов A такое, что $\{a, b, c\} \subseteq A$, $|A| = n$, $n \geq 3$. Обозначим $Q = \{\{a\}, \{a, b\}, \{a, c\}\}$ семейство его подмножеств.

Определим функцию выбора C так: $C(Z) = \{a\}$ при $Z \in Q$; $C(Z) = \varnothing$ при $Z \subseteq A$, $Z \notin Q$, $Z \neq \varnothing$. Тогда $C \in E_n$.

Покажем, что для функции C на A выполняется условие наследования. Рассмотрим два случая.

Случай 1: $X \subseteq A$, $X \notin Q$, $|X| \geq 2$. По определению функции C имеем $C(X) = \varnothing$, поэтому для любого непустого $Y \subset X$ получим $Y \cap C(X) = \varnothing$. Следовательно, $Y \cap C(X) \subseteq C(Y)$.

Случай 2: $X \in Q$. По определению функции C имеем $C(X) = \{a\}$. Есть ровно четыре пары X и непустого $Y \subset X$: 1) $X = \{a, b\}$ и $Y = \{a\}$; 2) $X = \{a, b\}$ и $Y = \{b\}$; 3) $X = \{a, c\}$ и $Y = \{a\}$; 4) $X = \{a, c\}$ и $Y = \{c\}$.

Для пар 1 и 3 $Y \in Q$, поэтому по определению функции C имеем $C(Y) = \{a\}$. Из $Y \cap C(X) = \{a\}$ следует $Y \cap C(X) \subseteq C(Y)$.

Для пар 2 и 4 $Y \cap C(X) = \varnothing$, поэтому $Y \cap C(X) \subseteq C(Y)$.

Анализ случаев 1 и 2 (случай $X \subset A$, $X \notin Q$, $|X| = 1$ не рассматриваем, так как при этом нет непустых $Y \subset X$) позволяет заключить, что для любых непустых $X \subseteq A$ и $Y \subset A$ при $Y \subset X$ имеет место $Y \cap C(X) \subseteq C(Y)$. Следовательно, для функции C на A выполняется условие наследования.

Покажем, что для функции C на A не выполняется условие согласия.

Пусть $X = \{a, b\}$, $Y = \{a, c\}$. По определению функции C имеем $C(X) = C(Y) = \{a\}$ (так как $X \in Q$, $Y \in Q$). Тогда $C(X) \cap C(Y) = \{a\}$ и $C(X \cup Y) = C(\{a, b, c\}) = \varnothing$ (так как $\{a, b, c\} \subseteq A$, $\{a, b, c\} \notin Q$). Для непустых $X \subset A$ и $Y \subset A$ не имеет места $C(X) \cap C(Y) \subseteq C(X \cup Y)$. Следовательно, для функции C на A не выполняется условие согласия.

Покажем, что для функции C на A не выполняется условие независимости.

Пусть $X = A$, $Y = \{a, b\}$. По определению функции C имеем $C(X) = \varnothing$ (так как $A \notin Q$) и $C(Y) = \{a\}$ (так как $Y \in Q$). При $C(X) \subseteq Y \subset X$ не имеет места $C(X) = C(Y)$. Следовательно, для функции C на A не выполняется условие независимости.

Так как для функции C на A не выполняется условие независимости, согласно [1, с. 34] для функции C на A не выполняется условие константности.

Итак, для рассматриваемой функции выбора $C \in E_n$ на таком множестве вариантов A, что $|A| = n$ (при любом натуральном $n \geq 3$), выполняется условие наследования и не выполняются условия константности, согласия, независимости. Следовательно, $C \in E_n^{(1000)}$, т.е. множество $E_n^{(1000)}$ непустое для любого натурального $n \geq 3$. ♦

Лемма 3. Для любого натурального $n \geq 3$ множество $E_n^{(0010)}$ непустое.

Доказательство. Пусть множество вариантов A такое, что $R \subset A$, $|A| = n$, $n \geq 3$, где $R = \{a, b\}$.

Определим функцию выбора C так: $C(R) = \{a\}$; $C(Z) = \varnothing$ при $Z \subseteq A$, $Z \neq R$, $Z \neq \varnothing$. Тогда $C \in E_n$.

Покажем, что для функции C на A не выполняется условие наследования.

Пусть $X = \{a, b\}$, $Y = \{a\}$. По определению функции C имеем $C(X) = \{a\}$ (так как $X = R$) и $C(Y) = \varnothing$ (так как $Y \subset A$, $Y \neq R$, $Y \neq \varnothing$). Тогда $Y \cap C(X) = \{a\}$. При $Y \subset X$ не имеет места $Y \cap C(X) \subseteq C(Y)$. Следовательно, для функции C на A не выполняется условие наследования.

Покажем, что для функции C на A выполняется условие согласия. Рассмотрим для непустых $X \subset A$ и $Y \subset A$ четыре случая.

Случай 1. $X = Y = R$. По определению функции C имеем $C(X) = C(Y) = \{a\}$. Тогда $C(X) \cap C(Y) = \{a\}$ и $C(X \cup Y) = C(R) = \{a\}$. Имеет место $C(X) \cap C(Y) \subseteq C(X \cup Y)$.

Случай 2. $X = R$, $Y \neq R$. По определению функции C имеем $C(X) = \{a\}$ и $C(Y) = \varnothing$. Тогда $C(X) \cap C(Y) = \varnothing$, поэтому имеет место $C(X) \cap C(Y) \subseteq C(X \cup Y)$.

Случай 3. $X \neq R$, $Y = R$. По определению функции C имеем $C(X) = \varnothing$ и $C(Y) = \{a\}$. Тогда $C(X) \cap C(Y) = \varnothing$, поэтому имеет место $C(X) \cap C(Y) \subseteq C(X \cup Y)$.

Случай 4. $X \neq R$, $Y \neq R$. По определению функции C имеем $C(X) = C(Y) = \varnothing$. Тогда $C(X) \cap C(Y) = \varnothing$, поэтому имеет место $C(X) \cap C(Y) \subseteq C(X \cup Y)$.

Анализ случаев 1–4 позволяет заключить, что для любых непустых $X \subset A$ и $Y \subset A$ имеет место $C(X) \cap C(Y) \subseteq C(X \cup Y)$. Следовательно, для функции C на A выполняется условие согласия.

Покажем, что для функции C на A не выполняется условие независимости.

Пусть $X = \{a, b\}$, $Y = \{a\}$. По определению функции C имеем $C(X) = \{a\}$ и $C(Y) = \varnothing$. При $C(X) \subseteq Y \subset X$ не имеет места $C(X) = C(Y)$. Следовательно, для функции C на A не выполняется условие независимости.

Так как для функции C на A не выполняется условие независимости, согласно [1, с. 34] для функции C на A не выполняется условие константности.

Итак, для рассматриваемой функции выбора $C \in E_n$ на таком множестве вариантов A, что $|A| = n$ (при любом натуральном $n \geq 3$), выполняется условие согласия и не выполняются условия наследования, константности, независимости. Следовательно, $C \in E_n^{(0010)}$, т.е. множество $E_n^{(0010)}$ непустое для любого натурального $n \geq 3$. ♦

Лемма 4. Для любого натурального $n \geq 3$ множество $E_n^{(1010)}$ непустое.

Доказательство. Пусть множество вариантов A такое, что $S \subset A$, $|A| = n$, $n \geq 3$, где $S = \{a\}$.

Определим функцию выбора C так: $C(S) = \{a\}$; $C(Z) = \varnothing$ при $Z \subseteq A$, $Z \neq S$, $Z \neq \varnothing$. Тогда $C \in E_n$.

Покажем, что для функции C на A выполняется условие наследования.

По определению функции C для любых непустых $X \subseteq A$ и $Y \subset A$ при $Y \subset X$ имеем $C(X) = \varnothing$. Тогда $Y \cap C(X) = \varnothing$. Имеет место $Y \cap C(X) \subseteq C(Y)$. Следовательно, для функции C на A выполняется условие наследования.

Покажем, что для функции C на A выполняется условие согласия. Рассмотрим для непустых $X \subset A$ и $Y \subset A$ четыре случая.

Случай 1. $X = Y = S$. По определению функции C имеем $C(X) = C(Y) = \{a\}$. Тогда $C(X) \cap C(Y) = \{a\}$ и $C(X \cup Y) = C(S) = \{a\}$. Имеет место $C(X) \cap C(Y) \subseteq C(X \cup Y)$.

Случай 2. $X = S$, $Y \neq S$. По определению функции C имеем $C(X) = \{a\}$ и $C(Y) = \varnothing$. Тогда $C(X) \cap C(Y) = \varnothing$, поэтому имеет место $C(X) \cap C(Y) \subseteq C(X \cup Y)$.

Случай 3. $X \neq S$, $Y = S$. По определению функции C имеем $C(X) = \varnothing$ и $C(Y) = \{a\}$. Тогда $C(X) \cap C(Y) = \varnothing$, поэтому имеет место $C(X) \cap C(Y) \subseteq C(X \cup Y)$.

Случай 4. $X \neq S$, $Y \neq S$. По определению функции C имеем $C(X) = C(Y) = \varnothing$. Тогда $C(X) \cap C(Y) = \varnothing$, поэтому имеет место $C(X) \cap C(Y) \subseteq C(X \cup Y)$.

Анализ случаев 1–4 позволяет заключить, что для любых непустых $X \subset A$ и $Y \subset A$ имеет место $C(X) \cap C(Y) \subseteq C(X \cup Y)$. Следовательно, для функции C на A выполняется условие согласия.

Покажем, что для функции C на A не выполняется условие независимости.

Пусть $X = A$, $Y = S$. По определению функции C имеем $C(X) = \varnothing$ (так как $X \neq S$) и $C(Y) = \{a\}$ (так как $Y = S$). При $C(X) \subseteq Y \subset X$ не имеет места $C(X) = C(Y)$. Следовательно, для функции C на A не выполняется условие независимости.

Так как для функции C на A не выполняется условие независимости, согласно [1, с. 34] для функции C на A не выполняется условие константности.

Итак, для рассматриваемой функции выбора $C \in E_n$ на таком множестве вариантов A, что $|A| = n$ (при любом натуральном $n \geq 3$), выполняются условия наследования и согласия, не выполняются условия константности и независимости. Следовательно, $C \in E_n^{(1010)}$, т.е. мно-

жество $E_n^{(1010)}$ непустое для любого натурального $n \geq 3$. ♦

Лемма 5. Если для функции $C \in E_n$ при натуральном $n \geq 2$ выполняется условие независимости, то выполняется условие константности.

Доказательство. Рассмотрим такое множество вариантов A, что $|A| = n$, $n \geq 2$.

Пусть для функции $C \in E_n$ на A выполняется условие независимости. Следовательно, для любых непустых $X \subseteq A$ и $Y \subset A$ при $C(X) \subseteq Y \subset X$ имеет место $C(X) = C(Y)$. Покажем, что при этом для C выполняется условие константности.

Предположим противное: для функции $C \in E_n$ на A выполняется условие независимости и не выполняется условие константности. Если условие константности для функции $C \in E_n$ на A не выполняется, то имеет место хотя бы один из двух случаев.

Случай 1. Существуют такие непустые $X \subseteq A$ и $Y \subset A$, что $Y \subset X$, $C(X) = \varnothing$ и $C(Y) \neq \varnothing$. Но тогда при $C(X) \subseteq Y \subset X$ не имеет места $C(X) = C(Y)$. Следовательно, для функции $C \in E_n$ на A не выполняется условие независимости. Это противоречит предположению.

Случай 2. Существуют такие непустые $X \subseteq A$ и $Y \subset A$, что $Y \subset X$, $Y \cap C(X) \neq \varnothing$ и $Y \cap C(X) \neq C(Y)$. Так как $Y \cap C(X) \neq \varnothing$ и $C \in E_n$, имеем $|C(X)| = 1$. Из $Y \cap C(X) \neq \varnothing$ и $|C(X)| = 1$ следует, что $Y \cap C(X) = C(X)$ и $C(X) \subseteq Y$. Из $Y \cap C(X) = C(X)$ и $Y \cap C(X) \neq C(Y)$ следует $C(X) \neq C(Y)$. Тогда при $C(X) \subseteq Y \subset X$ не имеет места $C(X) = C(Y)$. Следовательно, для функции $C \in E_n$ на A не выполняется условие независимости. Это противоречит предположению.

Анализ случаев 1 и 2 позволяет заключить, что предположение «для функции $C \in E_n$ на A выполняется условие независимости и не выполняется условие константности» при натуральном $n \geq 2$ не верно. Следовательно, если для функции $C \in E_n$ при натуральном $n \geq 2$ выполняется условие независимости, то выполняется условие константности. ♦

Лемма 6. Для любого натурального $n \geq 3$ множество $E_n^{(1111)}$ непустое.

Доказательство. Обозначим множество вариантов $A = \{a_1, a_2, \ldots, a_n\}$, где $n \geq 3$.

Определим функцию выбора C так: $C(\{a_n\}) = \varnothing$; $C(\{a_i\}) = \{a_i\}$, $C\left(\left\{a_i, a_{k_1}, a_{k_2}, \ldots, a_{k_j}\right\}\right) = \{a_i\}$ при $i = \overline{1, n-1}$; $k_t \in \{i+1, i+2, \ldots, n\}$; $t = \overline{1, j}$; $j \in \{1, 2, \ldots, n-1\}$. Тогда $C \in E_n$.

Покажем, что для функции C на A выполняется условие независимости.

Рассмотрим для произвольных непустых $X \subseteq A$ и $Y \subset X$ два случая.

Случай 1. $Y = \{a_n\}$. Тогда из $Y \subset X$ следует $|X| > 1$, поэтому по определению функции C имеем $C(X) \neq \{a_n\}$. Следовательно, не имеет места $C(X) \subseteq Y$.

Случай 2. $Y \neq \{a_n\}$. Тогда по определению функции C имеем $|C(Y)| = 1$ и (поскольку $Y \subset X$) $|C(X)| = 1$. Пусть $C(X) = \{x\}$. Возможны два случая.

Случай 2А. $C(Y) = \{x\}$. Тогда $x \in Y$. Следовательно, $C(X) \subseteq Y$ (так как $C(X) = \{x\}$). Таким образом, при $C(X) \subseteq Y \subset X$ имеет место $C(X) = C(Y)$.

Случай 2Б. $C(Y) \neq \{x\}$. Тогда по определению функции C имеем $x \notin Y$, так как при $x \in Y$ и $Y \subset X$ из $C(X) = \{x\}$ следовало бы $C(Y) = \{x\}$. Поскольку $x \notin Y$, не имеет места $C(X) \subseteq Y$ (так как $C(X) = \{x\}$).

Анализ случаев 1 и 2 позволяет заключить, что для любых непустых $X \subseteq A$ и $Y \subset A$ при $C(X) \subseteq Y \subset X$ имеет место $C(X) = C(Y)$. Следовательно, для функции C на A выполняется условие независимости.

Так как для функции $C \in E_n$ выполняется условие независимости, для нее согласно лемме 5 при натуральном $n \geq 2$ (а значит, и при $n \geq 3$) выполняется условие константности.

Так как для функции C на A выполняется условие константности, согласно [1, с. 34] для функции C на A выполняются условия наследования и согласия.

Итак, для рассматриваемой функции выбора $C \in E_n$ на таком множестве вариантов A, что $|A| = n$ (при любом натуральном $n \geq 3$), выполняются условия наследования, константности, согласия и независимости. Следовательно, $C \in E_n^{(1111)}$, т.е. множество $E_n^{(1111)}$ непустое для любого натурального $n \geq 3$. ♦

Установим, для каких $k \in K$ при любых натуральных $n \geq 3$ множества $E_n^{(k)}$ пустые.

Лемма 7. Для любого натурального $n \geq 3$ множества $E_n^{(0001)}$, $E_n^{(0011)}$, $E_n^{(0100)}$, $E_n^{(0101)}$, $E_n^{(0110)}$, $E_n^{(0111)}$, $E_n^{(1001)}$, $E_n^{(1011)}$, $E_n^{(1100)}$, $E_n^{(1101)}$, $E_n^{(1110)}$ пустые.

Доказательство. Согласно [1, с. 34], если для любой функции выбора выполняется условие константности, то выполняются условия наследования, согласия, независимости. Следовательно, нет функций выбора с конфигурациями условий 0100, 0101, 0110, 0111, 1100, 1101, 1110. Каждая функция из множества E_n — это функция выбора, поэтому для любого натурального $n \geq 3$ множества $E_n^{(0100)}$, $E_n^{(0101)}$, $E_n^{(0110)}$, $E_n^{(0111)}$, $E_n^{(1100)}$, $E_n^{(1101)}$, $E_n^{(1110)}$ пустые.

Согласно лемме 5, если для функции из множества E_n при натуральном $n \geq 2$ выполняется условие независимости, то выполняется условие константности. Следовательно, в множествах E_n при натуральном $n \geq 2$ (а значит, и при $n \geq 3$) нет функций с конфигурациями условий 0001, 0011, 1001, 1011. Таким образом, для любого натурального $n \geq 3$ множества $E_n^{(0001)}$, $E_n^{(0011)}$, $E_n^{(1001)}$, $E_n^{(1011)}$ пустые. ♦

Теорема 1. Для любого натурального $n \geq 3$ семейство $\left\{E_n^{(0000)}, E_n^{(0010)}, E_n^{(1000)}, E_n^{(1010)}, E_n^{(1111)}\right\}$ — это *разбиение* множества E_n на непустые подмножества.

Доказательство. Согласно леммам 1, 3, 2, 4, 6 соответственно для любого натурального $n \geq 3$ множества $E_n^{(0000)}$, $E_n^{(0010)}$, $E_n^{(1000)}$, $E_n^{(1010)}$, $E_n^{(1111)}$ непустые. Согласно лемме 7 для любого натурального $n \geq 3$ множества $E_n^{(0001)}$, $E_n^{(0011)}$, $E_n^{(0100)}$, $E_n^{(0101)}$, $E_n^{(0110)}$, $E_n^{(0111)}$, $E_n^{(1001)}$, $E_n^{(1011)}$, $E_n^{(1100)}$, $E_n^{(1101)}$, $E_n^{(1110)}$ пустые. Таким образом, $E_n^{(0000)} \bigcup E_n^{(0010)} \bigcup E_n^{(1000)} \bigcup E_n^{(1010)} \bigcup E_n^{(1111)} = E_n$.

Множества $E_n^{(0000)}$, $E_n^{(0010)}$, $E_n^{(1000)}$, $E_n^{(1010)}$, $E_n^{(1111)}$ по определению попарно не пересекаются.

Следовательно, для любого натурального $n \geq 3$ семейство $\left\{E_n^{(0000)}, E_n^{(0010)}, E_n^{(1000)}, E_n^{(1010)}, E_n^{(1111)}\right\}$ — это *разбиение* множества E_n на непустые подмножества. ♦

Определим число всех функций одиночно-пустого выбора для n-элементного ($n \geq 2$) множества вариантов.

Теорема 2. Для любого натурального $n \geq 2$ имеет место $\left|E_n\right| = \prod_{k=1}^{n} (k+1)^{C_n^k} - \prod_{m=1}^{n} m^{C_n^m} - 1$.

Доказательство. Рассмотрим произвольное натуральное $n \geq 2$.

У n-элементного множества есть ровно C_n^k k-элементных подмножеств. Пусть для каждого такого подмножества X при $1 \leq k \leq n$ так определено значение $C(X)$ функции выбора, что $\left|C(X)\right| \in \{0,1\}$. Множество всех таких функций выбора для n-элементного множества вариантов обозначим F_n.

Для каждого k-элементного подмножества X есть ровно $(k+1)$ различных значений $C(X)$ функций $C \in F_n$. Следовательно, по комбинаторному правилу произведения для всех C_n^k k-элементных подмножеств n-элементного множества имеем ровно $(k+1)^{C_n^k}$ различных наборов значений функций из F_n. Тогда по этому правилу получим $\left|F_n\right| = \prod_{k=1}^{n} (k+1)^{C_n^k}$.

Пусть для каждого m-элементного подмножества X n-элементного множества при $1 \leq m \leq n$ так определено значение $C(X)$ функции выбора, что $\left|C(X)\right| = 1$. Множество всех таких функций выбора (их называют функциями одиночного выбора [1]) для n-элементного множества вариантов обозначим I_n.

Для каждого m-элементного подмножества X есть ровно m различных значений $C(X)$ функций $C \in I_n$. Следовательно, по комбинаторному правилу произведения для всех C_n^m m-элементных подмножеств n-элементного множества имеем ровно $m^{C_n^m}$ различных наборов значений функций из I_n. Тогда по этому правилу получим $\left|I_n\right| = \prod_{m=1}^{n} m^{C_n^m}$.

Пусть для каждого p-элементного подмножества X n-элементного множества при $1 \leq p \leq n$ так определено значение $C(X)$ функции выбора, что $C(X) = \varnothing$. Для любого натурального $n \geq 2$ есть ровно одна такая функция (функция пустого выбора). Множество из одной такой функции для n-элементного множества вариантов обозначим Z_n. Тогда $\left|Z_n\right| = 1$.

Для любого натурального $n \geq 2$ имеем $F_n = Z_n \bigcup I_n \bigcup E_n$; $Z_n \bigcap I_n = \varnothing$; $Z_n \bigcap E_n = \varnothing$;

$I_n \cap E_n = \varnothing$. Следовательно, $|F_n| = |Z_n| + |I_n| + |E_n|$. Тогда $|E_n| = |F_n| - |I_n| - |Z_n|$. Получим $|E_n| = \prod_{k=1}^{n} (k+1)^{C_n^k} - \prod_{m=1}^{n} m^{C_n^m} - 1$. ♦

Экспериментальные исследования. Согласно теореме 2 имеем $|E_2| = 9$. Для множества вариантов $A = \{a, b\}$ значения всех девяти функций из множества E_2 приведены в таблице 1.

