СОДЕРЖАНИЕ

В.П. КОРЯЧКО, А.И. ТАГАНОВ, Р.А. ТАГАНОВ
Рязанский государственный радиотехнический университет

ГИБРИДНЫЕ АЛГОРИТМЫ И БИОНИЧЕСКИЕ МЕТОДЫ ДЛЯ АНАЛИЗА РИСКОВ И УПРАВЛЕНИЯ ПРОГРАММНЫМИ ПРОЕКТАМИ

Рассматриваются теоретические вопросы построения гибридных алгоритмов и бионических методов, основанных на использовании моделей и методов теории математического программирования в условиях неопределенности (неопределенного программирования) для решения широкого круга задач по нечеткому анализу рисков и нечеткому управлению программными проектами.

Введение

Практика инженерии программных систем (ПС) свидетельствует, что программные проекты являются достаточно сложными объектами в техническом и организационном выполнении. В жизненном цикле программных проектов присутствуют многочисленные факторы неопределенности и связанные с этим риски проектов, которыми надо управлять посредством применения соответствующих процессно-ориентированных методов, содержащих в своем составе проблемно-ориентированные интеллектуальные процедуры поддержки принятия решений, гибридные алгоритмы, бионические методы и др. [1,2].

Под неопределенным программированием здесь будем понимать теорию оптимизации в неопределенной среде. Основные направления неопределенного программирования включают в себя: стохастическое программирование, нечеткое МП, неточное МП, нечетко-случайное программирование, случайно-нечеткое программирование, случайно-неточное программирование, неточно-случайное программирование, нечетко-неточное программирование, неточно-нечеткое программирование, бислучайное программирование, бинечеткое программирование, бинеточное программирование и неопределенное программирование с множественной неопределенностью [6].

Для того чтобы можно было строить точно выраженные модели нечеткого программирования, в работе [3] представлена серия нечетких моделей ожидаемого значения (EVM-моделей), в основе которых лежит подход, предполагающий выбор решения с максимальным ожидаемым доходом.

В статье далее вводятся основные понятия, относящиеся к нечетким EVM-моделям, а также осуществляется необходимая интеграция нечеткого имитационного моделирования, нейронных сетей и генетических алгоритмов для того, чтобы получить целую серию гиб-

ридных алгоритмов, позволяющих работать с нечеткими EVM-моделями при решении многих задач анализа рисков и управления программными проектами [1,2, 7-10].

1. Нечеткие модели ожидаемого значения

Чтобы получить решение при решении задач управления проектами [2], обеспечивающее максимум ожидаемого значения дохода, можно воспользоваться следующей однокритериальной нечеткой EVM-моделью:

$$\begin{cases} \max \mathrm{E}\left[f(x,\xi)\right] \\ \text{при ограничениях :} \\ \quad \mathrm{E}\left[g_i(x,\xi)\right] \le 0, \quad j = 1,2,...,p, \end{cases} \tag{1.1}$$

где x – некоторый вектор решений, ξ – некоторый нечеткий вектор, $f(x,\xi)$ – функция дохода, а $g_i(x,\xi)$ – функции-ограничения, $j = 1,2,...,p$.

В ряде случаев приходится иметь дело с набором функций дохода (критериев). Соответственно, в таком случае имеет место задача многокритериального программирования с нечетким ожидаемым значением:

$$\begin{cases} \max \left[\mathrm{E}\left[f_1(x,\xi)\right], \ \mathrm{E}\left[f_2(x,\xi)\right], ..., \ \mathrm{E}\left[f_m(x,\xi)\right]\right] \\ \text{при ограничениях :} \\ \quad \mathrm{E}\left[g_i(x,\xi)\right] \le 0, \quad j = 1,2,...,p, \end{cases} \tag{1.2}$$

где $f(x,\xi)$ – функции дохода, $i = 1,2,...,m$.

Для того чтобы обеспечить получение компромиссного решения при многих конфликтующих показателях, можно ввести следующую модель целевого программирования с нечеткими ожидаемыми значениями:

$$\begin{cases} \min \sum_{j=1}^{l} P_j \sum_{i=1}^{m} \left(u_{ij} d_i^+ + u_{ij} d_i^-\right) \\ \text{при ограничениях :} \\ \quad \mathrm{E}\left[f_i(x,\xi)\right] + d_i^- - d_i^+ = b_i, \quad i = 1,2,...,m, \\ \quad \mathrm{E}\left[g_j(x,\xi)\right] \le 0, \qquad j = 1,2,...,p, \\ \quad d_i^+, d_i^- \ge 0, \qquad i = 1,2,...,m, \end{cases} \tag{1.3}$$

где P_j – заданные некоторым образом коэффициенты преимущественного приоритета, выражающие относительную важность различных

целей, $P_j >> P_{j+1}$, для всех j; u_{ij} – весовые коэффициенты, отвечающие положительному отклонению для цели i с приписанным ей приоритетом j; v_{ij} – весовые коэффициенты, отвечающие отрицательному отклонению для цели i с приписанным ей приоритетом j; d_i^+ – положительное отклонение от назначенного уровня i – й цели, определяемое как

$$d_i^+ = \left[\mathrm{E}\left[f_i(x,\xi)\right] - b_i\right] \vee 0, \tag{1.4}$$

d_i^- – отрицательное отклонение от назначенного уровня i – й цели, определяемое как

$$d_i^- = \left[b_i - \mathrm{E}\left[f_i(x,\xi)\right]\right] \vee 0, \tag{1.5}$$

f_i – некоторая функция в целевых ограничениях; g_i – некоторая функция в реальных ограничениях; b_i – назначенный уровень, соответствующий i – й цели; l – число приоритетов; m – число целевых ограничений; p – число реальных ограничений.

2. Алгоритмические основы метода нечеткого программирования

Для того чтобы решать задачи [1,2, 7-10] с использованием EVM-модели, требуется подготовить множество вход-выходных данных для $\mathrm{E}\left[f(x,\xi)\right]$ с помощью следующего алгоритма нечеткого имитационного моделирования.

2.1. Алгоритм 1. Нечеткое моделирование

Шаг 1. Задать $E = 0$.

Шаг 2. Сформировать случайным образом $u_{1j}, u_{2j}, ..., u_{nj}$ из ε – уровневых множеств для $\xi_1, \xi_2, ..., \xi_n$, обозначить $u_j = (u_{1j}, u_{2j}, ..., u_{nj})$, $j = 1,2,...,m$, соответственно, где ε – некоторое достаточно малое число.

Шаг 3. Задать $a = f(u_1) \wedge f(u_2) \wedge ... \wedge f(u_m)$, $b = f(u_1) \wedge f(u_2) \wedge ... \wedge f(u_m)$

Шаг 4. Сформировать случайным образом число r из интервала $[a, b]$.

Шаг 5. Если $r \ge 0$, тогда $E \leftarrow E + \mathrm{Cr}\{f(\xi) \ge r\}$.

Шаг 6. Если $r < 0$, тогда $E \leftarrow E - \mathrm{Cr}\{f(\xi) \le r\}$.

Шаг 7. Повторить шаги с четвертого по шестой N раз

Шаг 8. $E[f(\xi)] = a \vee 0 + b \wedge 0 + E \cdot (b - a)/N$.

Затем для аппроксимации неопределенной функции $E[f(x, \xi)]$ производится обучение нейронной сети. Обученная сеть встраивается в генетический алгоритм, что приводит к получению мощного гибридного алгоритма, основные шаги которого могут быть описаны ниже следующим образом.

2.2. Алгоритм 2. Гибридный алгоритм

Шаг 1. Для функций, содержащих неопределенность, сформировать обучающий набор пар «вход-выход» вида

$U : x \to E[f(x, \xi)]$

с использованием нечеткого моделирования (Алгоритм 1).

Шаг 2. Провести с использованием полученного набора данных обучение нейронной сети, которая будет аппроксимировать функции, содержащие неопределенность.

Шаг 3. Инициализировать *pop_size* хромосом, допустимость которых может быть проверена с использованием обученной нейронной сети [6].

Шаг 4. Модифицировать хромосомы с помощью операций кроссинговера и мутаций, с помощью обученной нейронной сети проверить допустимость потомков.

Шаг 5. Рассчитать целевые значения для всех хромосом, используя обученную нейронную сеть.

Шаг 6. Вычислить приспособленность каждой из хромосом с помощью функции оценки, основанной на ранжировании, используя найденные целевые значения.

Шаг 7. Произвести селекцию хромосом, используя случайный выбор.

Шаг 8. Повторить шаги с четвертого по седьмой заданное число раз.

Шаг 9. Выдать наилучшую из хромосом в качестве оптимального решения.

3. Алгоритмические основы нечеткого программирования с возможностными ограничениями

Аналогично стохастическому программированию с вероятностными ограничениями, нечеткое программирование с возможностными ограничениями дает в распоряжение лица, принимающего решения, средства, позволяющие формулировать целевые функции и ограничения в терминах возможности их достижения [2,6].

3.1. Возможностные ограничения

Пусть x — вектор решений, ξ — некоторый нечеткий вектор, $f(x,\xi)$ — функция дохода, $gj(x,\xi)$ — функции-ограничения, j = 1,2,... ,p. Поскольку нечеткие ограничения $g_j(x,\xi) \le 0, \quad j = 1,2,...,p$, не определяют какого-либо детерминированного множества возможных решений, вполне естественной представляется идея использовать желательную возможность α (называемую доверительным уровнем) удовлетворения нечетким ограничениям. Тогда получим возможностное ограничение в следующей форме:

$$Pos\{g_j(x,\xi) \le 0, \quad j = 1,2,...,p\} \ge \alpha \qquad (3.1)$$

Иногда можно использовать следующие отдельные возможностные ограничения

$$Pos\{g_j(x,\xi) \le 0\} \ge \alpha_j, \quad j = 1,2,...,p \qquad (3.2)$$

где α_j - доверительные уровни для j = 1,2,... ,p.

3.2. Максимаксное программирование с возможностными ограничениями

Следуя идеям стохастического программирования с вероятностными ограничениями в [6] предложен ряд моделей нечеткого программирования с возможностными ограничениями. Для задач [1,2,7-10] в тех случаях, когда требуется максимизировать оптимистическое значение критериальной функции, получаем следующую формулировку модели нечеткого программирования с возможностными ограничениями для однокритериального случая,

$$\begin{cases} \max \bar{f} \\ \text{при ограничениях :} \\ Pos\{f(x,\xi) \ge \bar{f}\} \ge \beta, \\ Pos\{g_j(x,\xi) \le 0,\ j = 1,2,...,p\} \ge \alpha, \end{cases} \qquad (3.3)$$

где α и β— предопределенные заранее доверительные уровни. Модель программирования с возможностными ограничениями (3.3) называется максимаксной вследствие того, что она эквивалентна следующей модели:

$$\begin{cases} \max\limits_{x} \max\limits_{\bar{f}} \bar{f} \\ \text{при ограничениях :} \\ Pos\{f(x,\xi) \ge \bar{f}\} \ge \beta, \\ Pos\{g_j(x,\xi) \le 0,\ j = 1,2,...,p\} \ge \alpha, \end{cases}$$

для которой более отчетливо виден ее максимаксный характер, где $\max \overline{f}$ есть β-оптимистическое значение критериальной функции.

Если в решаемой задаче несколько показателей (критериев), то получаем задачу многокритериального программирования с возможностными ограниче-ниями :

$$\begin{cases} \max[\overline{f_1}, \overline{f_2}, ..., \overline{f_m}] \\ \text{при ограничениях :} \\ \text{Pos}\{f_i(x,\xi) \geq \overline{f_i}\} \geq \beta_i, \quad i = 1,2,...,m, \\ \text{Pos}\{g_j(x,\xi) \leq 0\} \geq \alpha_j, \quad j = 1,2,...,p, \end{cases} \tag{3.4}$$

где $\alpha_j, j = 1,2,...,p,$ $\beta_i, i = 1,2,...,m$ - предопределенные заранее доверительные уровни. Задача нечеткого многокритериального программирования с возможностными ограничениями (3.4) эквивалентна следующей максимаксной задаче

$$\begin{cases} \max\limits_{x}[\max\limits_{f_1} \overline{f_1}, \max\limits_{f_2} \overline{f_2}, ..., \max\limits_{f_m} \overline{f_m}] \\ \text{при ограничени ях :} \\ \text{Pos}\{f_i(x,\xi) \geq \overline{f_i}\} \geq \beta_i, \quad i = 1,2,...,m, \\ \text{Pos}\{g_j(x,\xi) \leq 0\} \geq \alpha_j, \quad j = 1,2,...,p, \end{cases}$$

где $\max \overline{f_i}$ представляют собой β_i-оптимистические значения для целевых функций $f_i(x,\xi), i = 1,2,...,m$, соответственно.

Для нечеткой системы принятия решений можно также сформулировать задачу миниминного целевого программирования с возможностными ограничениями, следуя структуре приоритетов и уровням целевых значений, установленных лицом, принимающим решение:

$$\begin{cases} \min \sum\limits_{j=1}^{l} P_j \sum\limits_{i=1}^{m} \left(u_{ij} d_i^+ + v_{ij} d_i^-\right) \\ \text{при ограничениях :} \\ \text{Pos}\{f_i(x,\xi) - b_i \leq d_i^+\} \geq \beta_i^+, \quad i = 1,2,...,m, \\ \text{Pos}\{b_i - f_i(x,\xi) \leq d_i^-\} \geq \beta_i^-, \quad i = 1,2,...,m, \\ \text{Pos}\{g_j(x,\xi) \leq 0\} \geq \alpha_j, \qquad j = 1,2,...,p, \\ d_i^+, d_i^- \geq 0, \qquad i = 1,2,...,m, \end{cases} \tag{3.5}$$

где P_j — коэффициент преимущественного приоритета, который выражает относительную важность различных целей, $P_j >> P_{j+1}$, для всех j; u_{ij} - весовые коэффициенты, соответствующие положительному отклонению для цели i с присвоенным ей приоритетом j; v_{ij} - весовые коэффициенты, соответствующие отрицательному отклонению для цели i с присвоенным ей приоритетом j; d_i^+ представляет собой β_i^+ - оптимистическое положительное отклонение от назначенного уровня i-й цели, определяемое как

$$\min\{d \vee 0 \mid \text{Pos}\{f_i(x,\xi) - b_i \leq d\} \geq \beta_i^+\}, \tag{3.6}$$

d_i^- представляет собой β_i^--оптимистическое отрицательное отклонение от назначенного уровня i-й цели, определяемое как

$$\min\{d \vee 0 \mid \text{Pos}\{b_i - f_i(x,\xi) \leq d\} \geq \beta_i^-\}, \tag{3.7}$$

f_i - функция в целевых ограничениях; g_i - функция в реальных ограничениях;

b_i - назначенный уровень, соответствующий i-й цели; l - число приоритетов;

m - число целевых ограничений; p - число действительных ограничений.

Замечание. Если нечеткий вектор ξ вырождается в четкий, тогда две возможности $\text{Pos}\{f_i(x,\xi) - b_i \leq d_i^+\}$ и $\text{Pos}\{b_i - f_i(x,\xi) \leq d_i^-\}$ будут всегда иметь значение 1, при условии что $\beta_i^+, \beta_i^- > 0$, а из

$$\text{Pos}\{f_i(x,\xi) - b_i \leq d_i^+\} \geq \beta_i^+, \quad d_i^+ \geq 0,$$
$$\text{Pos}\{b_i - f_i(x,\xi) \leq d_i^-\} \geq \beta_i^-, \quad d_i^- \geq 0,$$

получаем, что

$$d_i^+ = \left[f_i(x,\xi) - b_i\right] \vee 0,$$
$$d_i^- = \left[b_i - f_i(x,\xi)\right] \vee 0.$$

Это совпадает со случаем четкого целевого программирования.

3.3. Минимаксное программирование с возможностными ограничениями

Фактически для задач управления проектами [1,2,7-10], модели максимаксного программирования с возможностными ограничениями представляют собой разновидность моделей оптимистического типа, которые максимизируют максимально возможное значение результата

(дохода). В данном случае рассматривается ряд минимаксных моделей программирования с возможностными ограничениями, предложенных в [4], в которых производится выбор альтернативы, обеспечивающей наилучшее из худших значений результата. Отметим также, что для наших целей различия между миниминной и максимаксной (или между минимаксной и максиминной) моделями не являются существенными.

Если требуется максимизировать пессимистическое значение получаемого результата при управлении проектами [1,2] получаем следующую минимаксную модель программирования с возможностными ограничениями в однокритериальной постановке:

$$\begin{cases} \max\limits_{x} \min\limits_{\bar{f}} \bar{f} \\ \text{ïðè îãðàíè÷åíè ÿõ:} \\ \text{Pos}\{f(x,\xi) \le \bar{f}\} \ge \beta, \\ \text{Pos}\{g_j(x,\xi) \le 0,\ j = 1,2,...,p\} \ge \alpha, \end{cases} \tag{3.8}$$

где $\min \bar{f}$ есть β-пессимистический доход.

В случае, когда критериев в решаемой задаче несколько, можно предложить следующую минимаксную модель многокритериального программирования с возможностными ограничениями:

$$\begin{cases} \max\limits_{x}[\min\limits_{\bar{f}_1} \bar{f}_1, \min\limits_{\bar{f}_2} \bar{f}_2, ..., \min\limits_{\bar{f}_m} \bar{f}_m] \\ \text{при ограничениях:} \\ \text{Pos}\{f_i(x,\xi) \le \bar{f}_i\} \ge \beta_i, \quad i = 1,2,...,m, \\ \text{Pos}\{g_j(x,\xi) \le 0\} \ge \alpha_j, \quad j = 1,2,...,p, \end{cases} \tag{3.9}$$

где α_j и β_i - доверительные уровни, а $\min \bar{f}_i$ представляют собой β_i-пессимистические значения для функций дохода $f_i(x,\xi), i = 1,2,...,m$, соответственно.

Согласно структуре приоритетов и целевым уровням, установленным лицом, принимающим решение, минимаксная модель целевого программирования с возможностными ограничениями запишется следующим образом:

$$\begin{cases} \min\limits_{x} \sum\limits_{j=1}^{l} P_j \sum\limits_{i=1}^{m} \left[u_{ij} \left(\max\limits_{d_i^+} d_i^+ \vee 0 \right) + v_{ij} \left(\max\limits_{d_i^-} d_i^- \vee 0 \right) \right] \\ \text{ïðè ñëåäóþùèõ îãðàíè÷åíèÿõ:} \\ \text{Pos}\{f_i(x,\xi) - b_i \ge d_i^+\} \ge \beta_i^+, \quad i = 1,2,...,m, \\ \text{Pos}\{b_i - f_i(x,\xi) \ge d_i^-\} \ge \beta_i^-, \quad i = 1,2,...,m, \\ \text{Pos}\{g_j(x,\xi) \le 0\} \ge \alpha_j, \quad j = 1,2,...,p, \end{cases} \tag{3.10}$$

где P_j - коэффициент преимущественного приоритета, который выражает относительную важность различных целей, $P_j >> P_{j+1}$, для всех j; u_{ij} - весовой коэффициент, отвечающий положительному отклонению для цели i с присвоенным ей приоритетом j; v_{ij} - весовой коэффициент, отвечающий отрицательному отклонению для цели i с присвоенным ей приоритетом j; $d_i^+ \vee 0$ представляет собой β_i^+-пессимистическое положительное отклонение от назначенного уровня i-й цели, определяемое как

$$\max\{d \vee 0 \mid \text{Pos}\{f_i(x,\xi) - b_i \ge d\} \ge \beta_i^+\}, \tag{3.11}$$

$d_i^- \vee 0$ представляет собой β_i^--пессимистическое отрицательное отклонение от назначенного уровня i-й цели, определяемое как

$$\max\{d \vee 0 \mid \text{Pos}\{b_i - f_i(x,\xi) \ge d\} \ge \beta_i^-\}, \tag{3.12}$$

b_i - назначенный уровень, соответствующий i-й цели; l - число приоритетов; т — число целевых ограничений.

Замечание . Если нечеткий вектор ξ вырождается в четкий, тогда две возможности $\text{Pos}\{f_i(x,\xi) - b_i \ge d_i^+\}$ и $\text{Pos}\{b_i - f_i(x,\xi) \ge d_i^-\}$ будут всегда иметь значение 1, при условии что $\text{Pos}\{f_i(x,\xi) - b_i \ge d_i^+\} \ge \beta_i^+$, $\beta_i^+, \beta_i^- > 0$, а $\text{Pos}\{b_i - f_i(x,\xi) \ge d_i^-\} \ge \beta_i^-$,

дают как следствие

$$d_i^+ \vee 0 = [f_i(x,\xi) - b_i] \vee 0,$$
$$d_i^- \vee 0 = [b_i - f_i(x,\xi)] \vee 0.$$

Это совпадает с задачей четкого целевого программирования.

3.4. Алгоритм 3. Гибридный алгоритм

Шаг 1. Сформировать набор данных, состоящий из вход-выходных пар для неопределенных функций вида

$$U_1 : x \to \mathrm{Pos}\{g_j(x,\xi) \le 0,\ \ j = 1,2,...,p\},$$

$$U_2 : x \to \max\{\bar{f} \mid \mathrm{Pos}\{f(x,\xi) \ge \bar{f}\} \ge \beta\},$$

с использованием нечеткого имитационного моделирования.

Шаг 2. Провести обучение нейронной сети для аппроксимации неопределенных функций, используя полученный на предыдущем шаге обучающий набор.

Шаг 3. Инициализировать *pop_size* хромосом, допустимость которых может быть проверена с помощью нейронной сети, обученной на предыдущем шаге.

Шаг 4. Модифицировать хромосомы, используя операции кроссинговера и мутации, в которых допустимость потомков может быть проверена с помощью обученных нейронных сетей.

Шаг 5. Рассчитать значения целевых функций для всех хромосом с помощью обученных нейронных сетей.

Шаг 6. Вычислить приспособленность каждой из хромосом, основываясь на значениях целевой функции.

Шаг 7. Выбрать случайным образом хромосомы.

Шаг 8. Повторить шаги с четвертого по седьмой заданное число раз.

Шаг 9. Объявить лучшую из хромосом оптимальным решением.

4. Алгоритмические основы метода нечеткого событийного программирования

Для этапа мониторинга рисков программного проекта, а также других этапов и задач управления проектами важными являются задачи событийного управления и оптимизации [2,7-10].

В развитие идеи событийного программирования (dependent-chance programming— DCP) в стохастической среде в работе [5] предложена теория нечеткого событийного программирования, которая основана на концепции выбора решения с максимальной возможностью появления заданного события. Для DCP вводятся понятия неопределенной среды, события, функции шансов и принципа неопределенности для нечеткого случая.. Для решения задач с нечеткими DCP-моделями необходимо соответствующее объединение нечеткого моделирования, нейронной сети и генетического алгоритма, приводящее к получению отвечающего данному случаю гибридного алгоритма.

4.1. Принцип неопределенности

Неопределенная среда [2,6], событие и вероятностная функция события представляют собой ключевые элементы концепции событийного программирования в стохастической среде. Переопределим эти понятия применительно к случаю нечеткой среды.

Определение 4.1. Под неопределенной средой (в данном случае — нечеткой средой) будем понимать нечеткие ограничения, представляемые как

$$g_j(x,\xi) \le 0, \quad j = 1,2,..., p, \tag{4.1}$$

где x — вектор решений, а ξ — некоторый нечеткий вектор.

Определение 4.2. Под событием будем понимать некоторую систему нечетких неравенств

$$h_k(x,\xi) \le 0, \quad k = 1,2,..., q, \tag{4.2}$$

где x — вектор решений, а ξ — некоторый нечеткий вектор.

Определение 4.3. Функция шансов некоторого события ε, характеризуемого посредством (11.2), определяется как мера возможности события ε, т. е.

$$f(x) = \mathrm{Pos}\{h_k(x,\xi) \le 0,\ \ k = 1,2,...,q\}, \tag{4.3}$$

с учетом неопределенной среды (4.1).

Концепции носителя, зависимого носителя, активного ограничения и зависимого ограничения - те же самые, что и в стохастическом случае. Таким образом, для каждого решения x и реализации ξ, про некоторое событие ε говорят, что оно согласовано с неопределенной средой, если выполняются следующие два условия:

$$h_k(x,\xi) \le 0, \quad k = 1,2,...,q;$$

$$g_j(x,\xi) \le 0, \quad j \in J,$$

где J —множество индексов всех зависимых ограничений. Чтобы вычислить функцию шансов нечеткого события, потребуется следующий принцип неопределенности.

Принцип неопределенности: Шансы некоторого нечеткого события — это возможность (или необходимость, правдоподобие) того, что событие согласовано с неопределенной средой.

Примем, что имеется т событий ε_i, характеризуемых соотношениями $h_{ik}(x,\xi) \le 0, \quad k = 1,2,...,q_i$ для i=1,2,…,m, в неопределенной

среде $g_j(x,\xi) \le 0, \quad j = 1,2,...,p$. Из принципа неопределенности следует, что возможностная функция i-го события ε_i в заданной неопределенной среде определяется как

$$f_i(x) = \mathrm{Pos}\left\{\begin{matrix} h_{ik}(x,\xi) \le 0, \quad k = 1,2,...,q_i \\ g_j(x,\xi) \le 0, \quad j \in J_i \end{matrix}\right\}, \qquad (4.4)$$

где J_i определяется через

$J_i = \left\{ j \in \{1,2,...,p\} \middle| g_j(x,\xi) \le 0 - \text{зависимое ограничение для } \varepsilon_i \right\}$,

для i=1,2,…,m.

4.2. Задачи событийного программирования

Типичная формулировка задачи событийного программирования в нечеткой среде [2,6] может быть записана следующим образом:

$$\begin{cases} \max \ \mathrm{Pos}\ \{h_k(x,\xi) \le 0,\ k = 1,2,...,\ q\} \\ \text{при ограничениях :} \\ g_j(x,\xi) \le 0, \quad j = 1,2,...,\ p, \end{cases} \qquad (4.5)$$

где x - n-мерный вектор решений, ξ - нечеткий вектор, событие ε характеризуется соотношением $h_k(x,\xi) \le 0, k = 1,2,...,q$,а неопределенная среда описывается нечеткими ограничениями $g_j(x,\xi) \le 0, \quad j = 1,2,...,p$.

В случае нечеткой среды это возможностная функция события.

Задача нечеткого событийного программирования (4.5) читается как «максимизация возможности нечеткого события $h_k(x,\xi) \le 0, k = 1,2,...,q$, с учетом неопределенной среды $g_j(x,\xi) \le 0, j = 1,2,...,p.$.»

Поскольку сложная система принятия решений обычно работает с целой совокупностью задач, вне всякого сомнения существует и множество потенциально возможных целевых функций. Типичная формулировка нечеткой многокритериальной задачи событийного программирования может быть записана следующим образом:

$$\begin{cases} \max \begin{bmatrix} \mathrm{Pos}\ \{h_{1k}(x,\xi) \le 0, k = 1,2,...,\ q_1\} \\ \mathrm{Pos}\ \{h_{2k}(x,\xi) \le 0, k = 1,2,...,\ q_2\} \\ ... \\ \mathrm{Pos}\ \{h_{mk}(x,\xi) \le 0, k = 1,2,...,\ q_m\} \end{bmatrix} \\ \text{ïðè \_ îãðàíè÷åíè ÿõ} \\ g_j(x,\xi) \le 0, j = 1,2,...,\ p, \end{cases}$$

где $h_{ik}(x,\xi) \le 0, k = 1,2,...,q_i$, представляют события ε_i для $i = 1,2,...,m,$ соответственно.

Целевое событийное программирование в нечеткой среде [2] может рассматриваться как некоторое расширение задачи целевого программирования применительно к сложным нечетким системам принятия решений. Когда заданы некоторые цели управления, целевая функция задачи может быть построена так, чтобы минимизировать отклонения, положительные, отрицательные, или и те и другие вместе, для некоторой заданной структуры приоритетов. Тогда можно сформулировать понятие нечеткой системы принятия решений как задачи целевого событийного программирования согласно структуре приоритетов и уровням целевых значений показателей качества, установленных лицом, принимающим решения на этапах управления проектами:

$$\begin{cases} \min \sum_{j=1}^{l} P_j \sum_{i=1}^{m} \left(u_{ij} d_i^+ + u_{ij} d_i^- \right) \\ \text{при \_ ограничениях :} \\ \mathrm{Pos}\left\{\begin{matrix} h_{ik}(x,\xi) \le 0 \\ k = 1,2,...,q_i \end{matrix}\right\} + d_i^- - d_i^+ = b_i, \quad i = 1,2,...,m, \\ g_j(x,\xi) \le 0, \qquad j = 1,2,...,p, \\ d_i^+, d_i^- \ge 0, \qquad i = 1,2,...,m, \end{cases}$$

где P_j — коэффициент преимущественного приоритета, который выражает относительную важность различных целей, $P_j >> P_{j+1}$, для всех j; u_{ij} — весовой коэффициент, отвечающий положительному отклонению для цели i с присвоенным ей приоритетом j; u_{ij} — весо-

вой коэффициент, отвечающий отрицательному отклонению для цели i с присвоенным ей приоритетом j; d_i^+ — положительное отклонение от назначенного уровня i-й цели; d_i^- —отрицательное отклонение от назначенного уровня i-й цели; g_i — функция в системных ограничениях; b_i — назначенный уровень i-й цели; l — число приоритетов; m — число целевых ограничений; p — число системных ограничений.

5. Метод нечеткого программирования с нечеткими решениями

Традиционно, модели математического программирования дают ответ в виде четкого (crisp) вектора решений такого, что некоторые целевые функции на нем достигают оптимальных значений. Однако в практических приложениях технологии управления программными проектами вместо четкого вектора решений необходимо использовать нечеткий вектор [1,2,7-10].

5.1. Модели с нечетким вектором решений

В работе [6] рассматривается более общий случай, который приводит к серии моделей максимаксного программирования с ограничениями на шансы (CCP-моделей) и с нечетким вектором решений. В [4] предложен ряд минимаксных CCP-моделей с нечетким вектором решений, а в [5] построена основа DCP-модели с нечетким вектором решений. В дополнение к этому, на случай нечеткого вектора решений распространена также и EVM-модель. Для того чтобы решать задачи, использующие эти модели, необходимо построить соответствующий вариант гибридного алгоритма, объединяющего средства нечеткого имитационного моделирования, нейронные сети и генетический алгоритм.

Для формирования набора вход-выходных пар для неопределенных функций вида можно воспользоваться средствами нечеткого имитационного моделирования. Поскольку процесс такого моделирования является весьма трудоемким, целесообразно аппроксимировать упомянутые неопределенные функции с помощью нечеткой нейронной сети.

5.2. Алгоритм 5. Гибридный алгоритм

Чтобы иметь возможность решать задачи нечеткого программирования общего вида с нечеткими решениями, можно использовать гибридный алгоритм, в котором соответствующим образом скорректированы применяемая в нем нейронная сеть, структура представления задачи, процедура инициализации, а также операции кроссинговера и мутации.

Первое, что требуется сделать, это сформировать набор нечетких вход-выходных пар данных для неопределенной функции. Для решения данной задачи воспользуемся средствами нечеткого имитационного моделирования. Затем на полученном таким образом наборе данных произведем обучение нейронной сети, аппроксимирующей требуемые неопределенные функции.

Для нечеткого решения необходимо построить соответствующую ему нечеткую хромосому, в которой каждый ген будет представлять собой нечеткую величину вместо обычного четкого числа. В качестве хромосомы будем использовать здесь нечеткий вектор $V = (\tilde{x}_1, \tilde{x}_2, ..., \tilde{x}_n)$. Он будет представлением нечеткого решения в задачах нечеткого программирования с нечеткими решениями, где компоненты $\tilde{x}_i$ выбираются из опорного набора нечетких величин X_i для $i = 1,2,...,n$, соответственно.

Нечеткие величины будем извлекать из опорного набора величин X_i случайным образом для каждого нечеткого гена $\tilde{x}_i$, формируя таким образом нечеткую хромосому $V = (\tilde{x}_1, \tilde{x}_2, ..., \tilde{x}_n)$, с проверкой ее допустимости с помощью ранее обученной нейронной сети.

Оператор кроссинговера для рассматриваемого случая продемонстрируем на паре (V_1, V_2). Вначале сформируем случайным образом целое число из диапазона между 1 и n, которое обозначим как n' и будем использовать в качестве точки кроссинговера. Затем между хромосомами V_1 и V_2 произведем обмен генами, находящимися после n'-го гена, что приводит к появлению двух потомков. Если с помощью обученной ранее нейронной сети можно показать, что полученные потомки являются допустимыми, тогда ими заменяются родительские хромосомы.

Для каждой выбранной родительской хромосомы $V = (\tilde{x}_1, \tilde{x}_2, ..., \tilde{x}_n)$ операция мутации осуществляется следующим образом. Случайным образом выбирается точка мутации как число n', лежащее между 1 и n. Затем из набора случайных величин $X_{n'}$ выбирается новая нечеткая величина, которая заменяет n'-й ген хромосомы V, что приводит к появлению новой хромосомы V'. Если нечеткая хромосома V' оказывается допустимой (это проверяется с помощью обученной ранее нечеткой нейронной сети), она принимается в состав набора хромосом решаемой задачи.

5.3. Алгоритм 6. Гибридный алгоритм

Шаг 1. Сформировать с помощью нечеткого имитационного моделирования набор вход-выходных пар для неопределенных функций, используемых в решаемой задаче.

Шаг 2. Обучить нечеткую нейронную сеть, аппроксимирующую неопределенные функции, с использованием полученного набора нечетких данных.

Шаг 3. Инициализировать *pop_size* нечетких хромосом, допустимость которых может быть проверена с помощью полученной ранее нечеткой нейронной сети.

Шаг 4. Модифицировать нечеткие хромосомы с привлечением операций кроссинговера и мутации, при этом допустимость потомков может быть проверена с помощью полученных ранее нечетких нейронных сетей.

Шаг 5. Вычислить значения целевых функций рассматриваемой задачи для всех нечетких хромосом с помощью полученной ранее нечеткой нейронной сети.

Шаг 6. Вычислить приспособленность каждой из нечетких хромосом, основываясь на значениях целевых функций для них.

Шаг 7. Произвести селекцию хромосом, используя случайный выбор.

Шаг 8. Шаги с четвертого по седьмой выполнить заданное число раз.

Шаг 9. Выдать значение наилучшей хромосомы в качестве оптимального нечеткого решения.

Заключение

Для решения широкого круга задач по нечеткому анализу рисков и нечеткому управлению программными проектами в статье рассмотрены теоретические вопросы построения гибридных алгоритмов и бионических методов, основанных на использовании моделей и методов теории математического программирования в условиях неопределенности, в число которых здесь вошли в основном методы нечеткого программирования [3-6,9].

Перспективными направлениями исследования применения неопределенного математического программирования для решения многих практических задач и в том числе задач по технологии управления программными проектами [1,2, 7-10] являются следующие: неточное математическое программирование, нечетко-случайное программирование, случайно-нечеткое программирование, случайно-неточное программирование, неточно-случайное программирование, нечетко-

неточное программирование, неточно-нечеткое программирование, бислучайное программирование, бинечеткое программирование, бинеточное программирование и неопределенное программирование с множественной неопределенностью [6-10].

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Корячко В.П., Таганов А.И. Программный метод управления рисками качества проекта информационной системы // Научно-технический журнал «Известия Белорусской инженерной академии». Выпуск 1(17)/4, 2004. - С. 168-179.

2. Таганов А.И. Методика анализа и сокращения рисков проектов сложных программных систем по характеристикам качества // Научно-технический журнал «Вестник РГРТУ». – Рязань: РГРТУ, 2010. Вып. 1. С. 77-82.

3. Liu, B, and Lai, Y.-K., Txpected value of fuzzy variable and fuzzy expected value models, IEEE Transactions on Fuzzy Systems, Vol. 10, No. 4, 445-450, 2002.

4. Liu, B, Minimax chanse constrained programming models for fuzzy decision systems, Information Sciences, Vol. 112, Nos. 1-4, 25-38, 1998.

5. Liu, B, Dependent-chance programming in fuzzy environments, Fuzzy Sets and Systems, Vol. 109, No. 1, 97-106, 2000.

6. Лю Б. Теория и практика неопределенного программирования /ер. С англ. – М.: БИНОМ. Лаборатория знаний, 2009. – 416 с.

7. Таганов А.И., Таганов Р.А. Методологические основы методов идентификации рисковых событий проекта // Научно-технический журнал «Вестник РГРТА». Рязань: РГРТА, 2003. Вып. 12. - С. 70-77.

8. Таганов А.И. Применение нечетких множеств для формализации процессов анализа и идентификации важности рисков программного проекта // Научно-технический журнал «Системы управления и информационные технологии». - Москва-Воронеж. Выпуск №4(30), 2007. - С. 46-51.

9. Таганов А.И., Таганов Р.А. Метод определения оптимальной альтернативы реагирования на этапе мониторинга рисков проекта // Научно-технический журнал «Вестник РГРТА». - Рязань: РГРТА, 2003. Вып. 11. - С. 115-118.

10. Таганов А.И., Таганов Р.А. Применение нечетких ситуационных моделей для идентификации рисков программного проекта // Научно-технический журнал «Системы управления и информационные технологии». - Москва-Воронеж. № 4.2(30), 2007. - С. 297- 303.

В.Г. АВЕРЦЕВ, С.В. АВЕРЦЕВ, Г.В. ИГОНИНА
Рязанский государственный радиотехнический университет

РАСЧЕТ ЗАМКНУТЫХ СЕТЕВЫХ МОДЕЛЕЙ С ПОМОЩЬЮ РАЗОМКНУТЫХ СЕТЕЙ ЭКВИВАЛЕНТНОЙ СТРУКТУРЫ

Рассматриваются замкнутые и разомкнутые линейные вероятностные сети. Вводится понятие их эквивалентности и предлагается расчет замкнутых сетей через посредство расчета разомкнутых эквивалентной структуры.

Супер-ЭВМ и большие ЭВМ (мэйнфреймы) в вычислительных центрах, вычислительных сетях часто функционируют в режиме пакетной обработки, основной целью которого является решение максимального числа задач в единицу времени, т.е. увеличение производительности [1]. При исследовании таких режимов, особенно на этапе системного проектирования ЭВМ, в качестве аналитических моделей часто используются замкнутые линейные вероятностные сети (ЗЛВС) [2]. Расчет ЗЛВС многократно сложнее разомкнутых. Причем трудоемкость этих расчетов резко возрастает с увеличением размерности модели (с увеличением числа узлов сети N и числа заявок M, циркулирующих в сети). Поэтому представляют интерес вопросы снижения трудоемкости расчета ЗЛВС большой размерности.

Предлагается расчет таких моделей посредством расчета разомкнутых сетей эквивалентной структуры. Пусть четырехузловая ЗЛВС задана в виде направленного размеченного графа (рис. 1а). Дуги графа помечены соответствующими вероятностями передач. Интенсивности обслуживания в узлах сети соответственно равны $\mu_1 = 10$ з/с, $\mu_2 = 5$ з/с, $\mu_3 = 2{,}5$ з/с, $\mu_4 = 2$ з/с. Введем искусственно нулевой узел на одной из дуг графа (например, на дуге, поток заявок по которой соответствует производительности λ_0 анализируемой ЭВМ) (рис. 1б). Чтобы введение нулевого узла не влияло на функционирование ЗЛВС, полагаем, что среднее время обслуживания заявки в этом узле $\tau_0 = 0$, т.е. заявки в нем не задерживаются.

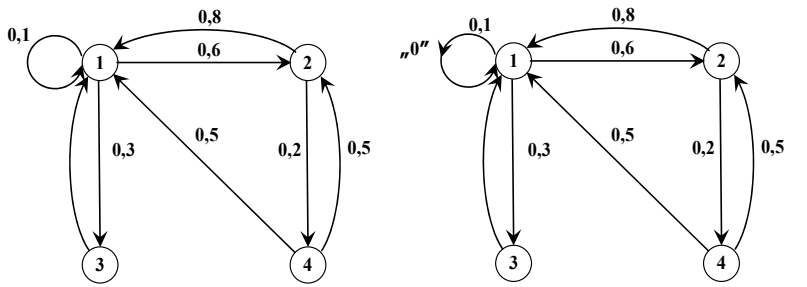

Рис. 1. Заданная модель ЗЛВС

Построим разомкнутую сеть, эквивалентную заданной ЗЛВС. Разомкнутую сеть будем считать эквивалентной заданной ЗЛВС, если она имеет такое же число узлов N, одинаковые значения вероятностей передач P_{ij} (см.графы), одинаковые значения интенсивностей обслуживания μ_i. Введенной нулевой точке в заданной ЗЛВС в разомкнутой сети ставится в соответствие внешний источник заявок с интенсивностью λ_0. Из сказанного следует, что граф эквивалентной разомкнутой сети имеет тот же вид, что и граф заданной ЗЛВС с нулевой точкой.

Далее в этой разомкнутой сети с помощью изменения значений интенсивности внешнего источника λ_0 подбирается такой режим, при котором среднее число заявок $m_1 + m_2 + \ldots + m_N = M_{cp} = M$, т.е. равно числу заявок, циркулирующих в ЗЛВС.

Затем рассчитываются характеристики эквивалентной разомкнутой сети, на основе которых определяются характеристики заданной ЗЛВС.

В качестве примера независимо друг от друга определим характеристики четырехузловых заданной ЗЛВС и разомкнутой приведенных на рис. Результаты расчетов сведены в таблицу, а для коэффициентов загрузки первого узла $\rho_1 = \lambda_1 / \mu_1$ построены графики. Понятно, что можно было бы определить только коэффициенты загрузки одного узла, а интенсивности входных (выходных) потоков и коэффициенты загрузки других узлов затем легко можно определить, зная коэффициенты передачи.

Табл. 1.

M \ ρ_i	$\rho_{1,P}$	$\rho_{2,P}$	$\rho_{3,P}$	$\rho_{4,P}$	$\rho_{1,3}$	$\rho_{2,3}$	$\rho_{3,3}$	$\rho_{4,3}$
1	0,19	0,253	0,227	0,125	0,237	0,317	0,286	0,16
2	0,31	0,414	0,372	0,207	0,376	0,502	0,453	0,251

3	0,392	0,522	0,471	0,261	0,465	0,621	0,557	0,31
4	0,454	0,606	0,545	0,302	0,527	0,707	0,633	0,352
5	0,5	0,667	0,599	0,333	0,571	0,76	0,689	0,379
10	0,617	0,822	0,664	0,411	0,678	0,903	0,739	0,453

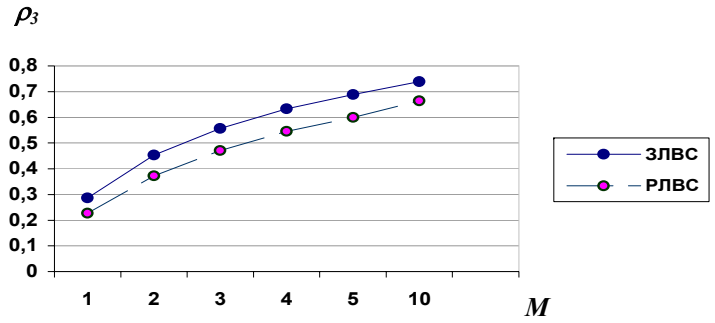

Рис. 2. Зависимость ρ_i от M четырехузловой ЗЛВС и эквивалентной РЛВС

Из таблицы и графиков видно, что значения ρ_i и λ_i (они пропорциональны ρ_i) для разомкнутой сети несколько меньше, чем соответствующие значения для ЗЛВС. Физически это объясняется тем, что заявка, вернувшаяся во внешний источник в разомкнутую сеть не возвращается, а очередная заявка поступает в нее только через время, равное $1 / \lambda_0$. В замкнутой же сети заявка, попавшая в нулевой узел, проскакивает его не задерживаясь. Следовательно, значение интенсивности λ_0 (как и всех других интенсивностей λ_i и коэффициентов загрузки) в ЗЛВС будут несколько больше.

Нетрудно заметить, что разность между характеристиками рассматриваемых сетей с увеличением числа узлов будет уменьшаться, так как значимость одного узла, в том числе и нулевого (или внешнего источника) при этом увеличении будет уменьшаться. Эта разность будет уменьшаться и при увеличении числа заявок M, циркулирующих в сети, так при увеличении $M_{ср} = M$ увеличивается интенсивность внешнего источника λ_0 и, следовательно уменьшается величина времени $1 / \lambda_0$.

Математически эту разность и указанное ее уменьшение можно оценить следующим образом.

Не умаляя общности результатов, будем полагать, что все узлы в сети одноканальные.

Известно [3], что коэффициенты загрузки всех узлов в сбалансированной ЗЛВС равны $\rho_i = M / (M + N - 1)$. Среднее число заявок в одноканальной системе массового обслуживания с неограниченной длиной очереди равно $\rho = \rho / (1 - \rho)$. Если понятие сбалансированности распространить на разомкнутую сеть, то из условий равенства коэффициентов загрузки всех узлов следует

$$m_1 + m_2 + \ldots + m_N = \rho_1 / (1 - \rho_1) + \rho_2 / (1 - \rho_2) + \ldots + \rho_N / (1 - \rho_N) = M_{ср} = M$$

или $N\rho/(1-\rho) = M$ откуда $\rho = \dfrac{M}{M+N}$, т.е.

коэффициент загрузки в сбалансированных разомкнутых сетях при одинаковых значениях $M_{ср} = M$ несколько меньше, чем в соответствующих замкнутых. А величина разности равна

$$\Delta_\rho = M / (M + N - 1) - M / (M + N) = M / (M + N - 1) \cdot (M + N).$$

Очевидно, что при больших M и (или) N эта разность меньше.

Однако сбалансированные ЗЛВС и без того рассчитываются просто, так как просто определяются коэффициенты загрузки их узлов. Поэтому в приведенном примере характеристики определяются для несбалансированной ЗЛВС (коэффициент сбалансированности $\delta_\varpi = 0{,}5$, т.е. коэффициент загрузки наименее загруженного узла в два раза меньше коэффициента загрузки узкого места [3]).

Так как при небольшой размерности модели (с числом узлов N и числом циркулирующих заявок M не более 4) трудоемкость расчета ЗЛВС ещё не велика, то определять ее характеристики через характеристики эквивалентной разомкнутой сети не имеет особого смысла. А для расчета ЗЛВС большей размерности имеет смысл ввести поправочный коэффициент $\kappa_П$.

В результате проведения многочисленных расчетов ЗЛВС и эквивалентных им разомкнутых сетей различной размерности и сопоставления их характеристик можно сделать следующие выводы.

При размерности ЗЛВС, равной 4, отличие характеристик ЗЛВС от эквивалентной сети составляет примерно 20 %.

При размерности модели ≥ 5 поправочный коэффициент $\kappa_П$ практически линейно уменьшается при увеличении размерности. В диапазоне размерности от 5 до 10, т.е., если $5 \le M \le 10$ и $5 \le N \le 10$,

поправочный коэффициент в среднем равен 1,1. При размерности модели более 10, но не более 20 κ_{II} равен в среднем 1,05. При размерности ЗЛВС более 20-ти поправочный коэффициент вообще можно не учитывать, так как характеристики заданной и эквивалентной сетей практически совпадают.

Таким образом, построив разомкнутую сеть, эквивалентную заданной ЗЛВС, и определив, например, коэффициент загрузки, хотя бы одного узла этой разомкнутой сети и умножив его затем на поправочный коэффициент можно определить характеристики исходной ЗЛВС. Погрешность полученных результатов при этом составляет 1-2 процента.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Пескова С.А., Кузин А.В., Волков А.Н. Сети и телекоммуникации. М.: Академия, 2008. 352 с.

2. Майоров С.А., Новиков Г.И., Алиев Т.И., Махарев Э.И., Тимченко Б.Д. Основы теории вычислительных систем. М.: Высшая школа, 1978. 400 с.

3. Аверцев В.Г., Игонина Г.В. Оценка эффективности мультипрограммного режима замкнутых структур. Межвуз. сб. науч. трудов. Информационные технологии и телекоммуникации в образовании и науке. Рязань, 1985. с. 10-14.

В.А. АСТАШИН

Рязанский государственный радиотехнический университет

МАТРИЧНЫЙ МЕТОД ВЫЧИСЛЕНИЯ ТАБЛИЦ МАРШРУТИЗАЦИИ В РАСПРЕДЕЛЕННОЙ СЕТИ

Рассмотрен вариант алгоритма вычисления таблиц маршрутизации для сети, использующей протокол маршрутизации по состоянию связей

Одна из главных задач, которую решает протокол маршрутизации при динамическом управлении сетью – это вычисление таблиц маршрутизации.

При вычислении таблиц матричным методом [1,2] используется специальная математическая операция умножения матриц. Некоторый элемент столбца m и строки j матрицы A, полученной умножением матриц B и L, вычисляется по формуле: A[m,j]=min{B[m,i]+L[i,j]} для i = 1,2,3..n, где n – число узлов сети.

Если известна матрица расстояний L, в которой записаны прямые связи между всеми узлами сети, то путем возведения этой матрицы в степень n-1 получим дистанционную матрицу. Компоненты этой матрицы будут равны кратчайшим расстояниям между узлами. Кроме того, умножив дистанционную матрицу на модифицированную матрицу расстояний, можно получить и матрицу путей. Компоненты полученных матриц содержат информацию, необходимую для централизованного вычисления таблиц маршрутизации.

Ниже предложен вариант децентрализованного вычисления таблиц маршрутизации. Полагаем, что каждый из узлов сети имеет полную информацию о состоянии связей в сети, на основании которой протокол строит матрицу расстояний L между узлами сети и вычисляет свои таблицы маршрутизации первого и второго выборов.

Работу алгоритма поясняют листинги фрагментов программы, написанной на зыке Турбо-Паскаля. Первоначально вычисляется таблица первого выбора (Листинг 1).

В программе используется двумерный массив L расстояний между узлами и два одномерных массива S1 иP1. Массив S1 инициализируется информацией, записанной в столбце m массива L, а массив P1 номерами узлов, непосредственно подключенных к узлу m, для которого вычисляется таблица маршрутизации.

В процессе выполнения основного цикла происходит обновление массивов S1 и P1. Вычисление прекращается, когда массив S1 не будет изменяться при выполнении цикла по параметру j.

В результате массив S1 будут содержать кратчайшие пути от выбранного узла m до остальных, а P1 - номера соседних узлов, на которые маршрутизатор должен отправлять пакеты в соответствии с адресами узлов назначения.

Листинг 1

```
for i := 1 to n do begin    {Инициализация массивов}
      S1[i]:=L[m,i];
      if S1[i]<max then P1[i]:=i
      else P1[i]:=0;
      end;
 repeat                {Основной цикл }
       flag:=0;
    for j:=1 to n do   {Выбор номера целевого узла}
       for i := 1 to n do
   if (i<>j) and (j<>m) then
```

```
 if ((S1[i]+L[i,j])<S1[j])  then begin   { Вычисляем массивы S1 и
P1}
  S1[j]:=S1[i]+L[i,j];  P1[j]:=P1[i];
 flag:=1
      end;
 until flag=0;
```

За следующий шаг программа вычисляет таблицу маршрутизации второго выбора. При этом используются готовые массивы S1 и P1. Для хранения промежуточных вычислений в программе выделены одномерные массивы S и P, а для индикации процесса заполнения таблицы маршрутизации - массив F. Фрагмент программы показан в листинге 2.

Алгоритм последовательно удаляет из матрицы L по одной связи, входящей в кратчайший маршрут первого выбора, вычисляет матричным методом массив S и заполняет массив маршрутов P. Если вычисленный массив P отличается от массива P1, то несовпадающие элементы массива P копируются в массив P2, а элементы массива S – в массив S2.

Затем восстанавливается удаленная из матрицы расстояний связь, и повторяется вычисление по параметру r. При этом заполнение таблицы маршрутизации фиксируется в массиве F. Процесс вычисления прекращается, когда массив F будет заполнен единицами. В результате выполнения второго шага алгоритма в массивах S2 и P2 будут записаны пути и маршруты второго выбора.

Следует отметить, что использование массива F позволяет существенно сократить время вычисления таблицы второго выбора по сравнению с простым перебором всех вариантов связей. Особенно преимущество проявляется в том случае, когда узел имеет малое число связей с соседними узлами. Например, если узел имеет одну связь, то таблица маршрутизации вычисляется за один проход тела цикла.

Листинг 2

```
 for i:=1 to n do F[i]:=0;      {Инициализация массива F}
 for r:=1 to n do               {Поиск нулевых компонент массива F}
 if (F[r]=0) and (r<>m) then begin
 buf:=L[m,P1[r]];              {Сохраняем удаляемую связь в буфере}
l[m,P1[r]]:=max;               {Удаляем связь из матрицы расстояний}
 for i := 1 to n do begin      {Инициализация  S и P }
 S[i]:=L[m,i]; if S[i]<max
then P[i]:=i else P[i]:=0
  end;
repeat                         {Вычисление  S и P}
```



```
 flag:=0;
for j:=1 to n do
    for i:=1 to n do
 if ((i<>j)and (j<>m)) then
       if (S[i]+L[i,j])<S[j]  then begin
 S[j]:=S[i]+L[i,j]; P[j]:=P[i]; flag:=1
                end;
 until flag=0;
for k:=1 to n do                {Заполняем  таблицы второго выбора}
 if P[k]<>P1[k] then begin  {Если не совпадают массивы P и P1}
P2[k]:=P[k]; S2[k]:=S[k]; F[k]:=1
          end;
 L[m,P1[r]]:=buf; {Восстанавливаем удаленную связь в матрице
L}
 end;  {Конец цикла по r}
```

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Камер, Дуглас. Компьютерные сети и Internet. Разработка приложений для Internet: Издательский дом “Вильямс”, 2002.

2. Лазарев В.Г. Саввин Г.Г. Сети связи, управление и коммутация М., Связь, 1973.

А.В. БАКУЛЕВ, М.А. БАКУЛЕВА
Рязанский государственный радиотехнический университет

АЛГОРИТМ ДОБАВЛЕНИЯ ДАННЫХ В ХРАНИЛИЩЕ

Рассматриваются вопросы актуализации хранилища данных, построенного на основе кратномасштабного представления.

На сегодняшний день основу большинства информационно-поисковых систем составляют базы данных (БД), построенные на основе реляционной модели [1, 2]. Основной идеей реляционной модели является нормализация с целью экономии ресурсов памяти [3]. Сложные по структуре и многообразию связей реляционные БД не отвечают требованиям производительности аналитических приложений и соответственно не могут выполнять функции информационной поддержки процедуры принятия решений, поэтому в современных информационных системах наиболее востребованы денормализованные БД – хранилища данных (ХД). ХД являются основным источником данных опера-

тивно-аналитических подсистем (OLAP-систем) систем поддержки принятия решений (СППР).

Одно из основных требовании, предъявляемых к ХД , – высокой производительность обработка запросов желательно в реальном масштабе времени [4]. Для решения задачи быстродействия выполнения аналитических запросов требуется пересмотреть структуру данных ХД.

Классическое представление данных, разработанное основателем теории ХД – Биллом Инмоном [5] обладает одним существенным недостатком: невозможностью производить суммирование данных по произвольным диапазонам, что часто бывает необходимым в реальных аналитических задачах. Однако идея сохранения всех возможных агрегированных данных является утопической, так как вызывает взрывное увеличение требуемых ресурсов памяти [6].

Как возможный способ решения задачи агрегирования (суммирования) предлагается метод кратномасштабного представления данных ХД, в частности, вейвлет сжатия.

Основная идея, которая стоит за всеми кратномасштабными методами заключается в том, чтобы представлять функции набором коэффициентов, каждый из которых дает некую ограниченную информацию о ней. Наиболее универсальный способ получения подобного разложения — вейвлет-преобразования над исходной функцией. Классическая форма кратномасштабного анализа разбивает ряд по базису вейвлетов, причем каждый вейвлет — это ни что иное, как масштабированная и сдвинутая копия единственной функции. Именно базис вейвлетов позволяет реализовать переход между различными уровнями иерархии [7].

Пусть временной ряд *W(t)* отображает численные показатели, содержащиеся в таблице фактов ХД. Мощность данного ряда $|W(t)| = n$, тогда количество уровней иерархии *p* вычисляется по формуле $p = \log_2 n$. Кратномасштабное представление данных в базисе Хаара выполняется по схеме, представленной на рис. 1:

<table>
<tr><td>$w_{0,1}$</td><td rowspan="2">$w_{1,1} = \frac{w_{0,1} + w_{0,2}}{2}$</td><td rowspan="4">$w_{2,1} = \frac{w_{1,1} + w_{1,2}}{2}$</td><td></td></tr>
<tr><td>$w_{0,2}$</td><td></td></tr>
<tr><td>$w_{0,3}$</td><td rowspan="2">$w_{1,2} = \frac{w_{0,3} + w_{0,4}}{2}$</td><td></td></tr>
<tr><td>$w_{0,4}$</td><td></td></tr>
</table>

<table>
<tr><td>…</td><td>…</td><td>...</td><td>…
$w_{p,m}$
…</td></tr>
<tr><td>$w_{0,n-3}$</td><td rowspan="2">$w_{1,\frac{n}{2}-1} = \frac{w_{0,n-3} + w_{0,n-2}}{2}$</td><td rowspan="2">$w_{2,\frac{n}{4}} = \frac{w_{1,\frac{n}{2}-1} + w_{1,\frac{n}{2}}}{2}$</td><td></td></tr>
<tr><td>$w_{0,n-2}$</td><td></td></tr>
<tr><td>$w_{0,n-1}$</td><td rowspan="2">$w_{1,\frac{n}{2}} = \frac{w_{0,n-1} + w_{0,n}}{2}$</td><td></td></tr>
<tr><td>$w_{0,n}$</td><td></td></tr>
</table>

Рис. 1. Кратномасштабное представление ряда

где $w_{p,m}$ — элемент с номером *m* на уровне разложения *p*.

Процедура добавления данных в ХД описывается следующим алгоритмом:

Начальные условия.

ХД содержит кратномасштабное представление ряда *W(t)* (рисунок 1). В ХД поступил новый ряд данных *S(t)*. Мощность данного ряда $|S(t)| = k$, тогда количество уровней иерархии *h* вычисляется по формуле $h = \log_2 k$.

Шаг 1. Производится вейвлет-разложение ряда *S(t)* по аналогии с исходным рядом *W(t)*.

Шаг 2. Пересчет коэффициентов разложения уровня *h* объединенных рядов *S(t)* и *W(t)*.

1) $x=1, y=1$;

2) $W_{hx} = \frac{W_{hy} + W_{hy+1}}{2}$;

3) если $x \geq \frac{n+k}{2^h}$, то переход к шагу 3.

4) $x=x+1$;

5) $y=y+1$.

Шаг 3. Проверка окончания алгоритма.

1) если $h \geq \log_2(n+k)$, то переход к шагу 4.

2) $h=h+1$, переход к шагу 2.

Шаг 4. Конец.

Основное преимущество данного представления заключается в том, что упраздняется необходимость заранее рассчитывать и хранить агрегированные данные, а также пересчитывать их при добавлении новых данных.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Дарвин Х., Дэйт К. Системы баз данных третьего поколения: Манифест//СУБД. – 1995. – № 2.

2. Зильбершатц А., Стоунбрейкер М., Ульман Д. Базы данных: достижения и перспективы на пороге 21-го столетия // СУБД. – 1996. № 3.

3. Дэйт К. Введение в системы баз данных. – М.: «Вильямс», 1999. – 848 с.

4. Klein H.K., Hirschheim R.A. A Comparative Framework of Data Modelling Paradigms and Approaches // The Computer Journal. – 1987. – No. 1 – P.8 – 15

5. Inmon B. Building the Data Warehouse. – New York: John Wiley & Sons, 1996.

6. Gupta H. Selection of Views to Materialize in a Data Warehouse. – In Proc. of the 6th Intl. Conf. on Database Theory. – 1997. – P.98 – 112

7. Столниц Э., ДеРоуз Т., Салезин Д. Вейвлеты в компьютерной графике. – Ижевск: НИЦ «Регулярная и хаотическая динамика», 2002. – 272 с.

Добеши И. Десять лекций по вейвлетам. – Ижевск: НИЦ «Регулярная и хаотическая динамика», 2001. – 464 с.

А.И. БАРАНЧИКОВ, Е.А. БАРАНЧИКОВА
Рязанский государственный радиотехнический университет

АНАЛИЗ ПОДХОДОВ К ПРОГРАММНОЙ РЕАЛИЗАЦИИ СТАТИСТИЧЕСКОГО ПОЧТОВОГО ФИЛЬТРА

Выбор критериев для оценки эффективности различных подходов к реализации почтовых статистических фильтров и применение выбранных критериев для практической реализации фильтров.

В последнее время процесс создания любой информационной системы (ИС) предприятия предваряет стадия всестороннего анализа подходов к разработке требуемого программного обеспечения, выбора средств его реализации, отвечающих заданным условиям, а также методологии разработки программного обеспечения. Как показывает

опыт, без предварительной тщательной проработки этих этапов, даже небольшие проекты вряд ли могут быть успешными. В данной статье предлагается анализ различных подходов к реализации программного статистического почтового фильтра, как части информационной системы малого предприятия с ограниченными ресурсами [1].

При написании программного обеспечения важно заранее определить перечень критериев, которым оно должно удовлетворять на выходе. Выбор каждого отдельного критерия обосновывается с той или иной точки зрения. На основе выбранных критериев определяются дальнейшие подходы к разработке программного обеспечения, производится выбор инструментальных средств, платформы разработки, возможных решений при написании отдельных частей ИС. Поскольку часто речь идет не просто об отдельной программе, а под программным обеспечением понимается целый комплекс средств, или даже целая информационная система, то заранее необходимо разбить такую систему на отдельные составляющие части, и отдельно для каждой части прорабатывать критерии и подходы к разработке.

К настоящему моменту разработано огромное количество программ фильтрации электронной корреспонденции. Но поскольку постоянно совершенствуются не только сами программы фильтрации, но и инструменты спамеров, то можно сказать, что процесс написания и совершенствования фильтров имеет постоянный характер. Определение четких критериев и разработка единых подходов к проектированию такого рода программ позволит ускорить этот процесс, уменьшить финансовые затраты как на их создание, так и поддержание уже существующего программного обеспечения в актуальном состоянии.

Рассмотрим и проанализируем различные подходы к написанию программного обеспечения на основе реализации предложенного автором алгоритма статистической фильтрации почты [2]. Представленный алгоритм может быть реализован с помощью следующего программного обеспечения, в состав которого входят:

1. Клиентское приложение для администрирования базы знаний, настройки критериев работы статистического фильтра, а также получения отчетных статистических данных по процессу фильтрации входящей электронной корреспонденции;
2. Интерфейс для взаимодействия с почтовым сервером;
3. Программа статистической фильтрации;
4. Модуль анализа текстовой информации;
5. Модуль принятия решений на основе проведённого анализа;

6. База знаний;
7. Программное обеспечение для интеграции почтового статистического фильтра и почтового клиента.

Каждый из этих модулей можно разбить на более мелкие составляющие, но при определении критериев к разработке, достаточно ограничится детализацией на этом уровне.

Определим основные критерии, используемые при разработке программного обеспечения.

Критерии, предъявляемые к конечному продукту:

1. Кроссплатформенность – характеризует платформонезависимость информационной системы во время ее эксплуатации, а также, что немаловажно, во время ее разработки. В последнее время все наибольшую популярность приобретают так Unix-подобные операционные системы, используемые как в качестве серверных платформ, так и в качестве клиентского программного обеспечения. В их состав входят широко распространенные почтовые серверы[3], такие как SendMail и PostFix, ориентированные в основном на работу под Unix системами. В разработке нашего почтового фильтра мы будем ориентироваться на его интеграцию почтовым сервером SendMail, предоставляющим возможность наиболее гибкой настройки [4].
2. Быстродействие – характеризует производительность системы, а также ее возможность работать в многопоточном режиме.
3. Минимизация затрат на модификацию программного обеспечения и его сопровождение.
4. Интегрируемость с уже существующими решениями – позволяет использовать уже существующее программное обеспечение при создании собственной информационной системы.
5. Надежность — количественные и качественные характеристики таких аспектов программного обеспечения, как: завершенность, устойчивость к дефектам, восстанавливаемость и доступность/готовность.
6. Защищенность— полнота использования доступных методов и средств защиты программного обеспечения от потенциальных угроз и достигнутой при этом безопасности функционирования информационной системы. Наиболее широко и детально методологические и системные задачи оценки комплексной защиты информационных систем изложены в трех частях стандарта ISO 15408:1999-1—3 «Методы и средства обеспечения безопасности. Критерии оценки безопасности информационных технологий».
7. Сопровождаемость - полнота и достоверность документации о состояниях программного обеспечения и его компонентов, всех предполагаемых и выполненных изменениях, позволяющей установить текущее состояние версий ПО в любой момент времени и историю их развития.