Таблица 1 — Значения функций одиночно-пустого выбора для 2-элементного множества вариантов

X	$C_1(X)$	$C_2(X)$	$C_3(X)$	$C_4(X)$	$C_5(X)$	$C_6(X)$	$C_7(X)$	$C_8(X)$	$C_9(X)$
$\{a\}$	$\{a\}$	$\varnothing$	$\{a\}$	$\varnothing$	$\varnothing$	$\{a\}$	$\{a\}$	$\varnothing$	$\varnothing$
$\{b\}$	$\varnothing$	$\{b\}$	$\{b\}$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\{b\}$	$\{b\}$
$\{a, b\}$	$\varnothing$	$\varnothing$	$\varnothing$	$\{a\}$	$\{b\}$	$\{a\}$	$\{b\}$	$\{a\}$	$\{b\}$

Проверка выполнения условий Чернова показывает, что каждая из функций C_4, C_5, C_7, C_8 имеет конфигурацию условий 0010, каждая из функций C_1, C_2, C_3 — конфигурацию условий 1010, каждая из функций C_6, C_9 — конфигурацию условий 1111. Следовательно, семейство $\{E_2^{(0010)}, E_2^{(1010)}, E_2^{(1111)}\}$ — это *разбиение* множества E_2 на непустые подмножества.

Согласно теореме 2 имеем $|E_3| = 839$, $|E_4| = 14909183$, $|E_5| = 37150108947578879$. Эти значения определяют возможности осуществления полного перебора всех функций одиночно-пустого выбора для определения их конфигураций условий.

Автором выполнен полный перебор на ЭВМ функций одиночно-пустого выбора для n-элементных множеств вариантов при $n = 3$ и $n = 4$. Для каждой такой функции определялась конфигурация условий. Результаты этого эксперимента приведены в таблице 2.

Таблица 2 — Число всех функций одиночно-пустого выбора, имеющих заданную конфигурацию условий

Конфигурация условий	Число функций одиночно-пустого выбора для n-элементного множества вариантов	
	$n = 3$	$n = 4$
0000	180	13110164
0010	566	1791982
1000	24	5148
1010	60	1849
1111	9	40

Заключение. В статье исследованы условия Чернова (условия наследования, константности, согласия, независимости) для функций одиночно-пустого выбора, значения которых — пустые или 1-элементные подмножества, причем хотя бы одно значение — пустое подмножество и хотя бы одно значение — 1-элементное подмножество.

В статье получены следующие основные результаты.

1. Для любого множества вариантов с тремя или более элементами доказано существование функций одиночно-пустого выбора с конфигурациями условий 0000, 0010, 1000, 1010, 1111 (леммы 1, 3, 2, 4, 6 соответственно). Для доказательства в каждой из этих лемм предложен пример функции одиночно-пустого выбора с соответствующей конфигурацией условий.

2. Для функций одиночно-пустого выбора на любом множестве вариантов с двумя или более элементами установлена взаимосвязь условий независимости и константности (лемма 5).

3. Для любого множества вариантов с тремя или более элементами доказано, что не существуют функции одиночно-пустого выбора с конфигурациями условий, отличными от 0000, 0010, 1000, 1010, 1111 (лемма 7).

4. Доказано (теорема 1), что любая функция одиночно-пустого выбора на множестве вариантов с тремя или более элементами имеет одну из *пяти* конфигураций условий: 0000, 0010, 1000, 1010, 1111 и такая функция существует для каждой из этих конфигураций.

5. Доказана теорема (теорема 2) о числе всех функций одиночно-пустого выбора для n-элементного множества вариантов (где $n \geq 2$).

6. Установлено, что любая функция одиночно-пустого выбора на 2-элементном множестве вариантов имеет одну из *трех* конфигураций условий: 0010, 1010, 1111. Определены такие функции для каждой из этих конфигураций (таблица 1).

7. Полным перебором на ЭВМ найдено число всех функций одиночно-пустого выбора с каждой из пяти возможных конфигураций условий (0000, 0010, 1000, 1010, 1111) для множеств вариантов с тремя и четырьмя элементами (таблица 2).

В прикладных задачах исследованные функции одиночно-пустого выбора могут быть моделями выбора объектов. При этом выполнение условий Чернова для функции выбора характеризует логически обоснованный выбор в такой модели.

Библиографический список

1. *Айзерман М.А., Алескеров Ф.Т.* Выбор вариантов: основы теории. — М.: Наука, 1990. — 240 с.

2. *Цветков И.А.* Функции одиночно-пустого выбора // Перспективы развития гуманитарных и технических систем: материалы всероссийской науч. конф. — Ч. 2. — Таганрог: изд-во ТТИ ЮФУ, 2011. — С. 77–79.

3. *Шоломов Л.А.* Логические методы построения и анализа моделей выбора // Прикладная дискретная математика. — 2009. — № 1. — С. 38–71.

4. *Chernoff H.* Rational Selection of Decision Functions // Econometrica. — 1954. — V. 22. — P. 422–443.

УДК 681.518

Л.А. Демидова

МОДЕЛИ ПРОГНОЗИРОВАНИЯ ВРЕМЕННЫХ РЯДОВ С КОРОТКОЙ АКТУАЛЬНОЙ ЧАСТЬЮ НА ОСНОВЕ МОДИФИЦИРОВАННОГО АЛГОРИТМА КЛОНАЛЬНОГО ОТБОРА

Предложены модели краткосрочного прогнозирования на основе модифицированного алгоритма клонального отбора, обеспечивающие повышение точности прогноза посредством автоматического формирования с помощью антител аналитических зависимостей, адекватно описывающих известные значения временного ряда. Разработана рекурсивная процедура формирования аналитических зависимостей, реализующая способ корректного кодирования антител с применением бинарных деревьев.

***Ключевые слова**: модель прогнозирования, алгоритм клонального отбора, временной ряд.*

Введение. Анализ временных рядов (ВР), описывающих процессы с короткой актуальной частью (порядка 20-30 значений), играет важную роль в решении многих практических задач, например, задач, связанных с прогнозированием тенденций рынка труда в России. При этом существенный интерес представляет разработка моделей краткосрочного прогнозирования, которые обеспечивали бы получение адекватной оценки предстоящих изменений и принятие соответствующих управленческих решений на основе известных значений ВР.

При реализации традиционных подходов к прогнозированию ВР, основанных, например, на использовании экстраполяционных описаний, возникает ряд сложных вопросов, связанных как с выбором модели прогнозирования, так и с оценкой свободных (неопределяемых автоматически) параметров модели, в частности, степени полинома или порядка авторегрессии.

Актуальной является проблема повышения точности прогнозирования процессов, представленных ВР с короткой актуальной частью. Для решения этой проблемы предлагается использовать хорошо зарекомендовавший себя при решении широкого спектра прикладных задач аппарат искусственных иммунных систем (ИИС), основанных на принципах естественной иммунной системы, реализующей модель адаптивного действия на локальном уровне и эмерджентность поведения на глобальном уровне и обладающей мощными и гибкими способностями обработки информации. Результативность применения аппарата ИИС, в частности, алгоритма клонального отбора и механизмов супрессии антител в решении задач интерполяции, экстраполяции и прогнозирования, доказанная в работах зарубежных исследователей [3], подтверждает перспективность использования ИИС для решения задачи прогнозирования ВР с короткой актуальной частью.

Теоретические исследования. В работе [2] для решения задачи краткосрочного прогнозирования предлагается использовать модели

прогнозирования k-го порядка на основе модифицированного алгоритма клонального отбора (МАКО), позволяющего при приемлемых временных затратах сформировать аналитическую зависимость, наилучшим образом описывающую известные значения ВР и обеспечивающую получение минимального значения аффинитета – средней относительной ошибки прогнозирования $AFER$ (Average Forecasting Error Rate):

$$AFER = \frac{\sum_{t=k+1}^{m} \left| \left(f(t) - d(t) \right) / d(t) \right|}{m} \cdot 100\%, \qquad (1)$$

где $f(t)$ и $d(t)$ – предсказанное и реальное значения для t-го отсчета времени; m – количество значений ВР (количество отсчетов времени).

Возможные варианты аналитических зависимостей в МАКО кодируются в виде антител Ab, которые должны осуществлять распознавание антигенов Ag – известных значений ВР. При реализации МАКО выбирается «лучшее» антитело Ab, обеспечивающее минимальное значение аффинитета $AFER$ [1, 2].

Антитело Ab представляет собой символьную строку, элементы которой выбираются из трёх предварительно заданных символьных алфавитов:

– алфавита арифметических операций $Operation = \{'+','-','*','/'\}$, то есть операций сложения, вычитания, умножения и деления;

– алфавита функционалов $Functional = \{'Q','S','C','E','L','\cdot'\}$, в котором символы $'Q','S','C','E','L'$ соответствуют математическим функциям «квадратный корень», «синус», «косинус», «экспонента», «натуральный логарифм», а символ $'\cdot'$ определяет отсутствие какой-либо математической функции;

– алфавита терминалов $Terminal = \{'a','b',...,'z','@'\}$, в котором символ '@' определяет некоторую константу, а символы $'a','b',...,'z'$ соответствуют аргументам искомой аналитической функции.

Предполагается, что арифметические операции являются двухместными, а применение функционала должно предшествовать применению арифметической операции.

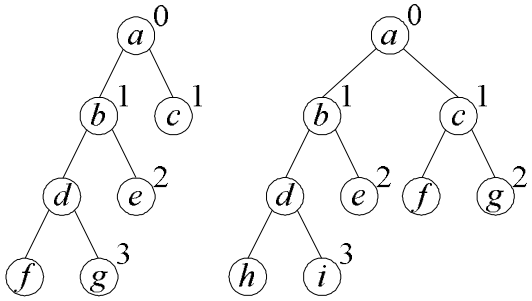
Количество терминальных позиций $Term$ в антителе определяется максимально возможным порядком модели прогнозирования [2]. Если максимально возможный порядок равен K, то это означает, что при прогнозировании значения ВР $d(t)$ на момент времени t могут использоваться K значений ВР: $d(t-K),\ldots, d(t-2), d(t-1)$. Тогда число «значащих» символов в алфавите терминалов определяется как K, а общее количество терминальных символов с учетом символа '@', определяющего некоторую константу, равно $K+1$. Следует отметить, что реальный порядок k модели прогнозирования удовлетворяет условию: $k \le K$ (ввиду возможного кратного вхождения некоторых терминальных символов в запись аналитической зависимости и возможного наличия константы).

Использование указанных выше трёх символьных алфавитов обеспечивает при реализации МАКО корректное преобразование случайным образом формируемых антител в аналитические зависимости. При этом предполагается, что структура антитела организована так, что может рассматриваться как бинарное дерево, определяющее некоторую аналитическую зависимость [2]. Для описания структуры антитела предлагается использовать строго бинарные деревья (СБД) двух видов, примеры которых приведены на рисунке 1, а, б, где рядом с узлами показаны значения их уровней. При этом строго бинарное дерево на рисунке 1, б является почти полным (ППСБД) [4]. Очевидно, что применение ППСБД (рисунок 1, б) позволяет строить более сложные аналитические зависимости, чем использование просто СБД (рисунок 1, а).

Для антител на основе СБД (рисунок 1, а) предлагается использовать рекурсивную процедуру формирования аналитических зависимостей произвольного максимального возможного порядка K, шаги реализации которой представлены на рисунке 2 [2]. При этом минимальный и максимальный номера позиций антитела, в которых стоят терминальные символы, равны $2 \cdot Term$ и $4 \cdot Term - 2$ соответственно, а длина антитела определяется максимальным номером позиции, в которой стоит терминальный символ.

В этом случае искомая аналитическая зависимость представляется в виде дерева, которое рекурсивно преобразуется в символьную строку посредством последовательной записи всех узлов, начиная слева направо и снизу вверх. При этом терминальные узлы (листья) могут содержать только символы из алфавита терминалов $Terminal$, а остальные узлы формируются из символов алфавита арифметических операций $Operation$ и алфавита функционалов $Functional$.

Пример формирования антитела на основе СБД для случая, когда $K = 3$ и алфавит терминалов имеет вид: $Terminal = \{'a','b','c','@'\}$, приведен на рисунке 3: в позициях 6, 8, 10 стоят символы из алфавита терминалов, в позициях 1, 3, 5, 7, 9 – из алфавита функционалов, а в позициях 2, 4 – из алфавита арифметических операций.



а б

Рисунок 1 – Примеры бинарных деревьев

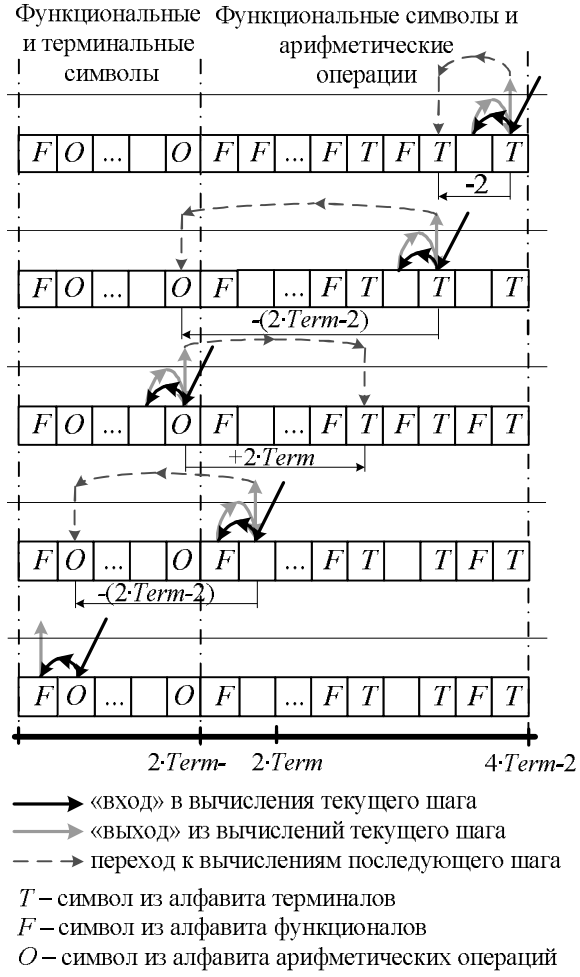


Рисунок 2 – Рекурсивная процедура формирования аналитических зависимостей

Так, если антитело имеет вид: $Ab = ('L'\ '*'\ 'C'\ '-'\ '.'\ '@'\ 'E'\ 'b'\ 'S'\ 'c')$, то оно порождает следующую аналитическую зависимость:

$$Function(b,c) = ln(cos(sin(c) - exp(b)) * const),$$

где *const* – константа, соответствующая символу '@' и определяемая случайным образом.

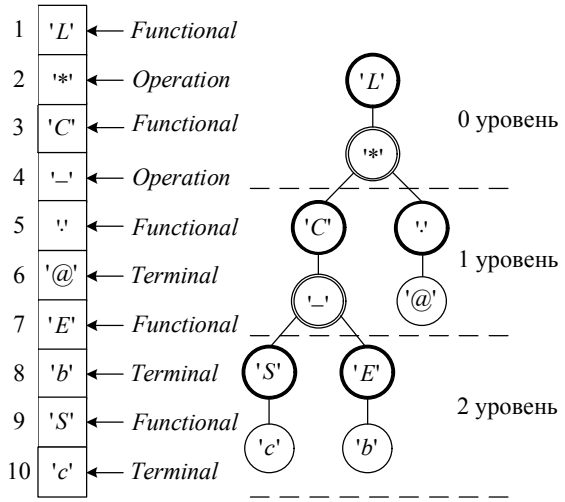


Рисунок 3 – Пример формирования антитела на основе строго бинарных деревьев

Для антител на основе ППСБД (рисунок 1, б) также может быть использована рекурсивная процедура формирования аналитических зависимостей (рисунок 2). Но ее применение в этом случае имеет ряд особенностей. Так как для антител на основе ППСБД максимально возможный порядок K всегда является нечетным числом, то само дерево может быть представлено в виде композиции (посредством последовательного выполнения некоторых арифметической операции из алфавита операций и математической функции из алфавита функционалов) одного левого поддерева максимально возможного порядка $K_{Л} = 3$ и некоторого количества n ($n \geq 1$) правых i-х ($i = \overline{1,n}$) поддеревьев максимально возможного порядка $K_{П}^{i} = 2$ ($K_{Л} + \sum_{i=1}^{n} K_{П}^{i} = K$). При этом количество терминальных позиций в левом и правых поддеревьях равно соответственно $Term_{Л} = 3$ и $Term_{П}^{i} = 2$ ($i = \overline{1,n}$), а длина антитела равна $(4 \cdot Term_{Л} - 2) + \sum_{i=1}^{n} (4 \cdot Term_{П}^{i} - 2) + 2 \cdot n$, то есть $10 + 8 \cdot n$. Тогда сама процедура кодирования антитела может быть реализована циклически в виде рекурсивного формирования частей антитела на основе соответствующих поддеревьев с последующей их композицией (слева направо и снизу вверх).

Пример формирования антитела на основе ППСБД для случая, когда $K = 5$ и алфавит терминалов имеет вид:
$Terminal = \{'a','b','c','d','e','@'\}$, приведен на

рисунке 4. В этом случае ППСБД состоит из двух поддеревьев: левого ($K_{Л}=3$) и правого ($K_{П}=2$) и, как легко заметить, левое бинарное поддерево является идентичным СБД, приведенному на рисунке 3. При этом длина антитела равна 18 и находится как сумма длин левого поддерева ($4\cdot Term_{Л}-2=10$), правого поддерева ($4\cdot Term_{П}-2=6$) и количества символов композиции (2).

Для обоих описанных выше вариантов формирования антител реализация МАКО осуществляется аналогичным образом. Ниже приведено описание МАКО, в котором антитела формируются на основе СБД.

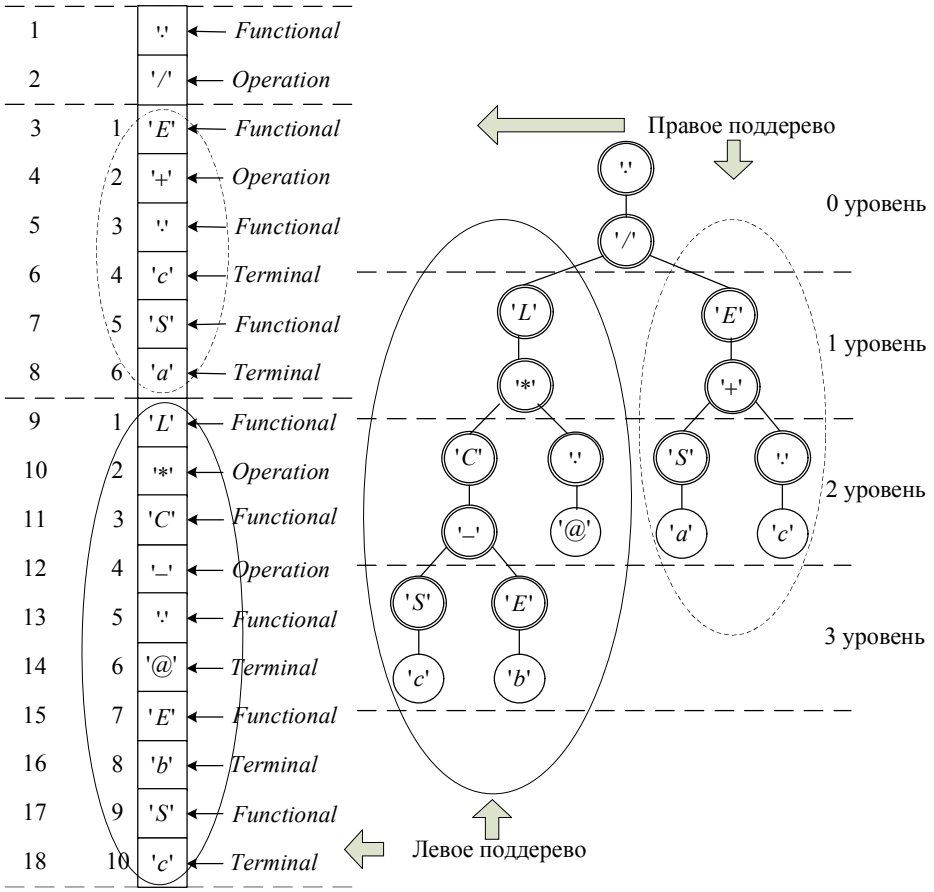


Рисунок 4 – Пример формирования антитела на основе почти полных строго бинарных деревьев

При реализации МАКО искомая аналитическая зависимость F кодируется в виде антитела Ab, которое должно обеспечивать распознание антигенов Ag, то есть элементов прогнозируемого ВР $d(t)$ ($t=\overline{1,m}$) таким образом, чтобы осуществлялась минимизация аффинитета, роль которого в данном случае играет средняя относительная ошибка прогнозирования $AFER$ вида (3).

МАКО включает подготовительную часть, реализующую формирование начальной популяции антител размером P, и итерационную часть, состоящую из следующих шагов: упорядочение антител по возрастанию значений аффинитета $AFER$; отбор и клонирование определенной доли «лучших» антител, имеющих наименьшие значения аффинитета $AFER$; гипермутация клонов антител; самоуничтожение клонов антител, «похожих» на другие клоны и антитела текущей популяции; вычисление аффинитета клонов антител и формирование новой популяции антител; супрессия полученной популяции; генерация новых антител и добавление их к текущей популяции до получения ее исходного размера P; проверка условия достижения заданного количества поколений G и завершение работы МАКО при выполнении этого условия, а в противном случае – повтор итерационной части [2].