Определим степень значимости каждого критерия, применительно ко всем модулям разрабатываемого почтового статистического фильтра (Табл. 1).

Табл. 1. Степень значимости критериев для различных модулей ИС

	Критерии						
Модули	1	2	3	4	5	6	7
1 Клиентское приложение	высокая	низкая	низкая	низкая	средняя	высокая	высокая
2 Интерфейс работы с сервером	высокая	высокая	высокая	высокая	высокая	высокая	средняя
3 Программа фильтрации	высокая	высокая	высокая	высокая	высокая	высокая	средняя
4 База знаний	высокая	высокая	средняя	средняя	высокая	высокая	средняя
5 ПО для интеграции с почтовым клиентом	высокая	средняя	средняя	высокая	высокая	средняя	средняя

Для удовлетворения данных критериев, важным является корректный выбор средств разработки ИС, таких как язык, среда разработки и СУБД. Такой выбор должен также определяться рядом требований.

Требования предъявляемые к языку и среде разработки:

1. Наличие средств для анализа текстовой информации.
2. Объектно-ориентированный язык.
3. Скорость разработки.
4. Быстродействие проектируемых решений.
5. Наличие средств и компонентов для реализации заданного функционала.
6. Бесплатность.
7. Кроссплатформенность.

Исходя из вышеперечисленных критериев, была выбрана бесплатная среда разработки с открытым исходным кодом NetBeans. В ее состав входят модули для работы с такими объектно-ориетированных языками, как Java, C++, Ruby и некоторые другие. NetBeans имеет удобный интерфейс и вспомогательный модули, повышающие производительность разработки (подсветка синтаксис, автоматическое дополнение программного кода, работа с системами контроля версий и прочее).

Язык C++ сохраняет производительность языка C с высокопроизводительной адресной арифметикой, однако имеет множественные расширения, в том числе, в сторону объектно-ориентированного программирования. Так, базовая система классов STL утверждена стандартами, что обеспечивает возможность компиляции исходного кода различными компиляторами в различных средах разработки под управлением множества операционных систем.

Изначально, SendMail был написан на языке C. Поэтому, библиотека Milter и Milter API также реализованы на этом языке. Множественные попытки портировать Milter API на другие языки (Ruby, Java и пр.) не являются пригодными для промышленной эксплуатации. Средства языка C++ позволяют использовать исходный API на языке C без дополнительных преобразований, что повышает надежность проектируемого программного продукта.

Выбор библиотеки для обработки текстовой информации в программе фильтрации остановился на PCRE, как наиболее распространенную для всех операционных систем, стабильно работающую, и входящую в стандартную поставку для семейства Unix-подобных операционных систем. Большое количество серверного программного обеспечения также используют данную библиотеку.

Для разработки клиентского приложения администрирования почтового фильтра наиболее удобным является язык Java. Приложения на Java не требуют перекомпиляции при переносе с одной операционной системы на другую. При этом они сохраняют одинаковый внешний вид пользовательского интерфейса. Приложение написанное на Java легко может быть представлено в виде аплета, что дает дополнительные преимущества при установке более чем на одной ЭВМ.

Средства разработки Java-приложений включают в себя мощную библиотеку классов, предоставляющих удобный и высокоуровневый доступ к базам данных, пользовательскому интерфейсу, сетевым протоколам (в том числе, прикладного уровня). Продуманная система классов позволяет существенно сократить сроки разработки программного обеспечения и упростить программный код, что, в свою очередь,

влечет уменьшение количества ошибок программирования.

Требования предъявляемые к СУБД:

1. Наличие средств для анализа текстовой информации.
2. Наличие серверного программирования.
3. Близость к стандартам SQL.
4. Быстродействие.
5. Бесплатность.
6. Кроссплатформенность.
7. Унифицированные средства аутентификации, такие как SASL.

В результате анализа СУБД наиболее подходящей была выбрана PostgreSQL. Это наиболее разработанная СУБД среди аналогичных продуктов с открытым исходным кодом. В данной СУБД присутствуют все необходимые для работы инструменты обработки данных, написания собственных функций, ограничения целостности. PostgreSQL постоянно пополняется новым функционалом и поддерживает многие расширения к стандарту SQL. Как показали исследования [5] данная СУБД имеет очень хороший оптимизатор запросов, что приближает ее быстродействие к таким дорогим промышленным СУБД, как Oracle. Также существуют готовые стандартные библиотеки для взаимодействия программ, написанных на C++ и Java , с базой данных, работающей на PostgreSQL. Достаточно распространенная в настоящее время СУБД MS SQL Server работает со своим собственным диалектом языка SQL, который далек от стандартов, имеет урезанное серверное программирование и более низкую производительность по сравнению с PostgreSQL, а также не является бесплатным, свободно распространяемым программным обеспечением, и поэтому не рассматривалась нами в качестве одного из возможных решений в выборе СУБД.

Для написания программного обеспечения по интеграции с почтовым клиентом, воспользуемся существующим решением, предоставляемым почтовым клиентом Kmail, в который встроен готовый интерфейс для взаимодействия пользователя и почтового сервера.

Таким образом, представленный подход позволяет реализовать гибкий настраиваемый программный комплекс, обеспечивающий надежную фильтрацию почты, а так же быструю адаптацию фильтра к изменяющимся условиям среды.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. А.И. Баранчиков, Е.А. Баранчикова Негативное влияние «спама» на работу малых информационных систем и основные методы борьбы с ним. // Вестник РГРТУ. 2007, № 21 с. 51-53.

2. Баранчикова Е.А. Способ фильтрации электронных почтовых сообщений // Вестник РГРТУ. 2009. №2. – С. 56—60.

3. Крупин Д. Идеальный почтовый сервер/ Д. Куприн// Компьютер Пресс [Электронный ресурс]. – Электрон. журн. – 2008. – №5 – Режим доступа: http://www.compress.ru/article.aspx?id=19066&iid=883

4. Welcome to sendmail.org [Электронный ресурс]. – URL: www.sendmail.org

5. Баранчиков П.А. Алгоритмы организации и модели ограничения доступа к отдельным записям таблиц реляционных баз данных.: Автореферат дис. канд. техн. наук. Рязань, 2009. - 16с.

В.А. БОГОНАТОВ, Д.А. ПЕРЕПЕЛКИН
Рязанский государственный радиотехнический университет

ВИРТУАЛЬНЫЙ СКАНИРУЮЩИЙ ЗОНДОВЫЙ МИКРОСКОП

Рассмотрены цели создания и основные особенности разработанной программы-эмулятора, ее отличия от «тяжелой» версии-прототипа, а так же принципы построения 3D-модели (поверхности) на основе снимка, сделанного микроскопом.

В настоящее время оборудование, подобное сканирующему зондовому атомно-силовому микроскопу, отличается высокой стоимостью и требует большой бережности в обращении, не считая немалой сложности в его настройке и использовании. Эти обстоятельства делают его неприменимым в образовательном процессе, например, для самостоятельной работы студентов.

В качестве программных систем, применяемых в настоящее время, для анализа снимка зондового микроскопа используется демонстрационная программа-прототип, сделанная в системе графического программирования LabVIEW. Данная программа отличается большим размером дистрибутива, а так же низкой стабильностью и скоростью работы.

Поэтому предложена программная система, лишенная вышеуказанных недостатков. Разработанная система визуализирует основные шаги работы с данным видом микроскопа, а именно: выбор образца, прицеливание лазера, спуск зонда, а так же получение снимка и его

анализ. Для удобства решения поставленная задача была разбита на несколько этапов.

Этап выбора образца заключается в определении (выборе) исследуемого вещества (материала) из представленного списка, отображаемого в виде древовидной структуры. Список формируется на основе базы данных, в которой хранятся доступные для исследования образцы (в виде совокупности специализированных файлов). Затем происходит переход к прицеливанию лазера, а в параллельном программном потоке начинается построение поверхности на основе снимка вещества для выбранного образца.

Основной задачей второго этапа является получение резонансной кривой специфического вида, а именно попадающей в заданные резонансные пределы. Положение кривой зависит от точности наведения лазера. Данный этап необходим для того, чтобы снимок, полученный в результате работы сканирующего атомно-силового микроскопа, получился достаточно четкий и не содержал «ошибок».

После наведения лазера программа переходит к следующей стадии – спуску зонда. На данном этапе происходит визуализация спуска зонда, как с помощью прямого отображения (анимации), так и с помощью графика, показывающего зависимость высоты зонда над поверхностью от времени.

На четвертом этапе после сканирования образца и получения снимка поверхности начинается анализ этого снимка, а именно построение разрезов поверхности в вертикальной плоскости по двум осям в виде графиков высот (рис. 1).

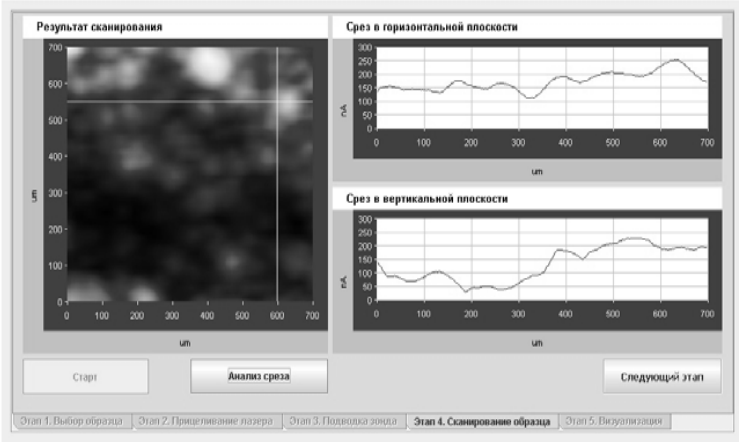

Рис. 1. Анализ снимка поверхности материала с построением ее разрезов в различных плоскостях

Далее происходит отображение параллельно построенной поверхности, сформированной на основе специализированного алгоритма обработки снимка, который базируется на том, что цветность пикселя полученного изображения прямо пропорциональна физической высоте точки на образце. Предложенная реализация на языке программирования Java лишена недостатков программы-прототипа, а использование технологии Java Applet сделало возможным ее размещение на любой html-странице. В частности, beta-версия программы уже размещена на нано-портале РГРТУ и успешно используется. В ближайшем будущем планируется доработка проекта: расширение функционала, подключение к базе данных, а так же эмуляция работы еще двух типов сканирующих микроскопов.

В.Д. ГАМАЗИН, А.И. СУХОРУК, Р.В. ШЕМЯКИН, О.П. БИРЮКОВА
г. Рязань, НПЦ завода «Красное знамя»

ЭКРАН ДЛЯ ОТОБРАЖЕНИЯ БОЛЬШИХ ФОРМАТОВ DVI/VGA ОТ ОДНОГО ИСТОЧНИКА ИНФОРМАЦИИ

Рассматриваются особенности построения экрана коллективного пользования на плазменных панелях переменного тока MIS-4220, MIS-4230, OPM-4250, обеспечивающие повышенное качество отображения больших форматов DVI/VGA от одного источника информации.

В настоящее время для отображения картографической, графической, табличной, телевизионной и др. информации используют экраны коллективного пользования с высоким качеством изображения. Улучшение характеристик экранов обеспечивается за счет совершенствования индикаторов и методов управления ими. Эти экраны имеют стандартный межмодульный интерфейс, позволяют управлять режимами работы основных узлов экрана и осуществлять контроль их состояния с помощью ЭВМ.

В статье рассматриваются особенности построения экрана коллективного пользования (ЭКП) на плазменных панелях переменного тока MIS-4220, MIS-4230, MIS-4250, обеспечивающие повышенное качество изображения от получаемой с ЭВМ информации в форматах DVI или VGA.

Информационное поле экрана построено по модульному принципу. Основу экрана составляет модуль индикации (МИ), представляющий собой газоразрядную индикаторную панель переменного тока

со схемой управления. Количество пикселей RGB люминофоров на одной панели равно 853x480. Входной информационный контроллер МИ осуществляет прием, ретрансляцию, коммутацию и масштабирование в соответствии с командами управления входной видеоинформации, поступающей от ЭВМ (DVI/VGA) или другого источника видеоинформации (Video, S-Video, DTV) и передачу ее в контроллер управления МИ для отображения.

Существующие способы соединения МИ по информационным входам DVI/VGA [1,2] предполагают подачу одной и той же информации (кадра) на все МИ экрана, а выбор «своего» участка кадра для каждого МИ осуществляется заданием его положения в кадре. При этом рекомендуемый формат отображения ограничен 1600x1200 [1] или 1706x960 [2] пикселями для DVI/VGA-входов при любом количестве МИ по горизонтали и вертикали, т.е. разрешающая способность МИ при отображении от одного источника информации уже при трех модулях по горизонтали не используется.

Для отображения больших форматов DVI/VGA от одного источника без потери качества в ЭКП вводится разбиение кадра на фрагменты с выводом каждого фрагмента на свой МИ в формате 853x480 пикселей, для улучшения качества VGA-изображения вводится преобразование входного аналогового VGA-формата в выходной цифровой DVI-формат, вводится также повышение частоты кадра с 38-60 Гц до 60-85 Гц.

Для повышения качества отображения оперативной информации, получаемой от ЭВМ в форматах: DVI до 3412x2880 в режиме «dual link» или 1920x1200 в режиме «single link» и VGA до 2048x1536 с 24-разрядной кодировкой цветности и частотой вертикальной развертки не менее 60Гц в схему ЭКП вводится контроллер экрана.

Контроллер экрана обеспечивает прием цифровой или аналоговой информации (DVI/VGA), преобразование кадра в шесть (или менее) фрагментов с разрешением до 853x480 пикселей каждый и подачу DVI-информации на входы шести (или менее) МИ. Если общее количество МИ в ЭКП больше шести, контроллер экрана преобразует кадр в шесть (или менее) фрагментов с разрешением до 1706x960, а к каждому его DVI-выходу подключается до четырех МИ (2x2). Для снижения мерцания изображения контроллер экрана увеличивает частоту кадра до 60 или 85 Гц. Контроллер экрана может использовать дополнительный режим для устранения остаточной статической информации МИ при длительном отображении одного и того же кадра.

Максимальное количество МИ в ЭКП равно 24.

ЭКП имеет систему звукового сопровождения видеоинформации как для работы в комплексе с ЭВМ (интерфейс сопряжения RS-232/485), так и в автономном режиме.

Несколько ЭКП могут быть объединены в видеостену с общим управлением (от одной управляющей ЭВМ) контроллерами экранов, системой звукового сопровождения (для видеостен на `MIS-4220, MIS-4230, MIS-4250`) и общей видеоинформацией Video, S-Video, DTV (для видеостен на `MIS-4220, MIS-4230`) для всех МИ видеостены, и имеющих различные источники DVI/VGA-информации, поступающей на входы контроллеров экрана от информационных ЭВМ.

Конструктивное исполнение табло и видеостены позволяет монтировать их на пол, платформу или в проём стены, обеспечивая подход к изделию спереди и сзади, для чего предусмотрены соответствующие узлы крепления конструктивов.

Программное обеспечение ЭКП (табло и видеостены) разработано для ОС LINUX/UNIX на языке программирования C++ в среде QT 3.3.X. Управление экранами (и табло, и видеостеной) осуществляется одной и той же программой с разными входными параметрами, что упрощает работу пользователя.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. A revolutionary MPDP. Infinitely Expandeble MPDP. User's Manual Mis-4220 / Mis-4220R / Mis-4230 / Mis-4230R //ORION PDP CO., LTD. www.oriondisplay.net.

2. A revolutionary MPDP. Infinitely Expandeble MPDP. User's Manual OPM-4250 / OPM-4250R //ORION PDP CO., LTD. www.oriondisplay.net.

Н.Н. ГРИНЧЕНКО, М.А. КОВРИГИНА
Рязанский государственный радиотехнический университет

ПРОГРАММНЫЕ СРЕДСТВА МОДЕЛИРОВАНИЯ МНОГОПОРОГОВЫХ ДЕКОДЕРОВ И ДРУГИХ АЛГОРИТМОВ КОРРЕКЦИИ ОШИБОК

В статье описываются программные средства моделирования многопороговых декодеров и построенных на их основе каскадных схем коррекции ошибок.

Введение

При проектировании современных систем телекоммуникаций одной из важнейших является задача обеспечения высокой достоверности передачи данных. К наиболее эффективным методам решения данной задачи следует отнести применение корректирующих кодов [1], в разработке которых теория помехоустойчивого кодирования в последние десятилетия достигла значительных успехов.

На сегодняшний день одними из лучших методов кодирования/декодирования по соотношению эффективности и сложности практической реализации являются многопороговые декодеры (МПД) [2], используемые для декодирования самоортогональных кодов (СОК). Однако в процессе исследования МПД были выявлены особенности МПД и используемых самоортогональных кодов, затрудняющие получение очень малых вероятностей ошибки вблизи пропускной способности канала.

Одним из вариантов решения данной проблемы является разработка каскадных схем кодирования, которые состоят из короткого сверточного кода [3], декодируемого с помощью оптимального декодера Витерби [4], и СОК, декодируемого с помощью МПД. При этом СОК являются внешним кодом, а короткий сверточный код – внутренним. Применение подобной схемы позволяет уменьшить вероятность ошибки декодирования и приблизить область эффективной работы к пропускной способности канала, поскольку даже в области больших шумов оптимальный декодер Витерби несколько уменьшает вероятность ошибки в канале передачи данных, в результате чего работающий после него МПД может обеспечить близкое к оптимальному декодирование используемого в нем СОК и, таким образом, существенно снижает вероятность ошибки декодирования. Для более эффективного исследования каскадных схем кодирования и самого МПД были разработаны программные средства, позволяющие моделировать их работу.

Структура и алгоритм работы программных средств

Программные средства моделирования МПД состоят из интерфейсного модуля, взаимодействующего с модулем имитации канала передачи данных, модулем имитации работы устройств кодирования и декодирования, модулем управления параметрами эксперимента, модулем отображения результатов эксперимента и модулем построения графиков. Кратко рассмотрим назначение и состав перечисленных модулей.

Модуль имитации канала передачи данных позволяет выбирать требуемую модель канала передачи данных, выполнять настройку ее параметров и использовать выбранную модель канала при моделировании. Модуль имитации работы устройств кодирования и декодирования позволяет выбирать требуемые алгоритмы для кодирования и декодирования данных, выполнять настройку параметров данных ал-

горитмов и использовать выбранные модели устройств кодирования и декодирования при моделировании. Модуль управления параметрами эксперимента позволяет настраивать параметры эксперимента и управлять ходом эксперимента. С помощью данного модуля обеспечивается требуемая точность результатов моделирования. Модуль отображения результатов эксперимента позволяет в удобной форме выводить результаты эксперимента для их дальнейшего анализа.

Опишем алгоритм работы программных средств на примере моделирования каскадной схемы, состоящей из МПД и кодека Витерби, при передаче данных по каналу с аддитивным белым гауссовским шумом (АБГШ) при использовании двоичной фазовой модуляции.

В процессе передачи данных по каналу с АБГШ и двоичной фазовой модуляцией, имитируемой программными средствами, от кодера поступают данные для передачи, которые представляют собой набор из 0 и 1 некоторой длины. Затем модулятор осуществляет отображение этого набора в передаваемый сигнал s, который и передается по каналу передачи данных. В канале передачи данных сигнал s искажается аддитивным белым гауссовским шумом n, в результате воздействия которого принимается сигнал $r=s+n$. Далее демодулятор на основе принятого сигнала r формирует решения относительно переданных данных b. Данные решения могут быть либо жесткими, либо мягкими.

При моделировании канала передачи данных, для настройки доступны следующие параметры:

– отношение сигнал/шум E_s/N_0;

– количество уровней квантования решения демодулятора;

– границы областей решения мягкого модема.

При моделировании каскадной схемы кодирования входная последовательность сначала кодируется блоковым самоортагональным кодом, затем внутренним коротким сверточным кодом. При функционировании кодер сверточного кода, заданный образующими полиномами, переходит из одного состояния в другое, определяемое содержимым его регистра сдвига, причем из каждого состояния кодер может перейти только в два других состояния. При моделировании программными средствами кодека сверточного кода необходимо определить следующие параметры:

– конструктивная длина кода;

– первый выходной полином;

– второй выходной полином.

При декодировании принятая из канала последовательность сначала декодируется декодером Витерби, который позволяет оптимально декодировать сверточные коды, а затем многопороговым декодером.

Основная идея алгоритма Витерби, используемого для декодирования сверточного кода, состоит в пошаговом сравнении всех путей по кодовой решетке с принятой из канала последовательностью и отбрасывании тех из них, которые точно будут находиться на большем расстоянии, чем другие пути. Для настройки работы декодера Витерби доступны следующие параметры:

– максимальная длина выживших путей по кодовой решетке;

– шаблоны выкалывания для первой и второй выходных ветвей.

В процессе моделирования работы каскадной схемы описываемыми программными средствами в пользовательском интерфейсе возможна настройка параметров блокового самоортогонального кода и многопорогового декодера. При задании параметров кодера необходимо определить количество информационных и проверочных ветвей, а также задать компоненты образующего полинома.

При задании параметров декодера устанавливается количество итераций декодирования, а также значения порогов, используемых на пороговых элементах этих итераций. Кроме этого определяется способ вычисления весов элементов синдромного регистра.

Определить объем эксперимента, вид передаваемой информационной последовательности, а также максимальное количество ошибок на выходе декодера, после которого следует прекратить эксперимент, предоставляется возможным в модуле управления параметрами эксперимента.

После настройки всех вышеописанных параметров можно переходить к проведению эксперимента, результаты которого отображаются на экран и содержат следующую информацию:

1) общее число переданных по каналу бит;

2) число ошибочно переданных бит;

3) вероятность ошибки в канале передачи данных;

4) число декодированных информационных бит;

5) число ошибочно декодированных бит;

6) вероятность ошибки декодирования;

7) общее количество переданных блоков;

8) время проведения эксперимента.

Для более удобного сравнения эффективности различных способов повышения достоверности в программе предусмотрена возможность построения графиков зависимостей вероятности битовой ошибки в канале и на выходе декодера от отношения сигнал/шум.

Заключение

Описанные программные средства необходимы для специалистов, занимающихся разработками цифровых сетей передачи данных.

Программные средства позволят им оценить возможность применения в разрабатываемых ими системах различных декодеров корректирующих кодов. Это создает возможность правильного проектирования всех узлов создаваемых новых коммуникационных систем с учетом требуемых уровней энергетической эффективности, сложности, скорости и надежности реализации, задержки принятия решения и других критериев выбора систем повышения достоверности.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Питерсон У., Уэлдон Э. Коды, исправляющие ошибки / Пер. с англ.; под ред. Р.П. Добрушина и С.И. Самойленко. – М.: Мир, 1976. – 594 с.
2. Золотарев В.В., Овечкин Г.В. Помехоустойчивое кодирование. Методы и алгоритмы. Справочник. М.: Горячая линия – Телеком, 2004. 126 с.
3. Нейфах А.Э. Сверточные коды для передачи дискретной информации. – М.: Наука, 1979. – 222 с.
4. Витерби А. Границы ошибок для сверточных кодов и асимптотически оптимальный алгоритм декодирования // Некоторые вопросы теории кодирования. М.: Мир, 1970. С. 142–165.

Н.Н. ГРИНЧЕНКО, Г.В. ОВЕЧКИН, В.Ю. ШАРОВАТОВ
Рязанский государственный радиотехнический университет

ПРОГРАММНЫЕ СРЕДСТВА МОДЕЛИРОВАНИЯ СИСТЕМ ПЕРЕДАЧИ ДАННЫХ

Рассматриваются программные средства моделирования систем передачи данных, которые могут быть использованы специалистами, занимающимися разработкой и исследованием цифровых сетей передачи данных. Описываются основные моменты разработки данных программных средств.

Исследование помехоустойчивости систем передачи данных существенно усложняется почти полным отсутствием доступных программных продуктов, позволяющих проводить их комплексный анализ. Существующее ПО (Matlab и др.) оказывается очень дорогим или сложным для освоения. Это вынуждает разработчиков аппаратуры передачи данных создавать собственные программные средства моделирования подобных систем. Очевидно, что данный подход обладает рядом существенных недостатков, главный из которых заключается в значительном времени разработки моделирующей программы. Поэтому чрезвычайно актуальной видится задача разработки общедоступных программных средств моделирования систем передачи данных.

Разрабатываемые программные средства необходимы специалистам, занимающимся разработками цифровых сетей передачи данных, и позволят оценить возможность применения в разрабатываемых ими системах различных декодеров корректирующих кодов. Это создает возможность правильного проектирования всех узлов создаваемых новых коммуникационных систем с учетом требуемых уровней энергетической эффективности, сложности, скорости и надежности реализации, задержки принятия решения и других критериев выбора систем повышения достоверности.

Перечислим основные требования, предъявляемые к проектируемым программным средствам:

- проведение моделирования системы передачи данных, включающей источник данных, кодер, модулятор, канал связи, демодулятор, декодер и приемник данных;
- возможность настройки параметров компонентов системы;
- возможность динамического подключения компонентов системы передачи данных;
- получение статистики моделирования;
- представление результатов моделирования в удобном для дальнейшего анализа текстовом и графическом виде.

Выделим некоторые особенности разработки. В качестве языка программирования был выбран C#, который наиболее полно реализует идеи объектно-ориентированного программирования из всех языков, поддерживаемых платформой .NET. Данный язык имеет ряд отличительных особенностей и преимуществ, использование которых значительно упрощает реализацию многих решений, которые имеют место в данном проекте, по сравнению с другими языками программирования.

Представленную разработку отличает возможность динамического подключения компонентов системы передачи данных, что позволит со временем увеличивать число разнообразных эффективных методов кодирования, доступных для исследования, без изменения уже разработанных средств. Поэтому в процессе разработки решался вопрос выбора средств, с помощью которых динамически к основной программе будут подключаться компоненты системы передачи данных, реализованные в виде библиотек классов (*.dll). В рамках данного вопроса рассматривались COM-технология и .NET Reflection. Выбор был сделан в пользу второго варианта, поскольку он является более удобным и быстрым в реализации, является более перспективным и наиболее просто позволит подключать к программе новые ком-

поненты (реализованные на любом .NET языке) без вмешательства в уже разработанные программные средства.

На основе анализа требований, определенных выше, была разработана структура программных средств, представленная на рис. 1.

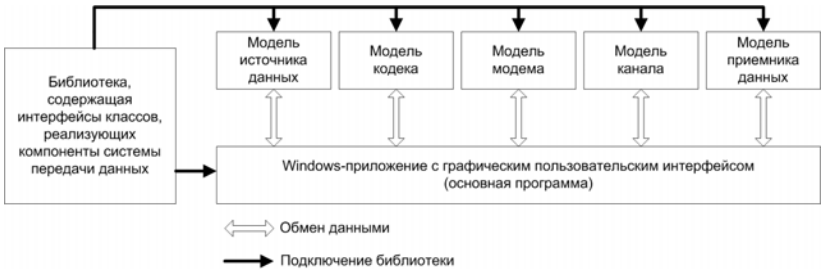

Рис. 1. Структура программных средств

Все модели компонентов системы передачи данных, подключаемые к основной программе, должны быть реализованы в виде библиотек классов (*.dll). В каждой из этих библиотек должен содержаться класс определенного компонента, наследованный от соответствующего интерфейса. Таким образом, в каждом подобном классе будет реализован набор определенных свойственных данному компоненту функций, которые обеспечат взаимодействие с основной программой.

Интерфейсы, от которых будут наследованы классы компонентов, должны содержаться в отдельной библиотеке (ChannelElements.dll). Помимо интерфейсов в этой библиотеке будут содержаться используемые в моделях компонентов вспомогательные классы:

- класс генератора нормально распределенных случайных чисел;
- класс комплексного числа.

Кроме моделей компонентов системы передачи данных на данную библиотеку должна ссылаться и основная программа для обеспечения взаимодействия последней с библиотеками компонентов.

При запуске основной программы должна происходить проверка на наличие библиотек компонентов в папках с определенными названиями (Channels, Data sources, Data receivers, Codecs, Modems), находящихся в одном каталоге с исполняемым файлом программы. Далее во время работы программы при выборе пользователем определенного компонента из списка имеющихся будет происходить загрузка этого компонента из библиотеки, после чего появится возможность использования его в качестве элемента моделируемой системы передачи данных.

После разработки структуры программных средств были определены списки функций (методов), которыми должны обладать модели элементов системы передачи данных. Данные группы методов было решено выделить в интерфейсы (по одному интерфейсу для каждого из классов элементов системы) и объединить в библиотеке ChannelsElements.dll. Таким образом, были выявлены следующие 5 интерфейсов:

- IChannel – интерфейс, который определяет функции, свойственные для модели физического канала;
- IDataSource – интерфейс, который определяет функции, свойственные для модели источника данных;
- IDataReceiver – интерфейс, который определяет функции, свойственные для модели приемника данных;
- ICodec – интерфейс, который определяет функции, свойственные для модели кодека (кодера/декодера);
- IModem – интерфейс, который определяет функции, свойственные для модели модема (модулятора/демодулятора).

Класс разрабатываемого компонента системы должен быть наследован от одного из этих интерфейсов (в зависимости от того, каким элементом системы передачи данных будет являться компонент), и, соответственно, содержать реализацию всех методов, определенных этим интерфейсом.

Перед началом моделирования необходимо выбрать компоненты, которые будут составлять исследуемую систему передачи данных, а также задать их параметры. Основным элементом процедуры моделирования является цикл, в рамках которого происходит взаимодействие с каждым из выбранных компонентом системы. Условия окончания эксперимента определяются путем задания соответствующих параметров у приемника, по инициативе которого и происходит окончание моделирования.

В рамках разработки программных средств моделирования было решено разработать следующие компоненты системы передачи данных:

- Источник данных, порождающий блок битов случайного содержания, состоящий из нулей и единиц и имеющий заданную длину.
- Кодек расширенного кода Хэмминга с возможностью выбора типа кода и использования мягких решений при декодировании.

- Кодек сверточного кода с декодером Витерби с возможностью определения порождающих полиномов и длины блока [1].
- Модем с двоичной фазовой модуляцией.
- Канал с аддитивным белым гауссовским шумом (АБГШ).
- Приемник данных, позволяющий задать в качестве своих параметров:
  1) максимальное количество передаваемых битов;
  2) максимальное количество передаваемых блоков;
  3) максимальное количество ошибочных битов;
  4) максимальное количество ошибочных блоков.

  Каждая из этих величин используется при проверке условий окончания эксперимента.

Набор этих компонентов может быть легко расширен без изменения уже разработанных программных средств.

Результатом моделирования являются статистика эксперимента (время и скорость эксперимента), вероятность ошибки в бите, вероятность ошибки в блоке, количество ошибочных битов и т.д. Также результаты могут быть представлены в виде графика зависимости вероятности ошибки в бите от отношения сигнал-шум. В качестве примера таких графических результатов приведем полученные с помощью разработанных программных средств графики для кодов Хэмминга и алгоритма декодирования Витерби, изображенные на рис. 2.

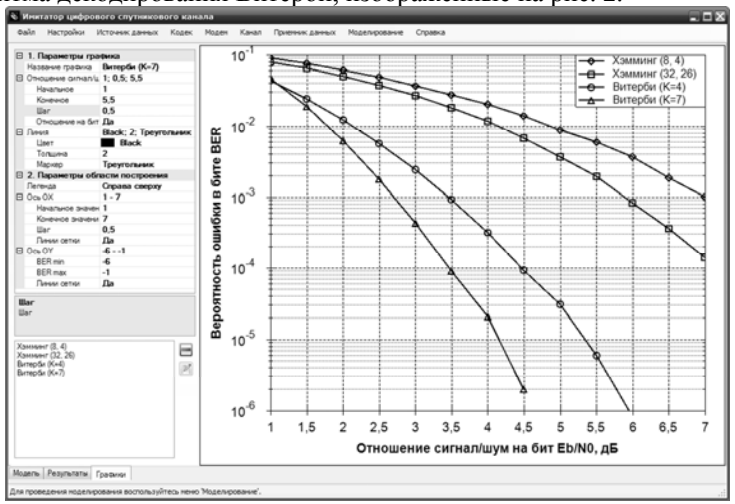

Рис. 2. Результаты моделирования кода Хэмминга и сверточного кода с декодированием Витерби в канале с АБГШ

Моделирование системы передачи данных с уже разработанными моделями кодеков вряд ли может быть полезно в реальных практических задачах, поскольку реализованные кодеки обладают слабой корректирующей способностью по сравнению с другими методами кодирования/декодирования. Тем не менее, программные средства с этими моделями кодеков могут быть полезны в процессе обучения.

В продолжение разработки программных средств планируется реализация кодека кодов с малой плотностью проверок на четность (LDPC-кодов), которые рекомендованы для исправления ошибок во многих стандартах передачи различного рода данных (в частности в стандарте DVB-S2). Также в перспективах и разработка других моделей кодеков, моделирование и исследование эффективности которых в сочетании с различными моделями каналов передачи может оказаться чрезвычайно полезным для разработчиков аппаратуры передачи данных и других специалистов, трудящихся в данной сфере.

Работа выполнена при финансовой поддержке РФФИ (грант 08-07-00078).

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Золотарев В.В., Овечкин Г.В. Помехоустойчивое кодирование. Методы и алгоритмы. Справочник. – М.: Горячая линия–Телеком, 2004 – 126 с.

Н.Н. ГРИНЧЕНКО, Т.Н. СОЛОВЬЕВА
Рязанский государственный радиотехнический университет

ЭФФЕКТИВНОСТЬ МНОГОПОРОГОВЫХ ДЕКОДЕРОВ В ДВОИЧНОМ СИММЕТРИЧНОМ КАНАЛЕ

Рассматриваются результаты моделирования работы многопорогового декодера в двоичном симметричном канале. Представлены характеристики, полученные для самоортогональных кодов с различной кодовой скоростью и кодовым расстоянием.

Введение. Передача информации по каналам связи существенно усложняется из-за помех и искажений в канале. Эффективным средством повышения достоверности передаваемой информации является помехоустойчивое кодирование.

В настоящее время широко используются такие системы кодирования, как коды Рида-Соломона, сверточные коды с декодированием по алгоритму Витерби, турбокоды и. др. [1..4]. Однако, для реализации данных алгоритмов требуются значительные вычисленные затраты,

большое количество операций, что приводит к увеличению времени кодирования и декодирования. Высокая сложность устройств кодирования и декодирования в этих системах затрудняет реализацию алгоритмов в реальном масштабе времени и существенно ограничивает скорость передачи информации.

На этом фоне метод коррекции ошибок, называемый многопороговым декодированием (МПД) [3,4], имеет ряд существенных преимуществ:

– способность исправлять большое число ошибок за пределами гарантированной корректирующей способности;

– крайне незначительная сложность процедуры декодирования;

– свойство разработанных МПД алгоритмов почти всегда достигать оптимальных решений при весьма высоких уровнях шума в канале связи;

– предельная легкость реализации МПД даже для очень длинных кодов, когда только и возможно достижение максимально допустимых значений эффективности кодирования.

Огромное преимущество МПД имеет перед всеми другими схемами декодирования по числу операций и возможность их полного распараллеливания при аппаратной реализации.

Эффективность МПД в ДСК. Рассмотрим вопросы эффективности многопороговых декодеров в двоичном симметричном канале передачи данных (ДСК). Опишем, каким образом проводилось моделирование процесса передачи данных по каналам такого типа.

При моделировании работы МПД в ДСК в соответствии с выражением

$$p_0 = Q\left(\sqrt{2RE_b / N_0}\right)$$

определялась вероятность ошибки p_0 в канале, соответствующая данному отношению сигнал/шум на бит E_b/N_0 (здесь отношение сигнал/шум является безразмерной величиной) при заданной кодовой скорости R. В процессе передачи по ДСК передаваемый кодовый бит b инвертировался с вероятностью p_0. Полученное значение поступало на вход МПД.

В процессе получения всех представленных далее результатов моделирования для обеспечения достаточной точности оценки вероятностей ошибки для каждой полученной точки графиков набиралась статистика до получения не менее чем 100 ошибочных блоков.

На рис. 1 представлены характеристики МПД в ДСК для различных блоковых самоортогональных кодов с кодовой скоростью R=1/2 и кодовым расстоянием d=9, характеризуемых различным уровнем размножения ошибок. На рисунке кривая «R=1/2» соответствует характеристикам МПД для кода с R=1/2 и n=1000, обладающего плохой устойчивостью к размножению ошибок.

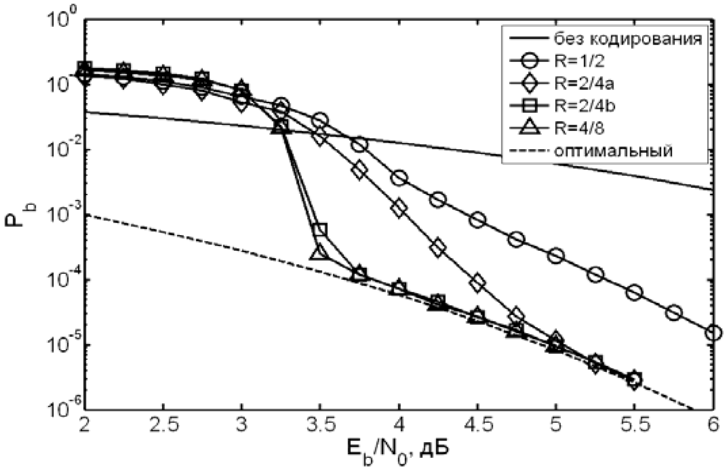

Рис. 1. Характеристики многопорогового декодера в ДСК для СОК с R=1/2, d=9 с различным размножением ошибок

Видно, что эффективность декодирования данного кода оказывается невысокой. Кривой «R=2/4a» отражены характеристики кода с переменными связями с R=2/4 и n=1200. В данном случае, начиная с отношения сигнал/шум 5.25 дБ, МПД выполняет близкое к оптимальному декодирование данного кода (вероятность ошибки оптимального декодера для СОК с R=1/2 и d=9 представлена на рисунке пунктиром). Эффективность МПД для более длинного СОК с R=2/4 и n=10400 представлена на рис. 1 кривой «R=2/4b». За счет использования более длинного кода, в большей степени устойчивого к размножению ошибок удалось приблизить границу области эффективной работы МПД, в которой он почти оптимально декодирует используемые с ним коды, к отношению сигнал/шум 3.75 дБ. Отметим, что переход к еще более длинным кодам, вероятность ошибки декодирования которых представлена кривой «R=4/8», приводит к незначительному улучшению характеристик.

Для примера на рис. 2 представлены характеристики МПД в ДСК для коротких блоковых самоортогональных кодов с кодовым расстоянием d=9 и длинных кодов с таким же кодовым расстоянием при различной кодовой скорости R.

Длинные коды выбраны в соответствии критерием минимизации эффекта размножения ошибок. При получении данных графиков использовалось от 5 до 15 итераций декодирования.

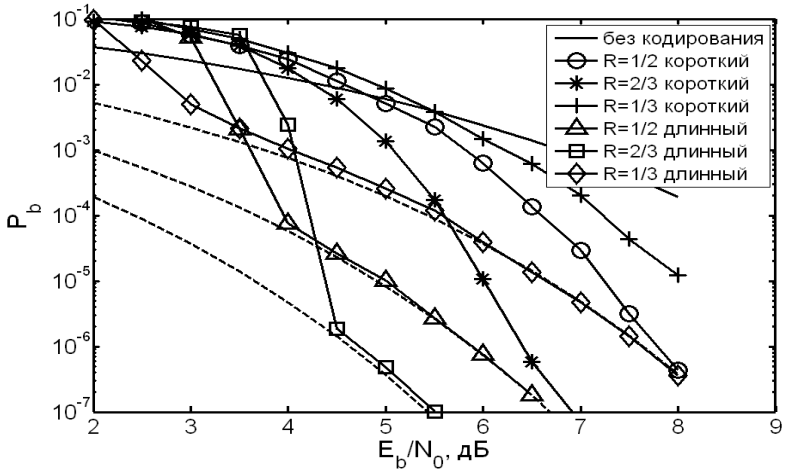

Рис. 2. Характеристики многопорогового декодера в ДСК для коротких и длинных СОК с d=9 и различной кодовой скоростью

Выводы. Как следует из представленных характеристик, применение МПД для декодирования плохо выбранных кодов с различными кодовыми скоростями не позволяет получить хороших характеристик, в то время как при использовании кодов с малым размножением ошибок обеспечивается почти оптимальное их декодирование, что позволяет увеличить ЭВК более чем на 2 дБ при P_b=10^{-5}. Также следует отметить, что характеристики МПД для кодов с меньшей кодовой скоростью R при одинаковом кодовом расстоянии d быстрее сходятся к оптимальным. Однако вероятность ошибки в области оптимального декодирования для кодов с меньшим R при одинаковом d оказывается выше. Это объясняется большими потерями в энергетике, требующейся для передачи большего числа проверочных символов.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Berrou C., Glavieux A., Thitimajshima P. “Near Shannon Limit Error-Correcting Coding and Decoding: Turbo-Codes,” // Proceeding of ICC’93, Geneva, Switzerland, May 1993. pp. 1064-1070.

2. Jin H., Khandekar A., McEliece R. “Irregular repeat-accumulate codes,” Proc. 2nd Int. Symp. on Turbo Codes and Related Topics, Brest, France, Sept. 2000. pp. 1–8.

3. Золотарев В.В., Овечкин Г.В. Помехоустойчивое кодирование. Методы и алгоритмы. Справочник. М.: Горячая линия – Телеком, 2004. 126 с.

4. Золотарёв В.В., Овечкин Г.В. Эффективные алгоритмы помехоустойчивого кодирования для цифровых систем связи // Электросвязь. 2003. № 9. С. 34–37.

Л.А. ДЕМИДОВА, Е.О. ИВАНОВА, Р.А. ТАГАНОВ
Рязанский государственный радиотехнический университет

ПРИМЕНЕНИЕ ГЕНЕТИЧЕСКИХ АЛГОРИТМОВ К АНАЛИЗУ РИСКОВ ПРОГРАММНЫХ ПРОЕКТОВ

Рассматривается подход к проблеме поиска «лучших» рисков, основанный на применении алгоритма кластеризации нечетких с-средних и генетического алгоритма.

Как показывает практика инженерии программных систем, современные программные проекты являются достаточно сложными объектами в техническом и организационном выполнении [5]. Для уменьшения рисков применяют обычно специальный процесс «Управление рисками», который в своем составе содержит ряд процессов: процесс планирования управления рисками; процесс идентификации рисков; процесс анализа рисков; процесс планирования реагирования на риски; процесс мониторинга и управления рисками.

Задачей процесса анализа рисков является выделение списка рисков, оказывающих наибольшее влияние на итоговые характеристики проекта, из большого количества рисковых событий, определенных на этапе идентификации рисков. Но на практике бывает сложно, а иногда невозможно определить зависимость различных процессов проекта от возможных рисков, действующих на проект, еще сложнее привести аналитическое описание такой зависимости, тем более на начальных этапах проекта. Эти обстоятельства значительно затрудняют применение на всех этапах проекта классических методов анализа рисков, так как большинство из них основывается на использовании априорной информации о реакции системы управления проектом на возникновение рисковых ситуаций в аналогичных, уже выполненных проектах.

В новых проектах действуют новые условия и не всегда методы, используемые в предыдущих проектах, позволяют адекватно выявлять и анализировать риски нового проекта и выявлять «лучшие» из них. В связи с этим встает задача построения методов анализа рисков, которые были бы способны выявлять риски проекта практически при полном отсутствии предположений о характере поведения процессов проекта на различных этапах.

Для решения задачи выбора «лучших» рисков программных проектов может быть применен кластерный анализ, задача которого заключается в нахождении некоторого теоретико-множественного разбиения исходного множества объектов на непересекающиеся подмножества таким образом, чтобы элементы, относимые к одному подмножеству, отличались между собой в значительно меньшей степени, чем элементы из разных подмножеств. Обладающие таким свойством подмножества объектов называются кластерами [2, 3].

Нахождение или выявление кластеров в исходном множестве объектов должно удовлетворять следующим требованиям [3]: каждый кластер должен представлять собой концептуально однородную категорию и содержать похожие объекты с близкими значениями характеристик; совокупность всех кластеров должна быть исчерпывающей, то есть охватывать все объекты исходного множества; кластеры должны быть взаимноисключающими, то есть ни один из объектов исходного множества не должен одновременно принадлежать двум различным кластерам.

Требование нахождения однозначной кластеризации для исходного множества объектов является достаточно грубым и жестким, особенно при решении плохо или слабо структурированных задач. Алгоритмы нечеткой кластеризации ослабляют это требование за счет введения в рассмотрение нечетких кластеров и соответствующих им функций принадлежности, принимающих значения из интервала [0, 1].

Алгоритм нечетких c-средних. В настоящее время наиболее известным алгоритмом нечеткой кластеризации, является алгоритм нечетких c-средних. Задача нечеткой кластеризации заключается в нахождении нечеткого разбиения или нечеткого покрытия исходного множества объектов, которые образуют структуру нечетких кластеров, присутствующих в анализируемых данных. Эта задача сводится к нахождению степеней принадлежности объектов искомым нечетким кластерам, определяющим в совокупности нечеткое разбиение или нечеткое покрытие исходного множества объектов [2, 3].

В общем виде задача нечеткой кластеризации имеет вид: найти нечеткое разбиение $R(X)=\{X_j \mid X_j \subseteq X\}$ множества объектов кластеризации X на заданное число c нечетких кластеров X_j ($j \in \{2,...,c\}$),обеспечивающее экстремум некоторой целевой функции $f(R(X))$ среди всех нечетких разбиений [2, 3].

Нечетким разбиением нечеткого множества X называется система нечетких подмножеств $R(X)=\{X_j \mid X_j \subseteq X\}$, если выполняются условия: $\bigcup\limits_j X_j = X$ ($X_j \in R$), $h_C < 1$, $C = X_l \cap X_m$, ($\forall X_l, X_m \in R$), где h_C – высота нечеткого множества C.

Пусть искомые нечеткие кластеры представляют собой нечеткие множества X_j, образующие нечеткое разбиение множества объектов кластеризации X.

Тогда условие $\bigcup\limits_j X_j = X$ ($X_j \in I$) принимает вид:

$$\sum_{j=1}^{c} u_j(x_i) = 1 \ (\forall x_i \in X), \tag{1}$$

где c – общее количество нечетких кластеров X_j, которое считается предварительно заданным ($c \in N$ и $c > 1$), $u_j(x_i)$ – функция принадлежности (ФП), определяющая нечеткую степень принадлежности объекта x_i кластеру X_j.

Алгоритм нечетких c-средних реализует минимизацию целевой функции вида [1, 3]:

$$J(U,V) = \sum_{j=1}^{c}\sum_{i=1}^{n}\left(u_j(x_i)\right)^m \cdot d_{ji}^2, \tag{2}$$

где $U=[u_j(x_i)]$ – нечеткое c-разбиение множества объектов X на основе ФП $u_j(x_i)$, определяющих степень принадлежности объекта x_i кластеру X_j; $V=(v_1,...,v_c)$ – центры кластеров; d_{ji} – расстояние между объектом x_i и центром кластера v_j; m – фаззификатор ($m \in R$, $m > 1$); c – количество кластеров X_j ($j \in \{2,...,c\}$); n – количество объектов кластеризации; $j = \overline{1,c}$; $i = \overline{1,n}$.

Пусть каждый из центров кластеров представляет собой вектор $v_j = (v_j^1, v_j^2, ..., v_j^q)$ в q-мерном нормированном пространстве R^q ($v_j^l \in R^q$). Обычно расстояние между объектом x_i ($i = \overline{1,n}$) и центром кластера v_j ($j = \overline{1,c}$) определяется как:

$$d_{ji} = \left(\sum_{l=1}^{q}\left(x_i^l - v_j^l\right)^2\right)^{\frac{1}{2}}, \tag{3}$$

где x_i^l – количественное значение по l-ой характеристике $p_l \in P$ для объекта $x_i = \left(x_i^1, x_i^2, ..., x_i^q\right) \in X$ (l-я координата i-го объекта); v_j^l – l-я координата центра j-го кластера; $i = \overline{1,n}$; $l = \overline{1,q}$.

В этом случае целевая функция (2) записывается как:

$$J(U,V) = \sum_{i=1}^{n}\sum_{j=1}^{c}\left(u_j(x_i)\right)^m \cdot \sum_{l=1}^{q}\left(x_i^l - v_j^l\right)^2 . \quad (4)$$

Функция принадлежности объекта x_i центру кластера v_j определяется через относительное расстояние между объектом x_i и всеми центрами кластеров v_t ($t = \overline{1,c}$) и может быть записана в виде [2, 3]:

$$u_j(x_i) = \frac{1}{\sum_{t=1}^{c}\left(\frac{d_{ji}}{d_{ti}}\right)^{\frac{2}{m-1}}} . \quad (5)$$

Координаты центров v_j искомых кластеров X_j ($j \in \{2,...,c\}$) по каждому l-ой характеристике рассчитываются по формуле:

$$v_j^l = \sum_{i=1}^{n}\left(u_j(x_i)\right)^m \cdot x_i^l \Big/ \sum_{i=1}^{n}\left(u_j(x_i)\right)^m , \ \forall j \in \{2,...,c\}, \ \forall p_l \in P . \quad (6)$$

В этом случае задача нечеткой кластеризации может быть сформулирована следующим образом. Для заданных множества объектов кластеризации X, количества кластеров c ($c \in N$ и $c > 1$) и фаззификатора m определить матрицу U значений ФП объектов кластеризации $x_i \in X$ кластерам X_j ($j \in \{2,...,c\}$), которые обеспечивают минимум целевой функции (4) и удовлетворяют ограничениям (1), (6)-(8):

$$\sum_{i=1}^{n} u_j(x_i) > 0 , \ (\forall j \in \{2,...,c\}); \quad (7)$$

$$u_j(x_i) \geq 0 , \ (\forall j \in \{2,...,c\}, \ \forall x_i \in X). \quad (8)$$

Алгоритм нечетких c-средних представляет собой итерационный алгоритм, последовательно улучшающий некоторое исходное нечеткое разбиение $R(X) = \{X_j \mid X_j \subseteq X\}$, формируемое обычно по какому-либо эвристическому правило. При этом на каждой итерации рекуррентно пересчитываются значения ФП объектов нечетким кластерам и их типичные представители – центры кластеров [2, 3].

Искомое решение о принадлежности некоторого объекта $x_i \in X$ ($i = \overline{1,n}$) кластеру j ($j = \overline{1,c}$) принимается в соответствии с правилом:

«Если ($u_j(x_i) > u_t(x_i)$) для $t = \overline{1,...,c}$ и $j \neq t$, (9)

То объект x_i относится к кластеру j».

Алгоритм нечетких c-средних определяет локально-оптимальное разбиение $R(X)$, описываемое совокупностью ФП $u_j(x_i)$ ($\forall x_i \in X$), и центры кластеров v_j^l ($\forall p_l \in P$) для $\forall j \in \{2,...,c\}$.

Для получения адекватных результатов нечеткой кластеризации необходимо многократное выполнение алгоритма нечетких c-средних при заданном количестве кластеров c для различных исходных разбиений $R(X) = \{X_j \mid X_j \subseteq X\}$ с целью принятия окончательного решения об искомой нечеткой кластеризации, определяемого по минимальному значению целевой функции для имеющихся разбиений.

Нечеткая кластеризация может быть выполнена для заданного (фиксированного) количества кластеров c или для произвольного количества кластеров c (например, $c_{min} \leq \tilde{n} \leq c_{max}$). В последнем случае задачей нечеткой кластеризации будет являться поиск оптимального количества кластеров c_{opt}, обеспечивающего наиболее адекватные результаты кластеризации, что определяется по минимальному значению целевой функции вида (4). Обычно в качестве показателя нечеткой кластеризации используется целевая функция алгоритма нечетких c-средних по формуле (4). Однако в настоящее время существуют подходы к оценке качества кластеризации, основанные на применении так называемых показателей качества кластеризации.

Одним из наиболее часто используемых и хорошо зарекомендовавших себя показателей качества кластеризации является индекс Се – Бени [6]:

$$XB = \frac{\sum_{j=1}^{c}\sum_{i=1}^{n}\left(u_j(x_i)\right)^2 \cdot \sum_{l=1}^{q}\left(x_i^l - v_j^l\right)^2}{n \cdot \min\limits_{t \neq j}\sum_{l=1}^{q}\left(v_t^l - v_j^l\right)^2}, \quad (10)$$

где c – количество кластеров; n – количество объектов; q – количество характеристик; $u_j(x_i)$ – ФП объекта x_i кластеру X_j; v_j^l – l-я координата центра j-го кластера; x_i^l – l-я координата i-го объекта; $i = \overline{1,n}$, $j = \overline{1,c}$; $t = \overline{1,c}$.

Индекс Се – Бени используется для оценки компактности и отделимости кластеров. Как показывает практическое применение ин-

декса Се – Бени, при хороших результатах нечеткой кластеризации $XB < 1$. При этом в качестве искомого количества кластеров $\tilde{n}$ необходимо выбирать то, для которого индекс XB принимает минимальное значение.

Генетический алгоритм. Использование генетического алгоритма (ГА) позволяет существенно сократить время поиска оптимального нечеткого разбиения, для которого значение функции соответствия будет экстремальным (минимальным или максимальным в зависимости от смыслового значения показателя качества кластеризации, используемого в качестве функции соответствия) [1, 5]. При этом использование ГА позволяет решить задачу поиска оптимальных результатов кластеризации с полиномиальной сложностью, в то время как применение классических методов поиска оптимальных результатов кластеризации, например, переборных, характеризуется экспоненциальной сложностью вычислений.

Для заданного количества кластеров $\tilde{n}$ хромосома может быть закодирована координатами центров всех кластеров или нечеткими степенями принадлежности объектов центрам кластеров – числами из интервала [0, 1]. Однако кодирование хромосом степенями принадлежности объектов центрам кластеров приводит к существенному увеличению длины хромосомы ввиду обычно большого количества объектов кластеризации n. При кодировании хромосомы координатами центрами кластеров длина хромосомы равна $\tilde{n} \cdot q$, где $\tilde{n}$ – количество кластеров, q – количество характеристик. При этом первые q элементов хромосомы соответствуют координатам центра первого кластера, вторые q элементов – координатам центра второго кластера и т.п. [1]. Диапазон изменения значения каждого гена (координаты центра кластера) определяется интервалом $[h_{min}, h_{max}]$, где h_{min} и h_{max} – минимальная и максимальная оценки используемой порядковой шкалы оценивания объектов.

Так как реализация алгоритма нечетких $\tilde{n}$-средних должна обеспечивать фактическое, а не формальное разбиение на заданное количество кластеров $\tilde{n} = c^*$, то при формировании начальной популяции хромосом и выполнении операций скрещивания и мутации хромосом хромосома, для которой нечеткая кластеризация выполняется на количество кластеров $\tilde{n}$ меньшее, чем c^* ($\tilde{n} < c^*$) должна быть признана «нежизнеспособной», а значение её функции соответствия необходимо положить равным некоторому максимальному числу Max, значение которого заведомо больше максимального значения функции соответствия. Это позволит исключить «нежизнеспособную» хромосому из популяции и повысить эффективность применения ГА [1].

Выбор родителей осуществляется на основе вероятностного отбора. Две выбранные таким образом хромосомы будут использоваться в качестве родителей при выполнении операции скрещивания. При этом при реализации ГА используется одноточечное скрещивание и одноточечная мутация. При выполнении операции скрещивания выбирается вероятность скрещивания R_c и генерируется случайное число N_c. Если $R_c > N_c$, то случайным образом выбирается точка скрещивания z ($z \in \{1,...,c \cdot q\}$) и выполняется скрещивание. При выполнении операции мутации выбирается вероятность мутации R_m и генерируется случайное число N_m. Если $R_m > N_m$, то случайным образом выбирается точка мутации z ($z \in \{1,...,c \cdot q\}$) и выполняется мутация.

Генетический алгоритм может быть записан в следующем виде.

1. Случайным образом создается популяция размером P в соответствии с формулой (1) для ФП объектов кластерам и формулой (6) для центров кластеров с проверкой условия «жизнеспособности» хромосом, то есть требования о разбиения на заданное количество кластеров: $\tilde{n} = c^*$.

2. При $g < G$ (G и g – максимальное и текущее количество поколений ГА соответственно) вычисляется значение функции соответствия (10) для каждой хромосомы и создается $R_c \cdot P/2$ пар хромосом-родителей.

3. Выполняются операции скрещивания и мутации для текущей популяции с проверкой условия «жизнеспособности» хромосом.

4. Создается новая популяция размером P, дополненная хромосомами-отпрысками в количестве $R_c \cdot P$, затем $R_c \cdot P$ хромосом с худшими значениями функции соответствия (10) отбрасываются. Если $g < G$, осуществляется переход к шагу 2. Если $g \geq G$, то работа ГА завершается и осуществляется переход к шагу 5.

5. Выбирается лучшая хромосома, которая минимизирует функцию соответствия (10). Для каждого объекта кластеризации определяются нечеткие степени принадлежности к нечетким кластерам с новыми центрами кластеров в соответствии с формулой (6).

Нечеткая кластеризация на основе ГА позволяет получить более адекватные результаты кластеризации с нахождением субминимума индекса Се – Бени по формуле (10), однако требует большого количества смен поколений G и большого размера популяции P, так как

качество кластеризации будет существенно зависеть от того, насколько хорошо выполнена инициализация центров кластеров.

Комбинированный метод нечеткой кластеризации. Комбинирование алгоритма нечетких c-средних и ГА позволяет значительно уменьшить количество поколений G, необходимое для получения адекватных результатов нечеткой кластеризации, и, следовательно, повысить эффективность применения ГА. Комбинированный метод нечеткой кластеризации предполагает поочередное выполнение алгоритма нечетких c-средних и ГА, реализуемое следующим образом [1].

1. Выполняется один шаг алгоритма нечетких c-средних при формировании хромосом начальной популяции размером P.

2. При $g < G$ (G и g – максимальное и текущее количество поколений ГА соответственно) выполняется один шаг ГА с реализацией операций скрещивания и мутации и вычислением значений функции соответствия по формуле (10) для новой популяции хромосом размером $(P + R_c \cdot P)$.

3. Для новой популяции размером $(P + R_c \cdot P)$, представленной хромосомами, закодированными координатами центров кластеров, выполняется один шаг алгоритма нечетких c- средних с вычислением новых значений ФП объектов центрам кластеров в соответствии с формулой (5), новых координат центров кластеров в соответствии с формулой (7). Затем реализуется вычисление новых значений ФП объектов центрам кластеров в соответствии с формулой (5) и вычисляются значения функции соответствия по формуле (10).

4. Из расширенной популяции размером $(2 \cdot P + R_c \cdot P)$, полученной путем объединения популяции размером P предыдущего шага и популяции размером $(P + R_c \cdot P)$ текущего шага, удаляются «нежизнеспособные» $(P + R_c \cdot P)$ хромосом с максимальными значениями функции соответствия по формуле (10). Если $g < G$, осуществляется переход к шагу 2. Если $g \geq G$, то работа ГА завершается и осуществляется переход к шагу 5.

5. Выбирается лучшая хромосома, минимизирующая функцию соответствия по формуле (10). Искомые координаты центров кластеров определяются на основе лучшей хромосомы. В качестве результирующих нечетких степеней принадлежности объектов центрам кластеров полагаются степени принадлежности объектов центрам кластеров, соответствующие лучшей хромосоме и уже вычисленные в ходе реализации комбинированного метода нечеткой кластеризации.

Для упорядочения кластеров от «лучшего» к «худшему» используется величина расстояния d_j от центра кластера $v_j = \left(v_j^1, v_j^2, ..., v_j^q\right)$ до «идеального объекта» с «идеальными» значениями оценок по всем q характеристикам. Например, если «идеальные» значения оценок описываются вектором (1, …, 1), то:

$$d_j = \sum_{l=1}^{q} \left(1 - v_j^l\right)^2 \quad (j = \overline{1,c}). \tag{11}$$

Таким образом, чем больше расстояние от «идеального объекта» до центра кластера, тем более «худшим» является кластер и попавшие в него объекты.

Применение комбинированого метода нечеткой кластеризации к задаче выбора «лучших» рисков программного проекта. Применение комбинированного метода нечеткой кластеризации к решения задачи выбора «лучших» рисков программного проекта может быть реализовано следующим образом. Для исходного множества рисков программного проекта, представленных кортежами вида $x_i = \left(x_i^1, x_i^2, ..., x_i^q\right) \in X$, где x_i^l – оценка степени влияния i-го риска на l-ю характеристику программного проекта; $i = \overline{1,n}$; $l = \overline{1,q}$, решается задача кластеризации рисков при $c \in \{2, ..., c_{max}\}$. При этом оптимальным количеством кластеров c_{opt} будет то, которое обеспечивает минмальное значение показателя качества кластеризации – индекса Се – Бени. Далее выполняется упорядочение кластеров (а именно, центров кластеров) в соответствии с формулой (1) по близости к «идеальному объекту». В качестве «лучших» лучших рисков программного проекта полагаются риски, попавшие к кластер, центр которого наиболее близок к «идеальному объекту».

Рис.1. Пример кластеризации

При необходимости может быть выполнен анализ и других кластеров, содержащих остальные риски, так как в соответствии с алгоритмом нечетких нечетких c-средних автоматически генерируются все c кластеров.

Следует отметить, что при реализации комбинированного метода нечеткой кластеризации все оценки степеней влияния рисков на характеристики программного проекта должны быть пронормированы таким образом, чтобы они находились в q-мерном кубе, что объясняется особенностями алгоритма нечетких нечетких c-средних.

Пример применения комбинированого метода нечеткой кластеризации. С использованием комбинированного метода нечеткой кластеризации был реализован поиск «лучших» рисков программного проекта для множества, содержащего 100 рисков, описываемых оценками степени влияния на характеристики: «длительность проекта», стоимость проекта» и «надежность проекта», при этом предполагалось, что чем ниже оценка степени влияния, тем лучше риск по данной характеристике. При этом оптимальное количество кластеров оказалось равным пяти. На рисунке маркерами ромбовидной формы показано множество «лучших» рисков.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Демидова Л.А., Кираковский В.В., Коняева Е.И. Классификация объектов жилого фонда на основе FCM-алгоритма и генетического алгоритма // Математическое и программное обеспечение вычислительных систем: межвуз. сб. науч. тр. / под ред. А.Н. Пылькина. – М.: Горячая линия – Телеком, 2008. – С. 21-32.

2. Демидова Л.А., Пылькин А.Н. Методы и алгоритмы принятия решений в задачах многокритериального анализа. – М.: Горячая линия – Телеком, 2007. – 232 с.: ил.

3. Леоненков А. Нечеткое моделирование в среде MATLAB и fuzzyTECH. – СПб.: БХВ-Петербург, 2003. – 736 с.: ил.

4. Таганов А.И. Методика анализа и сокращения рисков проектов сложных программных систем по характеристикам качествам // Вестник РГРТУ. № 1 (выпуск 31). – Рязань, 2010. – С. 77-82.

5. Ярушкина Н.Г. Основы теории нечетких и гибридных систем: Учеб. пособие. – М.: Финансы и статистика, 2004. – 320 с.: ил.

6. Xei X.L., Beni G.A. Validity Measure for Fuzzy Clustering // IEEE Trans. on Pattern Analysis and Machine Intellegence 3 (8). – 1991. – P. 841–846.

Н.Б. ЖЕРЕБЦОВ
Рязанский государственный радиотехнический университет

ПРОБЛЕМЫ МАРШРУТИЗАЦИИ В СЕТЯХ IP

Рассматриваются проблемы, возникающие при маршрутизации пакетных данных в современных сетях IP.

Трафик пакетных данных в последнее время стал для телекоммуникационных операторов заметным источником дохода, поэтому сети IP эксплуатируются все активнее. В погоне за прибылью операторы стараются выжать из сети максимум, что она может, а значит, разработка новых методов, которые позволяют наилучшим образом использовать ресурсы сетей IP, приобретают все большее значение.

Алгоритмы маршрутизации в сетях IP делятся на две группы: алгоритмы маршрутизации для продвижения пакетов в пределах одной автономной системы и алгоритмы маршрутизации для передачи информации между автономными системами. Указанные алгоритмы решают основные задачи маршрутизации. Однако остаются проблемы, которые еще требуют своего решения. Это объясняется необходимо-

стью наилучшего использования ресурсов сети с целью повышения показателей качества ее работы.