При этом в каждом поколении g ($g\le G$) при реализации МАКО за счет использования процедуры супрессии осуществляется поддержание максимально возможного разнообразия антител в популяции размером P.

Целью подготовительной части МАКО является получение популяции случайным образом сформированных антител, «непохожих» друг на друга.

Предполагается, что два антитела Ab_1 и Ab_2 отличаются друг от друга («непохожи» друг на друга), если все символы (из алфавитов арифметических операций, функционалов и терминалов) в одинаковых позициях кодов антител различны. Исключение составляют пары символов '@' (из алфавита терминалов), определяющие константы, которые могут иметь одинаковые или разные значения. В этом случае при наличии символов '@' в одинаковых позициях кодов антител Ab_1 и Ab_2 производится дополнительная проверка на равенство соответствующих констант, хранящихся во вспомогательном массиве: если эти константы не совпадают, то антитела полагаются различными. В общем случае понятие «похожие» антитела предполагает совпадение некоторого количества символов в кодах сравниваемых антител.

Процедура сравнения кодов двух антител, с использованием которой следует определять и уничтожать антитела (или клоны), «похожие» на уже сформированные ранее, может быть описана следующим образом [2].

Пусть при сравнении кодов двух антител Ab_1 и Ab_2 число совпадений пар символов не меньше, чем $S_d \le S$, где S – длина антитела (например, $S = 10$ для антитела, приведенного на рисунке 3), S_d – порог самоуничтожения антитела (порог «естественной смерти»), определяющий количество совпадений при попарном сравнении символов в одинаковых позициях кодов антител, при котором антитело Ab_2 считается «похожим» на Ab_1 и должно быть уничтожено.

Тогда возможны следующие действия:

– если во всех совпадающих позициях кодов антител Ab_1 и Ab_2 отсутствует символ '@', следует уничтожить антитело Ab_2;

– если хотя бы в одной из совпадающих позиций кодов антител Ab_1 и Ab_2 присутствует символ '@', необходимо произвести дополнительную проверку на равенство констант $c_{1,(j/2-2)}$ и $c_{2,(j/2-2)}$, которые могут соответствовать четным j-м позициям ($2 \cdot Term \le j \le 4 \cdot Term - 2$) в антителах (например, для антитела Ab, приведенного на рисунке 3, $j = 6,8,10$); в случае равенства всех пар $c_{1,(j/2-2)}$ и $c_{2,(j/2-2)}$ для четных j-х позиций ($2 \cdot Term \le j \le 4 \cdot Term - 2$) следует уничтожить антитело Ab_2, а при несовпадении хотя бы одной пары констант $c_{1,(j/2-2)}$ и $c_{2,(j/2-2)}$ для четных j-х позиций ($2 \cdot Term \le j \le 4 \cdot Term - 2$) антитело Ab_2 следует добавить к популяции уже сформированных антител.

Необходимо отметить, что случайно сформированное антитело Ab может не содержать ни одной константы в своем составе, а максимально возможное количество констант в составе антитела равно максимально возможному порядку модели прогнозирования K (например, $K = 3$ для антитела, приведенного на рисунке 3), однако, в этом случае и сама аналитическая зависимость будет задаваться некоторой константой с учетом применяемых к K константам арифметических операций и математических функций.

Начальная популяция размером P, сформированная с применением процедуры сравнения кодов антител при $S_d = S$, будет состоять из «непохожих» между собой антител.

При этом одновременно с популяцией антител Ab формируется массив констант $Constant$, содержащий «значащие» константы, соответствующие символу '@' в коде антитела, или «тривиальные» (одинаковые) константы, выбираемые в каждом конкретном случае индивидуально и соответствующие «значащим» терминальным символам в коде антитела, формируемых антител. При этом количество строк в массиве констант $Constant$ равно размеру популяции P, а количество столбцов равно максимально возможному порядку модели K.

Далее более подробно рассмотрены шаги, выполняемые при реализации итерационной части МАКО.

Шаг 1. Вычисление аффинитета и упорядочение антител. Аффинитет $AFER$ для каждого антитела вычисляется как средняя относительная ошибка прогнозирования в соответствии с формулой (1), в которую подставляются данные имеющегося ВР $d(t)$ и значения $f(t)$, полученные с использованием аналитической зависимости F, определяемой этим антителом.

Так как применение некоторых математических функций («натуральный логарифм» и «квадратный корень»), а также арифметической операции деления не всегда возможно к операндам (аргументам функций и константам), задаваемым с помощью случайно сформированных антител, то для исключения таких антител из популяции и поддержания ее «жизнеспособности» предлагается присваивать им максимально возможное значение аффинитета, равное 100,

что в данном случае соответствует максимально возможному значению средней относительной ошибки прогнозирования $AFER = 100$ %.

При упорядочении антител по возрастанию аффинитета «лучшие» антитела Ab, имеющие меньшие значения аффинитета $AFER$, будут располагаться в начале списка упорядочения антител.

Вычисление аффинитета $AFER$ для всех антител Ab реализуется только в случае реализации первого поколения МАКО (при $g = 1$). При $g > 1$ вычисление аффинитета реализуется лишь для новых антител, возможно добавленных к популяции антител после выполнения процедуры супрессии.

Шаг 2. Отбор и клонирование «лучших» антител. Пусть pq – доля антител в популяции, подлежащих клонированию. Тогда количество «лучших» антител, находящихся в начале списка и подлежащих клонированию, может быть вычислено по формуле:

$$Clone = round(pq \cdot P), \qquad (2)$$

где $round(\cdot)$ – функция округления вещественного числа к ближайшему целому числу.

Количество клонов $W(i)$ для каждого i-го антитела Ab_i ($i = \overline{1, Clone}$) является уникальным и зависит от его аффинитета $AFER_i$: чем меньше аффинитет, тем большее количество клонов генерируется:

$$W(i) = round(Q \cdot P / i), \qquad (3)$$

где Q – коэффициент размножения клонов, представляющий собой некоторое целое число (обычно $1 \le Q \le 30$).

Для антитела, содержащего в своих позициях хотя бы одну константу, предлагается производить дополнительную генерацию еще $W(i)$ клонов, чтобы предусмотреть большее количество вариантов констант в записи искомой аналитической зависимости.

Параллельно с популяцией клонов Cl создается популяция констант клонов (массив констант клонов) $ClConstant$, при этом производится дублирование соответствующих строк из популяции констант $Constant$.

Пусть общее количество полученных клонов Cl антител равно ClP.

Шаг 3. Гипермутация клонов антител. В процессе реализации МАКО производится гипермутация символов в некоторых позициях «лучших» антител, выбранных для клонирования. Вероятность изменения каждого символа антитела задается с помощью коэффициента мутации pm, значение которого вычисляется для каждой генерации на основе двух величин: коэффициента гипермутации pgm и скорости гипермутации $pgmV$ ($pgm < 1$, $pgmV < 1$).

Для первого поколения антител коэффициент мутации pm принимается равным pgm: $pm = pgm$. Далее в каждом следующем поколении значение коэффициента мутации pm уменьшается до некоторого порогового значения $pmMin$ (например, $pmMin = 0{,}1$) со скоростью $pgmV$: $pm = pm \cdot pgmV$. Если в некотором поколении оказывается, что $pm < pmMin$, то коэффициент мутации pm опять полагается равным $pm = pgm$.

Процедура гипермутации символов в клоне Cl антитела Ab производится следующим образом. Для каждого j-го символа ($j = \overline{1{,}4 \cdot Term - 2}$) клона Cl генерируется случайное вещественное равномерно распределенное число r_j из отрезка [0,1]. Если выполняется условие: $r_j \le pm$, то реализуется мутация j-го символа клона Cl, при этом новый символ выбирается случайным образом из алфавитов арифметических операций, функционалов и терминалов (в зависимости от того, какому алфавиту принадлежит j-й символ), в противном случае, если выполняется условие: $r_j > pm$, j-й символ не изменяется.

Если в процессе мутации символа в j-й позиции i-го клона ($i = \overline{1, ClP}$) производится замена «значащего» терминального символа на символ '@' (или замена символа '@' на символ '@'), то в массиве констант клонов $ClConstant$ в i-й строке выполняется замена «значащей» (или «тривиальной») константы $ClConstant_{i,(j/2-2)}$ на новое случайно сгенерированное вещественное число $randConstant = Min + r \cdot Max$, где Min и Max – соответственно минимально и максимально возможные значения вещественной константы, задаваемые с учетом диапазона изменения значений ВР, r – случайное вещественное равномерно распределенное число из отрезка $[0,1]$. Если в процессе мутации символа в j-й позиции i-го клона ($i = \overline{1, ClP}$) производится замена символа '@' на «значащий» терминальный символ, то в массиве констант клонов $ClConstant$ в i-й строке выполняется

замена «значащей» константы $ClConstant_{i,(j/2-2)}$ на «тривиальную» константу.

Шаг 4. Самоуничтожение клонов антител. В популяции клонов Cl размером ClP, полученной в результате клонирования антител и их гипермутации, следует выявить «похожие» клоны с целью их последующего удаления. Эти действия выполняются в два этапа с использованием рассмотренной выше процедуры сравнения кодов двух антител.

Первый этап самоуничтожения клонов Cl выполняется над исходной популяцией клонов размером ClP, при этом выполняется сравнение кодов клонов Cl между собой при заданном пороге самоуничтожения S_d (процедура сравнения «клон-клон» $Cl - Cl$) и уничтожение «похожих» клонов.

Пусть размер популяции клонов Cl, полученной в результате первого этапа самоуничтожения клонов, равен $ClClP$.

На втором этапе самоуничтожения клонов Cl производится сравнение клонов Cl, оставшихся после применения процедуры $Cl - Cl$, с антителами Ab текущей популяции антител (процедура сравнения «клон-антитело» $Cl - Ab$) и уничтожаются клоны Cl, коды которых «похожи» на коды антител Ab.

Пусть размер популяции клонов Cl, полученной в результате второго этапа самоуничтожения клонов, равен $ClAbP$.

Параллельно с самоуничтожением «похожих» клонов выполняется и самоуничтожение соответствующих им строк в массиве констант клонов $ClConstant$.

Шаг 5. Вычисление аффинитета клонов антител. Для популяции клонов Cl, полученной в результате самоуничтожения «похожих» клонов, вычисляются значения аффинитета $AFER$ клонов.

Шаг 6. Формирование новой популяции антител. Текущая популяция антител размером P объединяется с популяцией клонов Cl размером $ClAbP$ и создается расширенная популяция антител размером $P + ClAbP$, которая упорядочивается по возрастанию значений аффинитета антител. Затем из расширенной популяции антител размером $P + ClAbP$ удаляются антитела с наибольшими значениями аффинитета $AFER$ в количестве $ClAbP$, что в итоге дает новую популяцию антител с тем же размером P, что и исходная популяция антител.

Шаг 7. Супрессия популяции антител. Для антител популяции размером P вычисляется среднее значение аффинитета $\overline{AFER}$ и выполняется процедура супрессии, позволяющая удалить из популяции часть антител, у которых значение аффинитета ниже порогового значения $S_s \cdot \overline{AFER}$, где S_s – коэффициент супрессии антител (например, $S_s = 0{,}98$), и поддержать таким образом максимально возможное разнообразие антител в популяции при одновременном сохранении в ней «лучшего» антитела с минимальным значением аффинитета $AFER_{min}$.

Пусть размер популяции антител, полученной в результате выполнения процедуры супрессии, равен AbP.

Шаг 8. Генерация новых антител и добавление их к текущей популяции антител. Если после выполнения процедуры супрессии выполняется условие: $AbP < P$, то реализуется генерация новых случайно сформированных антител («непохожих» на уже имеющиеся антитела текущей популяции) в количестве $P - AbP$ и их добавление к популяции антител размером AbP. При формировании новых антител выполняется сравнение кодов антител между собой при заданном пороге самоуничтожения S_d (процедура сравнения «антитело-антитело» $Ab - Ab$) и уничтожение «похожих» антител.

Шаг 9. Проверка условия окончания МАКО. Проверка условия окончания МАКО осуществляется по количеству текущих поколений g. Если номер текущего поколения g равен заданному максимальному количеству поколений G, то работа МАКО завершается. В противном случае, если $g < G$, номер текущего поколения g увеличивается на единицу и осуществляется переход к шагу 1 итерационной части МАКО для реализации следующего поколения.

В случае формирования антител на основе ППСБД приведенное выше описание МАКО незначительно меняется (с учетом особенностей формирования антител на основе ППСБД).

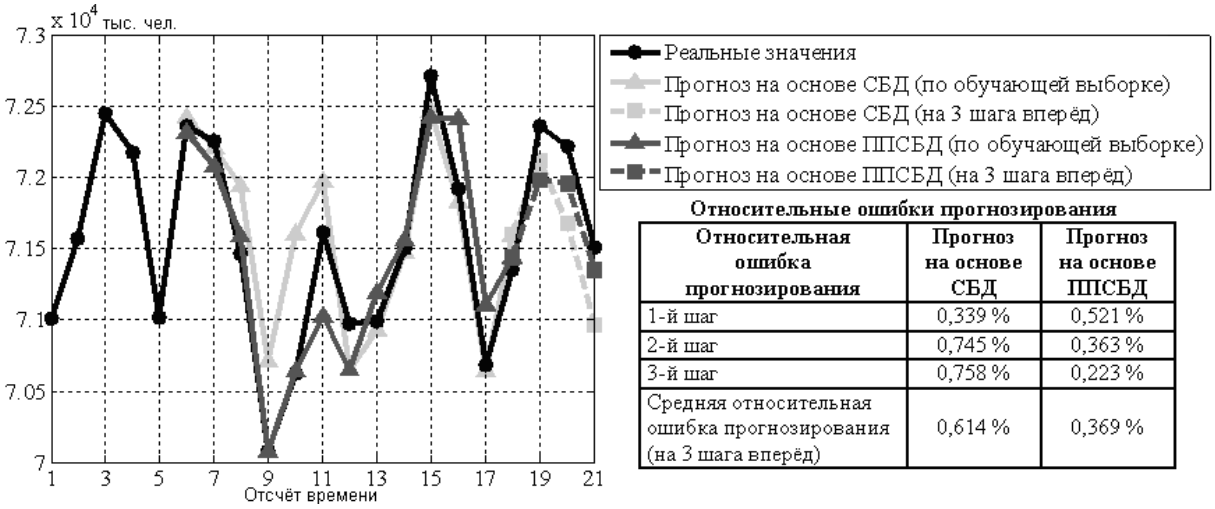
Экспериментальные исследования. Предложенные модели прогнозирования на основе МАКО были использованы для решения задачи краткосрочного прогнозирования на 3 шага вперед при прогнозировании тенденций рынка труда в России, в частности, для прогнозирования ВР, описывающего «экономически активное население».

Для построения моделей прогнозирования на основе МАКО использовалось 18 известных значений ВР (для отсчетов времени с февраля 1999 года по май 2003 года в точках отсчета: февраль, май, август и ноябрь), а оценка

качества прогнозирования выполнялась для 3 значений ВР (август 2003 года, ноябрь 2003 года и февраль 2004 года), то есть на 3 шага вперед.

При реализации МАКО были реализованы 2 варианта кодирования антител: в первом случае антитела формировались на основе СБД, а втором – на основе ППСБД. В обоих случаях максимально возможный порядок K и реальный порядок k моделей равны соответственно 7 и 5.

При этом в первом и втором случаях были получены значения средней относительной ошибки прогнозирования $AFER$, равные 0,410 % и 0,322 % соответственно, а средняя относительная ошибка прогнозирования на 3 шага составила 0,614 % и 0,369 % соответственно (рисунок 5).



Относительные ошибки прогнозирования

Относительная ошибка прогнозирования	Прогноз на основе СБД	Прогноз на основе ППСБД
1-й шаг	0,339 %	0,521 %
2-й шаг	0,745 %	0,363 %
3-й шаг	0,758 %	0,223 %
Средняя относительная ошибка прогнозирования (на 3 шага вперёд)	0,614 %	0,369 %

Рисунок 5 – Результаты прогнозирования

«Лучшие» антитела имеют соответственно вид:

'0+0*C/L/Q*C*0d0@LdCeSgLfSc', где символ '@' заменяется константой 852,666;

'0*0+QeCe0+0/QgSbC*S*QcCf0?', где символ '@' заменяется константой 0,922,

а аналитические зависимости записываются соответственно как:

$$F\big(d(t-5), d(t-4), d(t-3), d(t-2), d(t-1)\big)=$$

$$=cos(ln(sqrt(cos(sin(d(t\text{-}5))*ln(d(t\text{-}2)))*sin(d(t\text{-}1)))/cos(d(t\text{-}3)))/ln(d(t\text{-}4)))*852{,}666+d(t\text{-}4);$$

$$F\big(d(t-5), d(t-4), d(t-3), d(t-2), d(t-1)\big)=$$

$$=(cos(sin(0{,}922*cos(d(t\text{-}2)))*sqrt(d(t\text{-}4)))+sin(d(t\text{-}5))/sqrt(d(t\text{-}1)))*(cos(d(t\text{-}3))+sqrt(d(t\text{-}3))).$$

Как показывает анализ результатов прогнозирования ВР, описывающего «экономически активное население», лучшие результаты были получены в случае формирования антител на основе ППСБД.

Заключение. Модели прогнозирования на основе МАКО позволяют повысить точность прогнозирования процессов, представленных ВР с короткой актуальной частью, существенно сократив время подбора аналитической зависимости, наилучшим образом описывающей известные значения ВР, и могут быть рекомендованы для решения задач краткосрочного прогнозирования (на 1-3 шага вперед).

Предлагаемый подход к проблеме прогнозирования процессов, представленных ВР с короткой актуальной частью, основанный на использовании рекурсивной процедуры формирования аналитических зависимостей, реализующей корректное кодирование антител с помощью бинарных деревьев в МАКО, и обеспечивающий формирование аналитической зависимости, адекватно описывающей известные значения ВР, может быть использован при разработке многофакторных моделей прогнозирования.

Библиографический список

1. *Бидюк П.И., Баклан И.В., Литвиненко В.И., Фефелов А.А.* Алгоритм клонального отбора для прогнозирования нестационарных динамических систем // Штучний інтелект. – 2004. – № 4. – С. 89-99.

2. *Демидова Л.А., Корячко А.В., Скворцова Т.С.* Модифицированный алгоритм клонального отбора для анализа временных рядов с короткой длиной актуальной части // Системы управления и информационные технологии. – 2010. – № 4.1 (42). – С. 131-136.

3. Искусственные иммунные системы и их применение / Под ред. Д. Дасгупты. – М.: ФИЗМАТЛИТ, 2006. – 344 с.

4. *Паклин Н.Б., Орешков В.И.* Бизнес-аналитика: от данных к знаниям. СПб.: Питер, 2009. – 624 с.: ил.

УДК 007:681.512.2

И.Ю. Каширин, С.А. Минашкин

ОНТОЛОГИИ ДЛЯ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ В ИНТЕРАКТИВНЫХ СЕРВИСАХ ИНФОРМАЦИОННЫХ СЕТЕЙ

Предлагается онотологическая модель представления знаний для повышения релевантости информационного поиска и развития сервисов в глобальных информационных сетях. Модель показала свою эффективность в таких практических разработках, как Internet-портал муниципальных услуг, портал системы общего образования, интерактивные сервисы электронной записи на очередь в дошкольные образовательные учреждения и удаленная запись на прием к врачу.

Ключевые слова: *онтологическая модель, Internet-сервис, информационный поиск, дескриптивная логика.*

Введение. Последние два десятилетия в развитии Internet-ресурсов ознаменовались появлением весьма мощных с точки зрения информационного наполнения ресурсов, которые принято именовать порталами. Портал представляет собой множество сайтов глобальной информационной сети, объединенных общей укрупненной тематикой или даже целой отраслью знаний.

Основой портала являются базы данных (БД) и базы знаний (БЗ) с соответствующими средствами актуализации.

При этом БД, как правило, содержит кроме данных CMS (Content Management System)[1] данные прикладных информационных хранилищ. Хранилища составляют основную ценность портала и относятся к поисковому пространству, востребованному конечным пользователем.

БЗ служит надстройкой, целью которой является четко классифицированное описание портала как полиморфического информационного ресурса [2] для повышения релевантности поиска [3] и гибкого управления его интерактивными сервисами [4].

В этой связи возникает настолько же актуальная, насколько и сложная задача описания статических информационных ресурсов и динамических Internet-сервисов средствами представления знаний, которые:

- не должны нарушать концептуальной целостности существующих протоколов глобальной сети;
- должны использовать базовую идею построения конструкций в языках разметки (XML, WSDL, RDF, OWL);
- должны быть действенным средством полиморфического описания информационных ресурсов.

Перечисленные требования позволят получить надежный, достаточно простой и эффективный механизм, упрощающий поиск информации и делающий такой поиск более направленным.

Цель работы: *получение новой онтологической модели представления знаний, использующей концепцию Semantic Web[5] с применением средств языков OWL-S и WSDL, повышающей релевантность информационного поиска и эффективность использования интерактивных Internet-сервисов.*

Теоретические исследования. В качестве главной теоретической концепции используется онтология как базовая модель знаний, описывающая систему оперирования этими знаниями на основе какой-либо формальной системы [6]. Для этого применяется дескриптивная логика.

Дескриптивная логика DL [6,7] как формальная система представляет собой следующую четверку:

$$DL =< ALC, A, P, T >,$$

где ALC – язык выражений логики, A – множество аксиом, P – множество правил вывода и T – множество теорем.