Эти протоколы используются в управляющем компоненте технологии MPLS – мультипротокольной коммутации по меткам. На основе маршрутной информации в каждом маршрутизаторе MPLS формируется таблица пересылок, и присваиваются метки для ускоренной передачи пакетов по технологии коммутации.

Протокол основой маршрутизации между автономными системами BGP-4 (протокол граничного шлюза) работает по алгоритму Беллмана-Форда иначе называемом алгоритмом дистанционно-векторной маршрутизации. Принцип вектора расстояний подразумевает выбор маршрута по кратчайшему расстоянию между системами, определяемого числом пересылок между маршрутизаторами.

При этом маршрутизатор изменяет информацию в своей таблице, если:

1) если в полученной таблице находится более короткий путь;

2) в его списке находится неизвестный получатель;

3) маршрутизация проходит через маршрутизатор, от которого пришла таблица, и расстояние от него до получателя изменилось. По данным таблицы, применяя алгоритм Беллмана-Форда, и рассчитывается значение стоимости, того или иного маршрута.

Рассмотрим наиболее популярный сегодня протокол маршрутизации в пределах одной автономной системы или как еще принято его называть протокол внутреннего шлюза IGP, а именно OSPF. В этом протоколе применен принцип оценивания состояния канала, при этом стоимость того, или иного канала оценивается с помощью различных метрик. Стоимость маршрута складывается из стоимости всех каналов, входящих в этот маршрут. Одной из самых популярных метрик является пропускная способность канала.

Протокол OSPF используют метрику, в которую входит всего несколько параметров: пропускная способность; величина задержек пакетов в очередях; надежность; стоимость передачи данных по каналу. Однако при этом не учитывается ряд очень важных для достижения оптимальных режимов работы сети параметров.

Например, величины окон неподтверждения правильности передачи пакетов, значения тайм-аутов, степень согласованности длин пакетов в разных автономных системах MTU (от которой зависит, будут ли выполняться достаточно длительные процедуры фрагментации и дефрагментации пакетов), особенности методов доступа на MAC-уровне, влияние соседних каналов в радиосетях и др.

Важным параметром, влияющим на время передачи пакетов, является величина окна неподтверждения пакета. Изменяя значение этого параметра, можно повысить эффективность передачи пакетов в зависимости от того, насколько надежен канал. Там где каналы надежные, например трансмиссия организована на оптике, можно сделать его величину небольшой, т.к. вероятность потери или искажения пакета незначительная. Там же, где вероятность потери или искажения пакетов высока, например в РРЛ-каналах, возможно даже увеличение этого окна, в связи с тем, что на работу данного вида трансмиссии влияет погода и другие внешние факторы.

Можно отметить главные проблемы, которые должны быть решены для повышения эффективности алгоритмов маршрутизации.

1) Для всех протоколов маршрутизации, остается неизменным правило локального предпочтения, заключающееся в том, что при окончательном выборе маршрута предпочтение отдается не пути с наилучшей метрикой, но более протяженному географически, а маршруту через ближайшие автономные системы. Это объясняется следующими факторами:

– необходимостью исключения из маршрута некоторых автономных систем по признаку допустимости транзитного трафика;

– необходимость корректировки маршрутов по соображениям обеспечения информационной безопасности. При этом могут быть исключены определенные автономные системы и запрещена передача на компьютеры с определенными IP-адресами;

– особенностями локальной технической политики сетевых администраторов автономных систем, которые формализуются в рамках нечеткой логики.

Этот процесс осуществляется установкой необходимых параметров на маршрутизаторе или запуском специального скрипта. Таким образом, вносится субъективный человеческий фактор в работу сети, который должен учитываться при совершенствовании процессов маршрутизации.

2) Еще одним из факторов, влияющим на степень перегрузки, является использование в TCP/IP правила длиннейшего подходящего маршрута. Оно заключается в том, что маршрутизатор, принимающий для каждой сети решение на основе префиксов различной длины, всегда будет выбирать маршрут с более длинной маской. В результате это вызывает перегрузки доменов в направлении, которых присутствуют более длинные маски.

3) Мы говорим термин «наилучший маршрут», подразумевая тем самым, что алгоритм маршрутизации находит самый короткий путь прохождения пакетного трафика. Практика показывает использование метрик не всегда приводит к нахождению оптимального маршрута.

Примером может служить «эффект рыбы», когда между двумя узлами маршрутизации есть два пути прохождения трафика с разным числом маршрутизаторов. Алгоритм маршрутизации находит кратчайший путь и пускает по нему весь трафик. На найденный кратчайший путь направляется весь трафик, и он становится перегруженным, хотя ненамного больший маршрут остается незагруженным.

4) Сегодня многие операторы совершенно оправдано разделяют трафик по его типу и назначают каждому типу трафика свои каналы. Это рационально с точки зрения обслуживания: для услуг в реальном времени нужны надежные каналы без задержек, а для услуг, которые не требовательны к задержкам, предоставляются менее надежные каналы с задержками. Каждый оператор стремится достигнуть стопроцентной утилизации своего оборудования, из этого следует, что каналы должны быть загружены до разумного предела. Но сегодня работу по перераспределению трафика по нужным каналам делает системный администратор и далеко не всегда, получается, загрузить каналы так, чтобы по ним шел однотипный трафик и они были загружены до определенного предела. В результате каналы получаются незагруженными или наоборот перегруженными. Дело осложняется тем, что группировка трафика осуществляется фактически по нечетким признакам, например, схожими требованиями по обеспечению безопасности, по близости конечных адресов и т.п.

5) Необходимо также помнить об информационной безопасности: существует информация, которая должна передаваться только по защищенным каналам, где возможность ее перехвата низка. Но существует и такая, для которой защищенность не является критически важным фактором. Это обстоятельство сейчас также учитывается только системными администраторами и решение о надежности того или иного канала принимается только по их субъективному видению. Этот довольно значимый фактор должен быть обязательно учтен при построении маршрута, как непосредственно влияющий на перераспределение трафика в точках маршрутизации. Часто он не учитывается, т.к. системные администраторы различных сетей не могут согласовать между собой, как лучше построить маршрут, а прописывают маршруты по собственному усмотрению.

На основании вышеизложенного можно выделить следующие актуальные задачи:

– нахождения метрик и условий, позволяющих отойти от традиционной логики в алгоритмах маршрутизации и использовать наряду с традиционными метриками метрики, основанные на использовании нечетких множеств;

– нахождения методов, позволяющие выполнять профилирование трафика на узлах маршрутизации и перераспределять его с использованием характерных для сетевых администраторов понятий, основанных на нечеткой логике;

– введения корректировок в алгоритмы маршрутизации для учета предпочтения по информационной безопасности;

Решение этих задач позволит обеспечить более высокие показатели качества работы IP-сети.

Ю.С. ЗАДОРОВА, В.А. ШИБАНОВ
Рязанский государственный радиотехнический университет

НАХОЖДЕНИЕ РАСПРЕДЕЛЕНИЯ АДРЕСОВ В СЕТИ, МИНИМИЗИРУЮЩЕГО ОБЩЕЕ ЧИСЛО ЗАПИСЕЙ В ТАБЛИЦАХ МАРШРУТИЗАЦИИ

Рассматривается возможность нахождения такого распределения сетевых адресов при заданной структуре сети, которое обеспечит минимальное количество записей в таблицах маршрутизации.

Одной из существенных проблем традиционной классовой маршрутизации является очень большое количество строк в таблицах маршрутизации, которые приходится хранить на маршрутизаторах (в некоторых случаях – тысячи и десятки тысяч). Это может приводить к увеличению времени поиска нужной строки в таблице маршрутизации, росту очередей пакетов в маршрутизаторах, и, в конечном счете, ведет к резкому снижению производительности всей компьютерной сети [1]. Технология бесклассовой маршрутизации Classless Interdomain Routing (CIDR) позволяет сократить число строк в таблице маршрутизации путем объединения сетей с одинаковыми старшими битами адреса сетей в блок, которому соответствует единственная строка в таблице маршрутизации. Например, строки для четырех сетей со смежными адресами 192.168.0.0/22, 192.168.1.0/22, 192.168.2.0/22 и 192.168.3.0/22 могут быть объединены в единую строку в таблице маршрутизации – 192.168.0.0/20. Такое объединение возможно только в случае совпадения направления следующего шага при передаче информации с текущего маршрутизатора. Таким образом, на возможность объединения строк в таблице маршрутизации и, соответственно, сокращения общего числа строк в таблице, влияет как структура сети и маршруты, считающиеся оптимальными, так и адреса, назначенные компьютерам и портам маршрутизаторов. В условиях корпоративной или городской

сети назначение тех или иных сетевых адресов выполняется администратором сети. Поэтому можно поставить задачу нахождения такого распределения сетевых адресов при заданной структуре сети, которое обеспечит минимальное количество записей в таблицах маршрутизации.

Сформулируем данную задачу в следующем виде. Структура сети задана в виде графа. Каждый узел графа соответствует маршрутизатору. Дуга графа соответствует каналу связи между маршрутизаторами. Каждому узлу графа приписывается вес n_i, равный числу компьютеров в подсети, подключенной к маршрутизатору. Если маршрутизатор является магистральным, то $n_i = 0$. На графе задана матрица предпочтительных путей из каждой сети в каждую сеть $\mathrm{M} = \lfloor \mu_{ij} \rfloor$. Также задан исходный диапазон сетевых адресов. Требуется найти такое распределение сетевых адресов между сетями компьютеров, подключенных к маршрутизаторам, которое обеспечит минимизацию значения общего числа записей в таблицах маршрутизации.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Олифер В.Г., Олифер Н.А. Компьютерные сети. Принципы, технологии, протоколы: Учебник для вузов. 3-е изд. – СПб.: Питер, 2005. – 960 с.

Г.В. ИГОНИНА, С.В. АВЕРЦЕВ
Московский институт экономики, менеджмента и права
Рязанский филиал

ВЫБОР ОПТИМАЛЬНОГО ЧИСЛА ОДНОТИПНЫХ УСТРОЙСТВ В СИСТЕМАХ КОЛЛЕКТИВНОГО ДОСТУПА

Определяется оптимальное число однотипных устройств в узлах разомкнутых структур и оценивается замена этих отдельных устройств многоканальными узлами.

При разработке и эксплуатации вычислительных систем, центров коллективного пользования, и, в частности, диалоговых систем и систем разделения времени часто встает задача комплектования их узлов типовыми устройствами, такими как процессоры, мультиплексоры, внешние устройства и т.д. [1]. Основной характеристикой систем коллективного доступа (СКД) является среднее время реакции, т.е. среднее время ответа.

Исследование таких систем нередко осуществляется с помощью

сетей массового обслуживания. В частности, на этапе системного проектирования таких объектов для их исследования часто используются разомкнутые линейные вероятностные сети.

При рассмотрении данного вопроса целесообразно проанализировать две постановки задачи [2].

В случае первой постановки задана интенсивность входного потока задач $\boldsymbol{\lambda_0}$, определяемая суммарным числом запросов пользователей. Заданы ограничения на стоимость $\boldsymbol{S \le S^*}$. Требуется выбрать такое число $\boldsymbol{K_i}$ однотипных устройств в каждом из узлов СКД, при котором среднее время ответа будет минимальным $\boldsymbol{U_0 = U_{0min}}$.

При второй постановке задачи минимизируемая функция и ограничения меняются местами. Задана интенсивность $\boldsymbol{\lambda_0}$ входного потока задач. Заданы ограничения на время ответа $\boldsymbol{U_0 \le U_0^*}$. Требуется определить такие значения $\boldsymbol{K_i}$, чтобы стоимость была минимальной.

Рассмотрим первую постановку задачи.

На основе параметров задач, для решения которых проектируется СКД, определяются потребности в ресурсах технических устройств и структура системы. Исходя из этой информации, определяются трудоемкости этапов вычислительного процесса $\boldsymbol{\theta_i}$, связанные с ***i***-м узлом. Построив сетевую модель, определяем вероятности передач $\boldsymbol{P_{ij}}$, интенсивности входных потоков в каждый из узлов $\boldsymbol{\lambda_i}$ и коэффициенты передачи $\boldsymbol{\alpha_i}$ [2]. Полагаем, что система состоит из ***N*** узлов. Каждый узел может состоять из $\boldsymbol{K_i}$ однотипных одноканальных устройств (***ij*** *= 1, 2,…,* ***N***). Так как устройства типовые, то известны стоимость $\boldsymbol{S_i}$ каждого из них и быстродействие $\boldsymbol{V_i}$. Тогда стоимость устройств в ***i***-ой группе равна $\boldsymbol{K_i S_i}$, а стоимость всей системы равна $S = \sum_{i=1}^{N} K_i S_i$.

Следовательно, ограничения принимают вид

$$\sum_{i=}^{N} K_i S_i \le S_i^* \qquad (1)$$

Каждому устройству в ***i***-м узле СКД ставится в соответствие одноканальная система массового обслуживания (СМО) в модели. Тогда время ответа для сетевой модели определяется выражением $U_0 = \sum_{i=1}^{N} \frac{\alpha_i}{\mu_i - \lambda_i}$, где $\mu_i = \frac{1}{\tau_i}$ интенсивность обслуживания, а τ_i – среднее время обслуживания [2]. Параметры модели μ_i определяются через параметры соответствующих устройств и равны $\mu_i = V_i / \theta_i$.

Поскольку значения V_i и θ_i фиксированы, то может оказаться, что $\lambda_i \geq \mu_i$. Тогда в i-ом узле необходимо использовать больше одного устройства. В этом случае полагаем, что входной поток заявок между устройствами i-го узла распределяется равномерно (при таком распределении время ответа минимально). С учетом этого

$$U_0 = \sum_{i=1}^{N} \frac{\alpha_i}{\mu_i - \lambda_i / K_i} = \sum_{i=1}^{N} \frac{K_i \alpha_i \tau_i}{K_i - \lambda_i \tau_i}.$$

Для определения оптимальных значений K_i, минимизирующих функцию U_0, составляем функцию Лагранжа

$$\Phi(\alpha_i; K_i, ..., K_N) = \sum_{i=1}^{N} \frac{K_i \alpha_i \tau_i}{K_i - \lambda_i \tau_i} + \delta \left(\sum_{i=1}^{N} K_i S_i - S^* \right),$$

где δ - неопределенный множитель Лагранжа.

Дифференцируя эту функцию по переменным K_i, и приравнивая производные к нулю, получаем N уравнений, а с учетом ограничений (1), $N + 1$ уравнение.

$$\begin{cases} \delta S_i + \dfrac{\alpha_i \lambda_i \tau_i^2}{(K_i - \lambda_i \tau_i)^2} = 0, \\ \sum\limits_{i=1}^{N} K_i S_i - S^* = 0. \end{cases}$$

Решая эту систему, получаем

$$K_{ionm} = \lambda_i \tau_i + \frac{S^* - \sum\limits_{i=1}^{N} \lambda_i \tau_i S_i}{\sqrt{S_i}} \cdot \frac{\lambda_i \tau_i}{\sum\limits_{i=1}^{N} \lambda_i \tau_i \sqrt{S_i}} \qquad (2)$$

Первый член $\lambda_i \tau_i = \lambda_i / \mu_i$ определяет минимальное число устройств в i-ом узле, которое может обеспечить обработку λ_i задач, поступающих на вход этого узла. Если выбрать число устройств в i-ом узле, равное $\lambda_i \tau_i$, то коэффициент загрузки будет равен единице, и, следовательно, не будет существовать установившегося режима. Значит, значение K_i должно быть больше $\lambda_i \tau_i$.

Величина $\lambda_i \tau_i S_i$ представляет собой минимально необходимую стоимость устройств i-го узла. Тогда $\sum\limits_{i=1}^{N} \lambda_i \tau_i S_i$ минимально необходимая стоимость всей системы. А величина $S^* - \sum\limits_{i=1}^{N} \lambda_i \tau_i S_i$ определяет стоимость, оставшуюся после того, как все узлы обеспечили минимально необходимым числом устройств. Обозначим ее через S_0. Второе слагаемое выражения (2) определяет, каким образом распределяется эта оставшаяся стоимость между узлами с тем, чтобы число устройств в каждом из них было бы больше значения $\lambda_i \tau_i$ и существовал установившийся режим, а время ответа было минимальным при заданной стоимости.

Значения K_{ionm}, определяемые выражением (2) могут оказаться нецелочисленными. В этом случае их следует округлить до ближайшего целого, большего величины $\lambda_i \tau_i$. При этом, значения K_{ionm}, меньшие единицы, округляются до величины ≥ 1.

Найдем минимальное время ответа U_{0min}. Для этого подставим значения K_{ionm} в выражение для определения U_0 и преобразуем его

$$U_{0\min} = \sum_{i=1}^{N} \frac{\alpha_i K_i \tau_i}{K_i - \lambda_i \tau_i} = \frac{\alpha_i \lambda_i \tau_i^2 + \dfrac{S_0}{\sqrt{S_i}} \cdot \dfrac{\alpha_i \lambda_i \tau_i^2}{\sum\limits_{i=1}^{N} \lambda_i \tau_i \sqrt{S_i}}}{\dfrac{S_i}{\sqrt{S_i}} \cdot \dfrac{\lambda_i \tau_i}{\sum\limits_{i=1}^{N} \lambda_i \tau_i \sqrt{S_i}}}.$$

В результате получаем

$$U_{0\min} = \sum_{i=1}^{N} \alpha_i \tau_i + \frac{\lambda_0}{S_0} \left(\sum_{i=1}^{N} \alpha_i \tau_i \sqrt{S_i} \right)^2 \qquad (3)$$

Первый член в выражении (3) определяет среднее время обработки одной заявки в узлах сети за всё время ее пребывания в этой сети. Но время пребывания (время ответа) суммируется из времени обработки и времени ожидания. Следовательно, второй член в выражении (3) определяет среднее время ожидания заявки в узлах сети. Очевидно, что чем больше заданная стоимость, а, следовательно, чем больше оставшаяся стоимость S_0, тем меньше будет время ожидания и тем меньше отличается U_{0min} от величины $\sum\limits_{i=1}^{N} \alpha_i \tau_i$.

Как уже отмечалось при второй постановке задачи заданы λ_0 и

ограничения на время ответа

$$U_0 = \sum_{i=1}^{N} \frac{K_i \alpha_i \tau_i}{K_i - \lambda_i \tau_i} \le U^* \qquad (4)$$

Для определения $\boldsymbol{K_{ionm}}$ минимизируется стоимость $S = \sum_{i=1}^{N} K_i S_i$.

Для решения задачи в этой постановке тоже составляем функцию Лагранжа

$$\Phi(\alpha_i; K_i, ..., K_N) = \sum_{i=1}^{N} K_i S_i + \delta \left(\frac{K_i \alpha_i \tau_i}{K_i - \lambda_i \tau_i} - U^* \right).$$

Находим производные $\partial \Phi / \partial K_i = 0$, решая полученные уравнения совместно с ограничениями (4), получаем

$$K_{ionm} = \lambda_i \tau_i + \frac{\tau_i \sqrt{\alpha_i \lambda_i S_i}}{\left(U^* - \sum_{i=1}^{N} \alpha_i \tau_i \right) S_i} \cdot \sum_{i=1}^{N} \tau_i \sqrt{\alpha_i \lambda_i S_i} \qquad (5)$$

Как уже отмечалось, величина $\sum_{i=1}^{N} \alpha_i \tau_i$ определяет среднее время обработки заявки в узлах за всё время пребывания заявки в сети. Ограничения на время ответа должны быть больше этой величины, так как время ответа может быть равным этой величине, только в случае если в сети обрабатывается только одна заявка, или, если число устройств в каждом из узлов бесконечно велико, т.е. если нет очередей, и, следовательно, время ожидания равно нулю. Но в реальных системах очереди есть.

Второе слагаемое в выражении(5) определяет величину добавки к минимально необходимому числу устройств $\boldsymbol{\lambda_i \tau_i}$ в ***i***-ом узле с тем, чтобы существовал установившийся режим, и выполнялись ограничения на время ответа. Из выражения (5) видно, что чем жестче ограничения, т.е. чем меньше разность $\left(U^* - \sum_{i=1}^{N} \alpha_i \tau_i \right)$, тем больше должна быть эта добавка.

Значения $\boldsymbol{K_{ionm}}$ могут оказаться нецелочисленными. В этом случае их следует определить аналогично округлению при первой постановке задачи.

Найдем минимальную стоимость СКД

$$S_{\min} = \sum_{i=1}^{N} K_{ionm} S_i = \sum_{i=1}^{N} \lambda_i \tau_i S_i + \sum_{i=1}^{N} \frac{\tau_i \sqrt{\alpha_i \lambda_i S_i}}{\left(U^* - \sum_{i=1}^{N} \alpha_i \tau_i \right)} \cdot \sum_{i=1}^{N} \tau_i \sqrt{\alpha_i \lambda_i S_i} \, .$$

В результате получаем

$$S_{\min} = \sum_{i=1}^{N} \lambda_i \tau_i S_i + \frac{\left(\sum_{i=1}^{N} \lambda_i \tau_i \sqrt{S_i} \right)^2}{\lambda_0 \left(U^* - \sum_{i=1}^{N} \alpha_i \tau_i \right)}.$$

Первый член этого выражения определяет минимально необходимую стоимость всей системы. В реальных СКД к этой минимально необходимой стоимости добавляется минимально необходимая добавка, определяемая вторым членом. Эта добавка обеспечивает выполнение заданных ограничений на время ответа. Очевидно, что чем жестче ограничения на время ответа, т.е. чем меньше $\boldsymbol{U^*}$, тем больше должна быть эта добавка.

В рассмотренной задаче при обеих постановках определялось оптимальное число отдельных устройств $\boldsymbol{K_i}$ в каждом из узлов СКД. А функционирование каждого из этих устройств отображалось в модели функционированием одноканальной системы массового обслуживания (СМО). Хотя целесообразнее было бы определять в каждом из узлов СКД оптимальное число устройств, имеющих общую очередь. Тогда в соответствие каждому такому узлу надо ставить многоканальную СМО в модели. Однако из-за сложности выражений времени ответа $\boldsymbol{t_0}$ для таких СМО в аналитическом виде определить оптимальное число каналов не удается. Но реальная задача от этого не становится менее значимой.

Для решения рассматриваемой задачи предлагается следующий подход.

На первом этапе предлагается определить оптимальное число $\boldsymbol{K_{ionm}}$ отдельных типовых одноканальных устройств как это предлагалось выше. Затем определить время ответа $\boldsymbol{t_0}$ для каждого из узлов и в случае применения отдельных устройств и в случае многоканальных устройств при конкретных значениях $\boldsymbol{\lambda_i}$ и $\boldsymbol{\mu_i}$.

В случае применения отдельных устройств в каждом из узлов модель будет представлять собой $\boldsymbol{K_i}$ одноканальных СМО с интенсивностью входного потока заявок λ_i / K_i (рис. 1).

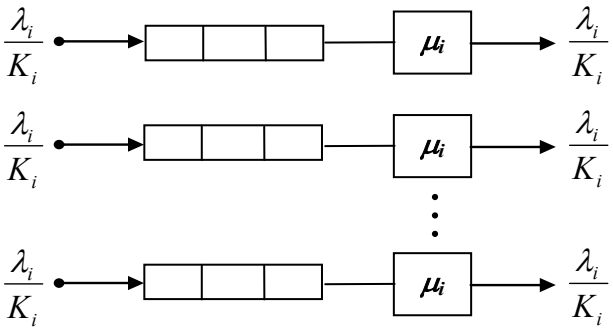

Рис. 1. Представление i-го узла одноканальными СМО

А в случае объединения этих же устройств общей очередью, модель будет иметь структуру многоканальной СМО с общей очередью с интенсивностью входного потока заявок λ_i (рис. 2).

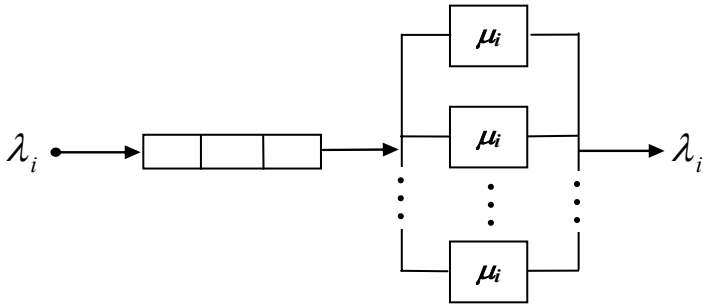

Рис. 2. Представление i-го узла многоканальной СМО

В первом случае время ответа определяется по выражению, уже используемому выше $t_{0i}(K_i) = \dfrac{1}{\mu_i - \lambda_i / K_i}$.

Во втором случае $t_{0i_{K_i}}$ определяется по известному выражению для многоканальных СМО с неограниченной длиной очереди.

Пусть, например, для i-го узла $\lambda_i = 20$ з/с, $\mu_i = 10$ з/с, тогда $\lambda_i / \mu_i = 2$ (Табл. 1). Значит K_i должно быть не меньше 3, т.е. $K_i \geq 3$. Значения $t_{0i}(K_i)$ и $t_{0i_{K_i}}$ в зависимости от числа устройств / каналов

сведены в таблицу и построены графики (рис. 3).

Табл. 1

Число устройств или каналов	$t_{0i}(K_i)$ (с)	$t_{0i_{K_i}}$ (с)
3	0,3	0,16
4	0,2	0,11
5	0,167	0,102
6	0,15	0,1005

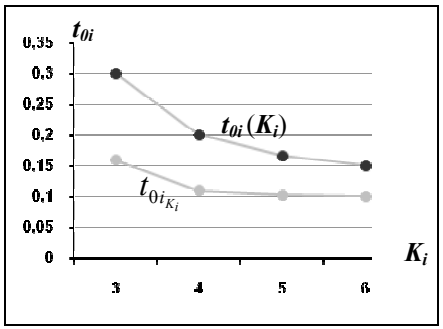

Рис. 3. Зависимости времени ответа от числа устройств или каналов

В смысле коэффициентов загрузки Z_i оба варианта эквивалентны ($Z_i = \lambda_i / \mu_i K_i$), а в смысле времени ответа t_0 (основной характеристики СКД), как видно из таблицы и графиков, они существенно отличаются. В случае применения многоканального варианта t_{0i} значительно меньше. Время обслуживания в обоих вариантах одинаково, так как одинаковы интенсивности обслуживания μ_i. Значит отличие во времени ожидания. Это не сложно объяснить физически. Если используются отдельные одноканальные устройства, то в каждое из них может поступить очередная заявка в тот момент, когда это устройство ещё занято обслуживанием предыдущей заявки. Значит, вновь поступившая заявка становится в очередь. В это же время какая-то часть других устройств может простаивать. В случае же многоканальности вновь пришедшая заявка становится в очередь только тогда, когда все каналы (устройства) заняты и, следовательно, время ожидания будет меньше. Особенно велико отличие во времени ответа при первоначально допустимом числе устройств / каналов. В данном примере это число равно трем. С увеличением числа устройств / каналов при постоянных значениях λ_i и μ_i это отличие уменьшается, так как уменьшается вероятность появления очереди и средняя длина очереди.

При увеличении соотношения $\lambda_i / \mu_i = \lambda_i \tau_i$, т.е. при увеличении минимального числа устройств / каналов указанное отличие растет. Это объясняется тем, что при этом растут коэффициенты загрузки и, следовательно, средняя длина очереди увеличивается.

Пусть $\lambda_i = 40$ з/с, $\mu_i = 10$ з/с, тогда $\lambda_i / \mu_i = 4$ (Табл. 2). Значит

первоначально допустимое число устройств / каналов равно 5, т.е. $K_i \geq 5$.

Табл. 2

Число устройств или каналов	$t_{0i}(K_i)$ (с)	$t_{0i_{K_i}}$ (с)
5	0,5	0,2
6	0,3	0,118
7	0,23	0,105
8	0,2	0,102

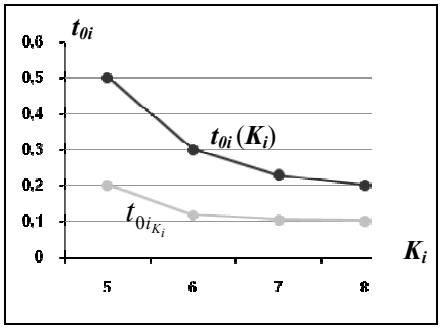

Рис. 4. Зависимости времени ответа от числа устройств или каналов

Из таблицы и графиков (рис. 4) видно, что отличие во времени ответа больше, чем в первом примере. При использовании 5 устройств / каналов отличие в 2,5 раза. А для того, чтобы получить $t_{0i}(K_i) = 0{,}2$ с можно использовать вместо восьми отдельных устройств только пять таких же устройств с единой общей очередью.

Рассматриваемое отличие во времени ответа и требуемое число устройств / каналов зависит от соотношения λ_i / μ_i, а время ответа в обоих вариантах зависит еще и от абсолютных значений λ_i и μ_i, так как время ответа, например, для одной отдельной СМО с очередью зависит от разности абсолютных значений λ и μ и определяется выражением $1/(\mu + \lambda)$.

Пусть $\lambda_i = 8$ з/с, $\mu_i = 2$ з/с, тогда $\lambda_i / \mu_i = 4$ (Табл. 3, рис. 5), т.е. соотношение λ_i / μ_i такое же, как и в предыдущем примере, а их абсолютные значения различны.

Табл. 3

Число устройств или каналов	$t_{0i}(K_i)$ (с)	$t_{0i_{K_i}}$ (с)
5	2,5	0,998
6	1,5	0,588
7	1,167	0,524
8	1	0,508

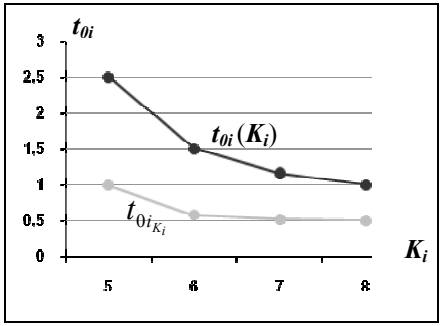

Рис. 5. Зависимости времени ответа от числа устройств или каналов

Из сопоставления последнего примера с предыдущим, видно, что при меньших абсолютных значениях λ_i и μ_i отличие во времени ответа и требуемое число устройств / каналов практически одно и то же, хотя абсолютное значение времени ответа, естественно, другое.

На основе изложенного предлагается следующая последовательность действий.

На первом этапе определить значение K_{ionm} по выражениям (2) или (5), в зависимости от постановки задачи. На втором этапе определить время ответа в каждом из узлов, где $K_{ionm} > 1$, для обоих вариантов. Затем выбрать нужное число устройств (каналов), в каждом из этих узлов, объединенных общей очередью, получив при этом или меньшее время ответа или (и) меньшее число устройств (меньшую стоимость). Как видно из приведенных примеров, экономия во времени ответа и стоимости получается весьма существенной.

Следует заметить, что если используются узлы с ограниченной длиной очереди (или вообще без очереди), то в случае многоканальности значительно уменьшается вероятность отказа, а, следовательно, увеличивается пропускная способность.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Пескова С.А., Кузин А.В., Волков А.Н. Сети и телекоммуникации. М.: Академия, 2008. 352 с.

2. Майоров С.А., Новиков Г.И., Алиев Т.И., Махарев Э.И., Тимченко Б.Д. Основы теории вычислительных систем. М.: Высшая школа, 1978. 400 с.

А.Н. КАБАНОВ, Г.И. НЕЧАЕВ,
Д.Н. ФОЛОМКИН, Ю.А. ЕГОРОВА, О.В. ПЕТРОВА
Рязанский государственный радиотехнический университет

ДВУХЭТАПНЫЙ МЕТОД ОПЕРАТИВНОГО СПЕКТРАЛЬНОГО АНАЛИЗА

Рассматривается двухэтапный метод оперативного спектрального анализа информационного процесса с использованием матричного уравнения.

Постановка задачи

В настоящее время остается актуальной задача создания новых методов обработки данных, которые наряду с выполнением требований заданной точности, обеспечивали бы повышение скорости обработки и давали возможность существенного сжатия информации для организации передачи данных и возможности последующего восстановления исходного сигнала [1,2]. Большое распространение получили спектральные методы обработки данных с адаптивной подстройкой ряда параметров выбранной базисной системы функций.