Семантика DL задается в терминах теории множеств на основании того, что любой концепт представляется множеством денотатов [2], а роль – суть бинарное отношение, т.е. множество пар (см. таблицу). В таблице символ I обозначает отображение на область интерпретации Δ^I, являющуюся универсумом денотатов предметной области. Семантика DL модифицируется

введением типизированных концептов:

$C \mid \bullet D$ или $C \mid \bullet D_1 \bullet D_2 ... \bullet D_n$ – концепт типа $(\bullet D_1 \bullet D_2 ... \bullet D_n)$,

$a : C \mid \bullet D$, или $a : C \mid \bullet D_1 \bullet D_2 ... \bullet D_n$ – денотат типа $(\bullet D_1 \bullet D_2 ... \bullet D_n)$.

Типизация концептов дает возможность использовать их полиморфическое (многозначное) толкование [2].

Приведем весьма упрощенный пример онтологического описания из предметной области “научные труды”.

Предполагается, что существует Internet-сервис, выбирающий по полиморфическому описанию предметной области научные статьи, интересующие конечного пользователя сервиса.

Семантика модифицированной логики DL

Конструкция	Синтаксис	Семантика	Пример
Атомарный концепт	A	$A^I \subseteq \Delta^I$	Программа
Атомарная роль	R	$R^I \subseteq \Delta^I \times \Delta^I$	Быть разделом
Типизированный концепт	$C \mid \bullet D$	$C^I \subseteq \{x \mid (x \mid \bullet D^I) \in \Delta^I\}$	Сервис | • Интерактивный Internet-ресурс • Информационный ресурс
Конъюнкция	$C \cap D$	$C^I \cap D^I$	Статья $\cap$ Научный труд
Дизъюнкция	$C \cup D$	$C^I \cup D^I$	Аннотация $\cup$ Название
Отрицание	$\neg C$	$\Delta^I \setminus C^I$	$\neg$ Беллетристика
Существование	$\exists R.C$	$\{x \mid \exists y(x,y) \in R^I \ \& \ y \in C\}$	$\exists$ Иметь интерактивное взаимодействие . Сервис
Всеобщность	$\forall R.C$	$\{x \mid \forall y(x,y) \in R^I \Rightarrow y \in C\}$	$\forall$ Иметь теоретическое значение . Научный труд
Ограничитель $\geq$	$\geq nR$	$\{x \parallel \{y(x,y) \in R^I\} \mid \geq n\}$	$\geq$ 7 Иметь ключевые слова
Ограничитель $\leq$	$\leq nR$	$\{x \parallel \{y(x,y) \in R^I\} \mid \leq n\}$	$\leq$ 1 Иметь автора

Опишем вначале информационные интересы конечного пользователя, *менеджера проекта*, в области интересов которого – статьи по автоматизации проектирования систем с банками данных и SQL-запросами.

Для построения онтологии определим концепты - базовые понятия предметной области. Для построения системы понятий онтологии выразим ее сначала в естественно-языковой форме.

1) Менеджер проекта - есть специалист и одновременно автор статей.

2) Автоматизированная система - есть программа как результат проекта и как тема для статьи.

3) Управление - процесс, которым занимается менеджер проекта, началом процесса является некоторая задача предметной области, а результатом – автоматизированная система.

4) qdPM - есть средство, являющееся инструментом для управления проектами.

5) Поиск - есть сервис для задачи поиска в хранилище публикаций. Конкретный экземпляр поиска – статьи по тематике qdPM.

Далее приводится соответствующее DL-описание предметной области как терминологическая часть БЗ, называемая TBox [7].

Менеджер проекта | • Специалист $\subseteq$ ($\exists$ Элемент. *Управление* | • Система),

Менеджер проекта | • Специалист $\subseteq$ ($\exists$ Элемент. *Наука* | • Автор статей),

Методы управления проектами | • Инструмент $\subseteq$ ($\exists$ Элемент. *Управление* | • Система),

Предметная область | • Аргументы $\subseteq$ ($\exists$ Элемент. *Управление проектами* | • Система),

Задача предметной области | • Начальная ситуация $\subseteq$ ($\exists$ Элемент. *Управление* | • Система).

Автоматизированная система | • Результат ⊆ (∃ Элемент. *Управление* | • Система).

Автоматизированная система | • Тема ⊆ (∃ Элемент. *Статьи* | • Научная статья).

qdPM ⊆ Программа), qdPM – проектирование ⊆ Проектирование,

Поиск ⊆ Сервис, Задача-поиска ⊆ Статьи,

Искомый документ ⊆ Статья | • Тема • qdPM -проектирование,

При правильно организованном хранилище научных трудов каждый из них может быть описан в фактологической части БЗ, называемой ABox [7,8].

“Иванов И.И. Основы qdPM – проектирование ” ⊆ Статья | • Тема • qdPM – проектирование,

”Семенов С.Ю. qdPM для профессионалов” ⊆ Статья | • Тема • qdPM – проектирование,

”Грошев А.И. qdPM – области применения и рынок продаж” | • Тема • qdPM –проектирование | • Тема • Рынок программного обеспечения.

Алгоритм унификации [8-16], проверяющий возможность существования индивидных отношений в онтологии, дает возможность определить множество релевантных запросу статей как экземпляров понятия «Искомый документ».

Типизированная онтология позволяет преодолеть трудность в определении релевантности, заключающуюся в том, с нужной ли точки зрения для пользователя рассматривается его предметная область в анализируемом документе. Например, один специалист рассматривает компьютерную программу как средство автоматизации учреждения, а другой, как один из аналогов для проектирования своей программы.

В то же время полиморфическая типизация концептов отсутствует в базовых языках описания онтологий, таких как RDF и OWL [11,12,13,14]. Для реализации предложенной концепции онтологии необходима практическая разработка инструментальных средств, позволяющих использовать преимущество нового подхода для повышения релевантности поиска.

Экспериментальные исследования. Алгоритм унификации OWL-выражений с полиморфическими термами впервые был получен в 2008 г. и подробно описан в [2]. Задача использования этого алгоритма в качестве одного из сервисов поисковых машин может решаться двумя способами:

- с помощью WSDL–описания сервиса для конкретной предметной области;

- с помощью OWL-S-описания такого сервиса.

Рассмотрим WSDL–описание, которое предполагает использование SOAP-протокола.

В языке WSDL 1.1 используется элемент `<wsdl:definitions>`, в рамках которого в пространстве имен WSDL определены один «пассивный» дочерний элемент и пять «активных» дочерних элементов, которые и составляют описание сервиса:

– `<wsdl:import>` - ссылка на документ WSDL с описаниями, подлежащими включению в этот документ;

– `<wsdl:types>` определяет типы XML или элементы, используемые для обмена сообщениями;

– `<wsdl:message>` определяет фактическое сообщение;

– `<wsdl:portType>` определяет абстрактный набор операций, осуществленных сервисом;

– `<wsdl:binding>` определяет реализацию `<wsdl:portType>` с помощью конкретных протоколов и форматов;

– `<wsdl:service>` определяет сервис в целом, как правило, включая один или несколько элементов `<wsdl:port>` с информацией доступа для элементов `<wsdl:binding>`.

В описании может задействоваться также элемент `<wsdl:document>`, который используется в целях документирования.

```
<wsdl:definitions ...
xmlns:tns="http://testpaper.com/paperi
nf/SearchServer"
targetNamespace=
"http://testpaper.com/paperinf/SearchS
erver">
<wsdl:document>Описание сервиса
определения статей.
</wsdl:document> <wsdl:types>
<xs:schema ...
targetNamespace=
"http://testpaper.com/paperinf/
SearchServer">
<xs:import namespace=
"http://testpaper.com/paperinf/types"
schemaLocation="Papers-types.xsd"/>
</xs:schema>
</wsdl:types>
  <wsdl:message
name="getPapersMessage">
<wsdl:part name="part" element
="tns:getPapers"/>
  </wsdl:message>
  <wsdl:message
name="getPapersResponseMessage">
    <wsdl:part name="part"
element="tns:getPapersResponse"/>
```

```
 </wsdl:message>
 <wsdl:message
name="addPapersMessage">
<wsdl:part name="part"
element="tns:addPapers"/>
</wsdl:message>
<wsdl:message name =
"addPapersResponseMessage">
<wsdl:part name="part"
element="tns:addPapersResponse"/>
/wsdl:message>
<wsdl:message
name="addDuplicateFault">
<wsdl:part name="fault"
element="tns:addDuplicate"/>
</wsdl:message>
<wsdl:portType
name="PapersServerPortType">
<wsdl:documentation>
Сервис реализации для помещения в
информационное хранилище новых статей.
</wsdl:documentation>
<wsdl:operation name="getPapers">
<wsdl:documentation>
Поиск статьи с необходимым описанием.
</wsdl:documentation>
<wsdl:input message=
"tns:getPapersMessage"/>
<wsdl:output message=
"tns:getPapersResponseMessage"/>
</wsdl:operation>
<wsdl:operation name="addPapers">
<wsdl:documentation>Добавление
статьи.</wsdl:documentation>
<wsdl:input message=
"tns:addPapersMessage"/>
<wsdl:output message=
"tns:addPapersResponseMessage"/>
<wsdl:fault message=
"tns:addDuplicateFault"
name="addDuplicateFault"/>
</wsdl:operation>
</wsdl:portType>
</wsdl:definitions>
```

Здесь в разделе <wsdl:definitions> определяется пространство имен с полиморфической типизацией. Предполагается также, что реализация сервиса включает поисковый алгоритм, описанный в [2].

Далее приведем сокращенное описание нового сервиса с помощью языка OWL-S.

Описание интеллектуального сервиса задается следующим образом:

```
<service:Service
rdf:ID="IntellPaperFinderService">
   <service:presents
rdf:resource="#IntellPaperFinderProfil
e"/>
   <service:describedBy
rdf:resource="#IntellPaperFinderProces
s"/>
   <service:supports
rdf:resource="#IntellPaperFinderGround
ing"/>
</service:Service>
```

Профиль сервиса (входные и выходные данные) описываются, например, так:

```
<profile:Profile
rdf:ID="IntellPaperFinderProfile">
   <service:presentedBy
rdf:resource="#IntellPaperFinderServic
e"/>
   <profile:serviceName
xml:lang="en">Intell Paper

Finder</profile:serviceName>
   <profile:textDescription
xml:lang="en">
```

Этот сервис возвращает информацию о статье, заголовок которой лучше всего соответствует строке, заданной через Intell-сервис.

```
</profile:textDescription>
<profile:hasInput
rdf:resource="#PaperName"/>
<profile:hasOutput
rdf:resource="#ItemInfo"/>
</profile:Profile>
```

Приведем также описание простейшего процесса передачи данных:

```
<process:AtomicProcess
rdf:ID="IntellPaperFinderProcess">
   <service:describes
rdf:resource="#IntellPaperFinderServic
e"/>

   <process:hasInput
rdf:resource="#PaperName"/>
   <process:hasOutput
rdf:resource="#ItemInfo"/>
</process:AtomicProcess>
```

Задание входных и выходных параметров процесса описывается следующим образом:

```
<process:Input rdf:ID="PaperName">
   <process:parameterType
rdf:datatype=
"&xsd;#anyURI">&xsd;#string
</process:parameterType>
<rdfs:label>Paper Name</rdfs:label>
</process:Input>
<process:Output rdf:ID="ItemInfo">
   <process:parameterType
rdf:datatype=
"&store;#StockItem
</process:parameterType>
 <rdfs:label>Item Info</rdfs:label>
</process:Output>
```

Выводы. Приведенные WSDL и OWL-S описания дают возможность использовать новую модифицированную онтологию с полиморфическим представлением концептов для повышения

релевантности информационного поиска в современных глобальных сетях.

Практика применения предложенной онтологии с соответствующим описанием сервиса показала свою эффективность при проектировании таких сервисов, как:

- Internet-портал муниципальных услуг (повышение релевантности поиска на 1,3 %);
- портал системы общего образования (повышение релевантности примерно на 4 %);
- интерактивные сервисы электронной записи на очередь в дошкольные образовательные учреждения и удаленная запись на прием к врачу (повышение релевантности на 0,5 %).

Все перечисленные Internet-ресурсы внедрены на серверах, арендуемых администрацией города Рязани.

Библиографический список

1. *Вильямсон Х.* Универсальный Dynamic HTML. Библиотека программиста / Вильямсон Х.-СПб.: Питер, 2001. - 304 с.

2. *Каширин Д.И., Каширин И.Ю., Пылькин А.Н.* Полиморфическое представление знаний в Semantic Web: Монография. М: Горячая линия- Телеком, 2009. 136 с.

3. *Манцивода А.В., Малых А.А.* Представление и обработка знаний в Интернете. Иркутский государственный университет, 2005. 103 с.

4. *Fensel D., Lausen H., Polleres A., Bruijn J., Stollberg M., Roman D., Domingue J.* Enabling Semantic Web Services. The Web Service Modeling Ontology. - Springer-Verlag, Berlin Heidelberg 2007 – 188 p.

5. *Bruijn J., Polleres A., Lara R., Fensel D.* OWL DL vs. OWL Flight: Conceptual modeling and reasoning on the SemanticWeb. In Proceedings of the 14th International World Wide Web Conference (WWW2005), Chiba, Japan, 2005.

6. *Каширин И.Ю., Коричнев Л.П.* Основы формального анализа интеллектуальных программных систем.-М.:Радио и связь, 1997.–160 с.

7. *Abdul-Ghafour S., Ghodous P., Shariat B., Perna E.* A Common Design-Features Ontology for Product Data Semantics Interoperability // IEEE/WIC/ACM International Conference on Web Intelligence (WI'07), 2007. - P. 443-446.

8. *Knight K.* Unification: A Multidisciplinary Survey. - ACM Computing Surveys. - 1989. - V. 21. – N 1. - P. 93-124.

9. *Fensel D., Harmelen F., Horrocks I., McGuinness D., Patel-Schneider P. F.* OIL: An Ontology Infrastructure for the Semantic Web. IEEE Intelligent Systems, 16(2), 2001. - 38-45pp.

10. *Abdul-Ghafour S., Ghodous P., Shariat B., Perna E.* A Common Design-Features Ontology for Product Data Semantics Interoperability // IEEE/WIC/ACM International Conference on Web Intelligence (WI'07), 2007. - P. 443-446.

11. *Abdul-Ghafour S., Ghodous P., Shariat B., Perna E.* A Common Design-Features Ontology for Product Data Semantics Interoperability // IEEE/WIC/ACM International Conference on Web Intelligence (WI'07), 2007. - P. 443-446.

12. *Bullinaria J.A., Huckle C.C.* Modelling lexical decision using corpus derived semantic representations in a connectionist network. In Proceedings of the Fourth Neural Computation and Psychology Workshop, 1997.

13. *Bryson J.,* at al. Agent-Based Composite Services in DAML-S. - 2002, - Режим доступа: www.cs.batch.ac.uk/~jjb/ftp/springer-daml.pdf

14. *Bruijn J., Polleres A., Lara R., Fensel D.* OWL DL vs. OWL Flight: Conceptual modeling and reasoning on the SemanticWeb. In Proceedings of the 14th International World Wide Web Conference (WWW2005), Chiba, Japan, 2005.

15. *Артемьева И.Л., Высоцкий В.И., Решетанко Н.В.* Модель онтологии предметной области (на примере органической химии) // НТИ, сер. 2. 2005. № 8. С. 19-27.

16. *Клещёв А.С., Москаленко Ф.М., Черняховская М.Ю.* Модель онтологии предметной области "Медицинская диагностика". В 2-х частях // НТИ. Сер. 2. Часть 1: 2005. № 12. С.1-7. Часть 2: 2006. № 2. С. 19-30.

УДК 519.179.2

А.П. Шибанов

МЕТОД ЭКВИВАЛЕНТНЫХ УПРОЩАЮЩИХ ПРЕОБРАЗОВАНИЙ GERT-СЕТЕЙ И ЕГО ПРИЛОЖЕНИЯ

Рассматриваются результаты теоретических исследований в области GERT-сетей и возможности их применения на практике. Излагаются основы метода эквивалентных упрощающих преобразований GERT-сети, построение иерархических GERT-сетей,GERT-сетей со старением заявок и GERT-сетей со сложными распределениями. Описывается использование компьютерных программ, реализующих метод эквивалентных упрощающих преобразований, для оптимизации производственных процессов и построения комбинированных систем имитационного моделирования.

Ключевые слова: *GERT-сеть, производящая функция моментов, характеристическая функция, старение заявок, сложные распределения, имитационное моделирование.*

Введение. За последние годы для моделирования и оптимизации технических систем все большее распространение получают GERT-сети (GERT − graphical evaluation and review technique), предложенные А. Прицкером [1]. Они являются разновидностью полумарковских моделей, но для случайных величин, характеризующих отдельные дуги, задается не только дисперсия, но и закон распределения. Выполнение отдельных операций в технической системе связывается с дугами GERT-сети, которые характеризуются аддитивными случайными величинами. Выбор каждой следующей дуги в общем случае – вероятностный. Для расчета выходных характеристик GERT-сетей используются производящие функции моментов случайных величин, характеризующих дуги. При использовании GERT-сетей, как правило, определяют значения переменных, связанных с первыми моментами распределения выходной величины относительно начала координат (математическое ожидание, дисперсию, коэффициенты асимметрии и эксцесса).

Одной из наиболее важных проблем теории GERT-сетей является получение точного распределения выходной величины GERT-сети. Ее решение открывает возможность получения ряда новых результатов:

– построение иерархических GERT-сетей;

– использование GERT-сетей с условными распределениями, в частности GERT-сетей со старением заявок;

– использование GERT-сетей со сложными распределениями (со случайным числом случайных слагаемых);

– решение задач выявления скрытого параллелизма в моделируемом процессе с целью его оптимизации за счет уменьшения дисперсии выходной величины процесса;

– создание комбинированных систем моделирования, в которых нижний иерархический уровень представлен GERT-сетями, а верхний – стандартными средствами имитационного моделирования.

Основной выходной характеристикой GERT-сети является эквивалентная производящая функция моментов $M_E(s)$. Для нахождения значений $M_E(s)$ используется топологическое уравнение Мейсона [1] для замкнутых потоковых графов. Формула Мейсона позволяет получить эквивалентную производящую функцию моментов GERT-сети через эквивалентные производящие функции моментов случайных величин, характеризующих петли разных порядков. Петлей первого порядка называется связная последовательность ориентированных дуг, каждый узел которых является общим ровно для двух дуг. Петля порядка n: множество n не связанных между собой петель первого порядка. С увеличением размерности GERT-сети резко возрастает число петель разных порядков. Методы анализа GERT-сетей, в которых используется уравнение Мейсона, являются полиномиально неразрешимыми. Для решения этой проблемы целесообразно искать такие подходы, в которых топологическое уравнение Мейсона не используется.

Базовый теоретический результат – метод эквивалентных упрощающих преобразований GERT-сети. Рассмотрим численный метод нахождения распределения времени прохождения GERT-сети, основанный на применении формулы обращения и интерполяции характеристической функции GERT-сети многочленом Лагранжа второй степени. Нахождение плотности распределения вероятностей выходной величины GERT-сети основано на преобразованиях структуры GERT-сети с пересчетом значений характеристических функций эквивалентных дуг в узлах интерполяции. Дуги GERT-сети могут характеризоваться произвольными дискретными и непрерывными законами распределения, а выходная величина GERT-сети должна быть непрерывной.

Преобразование исходной GERT-сети к эквивалентной дуге. По известным W-функциям двух дуг находим характеристические функции фрагментов: последовательно соединенные дуги, параллельные дуги, дуга и петля первого порядка, соединяющая выход и вход узла.

Структуру GERT-сети G отобразим в матрице весов $\mathbf{R}(G)$ с n узлами. Каждый элемент матрицы r_{ij} содержит: вероятность p_{ij} выбора дуги (i,j) (i – номер строки, j – номер столбца, $i, j = \overline{1,n}$) и массивы значений действительных $\mathrm{Re}\,\chi_{ij}(\zeta)$ и мнимых $\mathrm{Im}\,\chi_{ij}(\zeta)$ частей характеристической функции дуги (i,j) в l узлах интерполяции. Возьмем произвольный узел GERT-сети v_k. В него входят дуги из узлов $v_1, \ldots, v_{k-1}$ и выходят дуги в узлы $v_{k+1}, \ldots, v_h$. Кроме того, имеется дуга (v_k, v_k). Узлы

$v_1, \ldots, v_{k-1}, v_k, v_{k+1}, \ldots, v_h$ и соединяющие их дуги составляют фрагмент G_1 GERT-сети. Матрица смежности $\mathbf{A_1}$ этого фрагмента выглядит следующим образом:

$$\mathbf{A}_1 = \begin{array}{c} \begin{matrix} v_1 & \ldots & v_{k-1} & v_k & v_{k+1} & v_{k+2} & \ldots & v_h \end{matrix} \\ \begin{bmatrix} 0 & \ldots & 0 & 1 & 0 & 0 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & \ldots & 0 & 1 & 0 & 0 & \ldots & 0 \\ 0 & \ldots & 0 & 1 & 1 & 1 & \ldots & 1 \\ 0 & \ldots & 0 & 0 & 0 & 0 & \ldots & 0 \\ 0 & \ldots & 0 & 0 & 0 & 0 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & \ldots & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{matrix} v_1 \\ \ldots \\ v_{k-1} \\ v_k \\ v_{k+1} \\ v_{k+2} \\ \ldots \\ v_h \end{matrix} \end{array}.$$

Выделим фрагмент GERT-сети, состоящий из узлов $v_k, v_{k+1}, \ldots, v_h$ и дуг $(v_k, v_k), (v_k, v_{k+1}), \ldots, (v_k, v_h)$. Суммируя W-функции дуг, составляющих всевозможные пути из узла v_k в узел v_{k+1}, получаем эквивалентную W-функцию $W_{E\,k,k+1}$ дуги (v_k, v_{k+1}): $W_{E\,k,k+1} = W_{k,k+1} + W_{k,k} W_{k,k+1} + W_{k,k}^2 W_{k,k+1} + \ldots =$ $= W_{k,k+1} / (1 - W_{k,k})$. Аналогично получаем эквивалентные W-функции $W_{E\,k,k+2}, \ldots, W_{E\,k,h}$ дуги $(v_k, v_{k+2}), \ldots, (v_k, v_h)$: $W_{E\,k,k+2} = W_{k,k+2} / (1 - W_{k,k}), \ldots$ $\ldots, W_{E\,k,h} = W_{k,h} / (1 - W_{k,k})$. Петля первого порядка (v_k, v_k) исключается из сети, а W-функции дуг $(v_k, v_{k+1}), \ldots, (v_k, v_h)$ заменяются эквивалентными функциями $W_{E\,k,k+1}, \ldots, W_{E\,k,h}$. Так как в GERT-сети выполняется условие аддитивности по дугам любого пути, то любой узел можно дублировать, порождая число его копий, равное числу дуг, входящих в узел. Каждая копия $v_{k_1}, \ldots, v_{k_{k-1}}$ узла v_k имеет то же множество выходных дуг $(v_k, v_{k+1}), \ldots, (v_k, v_h)$ и соответствующих им вероятностей исполнения $p_{k+1}, \ldots, p_h$, что и узел v_k. При этом в каждую копию узла v_k из множества $v_1, \ldots, v_{k-1}$ входит одна дуга. Назовем такое дублирование дублированием узла v_k *с сохранением выходных дуг*. Полученный после дублирования фрагмент обозначим через G_2 с матрицей смежности

$$\mathbf{A}_2 = \begin{array}{c} \begin{matrix} v_1 & \ldots & v_{k-1} & v_{k_1} & \ldots & v_{k_{k-1}} & v_{k+1} & v_{k+2} & \ldots & v_h \end{matrix} \\ \begin{bmatrix} 0 & \ldots & 0 & 1 & \ldots & 0 & 0 & 0 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & \ldots & 0 & 0 & \ldots & 1 & 0 & 0 & \ldots & 0 \\ 0 & \ldots & 0 & 0 & \ldots & 0 & 1 & 1 & \ldots & 1 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & \ldots & 0 & 0 & \ldots & 1 & 1 & 0 & \ldots & 1 \\ 0 & \ldots & 0 & 0 & \ldots & 0 & 0 & 0 & \ldots & 0 \\ 0 & \ldots & 0 & 0 & \ldots & 0 & 0 & 0 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & \ldots & 0 & 0 & \ldots & 0 & 0 & 0 & \ldots & 0 \end{bmatrix} \begin{matrix} v_1 \\ \ldots \\ v_{k-1} \\ v_{k_1} \\ \ldots \\ v_{k_{k-1}} \\ v_{k+1} \\ v_{k+2} \\ \ldots \\ v_h \end{matrix} \end{array}$$

Столбцы $v_{k_1}, \ldots, v_{k_{k-1}}$ матрицы $\mathbf{A_2}$ содержат по одной единице. Из фрагмента G_2 выделим подграфы $\widetilde{G}_{k_1}, \ldots, \widetilde{G}_{k_{k-1}}$, состоящие из узлов: $\{v_1, v_{k_1}, v_{k+1}, \ldots, v_h\}, \ldots, \{v_{k-1}, v_{k_{k-1}}, v_{k+1}, \ldots, v_h\}$. Их структура одинакова и дальнейшие их преобразования идентичны. Поэтому рассмотрим преобразования только подграфа $\widetilde{G}_{k_1}$. Его матрица смежности

$$A_{k_1} = \begin{array}{c} \begin{matrix} v_1 & v_{k_1} & v_{k+1} & v_{k+2} & \ldots & v_h \end{matrix} \\ \begin{bmatrix} 0 & 1 & 0 & 0 & \ldots & 0 \\ 0 & 0 & 1 & 1 & \ldots & 1 \\ 0 & 0 & 0 & 0 & \ldots & 0 \\ 0 & 0 & 0 & 0 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & 0 & 0 & \ldots & 0 \end{bmatrix} \begin{matrix} v_1 \\ v_{k_1} \\ v_{k+1} \\ v_{k+2} \\ \ldots \\ v_h \end{matrix} \end{array}.$$

Скопируем узел $v_{k_1} \in \widetilde{G}_{k_1}$ так, чтобы из каждой копии узла $v_{k_{1,1}}, \ldots, v_{k_{1,h}}$ в каждый из узлов $v_{k+1}, \ldots, v_h$ входила бы одна дуга. Узел v_1 соединяется дугами с узлами $v_{k_{1,1}}, \ldots, v_{k_{1,h}}$. Назовем такое дублирование дублированием узла v_{k_1} *с копированием входных дуг*, после чего подграф $\widetilde{G}_{k_1}$ трансформируется в подграф $\breve{G}_{k_1}$ с матрицей смежности

$$v_1 \quad v_{k_{1,1}} \; v_{k_{1,2}} \quad \ldots \quad v_{k_{1,h}} \, v_{k+1} \, v_{k+2} \ldots \, v_k$$

$$\breve{\mathbf{A}}_{k_1}=\begin{bmatrix} 0 & 1 & 1 & \dots & 1 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 & 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & 0 & 0 & 0 & \dots & 1 \\ 0 & 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 \end{bmatrix} \begin{matrix} v_1 \\ v_{k_{1,1}} \\ v_{k_{1,2}} \\ \dots \\ v_{k_{1,h}} \\ v_{k+1} \\ v_{k+2} \\ \dots \\ v_k \end{matrix}.$$

Каждый из множества узлов $v_{k_{1,1}},\dots,v_{k_{1,h}}$ подграфа $\breve{G}_{k_1}$ имеет одну входящую и одну выходящую дуги. Аналогично каждый узел из множеств $\left\{v_{k_{2,1}},\dots,v_{k_{2,h}}\right\},\dots,\left\{v_{k_{h,1}},\dots,v_{k_{h,h}}\right\}$ подграфов $\breve{G}_{k_2},\dots,\breve{G}_{k_h}$ также имеет одну входящую и одну выходящую дуги. Последовательные дуги заменяются эквивалентными, и узел v_k исключается из GERT-сети. При этом размерность матрицы $\mathbf{A}(G)$ уменьшается на единицу.

Любая GERT-модель приводится к единственной эквивалентной дуге по следующему алгоритму:

1) произвольно выбирается узел v_k, отличный от источника и стока. Если имеется дуга (v_k,v_k), то значения характеристических функций дуг $(v_k,v_{k+1}),\dots,(v_k,v_h)$, выходящих из узла v_k, пересчитываются на эквивалентные, и дуга (v_k,v_k) исключается;

2) узел v_k дублируется *с сохранением выходных дуг*;

3) каждый продублированный в пункте 3 узел дублируется *с копированием входных дуг*;

4) последовательные дуги заменяются эквивалентными, и узел v_k исключается из GERT-сети;

5) если в процессе преобразований в GERT-сети образуются последовательные или параллельные дуги, то они заменяются эквивалентными дугами с пересчетом значений их характеристических функций;

6) если GERT-сеть G не приведена к эквивалентной дуге, то выполняется переход на пункт 1, в противном случае алгоритм заканчивается.

Вычисление плотности распределения вероятностей выходной величины GERT-сети. Рассматривается численный метод нахождения непрерывной плотности распределения вероятностей времени прохождения GERT-сети при условии, что множество распределений, которыми могут характеризоваться отдельные дуги модели, включает в себя наиболее часто используемые на практике распределения: дискретное, биномиальное, пуассоновское, геометрическое, отрицательное биномиальное, равномерное, экспоненциальное, гамма- и нормальное распределения. Кроме этого, можно использовать и непрерывные распределения произвольного вида, заданные на ограниченном интервале.

Перейдем от эквивалентной W-функции $W_E(s)$ GERT-сети к ее характеристической функции $\chi_E(\zeta)$ и используем формулу обращения. Плотность распределения вероятностей выходной величины GERT-сети

$$f(x)=\frac{1}{2\pi}\int_{-\infty}^{\infty}e^{-i\zeta x}\chi_E(\zeta)d\zeta. \qquad (1)$$

Функция $\chi_E(\zeta)$ находится с использованием метода эквивалентных упрощающих преобразований, в результате проведения которых остается единственная эквивалентная дуга GERT-сети.

Вычислим значения плотности распределения вероятностей $f(x)$ в точке $x=0$. При этом формула (1) принимает вид

$$f(0)=\frac{1}{2\pi}\int_{-\infty}^{\infty}\chi_E(\zeta)\,d\zeta.$$

Выполним интерполирование функции $\chi_E(\zeta)$ на отрезках $\left[d_k-\varepsilon,d_k+\varepsilon\right]$, $k=\overline{1,p}$ многочленом Лагранжа $L_2(\zeta)$ второй степени с равноотстоящими узлами (d_k – середина отрезка интерполяции) в пределах от $\left[-L,L\right]$. Если случайная величина ξ имеет плотность $f(x)$, то $\left|\chi_E(\zeta)\right|\to 0$ при $\zeta\to\pm\infty$. В целом же скорость стремления к нулю функции $\left|\chi_E(\zeta)\right|$ при $\zeta\to\pm\infty$ тем выше, чем выше гладкость $f(x)$. В результате имеем $\chi_{Ek}(\zeta)\approx L_{2k}(\zeta)$ и

$$f(0)\approx\frac{1}{2\pi}\sum_{k=1}^{p}\int_{d_k-\varepsilon}^{d_k+\varepsilon}L_{2k}(\zeta)\,d\zeta=$$

$$=\frac{1}{2\pi}\sum_{k=1}^{p}\int_{d_k-\varepsilon}^{d_k+\varepsilon}\left(U_{0k}+U_{1k}\zeta+U_{2k}\zeta^2\right)d\zeta, \qquad (2)$$

где U_{0k},U_{1k},U_{2k} – значения коэффициентов при степенях ζ многочлена Лагранжа $L_{2k}(\zeta)$ функции $\chi_{Ek}(\zeta)$ на отрезке $\left[d_k-\varepsilon,d_k+\varepsilon\right]$.

Поскольку мнимая часть характеристической функции является нечетной, то

$$f(0)\approx\frac{1}{2\pi}\int_{-L}^{L}\chi_E(\zeta)\,d\zeta=\frac{1}{2\pi}\int_{-L}^{L}\chi_{ER}(\zeta)\,d\zeta\ .$$

Выполняя интегрирование формулы (2), получаем

$$f(0)\approx\frac{\varepsilon}{\pi}\sum_{k=1}^{p}\left(U_{0k}+U_{1k}\,d_k+U_{2k}\,\frac{3d_k^{\,2}+\varepsilon^2}{3}\right).$$

Найдем значения плотности распределения вероятностей $f(x_\tau)$ в точках $x_\tau, \tau=\overline{1,q}$. Будем последовательно смещать плотность распределения по оси x на заранее заданную величину. Значения плотности распределения вероятностей находятся посредством рекуррентного умножения значений характеристической функции $\chi_E(\zeta)=\chi_{ER}(\zeta)+i\,\chi_{EI}(\zeta)$, вычисленных на предыдущем шаге, на величину $\exp(i\zeta T)=$ $=\cos\zeta T+i\sin\zeta T$, где $T-$ шаг изменения аргумента x.

Асимптотическая временная сложность алгоритма есть $O(n^4)$, где n – число узлов GERT-сети.

Предложенный численный метод можно использовать для нахождения закона распределения времени прохождения иерархической GERT-сети. Соединяя нужным образом источники и стоки сетей нижележащего уровня, получаем GERT-сеть более высокого уровня иерархии. После выполнения упрощающих преобразований фрагментов GERT-сети сохраняются значения их характеристических функций в узлах интерполяции при $x=0$. Эта сеть, в свою очередь, методом упрощающих преобразований приводится к эквивалентной дуге. После этого может быть вычислена плотность распределения времени прохождения иерархической GERT-сети при любых значениях аргумента x.

Основные результаты данного раздела опубликованы в работе [2]. Важность полученного результата отмечается в работах [3 – 5]. Дополнительная информация о методе содержится в работах [6 – 8].

GERT-сети с условными распределениями. Среди множества узлов сети выделяются такие узлы, в которых выполняется анализ плотности (или функции) распределения вероятностей времени движения заявки до попадания в данный узел. Полагаем, что безусловные функции распределения дуг, для которых проверяются условия попадания в интервалы, являются экспоненциальными. Рассмотрим три частных случая.

1. Найдена производящая функция моментов распределения $M^{(c)}(s)$ при условии, что заявки не устарели до момента a_1, $P\{\xi\le a_1\}$:

$$M^{(c)}(s)=\frac{\lambda}{\lambda-s}\cdot\frac{1-e^{-a_1(\lambda-s)}}{1-e^{-a_1\lambda}}. \qquad (3)$$

2. Производящая функция моментов распределения $M^{(c)}(s)$ при условии, что заявки попали в заданный интервал времени $[a_{k-1},\,a_k]$, $P\{a_{k-1}\le\xi\le a_k\}$, определяется выражением

$$M^{(c)}(s)=\frac{\lambda}{\lambda-s}\cdot\frac{e^{-a_1(\lambda-s)}-e^{-a_2(\lambda-s)}}{e^{-a_1\lambda}-e^{-a_2\lambda}}. \qquad (4)$$

3. Производящая функция моментов распределения при условии, что заявки *не уложились в заданное время* a_n, $P\{\xi\ge a_n\}$, равна

$$M^{(c)}(s)=\frac{\lambda}{\lambda-s}\cdot e^{a_n s}. \qquad (5)$$

Использование дуг с условными производящими функциями моментов (3) – (5) позволяет моделировать операции над объектами, в которых проводится контроль на попадание случайного события в заданные интервалы. По результатам этого контроля выполняются определенные действия, например исключение из дальнейшего рассмотрения объектов, для которых условия (3) – (5) оказались не выполнены. Для практического использования GERT-сетей со старением заявок необходимо выполнить переход от производящих функций моментов $M^{(c)}(s)$ к соответствующим характеристическим функциям $\chi^{(c)}(\varsigma)$. Более подробно тематика применения GERT-сетей со старением информации излагается в [9, 10].

GERT-сети со сложными распределениями. При моделировании часто возникает необходимость использования распределений сумм случайного числа случайных слагаемых. Например, можно найти распределение случайного времени передачи множества подряд посылаемых по телекоммуникационному каналу кадров, если известно распределение числа кадров в этом множестве и распределение времени передачи одного кадра. То есть рассматривается прохождение через GERT-сеть не одной заявки, а множества заявок, которое представляет собой достаточно длинное сообщение.

Пусть ξ_1, $\xi_2\ldots$ – независимые случайные величины с одним и тем же распределением F и характеристической функцией $\chi(\zeta)$. Пусть N – случайная целочисленная величина с произ-

водящей функцией $A(s)=\sum p_k s^k$, не зависящая от всех ξ_j. Тогда сумма $\xi_1+\ldots+\xi_N$ имеет характеристическую функцию $\omega(\zeta)=A(\chi(\zeta))$. Аналогично для производящей функции моментов имеем $\widetilde{M}(s)=A(M(s))$.

Примеры. Производящая функция распределения Пуассона равна $\exp[\lambda(s-1)]$. Подставляя в это выражение вместо s формулу для производящей функции моментов случайной величины ξ, распределенной по закону Эрланга с параметрами λ и l, получаем:

$$\widetilde{M}(s)=\exp\left(\lambda\left\{\left[\lambda^l/(\lambda-s)^l\right]-1\right\}\right).$$

Если величина N распределена по биномиальному закону с параметрами p, $q=1-p$ и n, то она имеет производящую функцию $(ps+q)^n$. Тогда, если ξ распределена равномерно,

$$\widetilde{M}(s)=\left\{p\left[\left(e^{as}-e^{bs}\right)/(a-b)s\right]+q\right\}^n.$$

Если величина N распределена по отрицательному биномиальному закону с параметрами p, $q=1-p$ и r, то она имеет производящую функцию $p^r/(1-qs)^r$. Если ξ имеет нормальное распределение, то справедливо выражение $\widetilde{M}(s)=p^r/\left(1-qe^{ms+0,5\sigma^2s^2}\right)^r$.

При решении практических задач сочетаний различных случайных величин N и ξ может встретиться достаточно много.

Для любой из вышеперечисленных производящих функций моментов $\widetilde{M}(s)$ с заменой переменных $s=i\zeta$ могут быть вычислены характеристическая функция $\omega(\zeta)$ и значения ее действительной и мнимой частей в узлах интерполяции. Распределения на основе суммы случайного числа случайных слагаемых можно использовать в GERT-сети наравне с основными наиболее часто применяемыми на практике распределениями. Для GERT-сети с расширенным множеством распределений может быть найдена плотность вероятности выходной величины GERT-сети с использованием метода эквивалентных упрощающих преобразований.

Экспериментальная часть. Для проверки корректности полученных научных результатов была создана компьютерная программа моделирования GERT-сетей "GERT Explorer". Программа обладает развитым графическим интерфейсом. Реализованы функции ввода исходной информации пользователем, корректировки модели, копирования фрагментов модели в буфер, масштабирования изображения на экране, перемещения изображения модели в пределах экрана, изменения формы и цветового изображения дуг и узлов, переименования узлов и дуг и т.п. Имеется возможность подготовки исходных данных в нескольких окнах. Реализован англоязычный интерфейс. Выдается необходимая статистическая информация о результатах моделирования в виде таблиц и графиков.

Программа была реализована в нескольких версиях на разных программно-технических платформах с постоянным наращиванием ее функциональных возможностей. Она неоднократно использовалась при проведении учебного процесса, опытно-конструкторских и научно-исследовательских работ.

Выявление внутреннего параллелизма в системе. Рассмотрим GERT-модель производственного процесса (рисунок 1). Характеристики дуг, отражающих время выполнения отдельных операций, приведены в таблице.

Разложим исходную GERT-сеть на параллельные компоненты, состоящие из дуг: ((1,2), (2,3), (3,4), (4,6)); ((1,2), (2,3), (3,5), (5,8), (8,6)); ((1,2), (2,7), (7,8), (8,6)). Эквивалентная W-функция GERT-сети равна сумме эквивалентных W-функций параллельных компонент:

$$W_E(s)=W_{E1}+W_{E2}+W_{E3},\ W_{E1}=W_1W_2W_3W_6,$$
$$W_{E2}=W_1W_2W_5W_7W_8,\ W_{E3}=W_1W_4W_9W_8.$$

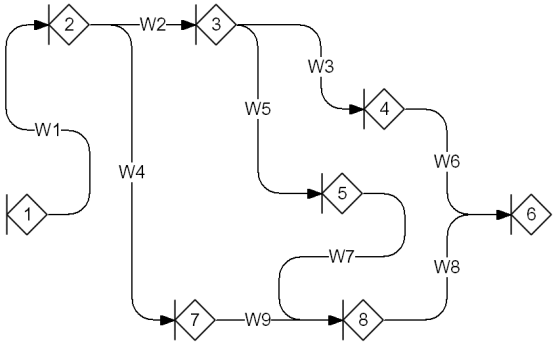


Рисунок 1 – Модель производственного процесса

Дуга	p	Распределение	$M(s)$
$W1$	1	Нормальное	$\exp\left(4s+0,5s^2\right)$
$W2$	0,6	Биномиальное	$\left(0,5e^s+0,5\right)^3$
$W3$	0,3	Равномерное	$\left(e^{2s}-1\right)/2s$
$W4$	0,4	Нормальное	$\exp\left(10s+0,5s^2\right)$
$W5$	0,7	Нормальное	$\exp\left(12s+0,5s^2\right)$
$W6$	1	Пуассона	$\exp[\exp(s)-1]$
$W7$	1	Биномиальное	$\left(0,5e^s+0,5\right)^4$
$W8$	1	Геометрическое	$0,5/\left(1-0,5e^s\right)$
$W9$	1	Равномерное	$\left(e^{2s}-e^s\right)/s$

Рисунок 2 – Плотности распределения времени прохождения исходной и оптимизированной GERT-сетей

На рисунке 2 изображена многомодовая плотность распределения выходной величины исходной GERT-сети, полученная с помощью разработанной компьютерной программы. Математическое ожидание времени ее прохождения равно 16,5 ед., а интервал “трех сигм” – 17 ед. Причиной достаточно большого разброса выходной случайной величины GERT-сети является наличие скрытого параллелизма процесса.

Рисунок 3 – Плотности распределения времени прохождения параллельных компонент GERT-сети

На рисунке 3 изображены плотности распределений отдельных параллельных компонент процесса. Длительности прохождения компонент, в общем случае, существенно различаются. Большинство случайных реализаций времени прохождения первой компоненты GERT-сети попадает в интервал (1,4; 12), второй и третьей компонент – в интервалы (15,2; 30,2) и (10,8; 26) соответственно. Чем в большей степени эти интервалы совпадают, тем меньше будет дисперсия времени прохождения GERT-сети. Таким образом, выявление внутреннего параллелизма в GERT-сети создает предпосылки для разработки алгоритмов оптимизации параметров системы за счет уменьшения дисперсии выходной величины GERT-сети. Если поставить цель уменьшения разброса выходной случайной величины, то можно, взяв, например, компоненту 3 за базовую (в этой компоненте ничего не изменяется), попытаться сдвинуть по оси времени две другие плотности распределения. Первая компонента обладает естественным запасом по времени. Поэтому ее плотность смещается на заданную величину вправо (используется задержка в 10 ед.). Пройти быстрее компоненту 3 можно лишь с использованием дополнительных ресурсов, например за счет модернизации, связанной с установкой более быстродействующего оборудования (в модели уменьшено математическое ожидание нормального распределения, характеризующего дугу (3,5) с 12 ед. до 8 ед.). Плотность распределения вероятностей времени исполнения оптимизированного производственного процесса приведена на рисунке 2. Интервал “трех сигм”, характеризующий разброс выходной случайной величины, сократился с 17 ед. до 5,67 ед.