В статье предлагается двухэтапный метод оперативного спектрального анализа с подстройкой масштабного коэффициента базисной системы функций. На первом этапе на основе аппроксимации исходного сигнала при двух различных масштабных коэффициентах определяется оптимальная оценка масштабного коэффициента базисной системы функций. На втором этапе на основе матричных преобразований оперативно определяется оценка спектрального разложения исходного процесса при выбранном масштабном коэффициенте. Такой подход позволяет повысить скорость обработки, осуществить значительное сжатие информации, так как для последующего восстановления требуется только коэффициенты разложения при двух различных масштабных коэффициентах. Метод является помехоустойчивым, так как нахождение коэффициентов разложения при двух различных масштабных коэффициентах представляет собой интегральные моменты исходного процесса. Под масштабом понимается временное растяжение или сжатие при обработке временных сигналов, или пространственное линейное преобразование при обработке изображений. Предложенный в статье двухэтапный подход был проверен с помощью ряда численных экспериментов при наличии помех типа белого шума и показал высокую эффективность по числу вычислительных операций и высокой помехоустойчивости метода.

Решение задачи

Этап №1. Описание алгоритма определения масштаба времени непрерывных информационных процессов

За критерий качества аппроксимации исходного процесса y(t) принимается квадратичный критерий

$$\mathrm{I} = \int_0^{\infty} (y(t) - \sum_{i=1}^{N} c_i \cdot \varphi_i(\frac{t}{T}))^2 dt \,. \qquad (1)$$

Можно, подбирая параметры c_i, $i = \overline{1, N}$; T, найти их значения, обеспечивающие минимум выбранного критерия. Однако этот путь сложный, длительный, так как потребует многократного численного вычисления интеграла (1).

Предлагается методика на основе оценок $\hat{C}_i$, определённых из условия минимума критерия (1) только при двух различных значениях $\hat{T}1$, а именно $\hat{T}1 = T21$ и $\hat{T}1 = T22$, определить «истинные» значения параметров процесса *T1*. При такой постанове задачи в качестве измеряемых величин выступают уже не значения наблюдаемого процесса, а оценки коэффициентов

$$\vec{Y}1^T = \left\{\hat{C}1(T21), \hat{C}2(T21), ..., \hat{C}N(T21)\right\} \qquad (2)$$

и

$$\vec{Y}2^T = \left\{\hat{C}1(T22), \hat{C}2(T22), ..., \hat{C}N(T22)\right\}. \qquad (3)$$

В матричном виде

$$\vec{Y}1 = \overline{F1} \cdot \vec{C}; \vec{Y}2 = \overline{F2} \cdot \vec{C}\,, \qquad (4)$$

где

$$\overline{F1} = \overline{A}^{-1}(T21) \cdot \overline{B}(T21, T1); \overline{F2} = \overline{A}^{-1}(T22) \cdot \overline{B}(T22, T1); \qquad (5)$$

$$\overline{B}(T_i, T_j) = \int_0^{\infty} \vec{\varphi}(\frac{t}{T_i}) \cdot \vec{\varphi}^T(\frac{t}{T_j}) dt\,;$$

$$\overline{A}(T) = \int_0^{\infty} \vec{\varphi}(\frac{t}{T}) \cdot \vec{\varphi}^T(\frac{t}{T}) dt\,.$$

Задавшись некоторым значением $T1 = \hat{T}1$, определим оценки $\hat{\bar{C}}1, \hat{\bar{C}}2$. Эти оценки зависит от двух параметров (*T21*, $\hat{T}1$), (*T21*, $\hat{T}1$).

$$\hat{\bar{C}}1(T21, \hat{T}1) = \bar{C}1 \cdot \vec{Y}1,$$

где

$$\bar{C}1 = \bar{F}1^{-1} = \bar{B}^{-1}(T21, \hat{T}1) \cdot \bar{A}(T21).$$

$$\hat{\bar{C}}2(T22, \hat{T}1) = \bar{C}2 \cdot \vec{Y}2,$$

где

$$\bar{C}2 = \bar{F}2^{-1} = \bar{B}^{-1}(T22, \hat{T}1) \cdot \bar{A}(T22).$$

За истинное значение T1 принимается значение $\hat{T}1$, при котором величина квадратичного критерия

$$Q = [\hat{\bar{C}}1(T21, \hat{T}1) - \hat{\bar{C}}2(T22, \hat{T}1)]^T \cdot [\hat{\bar{C}}1(T21, \hat{T}1) - \hat{\bar{C}}2(T22, \hat{T}1)]$$

$$= \sum_{i=1}^{N} \Delta C_i^2 \qquad (6)$$

принимает минимальное значение.

Этап №2. Формирование матричных уравнений для определения коэффициентов разложения при оптимальном масштабном коэффициенте

На основе уравнений (4), (5) можно записать матричное соотношение:

$$W*C=Y, \qquad (7)$$

где $W^T = \left[\overline{F1} / \overline{F2}\right]$; $Y^T = \left[\overrightarrow{Y1} / \overrightarrow{Y2}\right]$.

Решение для оценок С можно получить из переопределенной системы (7) методом наименьших квадратов. Использование переопределенной системы уравнений позволяет повысить помехоустойчивость оценок.

Демонстрационный пример применения двухэтапного метода оперативного спектрального анализа

В качестве наблюдаемого принят следующий процесс:

$$Y(t) = \sum_{i=1}^{5} c_i \varphi_i(t),$$

где

$$\varphi_i(t) = \sqrt{\frac{2i}{T}} \sum_{k=1}^{i} (-1)^{k+i} \cdot \frac{(i-k+1) \cdot (i-k+2) \ldots (i+k-1)}{k! \cdot (k-1)!} \cdot e^{-\frac{k \cdot t}{T}}$$

, T = 5, C_i = {4 -2 1 3 1}^{T}.

Оценки $\hat{C}_i$, определённые при двух различных значениях $\hat{T}1$, а именно $\hat{T}1 = T21 = 1$и $\hat{T}1 = T22 = 10$, представлены в таблице 2.

Таблица 2.

T	**C1**	**C2**	**C3**	**C4**	**C5**
1	2,16	0,78	1,79	-2,265	1,587
10	4,505	-0,442	-0,655	-0,175	3,504

Оптимальный масштаб T, найденный на основе минимизации критерия (6), равен **5**.

Матричное уравнение для определения коэффициентов разложения при оптимальном масштабном коэффициенте имеет вид:

$$\begin{pmatrix} 0{,}736 & 0{,}614 & 0{,}281 & 0{,}056 & 0{,}004 \\ -0{,}397 & 0{,}142 & 0{,}603 & 0{,}575 & 0{,}328 \\ 0{,}248 & -0{,}175 & -0{,}322 & 0{,}081 & 0{,}527 \\ -0{,}222 & 0{,}221 & 0{,}194 & -0{,}264 & -0{,}318 \\ 0{,}146 & -0{,}164 & -0{,}093 & 0{,}213 & 0{,}122 \\ 0{,}943 & -0{,}267 & 0{,}143 & -0{,}080 & 0{,}150 \\ 0{,}333 & 0{,}754 & -0{,}403 & 0{,}136 & -0{,}274 \\ 0{,}000 & 0{,}594 & 0{,}470 & -0{,}162 & 0{,}565 \\ 0{,}000 & 0{,}086 & 0{,}733 & -0{,}097 & -0{,}447 \\ 0{,}000 & 0{,}000 & 0{,}252 & 1{,}032 & 0{,}166 \end{pmatrix} * \begin{matrix} c1 \\ c2 \\ c3 \\ c4 \\ c5 \end{matrix} = \begin{pmatrix} 2{,}160 \\ 0{,}780 \\ 1{,}790 \\ -2{,}265 \\ 1{,}587 \\ 4{,}505 \\ -0{,}442 \\ -0{,}665 \\ -0{,}175 \\ 3{,}504 \end{pmatrix}$$

Коэффициенты разложения

$C = \{4{,}001 \quad -2{,}015 \quad 0{,}986 \quad 3{,}004 \quad 0{,}990\}^T$.

Результат аппроксимации $Y_{апр}(t)$ на основе двухэтапного метода представлены на рис. 1, рис. 2.

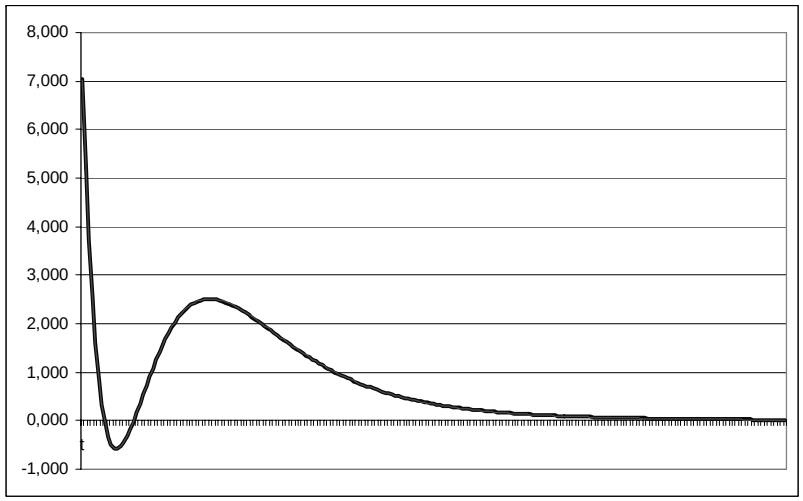

Рис. 1. Результат аппроксимации на основе двухэтапного метода

Относительная ошибка аппроксимации:

$$\sigma = \frac{\int_0^\infty (y(t) - y_{apr}(t))^2 dt}{\int_0^\infty y(t)^2 dt}. \quad \sigma = 0{,}00002.$$

При наличии помех типа белого шума с интенсивностью 0,05 результат показан на рис. 2.

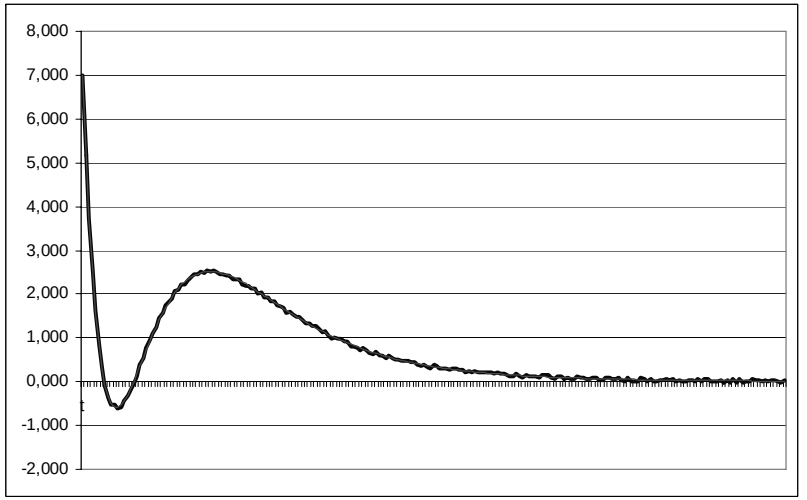

Рис. 2. Результат аппроксимации при наличии помехи

Относительная ошибка аппроксимации:

$$\sigma = \frac{\int_0^\infty (y(t) - y_{apr}(t))^2 dt}{\int_0^\infty y(t)^2 dt}. \quad \sigma = 0{,}000503.$$

Выводы: Предложенный метод отличается высокой универсальностью, что обеспечивается полным набором аналитических соотношений в пространстве коэффициентов разложения для различных базисных систем функций с подстройкой масштабного коэффициента. Метод может применяться при решении задач обработки данных неза-

висимо от их физической природы. Формирование переопределенной системы уравнений на втором этапе решения задачи позволяет повысить помехоустойчивость искомых оценок.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Идентификация и диагностика систем: учебник для студ. выш. учеб. заведений / А.А.Алексеев, Ю.А.Кораблев, М.Ю.Шестопалов. – М.: издательский центр «Академия», 2009 – 352с.

2. Методы и средства идентификации динамических объектов / А.А.Бессонов, Ю.В.Загашвили, А.С.Маркелов. - Л.: Энергоатомиздат. Ленингр. отд-ние, 1989. – 280с.

М. В. КОЛМЫКОВ

Рязанский государственный университет им. С. А. Есенина

ПРИМЕНЕНИЕ НЕЙРОПРОЦЕССОРА NM 6403 В ЗАДАЧАХ СОКРАЩЕНИЯ ИЗБЫТОЧНОСТИ ВИДЕОИНФОРМАЦИИ

Рассматривается проблема кодирования изображений на бортовых космических системах сбора информации. Приводится метод сокращения времени кодирования с использованием нейропроцессора.

В современных космических системах ДЗЗ актуальной остается проблема разработки бортовых космических систем сбора информации (БКССИ) с обязательным сжатием космических изображений и использовании обычного радиоканала для ее передачи на наземные пункты приема. Решение такой проблемы значительно снижает объем передаваемой информации, снижает технические требования на пропускную способность радиоканала, что позволяет использовать обычные радиоканалы.

Для решения рассматриваемой проблемы сжатия видеоизображения в работе предлагается использовать нейронные сети (НС) с целью повышения производительности БКССИ и повышении объемов обрабатываемой информации с возможностями гибкой перестройки БКССИ посредством обучения НС в реальном времени. При этом за счет использования многослойности НС можно дополнительно улучшить некоторые технические параметры системы: уточнение границ, контрастирование, выравнивание освещенности, удаление мелких локальных образований, подавление некоторых шумов, помех и искажений.

При этом спектр сигнала превращается в отфильтрованный сигнал. Путем настройки промежуточного и конечного слоя можно гибко менять параметры НС, что позволит осуществлять кроме фильтрации, уточнение границ, контрастирование, удаление мелких локальных образований, выравнивание освещенности и др. Поэтому в процессе проектирования НС необходимо минимизировать количество слоев в НС, количество нейронов и связей между ними при аппаратной реализации НС. Эта задача решается на уровне структурной организации нейросети, при выборе которой учитывать следующие аспекты:

-способность сети к обучению, т.е. возможность научить систему распознавать требуемое количество объектов, причем чем больше в сети слоев l и нейронов n, тем выше ее способности и потребности в аппаратных ресурсах;

- быстродействие, которое достигается уменьшением сложности сети - чем меньше нужно аппаратных ресурсов, тем быстрее осуществляется работа НС.

Методика связана с разложением X_n по базису $\{b_n\}$, пространства с использованием известных базисных функций Фурье, Уолша, Хаара и др. При этом необходимо решать задачу обратного спектрального преобразования для вычисления X_n. Последняя методика очень хорошо укладывается на нейронные сети и является эквивалентной оптимальному разложению Карунена - Лоэва с минимальным числом коэффициентов разложения. Для реализации указанной методики в общем случае требуется однослойная электронная вычислительная схема, в которой каждый слой содержит n_r нейронов - сумматоров с весовыми коэффициентами w_{ip}, где r - номер слоя, w_{ip} - весовой коэффициент нейрона и p - номер нейрона в слое.

Таким образом, для построения НС необходимо решить задачу оптимизации структуры по вышеуказанным критериям. Данная задача может быть решена на базе современного отечественного нейропроцессора Л1879ВМ1. Он сочетает в себе черты двух современных архитектур: VLIW и SIMD, которые позволяют существенно увеличить производительность процессора с большими потоками данных и матричными операциями. При этом основным функциональным элементом является векторный сопроцессор (VCP) с операндами переменной длины, представляющий собой матричное векторное устройство 64*64 ячеек с набором регистров общего назначения. Структура Л1879ВМ1 позволяет произвольно разделить матрицу на столбцы и строки, а в образовавшиеся после разделения макроячейки загружаются весовые

коэффициенты w_{ip}, а на вход матрицы подается вектор входных данных $X = \{x_1, x_2, ..., x_n\}$, каждому элементу которого соответствует строка матрицы, ширина которой в битах определяется разрядностью данного элемента входных данных.

После загрузки рабочей матрицы непосредственно приступают к вычислениям. Для этого необходимо сначала задать адреса входного и выходного буферов. При этом желательно, чтобы эти буферы располагались в разных блоках памяти, находящихся на разных шинах данных. В этом случае операции чтения входного буфера и запись выходного могут выполняться параллельно. Каждая векторная инструкция содержит внутренний счетчик циклов, количество выполняемых элементарных циклов лежит в пределах от 1 до 32. Эти элементарные циклы описывают выполнение одной и той же операции над потоком данных, т.е. реализация архитектуры SIMD. Максимальное количество данных, обрабатываемых одной векторной командой, определяется глубиной внутренних буферов памяти процессора и равна 32 64-х разрядным словам.

Выигрыш по времени выполнения вычислительных операций на нейропроцессоре достаточно значителен по сравнению с DSP процессорами за счет естественной параллельности нейросетевого алгоритма.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Нейрокомпьютеры в системах обработки изображений. Кн. 7: Коллективная монография / Общая ред. А. И. Галушкина. – Радиотехника, 2003.- 192 с.: ил.

2. С. В. Мушкаев, С. В. Ландышев. Применение процессора NM6403 (Л1879ВМ1) для сжатия изображений // Цифровая обработка сигналов, 2002 № 1. С. 12 – 18.

3. Очин Е. Ф. Вычислительные системы обработки изображений.- Л.: Энергоатомиздат. Ленингр. отд-ние, 1989. – 136 с.: ил.

4. В. Н. Ручкин Применение нейропроцесоров семейства Л1897 в задачах обработки изображений

А.В. КОРЖАВИН
Рязанский государственный радиотехнический университет

ПРОГРАММНЫЕ СРЕДСТВА ИССЛЕДОВАНИЯ ЭФФЕКТИВНОСТИ МНОГОПОРОГОВЫХ ДЕКОДЕРОВ САМООРТОГОНАЛЬНЫХ КОДОВ

Рассмотрены принципы работы многопорогового декодера (МПД) самоортогональных кодов. Описаны программные средства для для моделирования МПД, представлены полученные с помощью программных средств результаты моделирования МПД в канале связи с некоррелированными релеевскими замираниями.

Динамичный переход нашей технологической цивилизации на цифровые системы обработки и передачи информации создает много проблем при проектировании современных систем информатики и телекоммуникаций. Одной из важнейших задач, которые при этом необходимо решать, является обеспечение высокой достоверности передачи данных.

Наиболее эффективным средством повышения достоверности цифровой информации при ее передаче по каналам с шумом является применение помехоустойчивого кодирования. Изучение наиболее перспективных методов кодирования по критерию «эффективность-производительность» показало, что наиболее предпочтительными для применения в высокоскоростных каналах связи являются многопороговые декодеры (МПД), которые много лет успешно развиваются российскими специалистами и относятся к наиболее простым и одновременно достаточно эффективным алгоритмам коррекции ошибок в каналах с большим уровнем шума [1]. В настоящее время алгоритмы типа МПД могут декодировать длинные коды почти так же, как и оптимальные переборные алгоритмы [2]. Однако сами МПД сохраняют при этом линейную сложность реализации. При одинаковой эффективности МПД выполняет примерно в 100 раз меньшее число операций, чем соответствующие турбо декодеры. При аппаратной реализации МПД может быть в некоторых случаях более быстрым, чем, например, турбо декодеры, почти на 3 десятичных порядка. По совокупности характеристик эффективности, сложности реализации и быстродействия к уровню алгоритмов МПД в ближайшее время не смогут даже приблизиться никакие методы, известные в настоящее время.

МПД используется для декодирования блоковых или сверточных самоортогональных кодов (СОК), кодер для которых является простейшим устройством, состоящим только из регистров сдвига и сумматоров по модулю 2 [2]. Достаточно простым является и сам МПД,

пример которого для блокового кода показан на рис. 1. Отметим, что МПД отличается от обычного порогового декодера, лежащего в основе схемы, только наличием разностного регистра, в котором отмечаются измененные на пороговом элементе информационные символы. Решения порогового элемента из разностного регистра затем используются другим пороговым элементом на следующей итерации декодирования.

На пороговом элементе каждой итерации МПД в процессе декодирования информационного символа u_k при использовании жесткого модема (соответствует случаю, когда демодулятор оценивает только значения принятых битов) выполняются следующие операции [2]:

1. Вычисляется сумма проверок (каждая из которых для случая жесткого модема равна 0 или 1), т.е. функция

$$L_k = \sum_{m=1}^{J} S_{g_m} + r_k,$$

где $J = d–1$ – количество проверок (ненулевых элементов порождающего полинома кода g); r_k – символ разностного регистра, относящийся к декодируемому символу u_k (равный 0 или 1); S_m – m-й элемент синдромного регистра, входящий в множество проверок относительно декодируемого символа u_k.

2. Если $L_k > T$, где $T \geq (d–1)/2$ – значение порога порогового элемента, то символ u_k, все связанные с ним проверки $\{S_{g_m}\}_{m=1,J}$ и символ r_k инвертируются.

3. Переход к декодированию следующего символа (п.1).

Заметим, что при применении мягкого модема (соответствует случаю, когда демодулятор оценивает не только значения принятых битов, но и их надежность) в МПД выполняются те же операции, но проверки на пороговых элементах суммируются уже с весами, определяющими надежность оценок принятых из канала битов. [3] Алгоритм работы одной итерации декодирования МПД представлен на рис. 2.

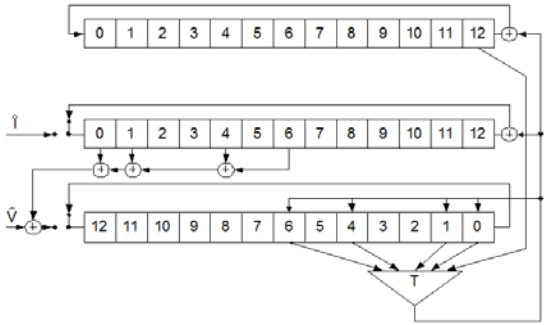

Рис. 1. Схема МПД блокового кода

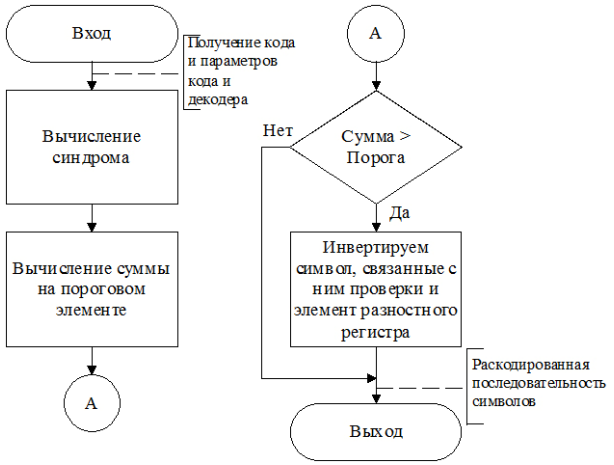

Рис. 2. Алгоритм работы МПД

Одним из способов для определения эффективности применения МПД является построение графиков зависимости вероятности ошибки декодирования символа от уровня шума в канале передачи данных, что позволяет оценить эффективность МПД при заданных параметрах модулятора, канала передачи данных, кодера и декодера. На сегодняшний день существует программное обеспечение, позволяющее выполнять моделирование МПД только в гауссовских каналах с аддитивным шумом. Однако такая модель канала часто не соответствует реальным каналам, в которых нередко наблюдается гораздо более сложный характер ошибок. Поэтому автором статьи были разработаны программные средства для моделирования МПД в каналах, содержащих не только аддитивную, но и мультипликативную компоненты.

Для разработки программных средств моделирования МПД была выбрана система MATLAB R2007a, обладающая широким набором функций, позволяющих моделировать различные системы передачи данных.

Разработанные программные средства позволяют проводить моделирование МПД в различных моделях канала передачи данных (с аддитивным белым гауссовским шумом, релеевскими и др. замираниями) при использовании разнообразных способов модуляции (BPSK, QPSK, QAM16, QAM64 и др.) как с жестким, так или мягким демодулятором сигнала. В процессе работы программные средства обеспечи-

вают построение графика зависимости вероятности ошибки декодирования принятого символа от уровня шума в канале передачи данных.

В качестве примера на рис. 3 представлены результаты моделирования МПД в канале связи с независимыми релеевскими замираниями и двоичной фазовой модуляцией, полученные при использовании разработанных программных средств. Здесь использовался МПД для блокового самоортогонального кода с кодовой скоростью 1/2 и длиной информационной части 5460 битов. Отметим, что ухудшение характеристик МПД в таких условиях составляет примерно на 1,5 дБ по сравнению с характеристиками аналогичного МПД в канале только с гауссовским шумом, что является достаточно хорошим результатом.

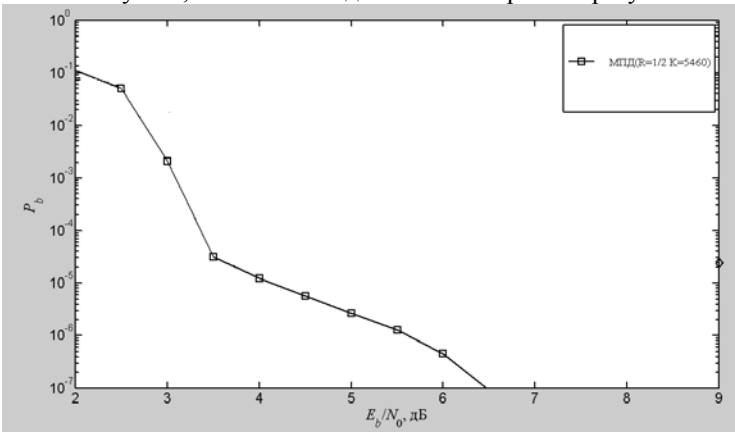

Рис. 3. Моделирование МПД в канале связи с замираниями

Разработанный пакет программ моделирования МПД существенно облегчит проведение исследований в области помехоустойчивого кодирования и снизит затраты на их проведение, что в будущем поможет существенно улучшить как качество, так и доступность высокоскоростной передачи данных для конечного пользователя.

*Работа выполнена при поддержке РФФИ (грант №08-07-00078).

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Зубарев Ю.Б., Овечкин Г.В. Обзор методов помехоустойчивого кодирования с использованием многопороговых декодеров // Электросвязь. М., 2008. №12. С.2–11.

2. Золотарев В.В. Теория и алгоритмы многопорогового декодирования. М.: Радио и связь, Горячая линия – Телеком, 2006.

3. Веб-сайты www.mtdbest.iki.rssi.ru и www.mtdbest.ru.

К.М. ЛОГУТКИНА
Рязанский государственный радиотехнический университет

АНАЛИЗ ФИНАНСОВОГО СОСТОЯНИЯ ПРЕДПРИЯТИЯ МЕТОДАМИ ТЕОРИИ НЕЧЕТКИХ МНОЖЕСТВ

Рассматривается подход к применению процедуры нечеткого вывода для формализации и автоматизации процесса анализа финансового состояния предприятия.

В настоящее время в связи со сложившейся ситуацией в мировой экономике особенно важным при осуществлении инвестиционной деятельности становится определение степени финансового здоровья или нездоровья объекта инвестирования. Ведь чем меньше предприятие имеет долгов, тем соответственно меньше у него возможности ответить по всем долгам, падает платежеспособность и доход инвестора.

В практике финансового анализа очень хорошо известен ряд показателей, характеризующих отдельные стороны текущего финансового положения предприятия, а также огромное количество моделей и методов анализа финансового состояния (модель Чессера, Лиса, Альтмана), однако большинство из них основаны на стохастическом подходе к анализу неопределенности. Стохастическая неопределенность имеет место в ситуациях, когда некоторое хорошо описанное событие может произойти, а может и не произойти.

При анализе финансового состояния имеет место не только стохастическая неопределенность, но и лингвистическая, которая возникает из-за неясности, нечеткости и субъективности описания и оценки рисковых событий. Поэтому при наличии лингвистической неопределенности стохастические модели сильно усложняют оценку финансового состояния предприятия и не всегда являются адекватными.

Теория нечетких множеств позволяет существенно усилить подход к анализу финансового состояния предприятия, объединяя учет количественных (финансовых) и качественных (индикаторных) показателей в анализе, причем рассматривая их не только в статике, но и в динамике; а также, учитывая все отмеченные недостатки существующих подходов.

Для того чтобы наилучшим образом изучить предметную область, рассматриваемую в данной статье, необходимо определить из каких процессов будет состоять анализ. К ним относятся:

1) Подготовка к анализу финансового состояния предприятия;
2) Идентификация финансовых показателей;
3) Качественный и количественный анализ финансовых показателей (или собственно анализ финансового состояния);

4) Планирование реагирования на динамику показателей;
5) Мониторинг и управление финансовыми показателями.

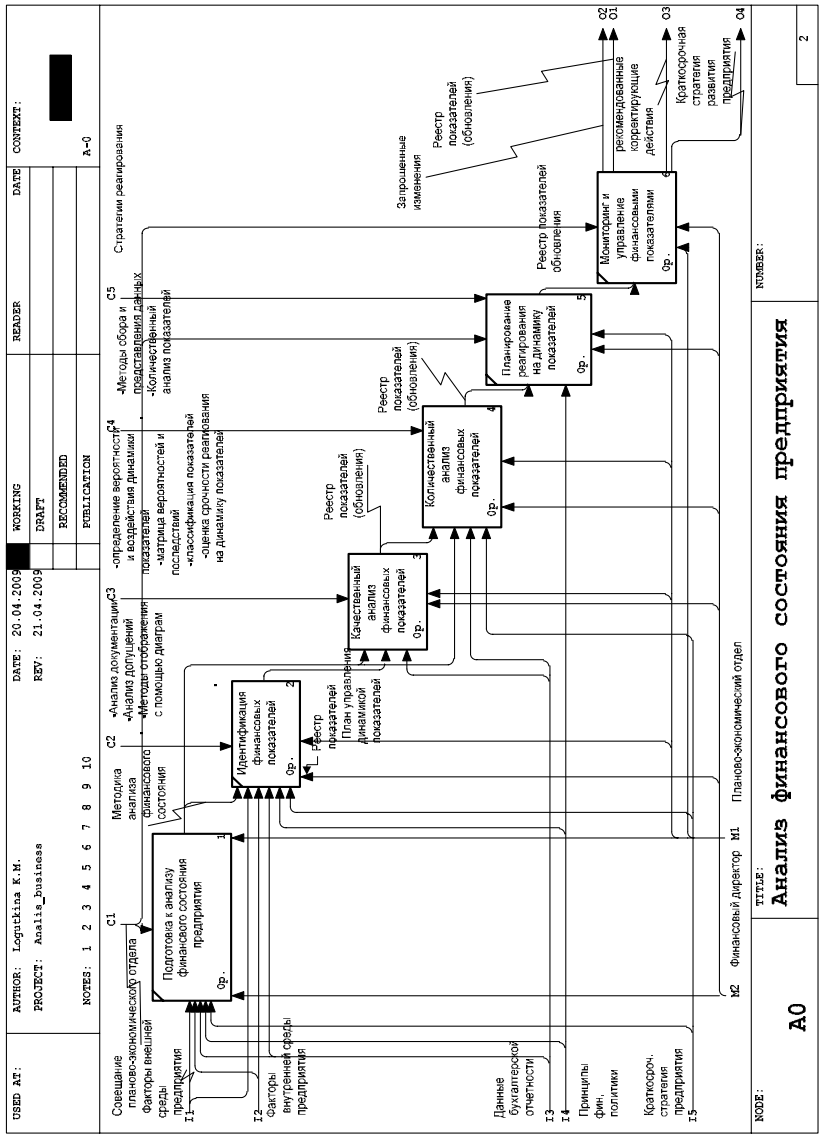

Рис. 1. Функциональная модель анализа финансового состояния предприятия в нотации IDEF0

На рис. 1 представлена функциональная модель взаимосвязи этих процессов в нотации IDEF0. Эти процессы взаимодействуют как друг с другом, так и с процессами из других областей знаний. В зависимости от потребностей предприятия в каждом процессе могут принимать участие один или несколько человек или групп.

Остановимся подробнее на процессе самого анализа. Центральной часть данного процесса является модуль нечеткого вывода, реализующий один из алгоритмов нечеткого вывода (Мамдани, Сугено и т.д.). Основные этапы работы алгоритма: формирование базы правил системы нечеткого вывода, фаззификация входных переменных, агрегирование подусловий, активизация подзаключений в нечетких правилах продукций, аккумулирование заключений нечетких правил продукций, дефаззификация.

Нечеткий вывод базируется в первую очередь на базе правил и функциях принадлежности, описывающих входные и выходные лингвистические переменные.

База правил нечетких продукций представляет собой конечное множество правил, согласованных относительно используемых в них лингвистических переменных и в данном случае представляется в форме структурированного текста:

ПРАВИЛО_1: ЕСЛИ "Условие_1" ТО "Заключение_1" (F1)
ПРАВИЛО_2: ЕСЛИ "Условие_2" ТО "Заключение_2" (F2)
. . .
ПРАВИЛО_n: ЕСЛИ "Условие_n" ТО "Заключение_n»" (Fn).