Знание точных распределений времени прохождения параллельных компонент GERT-сети, полученных с использованием численного метода эквивалентных упрощающих преобразований, позволяет с высокой степенью точности найти интервалы, в пределах которых изменяются выходные величины каждой из параллельных компонент. На практике это позволяет выработать меры для максимально возможного совмещения рассматриваемых интервалов.

Визуальные системы имитационного моделирования с использованием GERT-сетей. Созданы системы имитации с функциями GERT-сетей, оснащенные развитым графическим интерфейсом с пользователем. Структурно нижний уровень иерархии системы моделирования представлен множеством GERT-моделей, выходные данные которых используются на следующем иерархическом уровне – в системе имитации. Функции распределения выходных величин GERT-сетей записываются в файловую систему в виде таблиц. Имитационная система считывает значения из этих таблиц и выполняет имитацию с использованием метода “обратной функции”. Выборки из случайных величин, характеризующих выходы GERT-сетей, используются при имитации: как задержи в обслуживающих приборах систем массового обслуживания; в блоках-генераторах для задания длин заявок и интервалов их следования; для реализации логических функций с переменными времени; для определения числа активных входов и выходов и т.п.

Использование GERT-сетей в системах имитации существенно расширяет функциональные возможности исследователей. Распреде-

ление случайной величины, используемой при имитации, является результатом анализа процесса, который состоит из последовательно выполняемых операций, характеризуемых случайными величинами, с вероятностным, в общем случае, выбором каждой новой операции. Распределение данной случайной величины получается на основе структурных эквивалентных преобразований GERT-сети сведением ее к единственной эквивалентной дуге. Альтернативой этому является использование метода Монте-Карло и для "вставок" из GERT-моделей в имитационной системе. А это приводит к существенному проигрышу как по точности конечных результатов, так и по затратам машинного времени.

Вопросы использования GERT-сетей в системах имитации более подробно рассмотрены в работе [11].

Заключение. Исследования по данному научному направлению были поддержаны Российским Фондом фундаментальных исследований, гранты № 07-07-00146-а, 11-07-00121-а.

Основные результаты исследований изложены в вышедшей в 2010 г. монографии "Основы теории GERT-сетей" [12].

Библиографический список

1. *A.A.B. Pritsker*. GERT" Graphical evaluation and review technique. Memorandum RM-4973-NASA, 1966.

2. *Shibanov A.P.* Finding the distribution density of the time taken to fulfill the GERT network on the basis of equivalent simplifying transformation // Automation and Remote Control. Plenum Press New York, NY, USA. February 2003 . Volume 64. Issue 2. P. 279 – 287.

3. *S.S. Hashemin, S.M.T. Fatemi Ghomi.* A hybrid method to find cumulative distribution function of completion time of GERT networks //Journal of Industrial Engineering International Islamic Azad University, Tehran. September 2005. Vol. 1, No. 1. P. 1 – 9.

4. *Javier Navascues Fernandez-Victorio.* Un modelo para la simulaciron hrıbrida de la produccion de software a medida en un entorno multiproyecto. Departamento de Lenguajes y Sistemas Informaticos. Sevilla, Julio de 2008. P. 218.

5. *S.S. Hashemin.* Fuzzy completion time for alternative stochastic networks // J. Ind. Eng. Int., ISSN: 1735-5702. 2010. 6 (11). P. 17 – 22.

6. *Шибанов А.П.* Применение моделей стохастической структуры для анализа вероятностно-временных характеристик алгоритмов преобразования информации в специализированных сетях // Телекоммуникации, № 3. 2002. С. 35 – 39.

7. *Шибанов А.П.* Нахождение плотности распределения времени исполнения GERT-сети на основе эквивалентных упрощающих преобразований // Автоматика и телемеханика, № 2. 2003. С. 117 – 126.

8. *Шибанов А.П.* Стохастическая модель канала связи // Вычислительные технологии, Т. 8. 2003. № 1. С. 111 – 116.

9. *Корячко В.П., Шибанов В.А., Ижванов Ю.Л., Шибанов А.П.* Нахождение распределения выходной величины GERT-сети со старением информации // Системы управления и информационные технологии, 2009. № 4 (38). С. 39 – 43.

10. *Корячко В.П., Шибанов А.П., Ижванов Ю.Л., Шибанов В.А.* Оценка времени передачи файла с учетом старения информации // Информационные технологии, 2010. № 10. С. 40 – 45.

11. *A.P. Shibanov.* A Software Implementation Technique for Simulation of Ethernet Local Erea Networks. Programming and Computing Software. Plenum Press New York, NY, USA. Volume 28 Issue 6, November-December 2002. P. 349 – 355.

12. Корячко В.П., Кравчук Н.В., Шибанов А.П., Шибанов В.А. Основы теории GERT-сетей. – М.: Горячая линия – Телеком, 2010. – 207 с.

УДК 681.324

Н.В. Скворцов, С.В. Скворцов, В.И. Хрюкин

СИНТЕЗ ДИАГНОСТИЧЕСКИХ ГРАФОВ ДЛЯ МНОГОПРОЦЕССОРНЫХ СИСТЕМ С АКТИВНОЙ ОТКАЗОУСТОЙЧИВОСТЬЮ

Сформулирована задача синтеза диагностических графов с экстремальными характеристиками. Показано, что ее решение обеспечивает сокращение общего числа взаимных тестовых проверок вычислительных модулей при организации активной отказоустойчивости многопроцессорных систем и сводится к поиску минимального покрытия булевой матрицы.

Ключевые слова: *многопроцессорная система, отказоустойчивость, диагностическая модель, задача покрытия.*

Введение. Активная отказоустойчивость представляет собой один из методов построения самодиагностируемых многопроцессорных систем, который базируется на отдельно выделенных процессах автоматического обнаружения отказа, его локализации и реконфигурации системы с последующим восстановлением вычислительного процесса в реальном времени [1, 2].

Реализация активной отказоустойчивости базируется на динамическом перераспределении вычислительных ресурсов в процессе решения прикладных задач, когда в системе выделяется некоторое число дополнительных вычислительных модулей (ВМ), которые в соответствии с определенной дисциплиной [3, 4] подключаются к основным модулям на некоторые периоды времени (такты контроля) и образуют пары ВМ, дублирующие вычисления. В каждой паре выделяют контролируемый (основной) и контролирующий (дополнительный) ВМ, а сравнение результатов, полученных этими модулями (например, некоторых контрольных сумм или непосредственно промежуточных данных), реализует одну элементарную проверку (ЭП).

Отказы ВМ устанавливаются одновременным анализом результатов всех ЭП, полученных за цикл контроля, т.е. за период времени между повторением ЭП для совпадающих пар модулей. При этом состав основных и дополнительных ВМ может изменяться в соответствии с реалиизуемой дисциплиной активной отказоустойчивости или вследствие деградации системы из-за отказов отдельных ВМ.

Диагностический граф (ДГ) описывает множество ЭП, выполненных за цикл контроля, и их результаты, позволяющие определить техническое состояние ВМ в реальном времени. В классическом определении [5] ДГ представляется как ориентированный граф $G=(U,D)$, вершинам $U=\{u_1, u_2, ..., u_n\}$ которого сопоставлено множество ВМ системы, а ребра задают пары модулей, выполняющих ЭП. Каждое ребро $(u_i,u_j)\in D$ взвешено булевым значением $s_{ij}\in\{0,1\}$ результата ЭП, причем его направление определяет функции ВМ: u_i является контролирующим, а u_j - контролируемым. Совокупность всех значений s_{ij}, полученных за цикл контроля, образует синдром S системы.

Дешифрация синдрома S позволяет идентифицировать отказавшие ВМ системы и осуществляется с использованием некоторой диагностической модели (ДМ), которая определяет зависимость результатов взаимной проверки модулей от их состояний [5-7]. ДМ задается четверкой булевых переменных (s_{rr}, s_{rf}, s_{fr}, s_{ff}), для которых первый индекс указывает состояние контролирующего, а второй – контролируемого модуля (r - исправен, f - отказал). Каждая переменная может принимать одно из трех значений $\{0,1,x\}$, причем $x\in\{0,1\}$ указывает на непредсказуемый результат ЭП. Наиболее известными являются модели *PMC* [5] и *BGM* [6], которые описываются как $(0,1,x,x)$ и $(0,1,x,1)$ соответственно.

Цель работы заключается в формализации задачи синтеза ДГ, обеспечивающего реализацию эффективной самодиагностики ВМ многопроцессорной системы за счет минимизации необходимого числа ЭП в цикле контроля, с учетом ограничений, которые накладываются ДМ, используемой при дешифрации синдрома.

Теоретические исследования. Будем считать, что рассматриваются только устойчивые неисправности, т.е. любой ВМ может находиться в одном из двух состояний (работоспособен или отказал), причем для нормального функционирования системы требуется не менее b работоспособных модулей, где $b < n$. Следовательно, в системе допускается наличие некоторого числа $z = n - b$ отказов, где z - степень отказоустойчивости.

Обозначим как Ф множество допустимых состояний системы, каждое из которых соответствует некоторому подмножеству отказавших модулей и определяется в виде

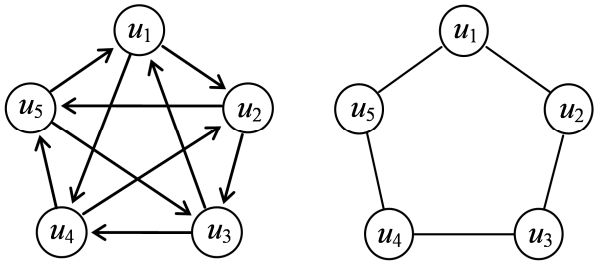
$$\Phi = F(0) \cup F(1) \cup ... \cup F(z),$$

где $F(k)$ - множество всех возможных подмножеств из Ф, содержащих k ($k=\overline{0,z}$) неисправных ВМ. Очевидно, что $F(0) = \{ F_1^{(0)} \}$ характеризует работоспособное (исправное) состояние, для которого $F_1^{(0)}=\varnothing$. Тогда $F(k)=\{ F_l^{(k)} \}$, где $k=\overline{1,z}$ и $l=\overline{1,C_n^k}$, соответствуют неисправным состояниям. Общее число состояний системы определяется как $|\Phi|=1+C_n^1+C_n^2+...+C_n^z$. Число t неисправных ВМ, которые могут быть выявлены в результате дешифрации синдрома без выполнения замены модулей, называется мерой параллельной диагностируемости и определяется структурой ДГ и используемой ДМ.

При организации активной отказоустойчивости многопроцессорных систем с магистральной организацией структура ДГ определяется реализуемой дисциплиной [3, 4]. Например, для неприоритетной дисциплины с переназначением модулей [4] в случае $n = 5$ и $b = 3$ распределение функций ВМ по 5 тактам одного цикла контроля показано в таблице 1, а соответствующий ДГ приведен на рисунке 1,а.

Таблица 1 – Распределение функций ВМ по тактам одного цикла контроля для $z = 2$

Такт контроля	Назначение вычислительных модулей		
	Основные	Дополнительные	Пары ВМ для реализации ЭП
1	u_1, u_2, u_3	u_4, u_5	$(u_5, u_1), (u_4, u_2)$
2	u_2, u_3, u_4	u_5, u_1	$(u_1, u_2), (u_5, u_3)$
3	u_3, u_4, u_5	u_1, u_2	$(u_2, u_3), (u_1, u_4)$
4	u_4, u_5, u_1	u_2, u_3	$(u_3, u_4), (u_2, u_5)$
5	u_5, u_1, u_2	u_3, u_4	$(u_4, u_5), (u_3, u_1)$



а **б**

Рисунок 1 – Примеры ДГ при n=5 и t=2 с минимальным числом связей: а - для модели *PMC*; б - для модели (0,1,1,1)

Диагностические графы, порождаемые различными дисциплинами активной отказоустойчивости при заданном значении z, могут отличаться числом и структурой проверочных связей, временем формирования (в тактах контроля), а также мерой параллельной диагностируемости t (для разных ДМ). Отсюда актуальной является задача синтеза ДГ $G=(U,D)$ с минимальным числом ребер $|D| \to \min$, дешифрация которого с использованием выбранной ДМ обеспечивает необходимое значение t.

При $t = 2$ для модели *PMC* показанный на рисунке 1,а диагностический граф имеет минимальное число ребер [8]. Однако для модели (0,1,1,1) этот ДГ является избыточным [9]. Заметим, что ДМ (0,1,1,1), (0,1,1,0) и $(0,1,1,x)$ относятся к симметричным моделям, для которых результат $s_{ij} \in \{0,1\}$ любой ЭП сохраняется при взаимном изменении функций ВМ (контролирующий - контролируемый).

При использовании симметричных ДМ диагностический граф является неориентированным, а каждое ребро $[u_i,u_j] \in D$ задает пару взаимоконтролирующих ВМ. Пример неориентированного ДГ, имеющего минимальное число ребер для модели $(0,1,1,1)$ при $n = 5$ и $t = 2$, показан на рисунке 1,б. Генерация такого ДГ может быть выполнена различными дисциплинами активной отказоустойчивости. В частности, при $z = 2$ за 5 тактов выполняются два цикла контроля, в каждом из которых синдром полностью обновляется (таблица 2), при $z = 1$ один цикл контроля включает 5 тактов (таблица 3).

Таблица 2 – Распределение функций ВМ по тактам двух циклов контроля для $z = 2$

Номер такта		Назначение вычислительных модулей		
Цикл № 1	Цикл № 2	Основные	Дополнительные	Пары ВМ для реализации ЭП
1	-	u_1, u_2, u_3	u_4, u_5	$[u_5, u_1], [u_4, u_3]$
2	-	u_2, u_3, u_4	u_5, u_1	$[u_1, u_2], [u_5, u_4]$
3	1	u_3, u_4, u_5	u_1, u_2	$[u_2, u_3], [u_1, u_5]$
-	2	u_4, u_5, u_1	u_2, u_3	$[u_3, u_4], [u_2, u_1]$
-	3	u_5, u_1, u_2	u_3, u_4	$[u_4, u_5], [u_3, u_2]$

Таблица 3 – Распределение функций ВМ по тактам одного цикла контроля для $z = 1$

Такт контроля	Назначение вычислительных модулей		
	Основные	Дополнительные	Пары ВМ для реализации ЭП
1	u_1, u_2, u_3	u_4, u_5	$(u_5, u_1), (u_4, u_2)$
2	u_2, u_3, u_4	u_5, u_1	$(u_1, u_2), (u_5, u_3)$
3	u_3, u_4, u_5	u_1, u_2	$(u_2, u_3), (u_1, u_4)$
4	u_4, u_5, u_1	u_2, u_3	$(u_3, u_4), (u_2, u_5)$
5	u_5, u_1, u_2	u_3, u_4	$(u_4, u_5), (u_3, u_1)$

Если выполнение ЭП неравноценно по затратам вычислительных ресурсов (машинное время, приоритеты дублируемых задач и контролируемых ВМ, пропускная способность магистралей системы и т.п.), то практический интерес представляет более общая форма задачи синтеза ДГ $G=(U,D)$ по критерию минимальной стоимости, где каждому ребру $(u_p,u_q) \in D$ или $[u_p,u_q] \in D$ сопоставляется некоторая суммарная оценка $c_{pq} \geq 0$ затрат на реализацию соответствующей ЭП (стоимость ребра).

Для решения представленной задачи предлагается использовать следующий подход, включающий два этапа. На первом этапе производится выбор дисциплины активной отказоустойчивости, которая порождает ДГ $G^*=(U,D^*)$, возможно избыточный, но обеспечивающий заданный уровень параллельной диагностируемости t. На втором этапе из графа G^* исключаются избыточные связи с целью получения диагностического графа G, ребра которого имеют минимальную суммарную стоимость. При этом сокращение общего числа ребер ДГ является частным случаем задачи при условии, что все ребра графа G^* имеют равные оценки стоимости.

В результате выполнения первого этапа для ДГ $G^*=(U,D^*)$ формируется таблица функций неисправностей системы [10], описываемая матрицей $\mathbf{T} = [\tau_{ij}]_{h \times m}$, где $h = |\Phi|$ и $m = |D^*|$. Каждая строка этой таблицы определяет возможное состояние системы, которое характеризуется подмножеством $F_l^{(k)}$ неисправных ВМ и соответствующим значением синдрома $S(F_l^{(k)})$.

Получение таблицы функций неисправностей производится с использованием некоторой ДМ для заданного значения t. Пример матрицы **T**, полученной при $n = 5$, $b = 3$, $t = 2$ с использованием модели *PMC* для ДГ, который показан на рисунке 1,а, приведен в таблице 4.

На втором этапе осуществляется исключение избыточных ребер ДГ с учетом следующего требования: если все строки матрицы **T** попарно различимы, то можно однозначно выполнить дешифрацию синдрома и установить текущее техническое состояние ВМ системы.

Для корректировки ДГ с целью получения его экстремальных характеристик перенумеруем ребра ДГ G^* в порядке расположения столбцов матрицы **T** и будем считать, что столбцу l соответствует ребро $d_l = (u_p, u_q)$ или $d_l = [u_p, u_q]$, результат ЭП $s_l = s_{pq}$ и ее стоимость $c_l = c_{pq}$.

Таблица 4 – Пример таблицы функций неисправностей для диагностической модели *PMC*

Модули $u_i \in F_l^{(k)}$	Результаты ЭП - синдром $S(F_l^{(k)})$									
	s_{12}	s_{14}	s_{23}	s_{25}	s_{31}	s_{34}	s_{42}	s_{45}	s_{51}	s_{53}
нет	0	0	0	0	0	0	0	0	0	0
$\{u_1\}$	x	x	0	0	1	0	0	0	1	0
$\{u_2\}$	1	0	x	x	0	0	1	0	0	0
$\{u_3\}$	0	0	1	0	x	x	0	0	0	1
$\{u_4\}$	0	1	0	0	0	1	x	x	0	0
$\{u_5\}$	0	0	0	1	0	0	0	1	x	x
$\{u_1, u_2\}$	x	x	x	x	1	0	1	0	1	0
$\{u_1, u_3\}$	x	x	1	0	x	x	0	0	1	1
$\{u_1, u_4\}$	x	x	0	0	1	1	x	x	1	0
$\{u_1, u_5\}$	x	x	0	1	1	0	0	1	x	x
$\{u_2, u_3\}$	1	0	x	x	x	x	1	0	0	1
$\{u_2, u_4\}$	1	1	x	x	0	1	x	x	0	0
$\{u_2, u_5\}$	1	0	x	x	0	0	1	1	x	x
$\{u_3, u_4\}$	0	1	1	0	x	x	x	x	0	1
$\{u_3, u_5\}$	0	0	1	1	x	x	0	1	x	x
$\{u_4, u_5\}$	0	1	0	1	0	1	x	x	x	x

Введем в рассмотрение матрицу $\boldsymbol{R} = [r_{kl}]_{g\times m}$ попарного сравнения строк таблицы функций неисправностей. Строки матрицы $\boldsymbol{R}$ задаются множеством $\Omega \subset \Phi\times\Phi$ всех пар возможных состояний системы:

$$\Omega = \{\sigma_k = (F_i^{(v)}, F_j^{(w)}):\ i \neq j;\ v \neq w;$$

$$v, w = \overline{0,t};\ i = \overline{1, C_n^v};\ j = \overline{1, C_n^w}\},$$

где число строк $g = |\Omega|$. Элементы матрицы $\boldsymbol{R}$ вычисляются с использованием операции $\oplus$ - «сложение по модулю два»:

$$r_{kl} = \begin{cases} \tau_{il} \oplus \tau_{jl}, \text{ если } \tau_{il}, \tau_{jl} \in \{0,1\}; \\ 0 \text{ - при } \tau_{il} = x \text{ или } \tau_{jl} = x, \end{cases}$$

где $i,j = \overline{1,h}$, причем $i \neq j$, и $l = \overline{1,m}$. В процессе вычислений совпадающие строки могут объединяться. Единичные элементы столбца $d_l \in D^*$ матрицы $\boldsymbol{R}$ задают подмножество

$$\Lambda_l = \{\sigma_k: r_{kl} = 1;\ k = \overline{1,g}\}$$

пар состояний (строк матрицы $\boldsymbol{R}$), которые отличаются результатом хотя бы одной ЭП.

Тогда для всех столбцов матрицы $\boldsymbol{R}$ получаем множество $\Lambda^* = \{\Lambda_1, \Lambda_2, ..., \Lambda_g\}$ различимых строк и задача синтеза ДГ $G=(U,D)$ по критерию минимальной стоимости, где $D \subseteq D^*$, сводится к поиску такого подмножества $\Lambda \subseteq \Lambda^*$, для которого достигается экстремальное значение целевой функции

$$\varphi = \sum_{l=1}^{m} c_l \xi_l \to \min$$

при ограничениях следующего вида

$$\sum_{l=1}^{m} \varepsilon_{kl} \xi_l \geq 1 \ \ (k = \overline{1,g}),$$

где целочисленные переменные ε_{kl} и ξ_l ($k = \overline{1,g}$; $l = \overline{1,m}$) могут принимать следующие значения:

$$\varepsilon_{kl} = \begin{cases} 1, \text{если } \sigma_k \in \Lambda_l; \\ 0, \text{если } \sigma_k \notin \Lambda_l; \end{cases} \quad \xi_l = \begin{cases} 1, \text{если } \Lambda_l \in \Lambda; \\ 0, \text{если } \Lambda_l \notin \Lambda. \end{cases}$$

Таким образом, синтез ДГ с экстремальными характеристиками может быть выполнен посредством поиска покрытия наименьшей стоимости булевой матрицы $\boldsymbol{R}$, т.е. такого подмножества столбцов в $\boldsymbol{R}$ с минимальной суммарной стоимостью, для которого каждая строка содержит единицу хотя бы в одном из выбранных столбцов.

Экспериментальные исследования. Задача построения минимального покрытия булевой матрицы относится к NP-полным и требует экспоненциальных затрат времени для ее решения. В работе предлагается эвристический последовательный алгоритм получения минимального покрытия, позволяющий находить приближенные решения достаточно высокого качества при полиномиальных вычислительных затратах [11].

Пусть задана булева матрица $\boldsymbol{R} = [r_{ij}]_{g\times m}$, где g – число строк и m – число столбцов. Требуется определить множество $P = \{j_1, j_2, \ldots, j_k\}$ индексов столбцов, такое, что для каждой i-й строки выполняется условие $\sum_{j\in P} a_{ij} \geq 1$ ($i = \overline{1,g}$), а общее количество таких столбцов $|P| \to \min$.

Другими словами, требуется найти такое покрытие двоичной матрицы минимальным количеством столбцов, чтобы в каждой строке

оставалась хотя бы одна единица.

Предлагаемый алгоритм выполняет не более m итераций (по числу столбцов матрицы $\boldsymbol{R}$), на каждой из которых множество P дополняется индексом очередного столбца, включаемого в формируемое покрытие. После выбора столбца из матрицы $\boldsymbol{R}$ вычеркиваются (заполняются нулями) все строки, которые на пересечении с этим столбцом имеют единичные элементы. Эти действия повторяются до тех пор, пока в матрице $\boldsymbol{R}$ остаются ненулевые элементы. Выбор очередного столбца производится по одному из двух следующих эвристических правил.

Правило 1. Выбирается столбец, которому соответствует строка с единственным ненулевым элементом. Если таких столбцов несколько, то столбец с минимальным индексом.

Правило 2. Выбирается столбец, обеспечивающий покрытие максимального количества строк, т.е. такой k-й столбец, для которого

$$\sum_{i=1}^{g} a_{ik} \to \max \ \ (k = \overline{1,m}).$$

Если таких столбцов несколько и их индексы составляют множество $M = \{k_1, k_2, \ldots, k_p\}$, то сначала вычисляются значения $E_k = \sum_{i=1}^{g}\sum_{j=1}^{m} a_{ik}a_{ij}$ для всех $k \in M$ и выбирается такой столбец, для которого $E_k \to \max$.

Другими словами, второе правило позволяет выбрать столбец, который при удалении соответствующих строк из матрицы $\boldsymbol{R}$ оставляет в ней минимальное количество единиц.

Последовательный алгоритм поиска минимального покрытия.

Данные: булева матрица $\boldsymbol{R}$.

Результаты: множество индексов столбцов P (покрытие матрицы).

Шаг 1. Положить $P = \varnothing$.

Шаг 2. Если в матрице $\boldsymbol{R}$ есть столбец, удовлетворяющий правилу 1, то зафиксировать его индекс k и перейти к шагу 4.

Шаг 3. Определить индекс k столбца матрицы $\boldsymbol{R}$ по правилу 2.

Шаг 4. Включить значение k в множество P и присвоить $r_{ij} = 0$, где $j = \overline{1,m}$, для всех значений i, таких, что $r_{ik} = 1$.

Шаг 5. Если $\boldsymbol{R} = \boldsymbol{O}$, где $\boldsymbol{O}$ – нулевая матрица, то конец алгоритма, иначе перейти к шагу 1.

Оценим вычислительную сложность этого алгоритма. Очевидно, что на каждой итерации (шаги 2-5) при просмотре столбцов выбирается лишь один претендент k на включение в покрытие P, а количество столбцов в формируемом покрытии в предельном случае может составлять m. При выполнении очередной итерации число просматриваемых столбцов уменьшается на один и в среднем составляет $m/2$. В худшем случае выбор каждого столбца производится по правилу 2, что приводит к выполнению на одной итерации порядка $m^2g/2$ действий. В целом выполняется не более m итераций, что в итоге дает оценку вычислительной сложности $O(m^3g)$.

Покажем особенности работы предложенного алгоритма на примере поиска минимального покрытия для приведенной ниже матрицы $\boldsymbol{R}$. При этом будем считать, что все столбцы (ребра ДГ) имеют равные оценки стоимости.

На первой итерации применяется правило 1, в соответствии с которым выбирается первый столбец, так как он покрывает строку 7, имеющую только один ненулевой элемент. После дальнейшего обнуления строк с индексами 1, 4, 7, имеющих на пересечении с выбранным столбцом единицы, получим матрицу $\boldsymbol{R}^{(1)}$.

$$\boldsymbol{R} = \begin{bmatrix} 1&0&0&0&1&0&0&1 \\ 0&0&1&0&0&1&0&0 \\ 0&0&0&0&1&0&0&0 \\ 1&0&0&0&1&0&1&1 \\ 0&1&1&0&0&0&1&1 \\ 0&1&0&0&1&1&0&0 \\ 1&0&0&0&0&0&0&0 \\ 0&0&0&0&0&1&0&1 \\ 0&1&0&1&0&1&0&1 \\ 0&0&0&1&1&0&0&0 \end{bmatrix}; \ \boldsymbol{R}^{(1)} = \begin{bmatrix} 0&0&0&0&0&0&0&0 \\ 0&0&1&0&0&1&0&0 \\ 0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&0&0&0 \\ 0&1&1&0&0&0&1&1 \\ 0&1&0&0&1&1&0&0 \\ 0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&1&0&1 \\ 0&1&0&1&0&1&0&1 \\ 0&0&0&1&1&0&0&0 \end{bmatrix}.$$

Результат второй итерации, на которой также по правилу 1 выбирается пятый столбец и обнуляются строки 3, 6, 10, показан в виде матрицы $\boldsymbol{R}^{(2)}$. На третьей итерации используется правило 2 и выбирается столбец 6, что в итоге дает матрицу $\boldsymbol{R}^{(3)}$.

$$\boldsymbol{R}^{(2)} = \begin{bmatrix} 0&0&0&0&0&0&0&0 \\ 0&0&1&0&0&1&0&0 \\ 0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0 \\ 0&1&1&0&0&0&1&1 \\ 0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&1&0&1 \\ 0&1&0&1&0&1&0&1 \\ 0&0&0&0&0&0&0&0 \end{bmatrix}; \ \boldsymbol{R}^{(3)} = \begin{bmatrix} 0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0 \\ 0&1&1&0&0&0&1&1 \\ 0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0 \end{bmatrix}.$$

При выполнении четвертой итерации по правилу 2 выбирается второй столбец, а полученная в результате матрица $\boldsymbol{R}^{(4)}$ становится нулевой. Таким образом, после четырех итераций алгоритма (из $m = 8$ возможных) найдено покрытие $P = \{1, 5, 6, 2\}$, которое, как нетрудно убедиться, является оптимальным.

Экспериментальные исследования предложенного алгоритма производились для 450

вариантов случайных матриц с размерами 50×10, 125×15 и 250×25 (по 150 экземпляров каждого размера). Степень заполнения этих матриц единицами варьировалась следующим образом: низкая (до 20 %); средняя (20 - 40 %); высокая (более 40 %).

Качество приближенных решений оценивалось относительным отклонением (в процентах)

$$\Delta = \frac{\varphi - \varphi_{opt}}{\varphi_{opt}} 100\%$$

значения целевой функции $\varphi = |P|$, полученного с помощью разработанного алгоритма, от оптимального результата φ_{opt}, который дает метод ветвей и границ [11] для этой же матрицы.

Точность полученных результатов в целом характеризует таблица 5, данные из которой детализируются в таблице 6, показывающей долю точных решений в зависимости от размера случайной матрицы и степени ее заполнения.

Таблица 5 – Доля приближенных решений (%) с погрешностью в пределах Δ

Оптимальное решение	Отклонение от оптимального решения		
	$\Delta \le 10$ %	$\Delta \le 15$ %	$\Delta \le 25$ %
76,9	79,6	88,4	98,9

Таблица 6 – Доля оптимальных решений (%)

Размер матрицы	Степень заполнения			В среднем по размеру
	Низкая	Средняя	Высокая	
50×10	95	97,5	77,5	90
125×15	100	57,7	65	74,2
250×25	85	45	70	66,6
В среднем по заполнению	93,3	66,7	70,8	76,9

Необходимо отметить, что для каждого сочетания характеристик случайных матриц (таблица 6) решено по 50 задач. Для этих же групп исходных данных в таблице 7 приведены средние показатели отклонений Δ приближенных значений целевой функции от оптимальных результатов.

Таблица 7 – Среднее значение погрешности Δ (%)

Размер матрицы	Степень заполнения			В среднем по размеру
	Низкая	Средняя	Высокая	
50×10	1,4	0,3	5,6	2,4
125×15	0	5,1	7,7	4,3
250×25	0,4	6,6	5,3	4,1
В среднем по заполнению	0,6	4,0	6,2	3,6

Таким образом, для 450 решенных задач результаты, полученные по разработанному алгоритму, совпали с оптимальными почти в 77 % случаев, причем для всех оставшихся приближенных решений средняя относительная погрешность составляет около 4 %. При этом с увеличением степени заполнения матрицы наблюдается небольшой рост средней относительной погрешности Δ результатов.

Скорость работы предложенного алгоритма практически не зависит от размера исследованных матриц и позволяет получать решения за секунды на персональной ЭВМ, имеющей двухядерный процессор Intel Core2 Duo T8100 с тактовой частотой 2,1 ГГц и оперативную память объемом 2 Гбайт. Для сравнения необходимо сказать, что при реализации метода ветвей и границ на базе этого же компьютера для матриц размером 250×25 на получение оптимального решения требуется несколько часов.

Заключение. Следует отметить, во-первых, универсальность предложенного подхода к синтезу ДГ с точки зрения используемой диагностической модели (асимметричной, например, *PMC* и *BGM*, или симметричной) и, во-вторых, возможность его применения как для минимизации числа проверочных связей (ребер ДГ), так и для уменьшения суммарных вычислительных затрат при реализации ЭП в процессе самодиагностирования многопроцессорной системы.

При этом качество решения задачи синтеза ДГ с минимальным числом ребер существенно зависит от эффективности используемого алгоритма поиска минимального покрытия. В частности, в работе [11] можно найти алгоритм решения этой задачи, основанный на методе ветвей и границ, который гарантирует получение точного решения, но требует больших временных затрат. Предложенный в статье приближенный последовательный алгоритм отличается высокой скоростью работы и достаточно хорошим качеством решений, что позволяет использовать его в режиме реального времени при выполнении реконфигураций отказоустойчивых многопроцессорных систем.

Библиографический список

1. *Иыуду К.А.* Надежность, контроль и диагностика вычислительных машин и систем. - М.: Высшая школа, 1989. – 216 с.

2. *Пархоменко П.П., Согомонян Е.С.* Основы технической диагностики. - М.: Энергия, 1981. – 320 с.

3. *Гершанов В.И., Скворцов С.В., Телков И.А.* Методы повышения отказоустойчивости вычислительных систем, основанных на принципе ассоциативной селекции потоков данных // Вопросы радиоэлектроники. Сер. Электрон. вычислит. техн. - 1992. - Вып. 7. - С. 50-58.

4. *Скворцов С.В.* Организация отказоустойчивых вычислений в магистрально-модульных многопроцессорных системах // Вестник Рязанской государственной радиотехнической академии. - 1996. - Вып. 1. - С. 27-32.

5. *Preparata F.P., Metze G., Chien R.T.* On the Connection Assignment Problem of Diagnosable Systems // IEEE Trans. Electron. Comput. - 1967. - V. EC-16. - № 6. - P. 848-854.

6. *Barsi F., Grandoni F., Maestrini P.* A Theory of Diagnosability of Digital Systems // IEEE Trans. Comput. - 1976. - V. C-25. - № 6. - P. 585-593.

7. *Крамаренко М.Б.* Модели диагностирования отказов параллельной вычислительной системы // Электронное моделирование. - 1989. - № 3. - С. 60-65.

8. *Баранов В.Г., Гладков В.В., Махалин Б.Н.* Математические модели для системного уровня диагностики неисправностей в мультипроцессорных системах // Обзоры по электронной технике. Сер. 8. Управление качеством, стандартизация, метрология, испытания. 1991. Вып. 2 (1601). 58 с.

9. *Корячко В.П., Скворцов С.В., Шувиков В.И.* Синтез оптимальных диагностических графов для симметричной модели дешифрации синдрома // Информационные технологии. - 1999. - № 12. - С. 32-37.

10. Основы технической диагностики. В 2-х книгах. Кн. 1. Модели объектов, методы и алгоритмы диагноза / Под ред. П.П. Пархоменко. - М.: Энергия, 1976. - 464 с.

11. *Кристофидес Н.* Теория графов. Алгоритмический подход. - М.: Мир, 1978. - 432 с.

УДК 681.31

В.Н. Ручкин

ТЕЛЕКОММУНИКАЦИОННЫЕ АСПЕКТЫ ПАРАДИГМ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

Излагаются предпосылки и парадигмы развития искусственного интеллекта как основного инструментария когнитивной теории и представления основных когнитивных процессов: мышления, языкознания и организации памяти, решение задач, творчества, расширения человеческого интеллекта с целью выявления общих телекоммуникационных аспектов передачи мысли и создания интеллектуальных структур на базе отечественных нейропроцессоров семейства NM 640X для реализации нового класса интеллектуальных телекоммуникационных систем.

Ключевые слова: *парадигмы искусственного интеллекта, коммуникация и телекоммуникации, передача мысли, решение задач мышления и «передачи» мышления, нейропроцессор и нейропроцессорная система, биокомпьютинг, нейроэмуляция, когнитивный нейрокомпьютер, нейрокомпьютерный интерфейс, интеллектуальная телекоммуникационная структура.*

Введение. Когнитивная теория рассматривает *парадигмы развития искусственного интеллекта* [1] как систему коммуникации и телекоммуникации, в которой мысли передаются посредством звуков, как в устной речи и музыке или символов, как в письменной речи и жестах, а также решения задач мышления и «передачи» мышления посредством когнитивных функций. При этом предпосылки и парадигмы развития искусственного интеллекта рассматриваются как основной инструментарий когнитивной теории, представление основных когнитивных процессов: мышления, языкознания и организации памяти, решение задач, творчества, расширения человеческого интеллекта, а также более глубокое изучение языка и организации памяти средствами и методами искусственного интеллекта.

Однако известно, что в познании - восприятии, нервных основах нейропознания, памяти, сознании, речи, решении проблем и других областях - все процессы протекают одновременно. Поэтому необходимо всестороннее изучение когнитивной психологии, включающей в себя оценку всех компонентов, сплетающихся в тончайший узор окружающего нас естественного «параллельного» мира.

Цель работы. Целью работы является выявление общих аспектов развития существующих парадигм искусственного интеллекта, мышления и «передачи» мышления для создания интеллектуальных структур реализации нового класса интеллектуальных телекоммуникационных систем.

Постановка задачи. В работе рассматриваются и анализируются парадигмы искусственного интеллекта с целью формирования общих аспектов представления знаний, метазнаний, мышления и «передачи» мышления и на этой базе создания современных интеллектуальных телекоммуникационных структур на базе нейропроцессоров и разработка алгоритмов опреде-

ления типа структуры нейропроцессорной системы (НПС). Для дальнейших исследований в качестве примера было выбрано семейство отечественных 64 битных нейропроцессоров NM640X.

Однако результаты исследования могут быть использованы и для других нейропроцессоров, соответствующих общим используемым здесь моделям и принципам функционирования нейрокомпьютеров.

Теоретические исследования

Обоснование исследования. В основе парадигм искусственного интеллекта лежат родовые черты, объединяющие принципы работы и обучения всех возможных систем искусственного интеллекта [1-3]. Главное, что их объединяет - нацеленность на обработку образов и различных структурированных данных-знаний и метазнаний. Эта особенность аналогична способу функционирования мозга согласно современной когнитивной теории. Поэтому с этой целью и производится анализ, формулируются парадигмы в концентрированном виде безотносительно к биологическим прототипам как способы обработки образов и данных. Эти общие сведения послужат фундаментом для более подробного разбора отдельных современных нейроархитектур телекоммуникационных систем мышления и «передачи» мышления. При этом сложность представления, формализации и воплощения принципов работы человеческого мозга предполагает наличие нескольких вариантов реализации парадигм ИИ, подобно функционированию самого мозга. В результате возникает уникальное многообразие или ансамбль современных систем искусственного интеллекта, которые необходимо реализовать современными вычислительными средствами – нейрокомпьютерами.

1. Классическая (алгоритмическая) парадигма. Полностью состоявшаяся на сегодня парадигма ИИ построена

а) как реализация в материальной среде операций булевой алгебры;

б) как реализация тезиса Черча-Тьюринга о том, что «любую интуитивно понятную задачу можно алгоритмизировать» [2,3].

В целом эту парадигму можно назвать алгоритмической. Объекты природы, социальные явления, операции мышления моделируются алгоритмами и при реальном обсчете на компьютере сведены к рекурсиям [4]. Использование вероятностных методов, теории игр, вычислительных методов при решении систем дифференциальных уравнений и пр. математических моделей (включая и существующие сегодня формальные нейронные сети) приводит к иллюзии, что, во-первых, удалось уйти от механистической детерминированности, а, во-вторых, с приемлемой точностью смоделировать любые реальные процессы.

Идеологами и основными исследователями состоявшейся парадигмы стали А. Черч, А. Тьюринг, С. Клини, А. Марков, А. Колмогоров и многие другие [5-6]. В данный момент мы наблюдаем сосуществование прошлой парадигмы с попытками отхода от нее в сторону цифрового моделирования динамики, самоорганизации и рефлексивности систем. Назовем это текущей парадигмой ИИ. В ней ставятся цели, выходящие за рамки предыдущей парадигмы, а также предпринимаются попытки выйти за ограниченные рамки алгоритмического подхода, используя, однако, алгоритмы как ведущее средство. Предполагается, что привнесение стохастических процессов, прежде всего случайных по времени транзакций между параллельными вычислительными процессами (подобно сети интернет [7]), создает качественно новый системный эффект с возможностью выхода в пространство невычислимых функций. Эта парадигма не является завершенной, исследования в ней активно ведутся на данный момент [8].

В рамках текущей парадигмы необходимо отличать реальную динамическую систему от формальной динамической системы. Формальная динамическая система является системой дифференциальных уравнений, удовлетворяющей условию единственности решения. Замкнутые кривые в фазовом пространстве соответствуют периодическим решениям системы дифференциальных уравнений. Качественное описание формальной динамической системы предполагает выделение притягивающих и отталкивающих точек фазового пространства. Существенными моментами поведения реальной динамической системы являются состояния, соответствующие изменению качественного описания. Момент возникновения нового качества в теории формальных динамических систем называется бифуркацией, которая определяется как малое изменение параметров дифференциальных уравнений, описывающих формальную динамическую систему. При этом постулируется, что этот малый скачок изменений параметров в моменты бифуркаций происходит случайно.

Иначе говоря, формальная динамическая система представляет собой множество автоматов (с непрерывным или дискретным временем). В положении равновесия формальная

динамическая система описывается одним определенным автоматом. В момент бифуркации происходит случайный переход от одного автомата к другому автомату. Такой подход может быть удовлетворительным при описании и объяснении простых реальных динамических систем, но он недостаточен при моделировании таких явлений, как мозг, поскольку случайность не может быть источником сложной самоорганизации и возникновения жизни (вероятность такой последовательности случайных событий практически равна нулю). Кроме этого, автомат не может обеспечить реализацию процесса мышления и рефлексии. Это означает, что формальное описание реальных динамических систем не объясняет работу мозга.

Поэтому моделирование формальных динамических систем на компьютере не только на дискретном, но и на непрерывном, не может служить основой для моделирования интеллекта. Ученые считают возможным, не дожидаясь исчерпания всего потенциала текущей синергетической парадигмы, начать переход к новой парадигме ИИ. Основаниями для такого решения служат результаты исследований логики естественного интеллекта (ЕИ) и исследование оснований алгоритмической парадигмы.

Далее рассматривается задача создания искусственной среды, моделирующей когнитивные процессы с качеством моделирования выше, чем при современных алгоритмических подходах. Аргументируется, что алгоритмически (при любой комбинации алгоритмов и при внесении случайности любого типа) когнитивные процессы, на самом деле, вообще не моделируются. Это является причиной расходимости в бесконечность вычислительных процессов при решении таких задач, как: выделение неизвестных объектов (в частности, распознавание образов), получение эффектов рефлексивности и самоорганизация системы. Выдвигаются технические требования к моделированию среды, адекватной ЦНС и когнитивным процессам естественного интеллекта.

2. Новая парадигма. Идея новой парадигмы искусственного интеллекта (НПИИ) состоит в создании искусственной (физической) среды со свойствами, обеспечивающими специфические системные эффекты телекоммуникации. В этой среде за счет дополнения алгоритмических методов моделирования ИИ другими методами, созданными в новой парадигме, предполагается обеспечить повышение качества решения когнитивных задач, таких как индуктивный вывод, понимание текстов на естественном языке (ЕЯ), самоорганизация и рефлексия системы.

В новой парадигме нарушение категории части/целое распространяется и на моделирование процессов: состояние включает переход, а переход включает состояние (S входит в A, а A входит в S), как это было в текущей парадигме.