Здесь через $F_i(i \in \{1,2,...,n\})$ обозначены коэффициенты определенности или весовые коэффициенты соответствующих правил. Эти коэффициенты могут принимать значения из интервала $[0,1]$.

Лингвистические переменные, которые используются в нечетких высказываниях подусловий правил нечетких продукций, являются входными лингвистическими переменными, а переменные, которые используются в нечетких высказываниях подзаключений правил нечетких продукций, считаются выходными лингвистическими переменными.

Входными переменными являются коэффициенты, описывающие состояние предприятия, выходная переменная – риск банкротства.

Для характеристики значений входных переменных $\beta_i, i = \overline{1,n}$ (где n – количество входных переменных) эксперты чаще всего используют следующие выражения:

1) очень низкий уровень показателя β_i,

2) низкий уровень показателя β_i,

3) средний уровень показателя β_i,

4) высокий уровень показателя β_i,

5) очень высокий уровень показателя β_i.

Таким образом, для входных лингвистических переменных терм-множеством является множество $T_1 =$ {Очень низкий, Низкий, Средний, Высокий, Очень высокий}.

Для выходной лингвистической переменной можно определить терм-множество $T_2 =$ {Предельный, Высокий, Средний Низкий, Незначительный}.

Для построения функций принадлежности ω_1 классифицируем значение показателя степени риска как критерий разбиения этого множества на нечеткие подмножества.

Табл. 1. Классификация текущего значения риска банкротства

Интервал значений ω_1	Классификация уровня параметра	Степень оценочной уверенности (функция принадлежности)
$0 < \omega_1 < 0.15$	риск банкротства незначителен	1
$0.15 < \omega_1 < 0.25$	риск банкротства незначителен	$\mu_5 = 10 \cdot (0.25 - \omega_1)$
	низкая степень риска банкротства	$1 - \mu_5 = \mu_4$
$0.25 < \omega_1 < 0.35$	низкая степень риска банкротства	1
$0.35 < \omega_1 < 0.45$	низкая степень риска банкротства	$\mu_4 = 10 \cdot (0.45 - \omega_1)$
	средняя степень риска банкротства	$1 - \mu_4 = \mu_3$
$0.45 < \omega_1 < 0.55$	средняя степень риска банкротства	1
$0.55 < \omega_1 < 0.65$	средняя степень риска банкротства	$\mu_3 = 10 \cdot (0.65 - \omega_1)$
	высокая степень риска банкротства	$1 - \mu_3 = \mu_2$
$0.65 < \omega_1 < 0.75$	высокая степень риска банкротства	1

$0.75 < \omega_1 < 0.85$	высокая степень риска банкротства	$\mu_2 = 10 \cdot (0.85 - \omega_1)$
	предельный риск	$1 - \mu_2 = \mu_1$
$0.85 < \omega_1 < 1.0$	предельный риск	1

Пример функции принадлежности для одной из входных лингвистических переменных представлен на рис. 2.

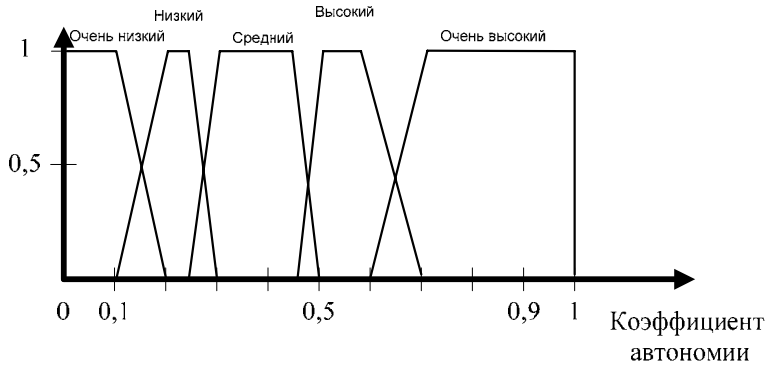

Рис. 2. Функция принадлежности одной из входных лингвистических переменных

Аналогично строится функция принадлежности для выходной лингвистической переменной.

Предложенная структура процесса анализа финансового состояния предприятия на основе применения процедуры нечеткого вывода расширяет возможности по формализации процесса управления риском банкротства при принятии управленческих решений по инвестированию и облегчает труд лиц, отвечающих за это решение.

Таким образом, описанный выше подход к анализу финансового состояния позволяет со всех сторон проанализировать деятельность предприятия; выявить слабые точки в его работе; повысить уровень знаний сотрудников планово-экономического отдела.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Недосекин А.О. Нечетко-множественный анализ рисков фондовых инвестиций / А.О. Недосекин. - СПб.: Сезам. 2002. 182 с.

2. Леоненков, А.В. Нечеткое моделирование в среде MATLAB и fuzzyTECH / А.В. Леоненков. - СПб.: БХВ-Петербург. 2005. 736 с.

А.И. ПЕРЕПЕЛКИН
Рязанский государственный университет имени С.А. Есенина

АЛГОРИТМ ПОВЫШЕНИЯ ПРОИЗВОДИТЕЛЬНОСТИ ТРАНСПОРТНОГО ПРОТОКОЛА ЗА СЧЕТ УМЕНЬШЕНИЯ РАЗМЕРА ПЕРЕДАВАЕМЫХ СЕГМЕНТОВ

Разработан алгоритм работы транспортного протокола, предотвращающий возникновение ошибок и снижающий потерю производительности соединения в условиях их появления.

В настоящее время передача данных по сети является важной составляющей существования информации. Огромное значение и распространение приобрела глобальная сеть Internet, постоянно включающая в себя все новые отдельные компьютеры и вычислительные сети. За производительность функционирования соединений отвечает транспортный уровень, на котором в большинстве случаев применяется протокол TCP [1]. Различными организациями и исследователями постоянно ведется работа по совершенствованию протокола. Основными его отрицательными качествами являются постоянное нахождение сети в состоянии перегрузки и существенное снижение производительности при появлении ошибок [2].

Анализ трафика на транспортном уровне выявил наличие периодичности в последовательности передаваемых и принимаемых сегментов в рамках конкретного соединения [3]. Выявление в каждый момент времени сценария [4] и его параметров, позволяет определить отклонение в условиях передачи. Существующие механизмы определения ошибок дополняются возможностью их прогнозирования. Отклонение от сценария позволяет заранее предпринимать необходимые действия для их избежания.

Для устранения приведенных недостатков разработан следующий алгоритм уменьшения размера передаваемых сегментов для оптимизации передачи.

Стандартный протокол TCP определяет появление ошибки по возникновению таймаута и приходу трех повторных подтверждений (ACK). Пропуск ожидаемых по сценарию ACK является дополнительным критерием возникновения сбоя в сети.

При наличии сценария введем число n – количество отправленных сегментов, после которых не были приняты ожидаемые в периоде подтверждения. При малом времени возврата (RTT) можно принять количество пропущенных ACK равным двум. При большом – увеличить до пяти. Данный параметр выбирается эмпирическим путем. В

случае большого времени реакции долгое ожидание подтверждения повлечет за собой передачу большого объема информации в условиях уже возникшей ошибки, особенно в несимметричных каналах.

Слежение за трафиком на предмет определения сценария необходимо продолжать и после пропуска ACK. Если в новом режиме такой пропуск является закономерным, это позволит сформировать новый сценарий и выбрать новое значение n.

После передачи n сегментов, отправитель уменьшает размер последующих, сохраняя интервал отправки, и продолжает посылать данные в пределах окна. После прихода ACK (с номером подтверждения больше последнего), размер пакетов восстанавливается. Если ACK не пришел, и передалось все окно, то применяется алгоритм медленного старта. Блок-схема данного алгоритма приведена на рис. 1.

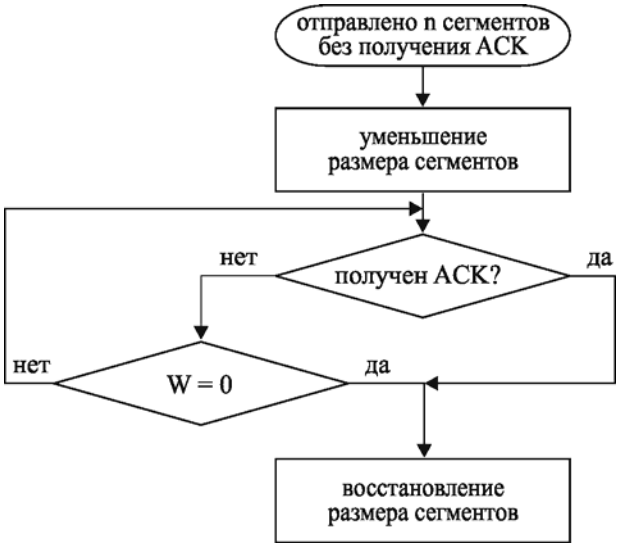

Рис. 1. Блок-схема алгоритма работы протокола при известном сценарии в случае пропуска подтверждений (W – размер окна)

Предлагаемый алгоритм, в условии отсутствия подтверждений, дает ряд преимуществ:

- уменьшается количество передаваемых данных при повторной передаче, если сегменты не доходят до адресата;
- снижается интенсивность трафика, что позволяет снизить нагрузку на маршрутизаторы, которые могли стать причиной задержки;

- в отличие от протокола TCP, применяющего алгоритм медленного старта, который резко уменьшает интенсивность передачи с последующим медленным наращиванием после прихода ожидаемого подтверждения, передача восстанавливается с прежним сценарием, либо формируется новый;

- уменьшенный трафик сокета освобождает канал передачи для других соединений того же узла;

- увеличивается интервал ожидания прихода подтверждений без начала повторной передачи данных, успешно переданных при первой передаче, но подтверждения на которые были задержаны;

- продолжение передачи вместо ее приостановки, дает возможность получить новый сценарий в новых условиях, когда задержка сегментов стала закономерной.

Уменьшение сегментов, а не простое внесение задержек в моменты их отправки, позволяет контролировать состояние сети с той же интенсивностью. Размер сегментов не изменяет закономерности отправки подтверждений принимающей стороной, так как ACK отправляется в ответ на сегмент любой длины. В результате временное появление задержек при передаче, также временно снижает интенсивность трафика без уменьшения производительности (как это происходит в предлагаемом алгоритме).

Например, пусть принимающая сторона объявила размер окна в 16К. Размер информационных данных сегмента составляет 536 байт. Скорость соединения – 3К/С.

После передачи нескольких периодов возникает переполнение по направлению к принимающей стороне; сегменты перестают доходить, но ранее отправленные сегменты продолжают поступать и на них отправляются подтверждения. В стандартном алгоритме ошибка не обнаруживается, пока не возникает таймаут на первый недошедший сегмент. За это время отправляется 12К данных, которые далее с медленного старта будут переданы повторно. Потери времени составляют как минимум 4 с.

При использовании алгоритмов, использующих сценарий, размер сегментов уменьшается в k = 2,5 раза. Дальнейшее увеличение k делает существенной долю заголовков протоколов стека по отношению к передаваемой полезной информации, меньшее – не дает выигрыша при применении алгоритмов и не обеспечивает быструю разгрузку переполнения.

При использовании сценария, анализ номеров подтверждений показывает, что сегменты доходят все с большей задержкой. При уменьшении трафика после 335 сегмента, на момент прихода подтверждения (сегмент 355) передастся 5К неподтвержденных данных, поэтому потери времени составят только 1,5 С. При передаче 1М информации ситуации, подобные описанной возникали 4 раза. Использование сценария позволяет сэкономить порядка 8% времени.

Использование предложенного алгоритма позволяет сгладить всплесковый характер трафика, предотвращая постоянную перегрузку, что характерно для большинства применяемых в настоящее время реализаций протокола TCP. При этом не требуется точный расчет RTT для контроля над состоянием сети. Для функционирования алгоритма необходимо ведение статистики по последним отправляемым и принимаемым сегментам. Анализ сетевого трафика показал возможность повышения производительности в случае возникновения ошибок.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. RFC 793 – Transmission Control Protocol. 1981.

2. Andrei Gurtov. TCP Performance in the Presence of Congestion and Corruption Losses. Master's Thesis. University of Helsinki, 2000.

3. RFC 2001 – TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms. 1997.

4. Федоров С. В., Перепелкин А. И. Определение параметров сетевого трафика протокола TCP // Информационные технологии в проектировании // Межвузовский сборник научных трудов. Рязань, 2004. С. 67 - 69.

Д.А. ПЕРЕПЕЛКИН
Рязанский государственный радиотехнический университет

КРИТЕРИИ ВЫБОРА ОПТИМАЛЬНОГО МАРШРУТА ПЕРЕДАЧИ ДАННЫХ В КОРПОРАТИВНЫХ СЕТЯХ

Рассматриваются критерии выбора оптимального маршрута передачи пакетов данных в протоколах маршрутизации корпоративных сетей.

Быстрый рост числа корпоративных сетей и развитие технологий передачи данных задают большие требования для обеспечения высокоскоростного и надежного обмена данными между узлами телекоммуникационной системы при жестких требованиях к задержкам информации. Точное определение оптимальных маршрутов передачи данных позволяет поддерживать корпоративную сеть в необходимом для пользователя состоянии.

Определение маршрута – это выбор последовательности транзитивных узлов и их интерфейсов, через которые надо передавать данные, чтобы доставить их адресату. Определение маршрута довольно сложная задача, особенно когда конфигурация сети такова, что между парой взаимодействующих сетевых интерфейсов существует множество путей. Чаще всего выбор останавливают на одном оптимальном по некоторому критерию маршруте. В качестве критериев оптимальности могут выступать, например, время и надежность доставки данных получателю по выбранному маршруту. На практике для снижения объема вычислений ограничиваются поиском не оптимального в математическом смысле, а рационального, то есть близкого к оптимальному маршруту.

Задача выбора еще более упрощается за счет того, что при анализе критерия оптимизации учитываются далеко не все факторы, влияющие на этот критерий. Например, если искать самый лучший с точки зрения времени доставки маршрут передачи данных, на время доставки сказываются топология сети (количество транзитивных узлов, которые должны пройти данные), скоростные характеристики каналов связи, их загруженность, надежность каналов и транзитных устройств, а также многие другие факторы [1].

В качестве количественной оценки критериев выбора оптимального маршрута передачи данных используются следующие показатели:

• Административная дистанция - это мера достоверности источника маршрута. Если маршрутизатор получил информацию о месте назначения более чем от одного протокола маршрутизации, то сравниваются значения административных дистанций и предпочтение отдается маршрутам, чья административная дистанция меньше;

• Метрика – это мера, используемая протоколом маршрутизации для вычисления наилучшего пути к любому данному месту назначения, если протокол изучает несколько путей к одному и тому же месту назначения. Каждый протокол маршрутизации использует различные метрики;

• Длина префикса.

В настоящее время в IP-сетях применяются протоколы маршрутизации, в которых маршрут выбирается по критерию кратчайшего расстояния. При этом расстояние измеряется в различных метриках. Чаще всего используется простейшая метрика – количество хопов, то есть количество маршрутизаторов, которые нужно преодолеть пакету до сети назначения. В качестве метрик применяются также пропускная

способность и надежность каналов, вносимые ими задержки и любые комбинации этих метрик.

Среди протоколов маршрутизации выделяют внутренние и внешние протоколы маршрутизации. Внутренние протоколы маршрутизации работают только в одной автономной системе. Примерами протоколов внутренней маршрутизации являются протоколы OSPF, RIP, IGRP и EIGRP. Внешние протоколы маршрутизации предназначены для обмена информацией между автономными системами. Примером протокола внешней маршрутизации является протокол BGP.

Протоколы внутренней маршрутизации делятся на две группы, каждая из которых связана с одним из следующих типов алгоритмов:

- Алгоритмы состояния каналов (Link State Algorithm, LSA);
- Дистанционно-векторные алгоритмы (Distance Vector Algorithm, DVA).

Алгоритм состояния каналов обеспечивает каждый маршрутизатор информацией, достаточной для построения точного графа связей сети. Все маршрутизаторы работают на основании одинаковых графов, что делает процесс маршрутизации более устойчивым к изменениям конфигурации. «Широковещательная» рассылка (то есть передача пакета всем непосредственным соседям маршрутизатора) используется здесь только при изменениях состояния связей, что происходит в надежных сетях не так часто. Вершинами графа являются как маршрутизаторы, так и объединяемые ими сети. Распространяемая по сети информация состоит из описания связей различных типов: маршрутизатор - маршрутизатор, маршрутизатор – сеть.

Протоколом, основанным на алгоритме состояния каналов стека TCP/IP, является протокол OSPF (Open Shortest Path First).

В протоколе OSPF каждый маршрутизатор самостоятельно решает задачу оптимизации маршрутов. В процессе выбора оптимального маршрута анализируется ориентированный граф сети. Выбор оптимального маршрута определяется по алгоритму Дейкстры.

Метрики выбранного пути могут характеризоваться следующими параметрами качества обслуживания (QoS):

1) пропускной способностью канала;
2) задержкой (время распространения пакета);
3) числом дейтограмм, стоящих в очереди для передачи;
4) загрузкой канала;
5) требованиями безопасности;
6) типом трафика;
7) числом шагов до цели;
8) возможностями промежуточных связей.

Протокол OSPF производит расчет метрики отдельно для каждого вида сервиса TOS (type of service). Число TOS определяет многообразие метрик, соответствующих видам сервиса, для данного канала. Последовательность описания метрик задается величиной кода TOS. Значения кодов TOS, принятых в протоколе OSPF приведены в табл. 1.

Табл. 1.

OSPF - код	TOS - коды	TOS (RFC 1349)
0	0000	Обычный сервис
2	0001	Минимизация денежной стоимости
4	0010	Максимальная надежность
8	0100	Максимальная пропускная способность
16	1000	Минимальная задержка

В протоколе OSPF используется принцип контроля состояния канала (link-state protocol), а метрика представляет собой оценку эффективности связи в этом канале: чем меньше метрика, тем эффективнее организация связи. В простейшем случае метрика маршрута может равняться его длине в пересылках (hops). Но в общем случае значения метрики могут определяться в гораздо более широком диапазоне [2].

Метрика, оценивающая пропускную способность канала, определяется, например, компанией Cisco, как количество секунд, нужное для передачи 100 Мбит. *Имеется следующая формула для вычисления метрики доставки информации через каналы сети OSPF:*

Метрика = 10^8 / скорость передачи в битах в секунду.

Метрика протокола OSPF для используемых в корпоративных сетях каналов передачи данных приведена в табл. 2:

Табл. 2.

Наименование канала	Скорость канала	Метрика канала
Канал 1	100 Мбит/с	1
Сеть Ethernet / 802.3	10 Мбит/с	10
Тракт E1	2,048 Мбит/с	48
Тракт T1	1,544 Мбит/с	65
Канал 2	64 Кбит/с	1562
Канал 3	56 Кбит/с	1785
Канал 4	19,2 Кбит/с	5208
Канал 5	9,6 Кбит/с	10416

Протоколами, основанным на дистанционно-векторном алгоритме стека TCP/IP, являются протоколы RIP (Routing Information Protocol), протокол IGRP (Interior Gateway Routing Protocol) и протокол EIGRP (Enhanced Interior Gateway Routing Protocol).

Протокол RIP представляет собой один из старейших протоколов обмена маршрутной информацией, однако он до сих пор широко распространен в корпоративных сетях. В этом протоколе все сети имеют номера (способ образования номера зависит от используемого в сети протокола сетевого уровня), а все маршрутизаторы - идентификаторы. Протокол RIP широко использует понятие «вектор расстояний». Вектор расстояний представляет собой набор пар чисел, являющихся номерами сетей и расстояниями до них в хопах. Вектора расстояний итерационно распространяются маршрутизаторами по сети, и через несколько шагов каждый маршрутизатор имеет данные о достижимых для него сетях и о расстояниях до них. Если связь с какой-либо сетью обрывается, то маршрутизатор отмечает этот факт тем, что присваивает элементу вектора, соответствующему расстоянию до этой сети, максимально возможное значение, которое имеет специальный смысл – «связи нет». Таким значением в протоколе RIP является число 16. Метрика представляет собой оценку качества связи в данной сети (на данном физическом канале); чем меньше метрика, тем лучше качество соединения. Метрика маршрута равна сумме метрик всех связей (сетей), входящих в маршрут. В простейшем случае в протоколе RIP метрика каждой сети равна единице, а метрика маршрута тогда просто является его длиной в хопах.

Преимуществом протокола RIP является его вычислительная простота, а недостатками - увеличение трафика при периодической рассылке широковещательных пакетов и не оптимальность найденного маршрута [3].

Протокол IGRP разработан компанией Cisco для больших сетей со сложной топологией и сегментами, которые обладают различной полосой пропускания и задержкой. Это внутренний протокол маршрутизации имеет некоторые черты сходства с протоколом OSPF.

Выбор оптимального маршрута в протоколе IGRP определяется по алгоритму Беллмана-Форда. IGRP использует несколько типов метрики выбранного пути, по одной на каждый вид QOS. Метрика характеризуется 32-разрядным числом. В однородных средах этот вид метрики вырождается в число шагов до цели. Маршрут с минимальным значением метрики является предпочтительным. Актуализация маршрутной информации для этого протокола производится каждые 90 секунд. Если какой-либо маршрут не подтверждает своей работоспособности в течение 270 сек., он считается недоступным. После семи циклов (630 сек) актуализации такой маршрут удаляется из маршрутных таблиц. IGRP аналогично OSPF производит расчет метрики для каждого вида сервиса (TOS) отдельно.

Метрика, используемая в IGRP, учитывает:

1) время задержки;

2) пропускную способность самого слабого сегмента пути;

3) загруженность канала;

4) надежность канала.

Время задержки предполагается равным времени, необходимому для достижения места назначения при нулевой загрузке сети. Расчет метрики производится для каждого сегмента пути. Время от времени каждый маршрутизатор широковещательно рассылает свою маршрутную информацию всем соседним маршрутизаторам. Получатель сравнивает эти данные с уже имеющимися и вносит, если требуется, необходимые коррекции. На основании вновь полученной информации могут быть приняты решения об изменении маршрутов. Одним из преимуществ IGRP является простота реконфигурации.

Оптимальный маршрут в протоколе IGRP определяется с использованием комбинированной метрики, вычисляемой по следующей формуле:

Метрика = $[(k1 / b_e) + (k2 * d_c)] * r$ **,** где:

1) k1 и k2 – константы;

2) b_e – пропускная способность канала * (1 - загрузка канала);

3) d_c – топологическая задержка;

4) r – относительная надежность (% пакетов, успешно передаваемых по данному сегменту пути).

По умолчанию определение метрики в протоколе IGRP выполняется по критериям пропускной способности и задержки, значения которых приведены в табл. 3:

Табл. 3.

Наименование канала передачи данных	Пропускная способность	Задержка
Спутник – 500 Мбит/с	20	200 000
Сеть Ethernet / 802.3 – 10 Мбит/с	1 000	100
Канал 1 – 1.544 Мбит/с	6 476	2 000
Канал 2 – 64 Кбит/с	156 250	2 000
Канал 3 – 56 Кбит/с	178 571	2 000
Канал 4 – 10 Кбит/с	1 000000	2 000
Канал 5 – 1 Кбит/с	10 000000	2 000

Пропускная способность в протоколе IGRP измеряется в величинах, обратных бит/сек, умноженных на 10^{10}. Например, если пропускная способность равна N Кбит/с, то ее измерением в IGRP будет 10000000 / N). Надежность измеряется в долях от 255 (255 соответствует 100%). Загрузка измеряется также в долях от 255, а задержка в десятках миллисекунд.

В протоколе IGRP маршрут, имеющий наименьшую комбинированную метрику, считается лучшим. При использовании такой схемы появляется возможность, используя весовые коэффициенты, адаптировать выбор маршрутов к задачам конечного пользователя [4].

Протокол EIGRP – дистанционно векторный протокол внутренней маршрутизации, разработанный фирмой Cisco на основе протокола IGRP той же фирмы. Протокол EIGRP – переработанный и улучшенный вариант IGRP, свободный от основного недостатка дистанционно-векторных протоколов – особых ситуаций с зацикливанием маршрутов – благодаря специальному алгоритму распространения информации об изменениях в топологии сети. Несмотря на то, что в общем случае протоколы состояния связей (OSPF) отрабатывают изменения в топологии сети быстрее, чем EIGRP, а также OSPF имеет ряд дополнительных возможностей, EIGRP более прост в реализации и менее требователен к вычислительным ресурсам маршрутизатора.

EIGRP-маршрутизатор обнаруживает своих соседей путем периодической рассылки сообщений «Hello». Эти же сообщения используются для мониторинга состояния связи с соседом (рассылаются каждые 5 секунд в сетях с большой пропускной способностью – например, для сетей Ethernet – и каждые 60 секунд в «медленных» сетях). Такой мониторинг позволяет рассылать в сети векторы расстояний не периодически, а только при изменении топологии сети.

EIGRP использует комплексное значение метрики, вычисляемое на основании показателей пропускной способности и задержки при передаче данных в сети. Также в расчет метрики могут быть включены показатели загрузки и надежности сети. В отличие от протокола RIP метрика в EIGRP не является фактором, ограничивающим размер системы. При получении от соседей векторов расстояний, маршрутизатор для каждой сети назначения не только выбирает соседа, через которого лежит кратчайший путь в эту сеть, но также запоминает и вероятных заместителей (feasible successors). Вероятным заместителем становится маршрутизатор, объявивший метрику маршрута от себя до данной сети меньшую, чем полная метрика установленного маршрута.

Формула вычисления метрики по умолчанию для протокола EIGRP определяется по следующей формуле:

Метрика = 256 * (10^7) / скорость передачи в битах в секунду + 256 (задержка).

Для решения задачи внешней маршрутизации разработан протокол BGP (Border Gateway Protocol). Используемая в настоящий момент версия этого протокола имеет номер 4.

Протокол BGP в своей работе использует подход под названием path vector, являющийся развитием дистанционно-векторного подхода. BGP-соседи рассылают (анонсируют, advertise) друг другу векторы путей (path vectors). Вектор путей, в отличие от вектора расстояний, содержит не просто адрес сети и расстояние до нее, а адрес сети и список атрибутов (path attributes), описывающих различные характеристики маршрута от маршрутизатора-отправителя в указанную сеть. Данных, содержащихся в атрибутах пути, должно быть достаточно, чтобы маршрутизатор-получатель, проанализировав их с точки зрения политики своей автономной системы, мог принять решение о приемлемости или неприемлемости полученного маршрута. В протоколе BGP в качестве метрики используется число шагов до цели, и время распространения маршрутной информации, причем, у разных маршрутизаторов может быть прописана разная маршрутная политика.

Характеристиками процедуры выбора маршрута в протоколе BGP являются:

• В таблице BGP хранятся все известные пути, а в таблице маршрутизации – лучшие;

• Пути выбираются на основании маршрутных политик;

• Пути не выбираются на основании пропускной способности.

На маршрутизаторе Cisco при отсутствии маршрутной политики, выбор маршрута происходит следующим образом (на каждый следующий шаг маршрутизатор переходит только при совпадении значений на предыдущем):

1) максимальное значение weight (локально для маршрутизатора);

2) максимальное значение local preference (для всей AS);

3) предпочесть локальный маршрут маршрутизатора (next hop = 0.0.0.0);

4) оптимальный маршрут через автономные системы;

5) минимальное значение origin code (IGP < EGP < incomplete);

6) минимальное значение MED (распространяется между автономными системами);

7) маршрут EBGP лучше, чем маршрут IBGP;

8) выбрать маршрут через ближайшего IGP-соседа;

9) выбрать самый старый маршрут для EBGP-пути;

10) выбрать маршрут через соседа с наименьшим BGP router ID;

11) выбрать маршрут через соседа с наименьшим IP-адресом.

Только лучший маршрут помещается в таблицу маршрутизации и анонсируется BGP-соседям [5].

В заключение хотелось бы отметить, что в большинстве случаев выбор метрики в протоколах маршрутизации корпоративных сетей зачастую носит неформальных характер, так как последнее решение всегда остается за администратором сети. Однако знание базовых принципов вычисления метрики в протоколах маршрутизации позволяет более точно определить оптимальные маршруты передачи данных, повысить эффективность функционирования корпоративных сетей и обеспечить конечному пользователю необходимый уровень качества обслуживания.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Олифер В.Г., Олифер Н.А. Основы компьютерных сетей – СПб.: Питер, 2009. – 352 с.

2. Томас, Том М. II. Структура и реализация сетей на основе протокола OSPF. 2-е изд.– Издательский дом «Вильямс», 2004. – 816 с.

3. Столлингс В. Современные компьютерные сети. 2-е изд. – СПб.: Питер, 2003. – 783 с.

4. Леинванд А., Пински Б. Конфигурирование маршрутизаторов Cisco. 2-е изд. – Издательский дом «Вильямс», 2001. – 368 с.

5. Хелеби С., Мак-Ферсон Д. Принципы маршрутизации в Internet. 2-е изд. – Издательский дом «Вильямс», 2001. – 448 с.

В.Е. РУДАКОВ, С.В. СКВОРЦОВ
Рязанский государственный радиотехнический университет

АЛГОРИТМ ПОСТРОЕНИЯ БАЗОВОГО МНОЖЕСТВА НЕЗАВИСИМЫХ ПУТЕЙ ПОТОКОВОГО ГРАФА

Предлагается алгоритм построения независимых путей потокового графа, ориентированный на использование в задачах структурного тестирования программных модулей. В основе алгоритма лежит метод перечисления всех путей ориентированного графа.

При разработке прикладных программ часто возникают прикладные задачи, для решения которых используется математический аппарат теории графов. К ним в первую очередь относятся задачи анализа потока управления в программе, тестирования, оценки сложности и др. [1].

В данной работе рассматривается задача определения базового множества линейно-независимых путей потокового графа, решение которой требуется при тестировании прикладных программных модулей по методу «белого ящика» [2]. Потоковый граф является моделью тестируемого программного модуля и представляет собой ориентированный граф $G = (X, U)$, где $X = \{x_1, x_2,\ldots, x_n\}$ - множество вершин, $U = \{u_1, u_2,\ldots, u_m\}$ - множество дуг вида $u_m = (x_i, x_j)$, $x_i, x_j \in X$. Вершины потокового графа соответствуют линейным участкам программы, включающим один или несколько операторов, а дуги отображают поток управления в программе. Различают предикатные и операторные вершины. Из операторной вершины выходит одна дуга, а из предикатной – две, что определяет условную передачу управления.

Пусть граф G имеет одну входную вершину s и одну выходную вершину t, где $s, t \in X$. Тогда некоторый k-й путь из s в t можно представить как последовательность $\mu_k[s, t] = (s, x_{k_1}, x_{k_2},..., x_{k_p}, t)$ вершин, соединенных дугами графа $(s, x_{k_1}), (x_{k_1}, x_{k_2}),..., (x_{k_p}, t) \in U$. Множество путей из вершины s в вершину t обозначим как $M_{st} = \{\mu_1, \mu_2,\ldots, \mu_q\}$.

Требуется определить базовое множество путей M_{st}, которое включает все независимые пути из s в t. Каждый путь $\mu_k \in M_{st}$ содержит дугу, не входящую ни в какой другой путь этого множества [1].

С точки зрения задачи тестирования программных модулей базовое множество путей M_{st} обладает следующими свойствами [2]. Во-первых, тесты, обеспечивающие анализ путей $\mu_k \in M_{st}$, гарантируют однократное выполнение всех операторов программы, включая каждое условие для истинного и ложного результатов проверки. Во-вторых, мощность этого множества, т.е. число путей $\mu_k \in M_{st}$, равна цикломатической сложности потокового графа $\nu(G)$, которую можно вычислить по формуле [2]

$$\nu(G) = m(G) - n(G) + 2, \qquad (1)$$

где $m(G)$ - число дуг, $n(G)$ - число вершин.

Второе свойство имеет важное прикладное значение, так как позволяет получить априорную оценку числа независимых путей, входящих в множество M_{st}. Например, для графа, показанного на рис. 1, имеем $\nu(G) = 5 - 5 + 2 = 2$, а базовое множество путей принимает вид $M_{st} = M_{ae} = \{(a, b, c, e), (a, b, d, e)\}$.