В современных компьютерах категория часть\целое полностью определяет все процессы, нарушение в них этой категории не допустимо. Поэтому в компьютерах целое всегда равно сумме своих частей и системные эффекты при работе с информацией в компьютерах отсутствуют. Когнитивные акты (с соответствующими системными эффектами) происходят в мышлении программистов, и в машину заносятся уже в преобразованном виде, в виде следов работы естественного интеллекта: постановленных задач, алгоритмов решения, критериев эффективности решения, параметров работы, вносимых в систему извне, интерпретации результатов и т.п. Сам же механизм ЕИ (обеспечивающий понимание текстов, распознавание образов, индуктивный вывод, рефлексию и т.п.) включает в себя единство противоположностей и нарушение категории части\целое. Попытка моделирования указанных свойств среды с помощью частично рекурсивных функций, к которым, как было показано Геделем [3], сводятся все численные методы, сталкивается с достаточно серьезными трудностями.

Авторы новой парадигмы полагают, что описанные особенности механизма реализации интеллектуальных операций и некоторых базовых операций нейронной сети не разрешимы рекурсивно.

В работе машины Тьюринга (МТ), которой эквивалентен любой алгоритм, четко разделены состояния ячеек ленты (операнды) и процедуры (операции), считывание информации («сенсоры») и запись информации («эффекторы»). Любые реальные процессы, моделируемые МТ (и компьютером), представляются как набор дискретных состояний и переходов между ними, даже если времена переходов ничтожно малы, а колоссальное число дискретных единиц «практически» создает эффект непрерывности.

Количественная малость, отличающая характеристики цифровых моделей процессов от характеристик процессов реальных создает иллюзию о качественной близости, эквивалентности и даже тождественности модели и реальности. Однако качественное отличие остается и дает о себе знать при составлении дифференциальных уравнений в перекрестных членах, которые не могут быть рассчитаны, но отбрасываются в силу своей малости, в кван-

товых эффектах единства волны и частицы, в нейронных эффектах единства активности и восприятия. А также в логических парадоксах, которые были описаны выше.

Должен быть сделан переход в самом фундаменте – к такому когнитивному акту, в котором нарушается действие категории части\целое. В этом акте объекты существуют одновременно и в их единстве, и в их раздельности. При этом сама идея циклических процессов останется, но будет переопределена. Поэтому в новой парадигме ИИ алгоритмы, реализованные на цифровых компьютерах, должны быть дополнены когнитивными процессами, моделируемыми материальной активной средой. При этом должна возникнуть альтернатива самому принципу работы МТ и архитектуры фон Неймана. В активной среде вместо тактов работы процессора с данными 1 и 0 должна реализовываться элементарная когнитивная единица самоорганизации, самодействия или, иначе, элементарный когнитивный акт, заключающий в себе единство противоположностей: операнда и операции, восприятия и воздействия.

Таким образом, целью новой парадигмы ИИ является разработка систем, способных получать решения «за рамками» взаимосвязанных утверждений: «если решение существует, оно может быть алгоритмически получено» и «алгоритмически полученное решение – существует, т.е., является решением». Нужно научиться получать неполные решения (такие, которые сами по себе не являются решениями, но становятся решениями, будучи дополнены), частные решения (те, верность которых не может быть проверена иначе, чем результатами исполнения решения), комплексы решений (т.е. многосторонние решения при взаимодействиях) и другие столь же нетривиальные виды решений.

3. Квазибиологическая парадигма Biocomputing. Биокомпьютинг или квазибиологическая парадигма [4]) (англ. Biocomputing) — биологическое направление в искусственном интеллекте, сосредоточенное на разработке и использовании компьютеров, которые функционируют как живые организмы или содержат биологические компоненты так называемые биокомпьютеры.

Родоначальником биологического направления в кибернетике является У. Мак-Каллок, а также последующие идеи М. Конрада, которые привели к направлению — биомолекулярная электроника. В отличие от понимания искусственного интеллекта по Джону Маккарти [6], когда исходят из положения о том, что искусственные системы не обязаны повторять в своей структуре и функционировании структуру и протекающие в ней процессы, присущие биологическим системам, сторонники данного подхода считают, что феномены человеческого поведения, его способность к обучению и адаптации есть следствие именно биологической структуры и особенностей ее функционирования.

Часто квазибиологической парадигме противопоставляют понимание искусственного интеллекта по Джону Маккарти, тогда говорят о восходящем (англ. Bottom-Up AI) ИИ, на котором базируется квазибиологическая парадигма. Однако существует и нисходящий (англ. Top-Down AI) ИИ — создание экспертных систем, баз знаний и систем логического вывода, имитирующих высокоуровневые психические процессы. В этом случае, как правило, говорят о рациональном ИИ

Парадигма «фон Неймана» является основой подавляющего большинства современных средств обработки информации. Она оптимальна, когда решаются массовые задачи достаточно низкой вычислительной сложности.

Квазибиологическая парадигма сегодня по своему содержанию и возможным приложениям значительно богаче, чем первоначальный подход МакКаллоха и Питса. Она находится в процессе развития и изучения возможностей создания на её основе эффективных средств обработки информации.

К. Заенер и М. Конрад сформулировали понятие индивидуальной машины в противоположность универсальному компьютеру «фон Неймана». Данное понятие базируется на следующих положениях.

1. Универсальная машина не может решать любую проблему так же эффективно, как машина, специально сконструированная для её решения.

2. Жесткая программа подразумевает последовательное выполнение операций, т.е. неэффективное использование вычислительных ресурсов.

3. Программу легко разрушить, если извне ввести случайные изменения. Поэтому невозможно шаг за шагом вносить малые изменения и постепенно менять структуру программы.

Поэтому основные особенности индивидуальной машины, следующие:

1) физическая структура машины определяет решение конкретной задачи;

2) эволюция машины после ввода управляющих стимулов приводит к такому состоянию и/или структуре машины, которые могут быть интерпертированы как решение искомой задачи.

Коннекционизм. В отличие от цифровых систем, представляющих собой комбинации про-

цессорных и запоминающих блоков, нейропроцессоры содержат память, распределённую в связях между очень простыми процессорами, которые часто могут быть описаны как формальные нейроны или блоки из однотипных формальных нейронов. Тем самым основная нагрузка на выполнение конкретных функций процессорами ложится на архитектуру системы, детали которой, в свою очередь, определяются межнейронными связями. Подход, основанный на представлении как памяти данных, так и алгоритмов системой связей (и их весами), называется коннекционизмом.

Как указывалось выше, отличительной чертой нейросетей является глобальность связей. Базовые элементы искусственных нейросетей - формальные нейроны - изначально нацелены на работу с широкополосной информацией. Каждый нейрон нейросети, как правило, связан со всеми нейронами предыдущего слоя обработки данных. На рисунке 1 иллюстрируются наиболее широко распространенная в современных приложениях архитектура многослойного персептрона).

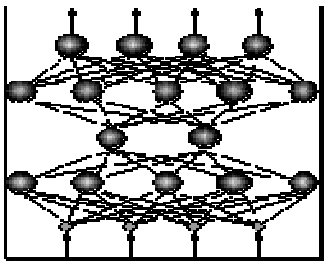


Рисунок 1 - Глобальность связей в искусственных нейросетях

Глобальность связей в искусственных нейросетях является главным отличием формальных нейронов от базовых элементов последовательных ЭВМ - логических вентилей, имеющих лишь два входа. В итоге универсальные процессоры имеют сложную архитектуру, основанную на иерархии модулей, каждый из которых выполняет строго определенную функцию. Напротив, архитектура нейросетей проста и универсальна. Специализация связей возникает на этапе их обучения под влиянием конкретных данных и определяет нейропроцессорную структуру интеллектуальной телекоммуникационной сети. Типичный формальный нейрон производит простейшую операцию - взвешивает значения своих входов со своими же локально хранимыми весами и производит над их суммой нелинейное преобразование (рисунок 2 - согласно модели МакКаллоха и Питса):

$$y = f(u), u = u_0 + \sum_i w_i x_i$$

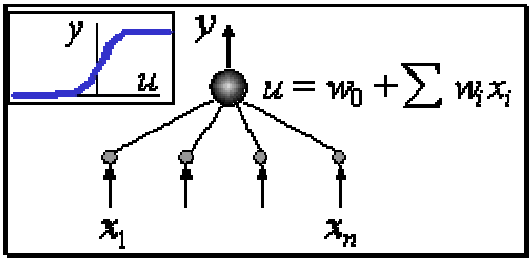


Рисунок 2 - Модедь МакКаллока и Питса

4. Нейроэмуляция. Комплекс программ, использующих принципы нейросетевой обработки данных и исполняемых на последовательных компьютерах.

Преимущества нейроэмуляторов. Преимущества таких *"виртуальных"* нейрокомпьютеров для относительно небольших задач очевидны.

1. Не надо тратиться на новую аппаратуру, если можно загрузить уже имеющиеся компьютеры общего назначения.

2. Пользователь не должен осваивать особенности программирования на спецпроцессорах и способы их сопряжения с базовым компьютером.

3. Универсальные ЭВМ не накладывают никаких ограничений на структуру сетей и способы их обучения, тогда как спецпроцессоры зачастую имеют ограниченный набор "зашитых" в них функций активации и достигают пиковой производительности лишь на определенном круге задач (нейроускорители). **Различают:**

1. Готовые нейропакеты представляют собой законченные независимые программные продукты, предназначенные для широкого класса задач, в основном - для предсказаний и статистической обработки данных. Большинство из имеющихся на рынке нейропакетов имеет *дружественный интерфейс пользователя, не требующий знакомства с языками программирования*.

2. Инструменты разработки нейроприложений отличают этот класс программного обеспечения - способностью генерировать "отчуждаемые" нейросетевые продукты, т.е. генерировать программный код, использующий обученные нейросети для обработки данных. Такой код может быть встроен в качестве подсистемы в любые сколь угодно сложные информационные комплексы.

3. Инструменты нейросетевого консалтинга. Нейросетевой консалтинг в процессе нейрокомпьютинга занимает особое место. Поэтому описание рынка нейропродуктов будет не полным без упоминания о нейроконсалтинге. Вместо того чтобы продавать готовые программы либо инструменты для их разработки, можно торговать и услугами.

5. Когнитивный нейрокомпьютер. На прошедшей в Портленде (штат Орегон) **Supercomputing Conference'09** фирма **IBM** заявила о существенном прогрессе в создании вычислительной системы, которая симулирует и эмулирует способность мозга чувствовать, воспринимать, действовать, взаимодействовать и познавать и при этом сравнима с мозгом по низкому энергопотреблению и размерам.

BlueMatter – новый алгоритм, созданный IBM Research в сотрудничестве со Стэнфордским университетом, использует суперкомпьютерную архитектуру BlueGene для неинвазивного измерения и отображения связей между всеми локусами коры и подкорки в мозге человека с помощью диффузной спектральной томографии.

Крупномасштабная симуляция деятельности коры головного мозга – новое междисциплинарное направление, объединяющее вычислительную неврологию, методологию симуляции и суперкомпьютеры – краеугольный камень возможности разработки когнитивного чипа.

6. Нейрокомпьютерный интерфейс. Нейрокомпьютерный интерфейс (НКИ) (называемый также прямой нейронный интерфейс, или мозговой интерфейс) - система, созданная для обмена информацией между мозгом и электронным устройством (например, компьютером). В однонаправленных интерфейсах внешние устройства могут либо принимать сигналы от мозга, либо посылать ему сигналы (например, имитируя сетчатку глаза при восстановлении зрения электронным имплантатом). Двунаправленные интерфейсы позволяют мозгу и внешним устройствам обмениваться информацией в обоих направлениях. В основе нейрокомпьютерного интерфейса, часто лежит метод биологической обратной связи.

Биологическая обратная связь (англ. *Biofeedback*) - технология, включающая в себя комплекс исследовательских, лечебных и профилактических физиологических процедур, в ходе которых пациенту посредством внешней цепи обратной связи, организованной преимущественно с помощью микропроцессорной или компьютерной техники, предъявляется информация о состоянии и изменении тех или иных собственных физиологических процессов.

Изучение оснований, на которых базируется нейрокомпьютерный интерфейс, уходит корнями в учение И.П. Павлова об условных рефлексах и регулирующей роли коры. Это научное направление возникло в самом начале 20-го века в Институте экспериментальной медицины (Санкт-Петербург). Развивая эти идеи, П.К. Анохин с 1935 г. показал, что принципу обратной связи принадлежит решающая роль в регулировании как высших приспособительных реакций человека, так и его внутренней среды. В результате была разработана теория функциональных систем, потенциал использования которой в нейрокомпьютерных интерфейсах далеко не исчерпан. Большой вклад внесли работы Н.П. Бехтеревой с 1968 по 2008 гг. по расшифровке мозговых кодов психической деятельности, продолжающиеся до настоящего времени её последователями, в том числе с позиций нейрокибернетики и офтальмонейрокибернетики.

Практические исследования. Большинство вышеперечисленных парадигм можно реализовать или смоделировать на базе современного отечественного семейства нейромикропроцессоров NM 640X [9, 10].

Рабочая матрица векторного процессора NM 640X имеет два входа *X* и *Y* (рисунок 3). На эти входы подаются данные, расположенные во внешней памяти, либо во внутренних буферах *ram* и *afifo*, работающих по принципу FIFO. Данные из буферов или из памяти могут быть поданы как на вход *X*, так и на *Y*. То есть, например, вектор 64-х разрядных слов, хранящийся в ram, может быть передан на обработку в операционный узел через вход *X* и/или *Y*. Для управления потоком данных из внешней памяти используется логический буфер *data*. В качестве входа *Y* может также быть использован векторный регистр *vr*. Кроме этого, в качестве входов могут выступать так называемые "нулевые" устройства, что означает, что данные на вход не поступают.

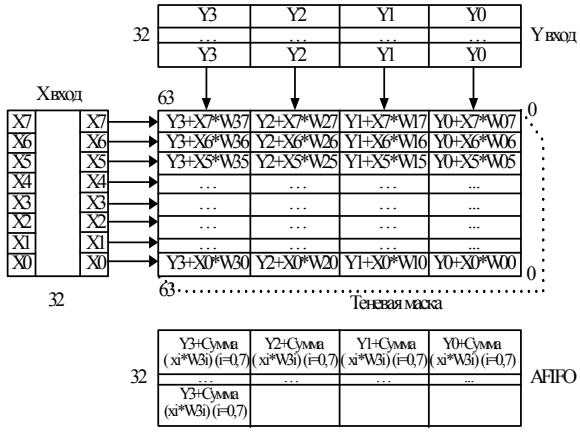


Рисунок 3 - Регистровая модель векторного процессора

Более детально рабочая матрица выполняет операции взвешенного суммирования согласно математической модели МакКаллоха и Питса (рисунок 4).

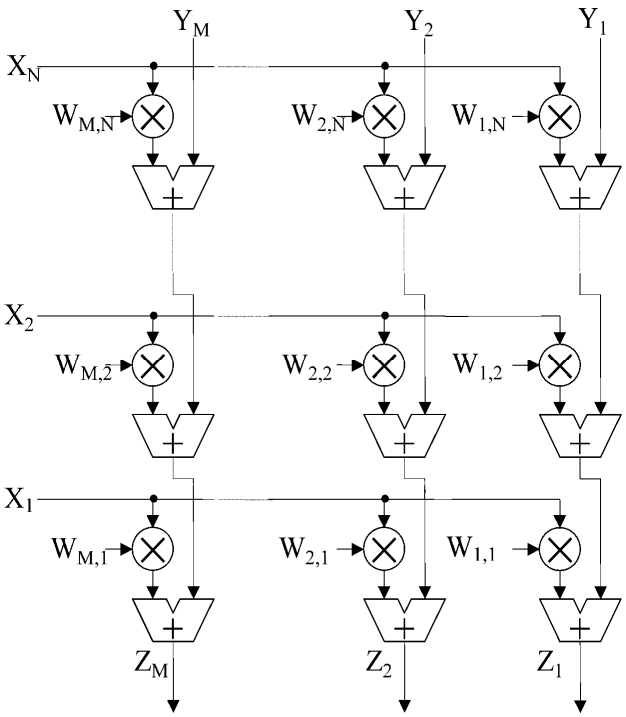


Рисунок 4 - Операция взвешенного суммирования

В макроячейках происходит умножение элемента входных данных X_i на весовой коэффициент W_{ij} и сложение со значением верхней ячейки с учётом входного сигнала Y_j, а для защиты от арифметического переполнения используется программируемая функция насыщения.

Операнды и выходные значения упаковываются в 64-разрядные слова. Все операции в матрице производятся параллельно за 1 такт. Загрузка весовых коэффициентов производится за 32 такта, однако для ускорения работы существует теневая маска, в которую весовые коэффициенты загружаются в фоновом режиме. Причем переключение теневой в рабочую матрицу происходит за один такт.

Важной особенностью векторного процессора является реализация работы с программно управляемыми операндами различной длины.

Для моделирования мультинейропроцессорных телекоммуникационных структур на базе нейропроцессоров NM 640X был разработан программный комплексе «НейроКС» [10].

Алгоритм определения связей элементов вычислительной структуры на базе нейропроцессоров с целью определения вида НПС реализован в подсистеме «Текстовой редактор для языка нейроассемблера» программного комплекса (рисунок 5).

После вставки директив в текстовом редакторе становится возможным режим «Генерация матрицы связи подпрограмм» для визуального представление матрицы M (рисунок 6).

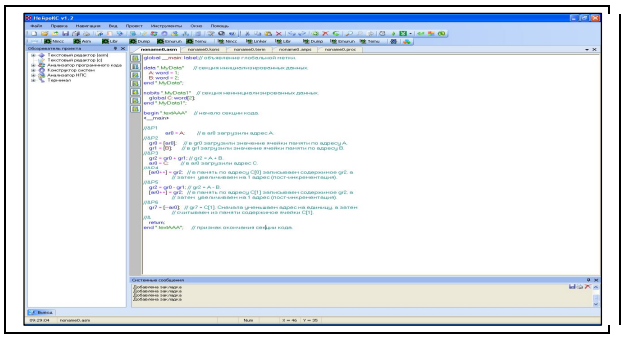


Рисунок 5 – Визуальное представление матрицы подсистемы «Текстовой редактор для языка Нейроассемблера»

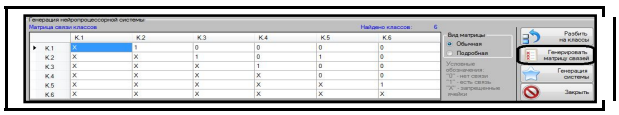


Рисунок 6 – Визуальное представление матрицы M

После генерации матрицы M появляется возможность генерации матрицы M' (рисунок.7).

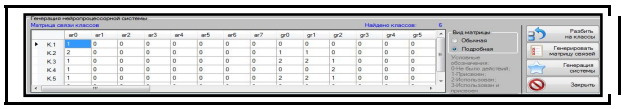


Рисунок 7 – Визуальное представление матрицы M

Алгоритм определения вида структуры НПС на основе описания связей ее элементов с целью использования оценок эффективности для этого вида структуры реализован также в подсистеме «Текстовой редактор для языка нейроассемблера». После вызова соответствующей процедуры на основании матрицы связей M' происходит открытие экземпляра подсистемы «Конструктор систем» для визуального представления нейронной телекоммуникационной структуры (рисунок 8).

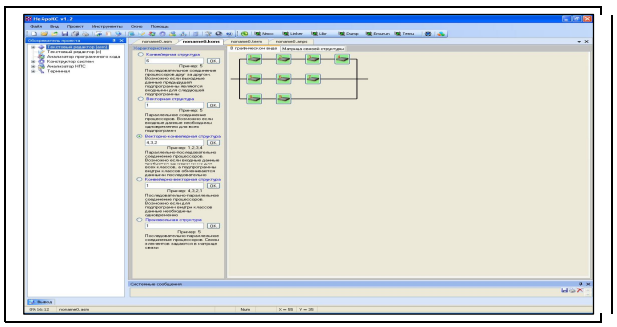


Рисунок 8 – Визуальное представление интеллектуальной телекоммуникационной системы

Заключение. Сочетание когнитивной теории и ИИ позволяет создавать достаточно простые, универсальные и альтернативные архитектуры интеллектуальных телекоммуникационных сетей, не требующих значительных аппаратных затрат.

Многообразие парадигм указывает на зна-

чительное расширение возможностей и повышение их эффективности за счет обучения и самообучения.

Программный комплекс «НейроКС» позволяют визуализировать телекоммуникационные аспекты организации интеллектуальных структур на базе отечественных нейропроцессоров семейства NM 640X.

Библиографический список

1. *Солсо Р.* Когнитивная психология. - СПб.: Питер, 2006. - 589 с.

2. *Тьюринг А.* Может ли машина мыслить. Физматгиз, 1960.

3. *Черч А.* Введение в математическую логику. В 2-х т.: пер. с англ. - М.: Издательство иностранной литературы, 1960.

4. *Эббинхауз Г.Д., Якобс К., Ман Ф.К., Хермес Г.* Машины Тьюринга и рекурсивные функции: пер. с нем. М., 1972.

5. Компьютер учится и рассуждает (ч. 1) // Компьютер обретает разум Artificial Intelligence Computer Images / под ред. *В. Л. Стефанюка.* - М.: Мир, 1990. - 240 с.

6. *McCarthy J.* What is Artificial Intelligence?, Stanford University, 2007.

7. *Калинкина Т.И., Костров Б.В., Ручкин В.Н.* Телекоммуникационные и вычислительные сети. Архитектура, стандарты и технологии.-СПб.: БХВ-Петербург, 2010. –288 с.

8. *Рассел С., Норвиг П.* Искусственный интеллект: современный подход = Artificial Intelligence: a Modern Approach / Пер. с англ. и ред. *К. А. Птицына.* - 2-е изд. - М.: Вильямс, 2006. - 140 с.

9. *Злобин В.К., Ручкин В.Н.* Нейросети и нейрокомпьютеры. СПб.: БХВ-Петербург, 2011. -256 с.

10. *Романчук В.А., Ручкин В.Н.* Разработка программных средств анализа нейропроцессорных систем // Вестник РГРТУ. 2010. №2. Вып.32. С.61-67.