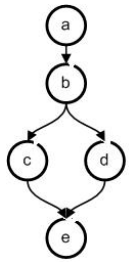

Рис. 1. Потоковый граф условного оператора

В работе [2] независимым называется любой путь, который включает новую вершину графа (оператор обработки или новое условие) и должен содержать дугу, не входящую в ранее определенные пути. Поэтому предлагается формировать все независимые пути в порядке от самого короткого к самому длинному, таким образом, чтобы каждый следующий путь содержал новую дугу. Для этого можно использовать метод перечисления всех путей графа [3], использующий матрично-алгебраический подход.

Пусть $P^{(r)} = [p_{ij}^{(r)}]_{n\times n}$ – матрица всех путей длины $r + 1$ в графе G, т.е. таких путей, которые включают r промежуточных вершин. Элементы этой матрицы определяются следующим образом

$$p_{ij}^{(r)} = p^{(r)}(x_i, x_j) = M_{ij} = \{\mu_k[x_i, x_j]\} = \sum_{k=1}^{q} x_{k_1} x_{k_2} ... x_{k_r}, \qquad (2)$$

и описывают множества M_{ij} путей $\mu_k[x_i, x_j]$, $k = \overline{1, q}$, в виде последовательностей промежуточных вершин (без начальной x_i и конечной x_j вершин), где x_{k_r} это r-я по порядку промежуточная вершина пути. Если путей длины $r + 1$ между x_i и x_j нет, то $p_{ij}^{(r)} = 0$. Слагаемые $x_{k_1} x_{k_2} ... x_{k_r}$ представляют собой композицию (сцепление) вершин, лежащих на некотором пути от x_i к x_j [3].

Матрица смежности графа G, является матрицей $P^{(0)}$, описывающей все пути без промежуточных вершин. Вычисление матриц $P^{(r)}$ для $r > 0$ выполняется последовательно с использованием операции умножения матриц

$$P^{(r)} = P^* \times P^{(r-1)}, \qquad (3)$$

где $P^* = [p_{ij}^*]_{n\times n}$ - вспомогательная матрица, в которой элемент $p_{ij}^* = x_j$, если $(x_i, x_j) \in U$, и $p_{ij}^* = 0$ в противном случае. Заметим, что

элемент $p_{ij}^{(r)}$ матрицы $P^{(r)}$ определяется как $p_{ij}^{(r)} = \sum_{k=1}^{n} P_{ik}^{*} P_{kj}^{(r-1)}$ и описывает все пути длины $r + 1$ из x_i и x_j, поскольку j-й столбец матрицы $P^{(r-1)}$ задает все пути длины r с начальной вершиной x_k ($k = \overline{1,q}$) и конечной x_j, а i-я строка матрицы P^* определяет конечные вершины x_j всех дуг $(x_i, x_j) \in U$. При этом сомножители в произведениях не переупорядочиваются и не используется степенная запись повторяющихся сомножителей, что позволяет не потерять информацию о порядке следования вершин.

Таким образом, используя формулу (3) можно сформировать ряд матриц $P^{(1)}$, $P^{(2)}$, …, $P^{(r-1)}$, в совокупности определяющих все пути с длиной меньше или равной r для любой пары вершин графа.

Пример 1. Выполним перечисление всех путей для графа, представленного на рис. 1. Матрицы P^* и $P^{(0)}$ для этого графа имеют вид:

$$P^{*} = \begin{bmatrix} 0 & b & 0 & 0 & 0 \\ 0 & 0 & c & d & 0 \\ 0 & 0 & 0 & 0 & e \\ 0 & 0 & 0 & 0 & e \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}; \; P^{(0)} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Вычисление $P^{(1)}$ и $P^{(2)}$ выполняется следующим образом:

$$P^{(1)} = P^{*} \times P^{(0)} = \begin{bmatrix} 0 & 0 & b & b & 0 \\ 0 & 0 & 0 & 0 & c+d \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}; \; P^{(2)} = P^{*} \times P^{(1)} = \begin{bmatrix} 0 & 0 & 0 & 0 & bc+bd \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Для матрицы $P^{(2)}$ в ячейке $p[a, e]$ имеем запись $bc+bd$, что означает получение путей $\mu_1[a, e] = (a, b, c, e)$ и $\mu_2[a, e] = (a, b, d, e)$.

Покажем далее использование рассмотренного метода перечисления путей для решения задачи нахождения базового множества независимых путей. Предлагаемый алгоритм выполняет последовательное получение путей возрастающей длины и выбор независимых путей, связывающих начальную и конечную вершины потокового графа исследуемой программы.

Алгоритм построения базового множества независимых путей

Данные: потоковый граф $G = (X, U)$, где $X = \{x_1, x_2, \ldots, x_n\}$ - множество вершин, $U = \{u_1, u_2, \ldots, u_m\}$ - множество дуг.

Результаты: базовое множество независимых путей $M_{st} = \{\mu_1, \mu_2, \ldots, \mu_q\}$ из начальной вершины s в конечную вершину t.

Шаг 1 (определение начальных значений).

Положим $M_{st} = \varnothing$. Рассчитаем цикломатическую сложность графа $\nu(G)$ по формуле (1). Зададим матрицу $P^{(0)}$, равную матрице смежности графа G, и определим матрицу $P^* = [\, p_{ij}^{*} \,]_{n \times n}$ с элементами

$$p_{ij}^{*} = \begin{cases} x_j, \text{ если } (x_i, x_j) \in U; \\ 0, \text{ если } (x_i, x_j) \notin U. \end{cases}$$

Шаг 2 (расчет матрицы путей длины $r + 1$ для $r = 1,2,3\ldots$).

Вычислим матрицу $P^{(r)}$ по формуле (3).

Шаг 3 (поиск независимых путей).

Просмотрим ячейку $p_{st}^{(r)}$ матрицы $P^{(r)}$. Если $p_{st}^{(r)} = 0$, то новый независимый путь из s в t не получен. Если $p_{st}^{(r)} \neq 0$, то выберем из этой ячейки $p_{st}^{(r)} = \sum_{k=1}^{q} x_{k1} x_{k2} \ldots x_{kr}$ описание независимых путей $\mu_k[s,\ t] = (s, x_{k_1}, x_{k_2}, \ldots, x_{k_p},\ t)$, где $k = \overline{1,q}$, и дополним множество $M_{st} = M_{st} \cup \{\mu_k[s,t], k = \overline{1,q}\}$.

Шаг 4 (проверка окончания работы алгоритма).

Если число найденных путей больше или равно $\nu(G)$, то завершаем работу алгоритма, в противном случае возвращаемся к шагу 2.

Пример 2. Рассмотрим построение базового множества независимых путей для потокового графа, показанного на рис. 2 и описывающего циклический фрагмент программы, приведенный на рис. 3.

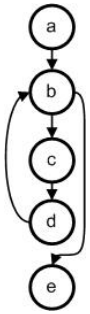

Рис. 2. Потоковый граф

```
a   begin
    randomize;
b   for i:=2 to 40 do
    begin
c   A[i]=random(1000);
    writeln('A[',i,']=',A[i]);
d   end;
e   readln();
    end.
```

Рис. 3. Фрагмент программы

Цикломатическая сложность этого графа оценивается как $\nu(G) = 2$, поскольку $m = n = 5$. Следовательно, базовое множество независимых путей M_{ae} должно содержать два пути $\mu_1[a, e]$ и $\mu_2[a, e]$, связывающих начальную вершину a и конечную вершину b. Матрица смежности $P^{(0)}$, вспомогательная матрица P^* и результат их умножения (матрица путей $P^{(1)}$) имеют вид:

$$P^{(0)} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}; \quad P^* = \begin{bmatrix} 0 & b & 0 & 0 & 0 \\ 0 & 0 & c & 0 & e \\ 0 & 0 & 0 & d & 0 \\ 0 & b & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}; \quad P^{(1)} = P^* \times P^{(0)} = \begin{bmatrix} 0 & 0 & b & 0 & b \\ 0 & 0 & 0 & c & 0 \\ 0 & d & 0 & 0 & 0 \\ 0 & 0 & b & 0 & b \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

После формирования матрицы $P^{(1)}$ в ячейке $p(a, e)$ имеем запись b, что определяет первый искомый путь $\mu_1[a, e] = (a, b, e) = abe$. Поскольку количество найденных путей меньше величины цикломатической сложности программы, сформируем очередную матрицу $P^{(2)}$ и затем $P^{(3)}$:

$$P^{(2)} = P^* \times P^{(1)} = \begin{bmatrix} 0 & 0 & 0 & bc & 0 \\ 0 & cd & 0 & 0 & 0 \\ 0 & 0 & db & 0 & db \\ 0 & 0 & 0 & bc & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}; \quad P^{(3)} = P^* \times P^{(2)} = \begin{bmatrix} 0 & bcd & 0 & 0 & 0 \\ 0 & 0 & cdb & 0 & cdb \\ 0 & 0 & 0 & dbc & 0 \\ 0 & bcd & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Очевидно, что эти матрицы не порождают новых путей из вершины a в вершину e, так как в обоих случаях $p(a, e) = 0$. Однако следующее умножение является результативным, так как $p(a, e) \neq 0$:

$$P^{(4)} = P^* \times P^{(3)} = \begin{bmatrix} 0 & 0 & bcdb & 0 & bcdb \\ 0 & 0 & 0 & cdbc & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & dbcd & 0 & 0 & 0 \\ 0 & 0 & bcdb & 0 & bcdb \end{bmatrix}.$$

Запись $bcdb$, которая содержится в ячейке $p(a, e)$ матрицы $P^{(4)}$, определяет второй искомый путь $\mu_2[a, e] = (a, b, c, d, b, e) = abcdbe$.

В заключение отметим, что предложенный алгоритм наглядно демонстрирует процесс поиска независимых путей, выполняя последовательное формирование всех путей в графе с постепенным увеличением их длины. Однако существенным недостатком является достаточно большая вычислительная сложность, связанная с выполнением матричной операции умножения. Необходимость получения результата такого умножения в символьном виде (композиция вершин), а также большая разреженность формируемых матриц, делают актуальной задачу разработки алгоритма поиска базового множества независимых

путей, который использует специальные списочные структуры для описания потокового графа и формируемых путей.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Евстигнеев В.А. Применение теории графов в программировании. – М.: Наука, 1985. – 352 с.

2. Орлов С. Технологии разработки программного обеспечения. - СПб.: Питер, 2002. - 464 с.

3. Кофман А. Введение в прикладную комбинаторику. - М.: Наука, 1975. 480 с.

А.Н. САПРЫКИН
Рязанский государственный радиотехнический университет

ГЕНЕТИЧЕСКИЙ АЛГОРИТМ БАЛАНСИРОВКИ НАГРУЗКИ ПОРТОВ КОММУТАТОРА

Предлагается решение задачи оптимизации структуры сети Ethernet с коммутатором, имеющим разноскоростные порты.

В данной работе рассматривается решение задачи оптимизации структуры сети Ethernet на основе коммутатора с разноскоростными портами. При оптимизации сети учитывается: количество компьютеров в сети; количество портов коммутатора; пропускная способность каждого порта; объем трафика между компьютерами.

Для решения задачи топологического синтеза оптимальной схемы соединения узлов коммутации с рабочими станциями локальной сети Ethernet при заданной пропускной способности каналов и известном трафике можно использовать комбинаторный подход. Однако число анализируемых вариантов настолько велико, что получить решение задачи проблематично. Поэтому нахождение оптимальной структуры сети производится на основе генетического алгоритма.

В алгоритме используется минимаксный критерий оптимизации, при этом выравниваются коэффициенты нагрузки на наиболее перегруженных входных и выходных портах коммутатора. В конечном итоге, это позволяет повысить производительность коммутатора, а также увеличить резервную полосу пропускания устройства.

Генетические алгоритмы – процедуры поиска, основанные на механизмах естественного отбора и наследования. В них используется эволюционный принцип выживания наиболее приспособленных особей.

Генетический алгоритм для решения данной задачи включает в себя следующие этапы:

- формирование случайным образом начальной популяции, состоящей из *M* особей;
- проверка жизнеспособности хромосом – оценка каждой хромосомы по фитнесс функции с целью выбраковки неприспособленных особей;
- селекция хромосом – формирование пар хромосом для применения генетических операторов; случайным образом выбираются родители $s_1, s_2, \ldots, s_n$ из популяции в соответствии с распределением вероятностей; отбор особей реализуется с помощью метода "*remainder stochastic sampling"*. Для каждой особи вычисляется отношение ее приспособленности к средней приспособленности популяции. Целая часть этого отношения указывает, сколько раз нужно записать особь в промежуточную популяцию, а дробная показывает вероятность ее повторного попадания в промежуточную популяцию. Реализуется такой способ отбора следующим образом: располагаем особи на рулетке так, что размер сектора каждой особи пропорционален ее приспособленности. Допустим, что у рулетки не одна стрелка, а *N*, причем они отсекают одинаковые сектора. Тогда один запуск рулетки выберет сразу все *N* особей, которые нужно записать в промежуточную популяцию. Такой способ иллюстрируется рис. 1:

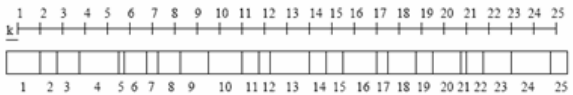

Рис. 1. Селекция хромосом

- применение генетических операторов – производится скрещивание и мутация; скрещивание выполняется с помощью оператора x_i – точечного кроссовера применительно к *i*-тому гену; при мутации производится модифицирование одного, случайно выбранного бита одного из генов хромосомы; оператор мутации необходим для "выбивания" популяции из локального экстремума и выполняет функцию предотвращения преждевременной сходимости. Один бит каждой особи популяции изменяется случайным образом с вероятностью 50%;
- формирование новой популяции;
- определяется условие остановки алгоритма, и если условие выполняется и найдено оптимальное решение, то производится остановка процесса эволюции, в противном случае весь процесс эволюции начинается заново; критерием остановки алгоритма служит так же

достижение заданного значения числа селекций или достижение некоторого заранее заданного критерия;

- выбор наилучшей хромосомы – если условие остановки генетического алгоритма выполняется, то фиксируется искомое решение, которое выводится на экран компьютера в виде таблиц и графиков.

Размер хромосом зависит от количества компьютеров в сети и от количества портов коммутаторов и в общем виде представляется в виде таблицы сопряжения элементов (табл. 1).

Количество хромосом в генотипе равно количеству портов коммутатора в рассматриваемой сети. Это приводит к громоздким вычислениям и сложному формированию начальной популяции. Поэтому вся цепочка хромосом сворачивается в одну хромосому, с использованием следующего кодирования $pc \in \{0, 1, ..., L\}$, где $L = \left(\sum_{i=1}^{n} psw \right)$ – указатель на порт коммутатора.

Табл. 1. Представление хромосомы в общем виде

	Порт 1	Порт 2	…	Порт *m*
Компьютер 1	1	0	…	0
Компьютер 2	0	0	…	1
…	…	…	…	…
Компьютер *n*		1	…	

Полученный алгоритм можно модифицировать таким образом, чтобы могла быть решена более сложная задача многоцелевой оптимизации. Она заключается в нахождении наилучшей структуры сети Ethernet путем разделения ее на подсети таким образом, чтобы по возможности равномерно был распределен сетевой трафик между различными сегментами сети. Ограниченная емкость коммутаторов при распределении компьютеров по сегментам становится одним из главных ограничений в задаче оптимизации.

В данном случае размер хромосом зависит от количества компьютеров в сети и от количества используемых коммутаторов и в общем виде представляется в виде таблицы сопряжения элементов, где наличие связи указывает 1, а ее отсутствие – 0 (табл. 2).

Табл. 2. Представление хромосомы при разделении сети Ethernet на подсети

	Коммутатор 1	Коммутатор 2	…	Коммутатор *m*
Компьютер 1	1	0	…	0
Компьютер 2	0	0	…	1
…	…	…	…	…
Компьютер *n*		1	…	
Коммутатор 1	0	1	…	1
…	…	…	…	…
Коммутатор *m*	1	0	…	0

Генетический алгоритм балансировки нагрузки портов коммутатора реализован в программе с использованием среды программирования Delphi.

Н.В. СКВОРЦОВ, С.В. СКВОРЦОВ, В.И. ХРЮКИН
Рязанский государственный радиотехнический университет

РЕШЕНИЕ ЗАДАЧИ ПОКРЫТИЯ ПРИ СИНТЕЗЕ ДИАГНОСТИЧЕСКИХ ГРАФОВ

Разработан и обоснован приближенный алгоритм построения двоичных таблиц, обеспечивающий их минимальное покрытие столбцами. Представлены основные результаты вычислительного эксперимента по исследованию предложенного алгоритма.

Решение многих задач, связанных с диагностированием многопроцессорных систем, требует построения двоичных таблиц, описывающих их возможные состояния [1]. Анализируя эти таблицы, можно определить текущее состояние системы и ее неисправные элементы. Размер подобной таблицы (число столбцов) определяет количество взаимных проверочных связей процессоров в цикле контроля технического состояния. Для ускорения поиска неисправностей следует уменьшать размеры таких таблиц, что приводит к необходимости решения задачи, связанной с определением минимального покрытия [2]. Эта задача 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.

Пусть задана булева матрица $A = [a_{ij}]_{m\times n}$, где m – число строк и n – число столбцов. Требуется определить множество столбцов с индексами $P = \{j_1, j_2, \ldots, j_k\}$, такое, что для каждой i-й строки ($i = 1,2,\ldots,m$) выполняется условие $\sum_{j\in P} a_{ij} \ge 1$, а общее количество таких столбцов минимально, т.е. $|P| \to \min$. Другими словами, требуется найти такое покрытие двоичной матрицы минимальным количеством столбцов, чтобы в каждой строке оставалась хотя бы одна единица.

Работа предлагаемого алгоритма включает не более n итераций (по числу столбцов матрицы A), на каждой из которых множество P дополняется индексом очередного столбца, включаемого в покрытие. После выбора столбца из матрицы A вычеркиваются (заполняются нулями) все строки, которые на пересечении с этим столбцом имеют единичные элементы. Указанные действия повторяются до тех пор, пока в матрице A остаются ненулевые элементы. Выбор столбца производится по одному из двух следующих правил.

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

Правило 2. Выбирается столбец, обеспечивающий покрытие максимального количества строк, т.е. такой k-й столбец, для которого $\sum_{i=1}^{m} a_{ik} \to \max$. Если таких столбцов несколько и их индексы составляют множество $M = \{k_1, k_2, \ldots, k_p\}$, то выбор осуществляется следующим образом. Вычисляются значения $E_k = \sum_{i=1}^{m}\sum_{j=1}^{n} a_{ik}a_{ij}$ для всех $k \in M$ и выбирается такой столбец, для которого $E_k \to \max$. Другими словами, выбирается столбец, который при удалении соответствующих строк из матрицы A оставляет в ней минимальное количество единиц.

Последовательный алгоритм решения задачи покрытия

Данные: булева матрица A.

Результаты: покрытие - множество P индексов столбцов.

1. Положить $P = \varnothing$.

2. Если есть столбец, удовлетворяющий правилу 1, то зафиксировать его индекс k и перейти к п.4.

3. Определить индекс k столбца по правилу 2.

4. Включить индекс k в множество P и присвоить $a_{ij} = 0$, где $j = 1, 2,\ldots, n$, для всех значений i, таких, что $a_{ik} = 1$.

5. Если $A \ne O$, где O – нулевая матрица, то перейти к п.1.

6. Конец алгоритма.

Оценим вычислительную сложность этого алгоритма. Очевидно, что на каждой итерации при просмотре столбцов выбирается лишь один претендент k на включение в покрытие P, а количество столбцов в покрытии в предельном случае может составлять n. На каждой итерации число просматриваемых столбцов уменьшается на 1 и в среднем составляет примерно $n/2$. В худшем случае выбор каждого столбца производится по правилу 2, что приводит к выполнению на одной итерации порядка $n^2m/2$ действий. В целом выполняется не более n итераций, что в итоге дает оценку $O(n^3m)$.

Пример. Найти минимальное покрытие таблицы, которая задана показанной ниже матрицей A. На первой итерации используется правило 1, в соответствии с которым выбирается первый столбец (он покрывает строку 7, имеющую один ненулевой элемент). После обнуления строк 1, 4, 7, имеющих на пересечении с этим столбцом единицы, получим матрицу $A^{(1)}$.

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

Результат второй итерации, где также используется правило 1, выбирается пятый столбец и обнуляются строки 3, 6, 10, показан в виде матрицы $A^{(2)}$. Третья итерация, на которой применяется правило 2 и выбирается столбец 6, в итоге дает матрицу $A^{(3)}$.

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

При выполнении четвертой итерации по правилу 2 выбирается второй столбец, а полученная в результате матрица $A^{(4)}$ становится нулевой. Таким образом, найдено покрытие $P = \{1, 5, 6, 2\}$, которое, как нетрудно убедиться, является оптимальным.

Экспериментальные исследования предложенного алгоритма для случайных матриц размером от 50×10 до 250×25 элементов показывают, что из 360 решенных задач полученные результаты являются оптимальными в 76 % случаев, а для оставшихся средняя погрешность составляет примерно 3 %.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Пархоменко П.П., Согомонян Е.С. Основы технической диагностики. М.: Энергия. 1981. 320 с.

2. Кристофидес Н. Теория графов. Алгоритмический подход. М.: Мир. 1978. 432 с.

А.И. ТАГАНОВ
Рязанский государственный радиотехнический университет

МОДЕЛИ СИСТЕМНОГО ПРОГНОЗИРОВАНИЯ РИСКОВ КАЧЕСТВА ПРОЕКТОВ СЛОЖНЫХ ПРОГРАММНЫХ СИСТЕМ

Рассмотрены модели инженерной и формализованной методик прогнозирования и сокращения рисков проектов сложных программных систем по характеристикам качества.

Теория и практика инженерии сложных программных систем (ПС) свидетельствует, что проекты ПС являются достаточно трудоемкими и сложными объектами в техническом и организационном выполнении. Жизненный цикл проектов сложных ПС содержит многочисленные факторы неопределенности и как следствие связанные с этим риски, которые часто приводят к «провалам» проектов сложных ПС или к неполному завершению проектов по запланированным (заданным) характеристикам качества ПС [1,2].

С позиций разработчиков и пользователей ПС риски проявляются как негативные последствия функционирования и применения ПС, которые способны вызвать ущерб системе, внешней среде или пользователю, в результате отклонения характеристик объектов или процессов от заданных требований заказчика, согласованных с разработчиками [2-4].

1. Модели системного прогнозирования рисков проектов сложных программных систем

Рассматриваемая модель системного анализа, прогнозирования и сокращения рисков проектов сложных ПС, основывается на результа-

тах проведенного исследования проблем управления рисками и аккумулирует здесь ряд положений, моделей, методов и стандартов, регламентирующих многие задачи управления программными проектами и в том числе управления рисками [1-5].

К настоящему времени разработано несколько моделей и стандартов для анализа и сокращения рисков в жизненном цикле ПС. Каждая из моделей имеет свои особенности, обусловленные свойствами и характеристиками объектов разработки. Модели отличаются спецификой интересов и квалификации их авторов, и охватывают широкий спектр реальных ситуаций проектирования ПС, в которых необходимо сокращение или исключение рисков проектов [1-3].

Для согласования многих подходов и моделей анализа, прогнозирования и сокращения рисков проектов ПС, предлагается модель многоэтапной процедуры выполнения процесса управления рисками качества проекта ПС, позволяющая во многом обобщить инженерные методики [1-5] и создать необходимую научную и методическую базу для формализации проектных задач по управлению рисками качества проектов сложных ПС [2,5].

Указанная модель включает следующие основные этапы анализа, прогнозирования и сокращения рисков проектов сложных программных систем:

Этап 1. Подготовка исходных данных для анализа, прогнозирования и управления рисками проекта ПС включает:

- описание системы, внешней среды и программной системы;
- определение целей, назначения и функций проекта ПС;
- разработка предварительных требований к функциональной пригодности и конструктивным характеристикам проекта ПС;
- формирование группы экспертов для анализа угроз и управления рисками.

Этап 2. Выделение, идентификация, анализ угроз и рисков ПС:

- выделение источников и угроз нарушения требований и ограничений ресурсов, определение критериев работоспособности проекта ПС;
- отбор и идентификация основных угроз и рисков проекта ПС;
- анализ причин, выделение категорий угроз и возможных последствий появления рисков функциональной пригодности проекта ПС;
- анализ причин, выделение категорий угроз и возможных последствий проявления рисков конструктивных характеристик проекта ПС;

- анализ причин, выделение категорий угроз и возможных последствий рисков ограничения доступных ресурсов проекта ПС.

Этап 3. Оценивание опасности угроз и рисков и выбор контрмер для их сокращения включает:

- оценивание возможных последствий, уровней потенциальных опасностей угроз и приоритетов категории рисков проекта ПС;
- выделение и упорядочение группы наиболее опасных, высокоприоритетных рисков проекта ПС;
- планирование методов и ресурсов реализации контрмер для сокращения опасных, приоритетных рисков проекта ПС;
- распределение ресурсов на контрмеры для сбалансированного сокращения интегрального риска проекта ПС;
- распределение ответственности специалистов за реализацию сокращения опасных рисков проекта ПС.

Этап 4. Сокращение или ликвидация опасных рисков ПС включает:

- реализация контрмер для сокращения интегрального риска и составление отчетов о состоянии проекта;
- корректировка требований к функциональной пригодности, конструктивным характеристикам и ограничениям ресурсов проекта ПС;
- регистрация результатов сокращения интегрального риска на очередном этапе проекта ПС.

Этап 5. Контроль, регистрация, мониторинг и утверждение допустимого интегрального риска ПС включает:

- контроль, отслеживание и мониторинг реализации сокращения интегрального риска по этапам проекта ПС;
- мониторинг состояния проекта и интегрального риска по этапам жизненного цикла ПС;
- документирование и утверждение допустимого интегрального риска по этапам жизненного цикла проекта ПС;
- оформление итоговых данных по результатам сокращения рисков ПС.

Согласно представленной модели в процессах анализа, идентификации, планирования, контроля, мониторинга, сокращения и документирования рисков проекта циркулирует рисковая информация, представленная как количественными, так и качественными значениями параметров проекта. Экспертные оценки и вербальные высказывания менеджеров программного проекта по ходу жизненного цикла проектов сложных ПС, отражают свойство нечеткости проектных данных для принятия управленческих решений по рискам.

Возникает объективная необходимость решения важных задач по разработке новых формализованных подходов, моделей и методов, реализующих нечеткие процессы переработки сложной рисковой информации в удобную для специалистов форму, позволяющую более качественно и оперативно решать многочисленные и важные задачи управления рисками проекта сложных программных систем.

2. Формальные модели прогнозирования и сокращения рисков проектов программных систем

Представленная выше многоэтапная модель анализа, прогнозирования и сокращения рисков проектов сложных ПС в основном основывается на известном инженерном и практическом опыте, на регламентированных процедурах управления рисками, руководствах и стандартах различных уровней [1-5] и создает основу для разработки формализованных моделей.

Степень формализации и автоматизации процедур поддержки принятия решений по рискам во многом способствует повышению результативности программных проектов. В связи с этим в работе рассматривается подход к построению формализованной методики управления рисками, основанной на использовании нечетких моделей, методов и процедур нечеткого вывода для автоматизации отдельных этапов анализа и прогнозирования рисков проектов ПС при входных данных, поступающих в виде нечетких вербальных высказываний об угрозах проекту по его этапам:

1) идентификация рисков программного проекта [6].
2) анализ рисков программного проекта [7,8].
3) планирование откликов и контрмер на риски [9].
4) мониторинг рисков программного проекта [5].

Механизмы или алгоритмы вывода являются важной частью формализованной методики и базовой архитектуры системы нечеткого вывода по рискам проекта. К настоящему времени на практике используется несколько алгоритмов получения заключений в системах нечеткого вывода. При этом проведенные исследования известных алгоритмов (Мамдани, Цукамото, Ларсена и др.), с учетом особенностей задач управления рисками программного проекта показали, что алгоритм Мамдани имеет ряд преимуществ, заключающихся в общности применяемых действий и отсутствии дополнительных требований к входным данным [8,9].

Выводы. Представленные на структурном уровне модели инженерной и формализованной методик прогнозирования и сокращения

рисков проектов сложных ПС обобщают и очерчивают круг проблем и задач, которые приходится решать специалистам и менеджерам проекта на практике для достижения желаемых характеристик качества проектов сложных программных систем посредством управления рисками. При этом практическая реализация предложенной модели процесса анализа, прогнозирования и сокращения рисков проектов сложных ПС, на основе применения процедур нечеткого вывода, расширяет возможности по формализации процессов управления рисками для тех случаев, когда входные данные процесса представлены в виде нечетких лингвистических высказываний специалистами по рискам проекта.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Фатрелл Р.Т., Шафер Д.Ф., Шафер Л.И. Управление программными проектами: достижение оптимального качества при минимальных затратах. Пер. с англ. - М.: Вильямс, 2003. - 740 с.

2.Липаев В.В. Анализ и сокращение рисков проектов сложных программных средств. - М.: СИНТЕГ, 2005. - 224 с.

3. ANSI/PMI 99-001-2004. Руководство к Своду знаний по управлению проектами (Руководство PMBOK). - 388 с.

4. ИСО/МЭК 9126-1-4: 2000-2004. Информационные технологии. Качество программных средств: Ч.1. Модель качества. Ч.2. Внешние метрики. Ч.3. Внутренние метрики. Ч.4. Метрики качества в использовании.

5. Корячко В.П., Таганов А.И. Программный метод управления рисками качества проекта информационной системы // Научно-технический журнал «Известия Белорусской инженерной академии». Выпуск 1(17)/4, 2004. С. 168 - 179.

6. Таганов А.И., Таганов Р.А. Методологические основы методов идентификации рисковых событий проекта. // Научно-технический журнал «Вестник РГРТА». Вып. № 12, 2003. С. 70 -77.

7. Таганов А.И. Применение нечетких множеств для формализации процессов анализа и идентификации важности рисков программного проекта // Научно-технический журнал. «Системы управления и информационные технологии». Москва-Воронеж. Выпуск №4.(30), 2007. С. 46-51.

8. Везенов В.И., Таганов А.И., Таганов Р.А. Применение процедуры нечеткого вывода для анализа рисков программного проекта // Научно-технический журнал. Москва-Воронеж. «Системы управления и информационные технологии». № 2(24), 2006. С. 34-39.

9. Таганов А.И., Таганов Р.А. Метод определения оптимальной альтернативы реагирования на этапе мониторинга рисков проекта. //

Научно-технический журнал «Вестник РГРТА». Вып. 11, 2003. С. 115 - 118.

А.И. ТАГАНОВ, Е.Н. ЖУРАВЛЕВА
Рязанский государственный радиотехнический университет

РАЗРАБОТКА СРЕДСТВ АВТОМАТИЧЕСКОЙ КЛАССИФИКАЦИИ РИСКОВ ПРОГРАММНОГО ПРОЕКТА МЕТОДАМИ НЕЧЕТКОЙ КЛАСТЕРИЗАЦИИ

Рассматривается решение задачи классификации рисков программного проекта методами нечеткой кластеризации.

Введение

Большой объем субъективной информации циркулирующей на этапах идентификации и анализа рисков программного проекта обуславливает необходимость проведения исследований и разработок по созданию методов и средств эффективной обработки рисковой информации для поддержки специалистов принимающих решения по управлению рисками проекта. Для решения этой трудно формализуемой задачи [1] предлагается способ, основанный на использовании методов автоматической классификации и нечеткого кластерного анализа [2-4]. Разработанные в рамках данного направления методы и алгоритмы приобретают в последнее время новое содержание в связи с исследованиями по теории возможностей [4-6]. В основе данной теории лежит нечетко-возможностная интерпретация неопределенности, что в значительной степени согласуется с исходными установками методологии анализа данных по рискам проекта [1,5].

1. Постановка задачи

Формальная постановка задачи нечеткого кластерного анализа рисков программного проекта выглядит следующим образом [1,2]. Пусть имеется определенное на этапе идентификации рисков конечное множество рисков $R = \{R_1, R_2, ..., R_n\}$, которое будем называть множеством объектов кластеризации. Также пусть имеются конечное множество характеристик программных проектов $P = \{P_1, P_2, ..., P_n\}$, на которые влияют потенциальные рисковые события и значения последствий влияния которых можно оценить.

Далее предполагаем, для каждого из объектов кластеризации (рисков проекта) некоторым образом измерены все признаки P в некоторой количественной шкале. Тем самым каждому из элементов

$R_i \in R$ поставлен в соответствие некоторый вектор $x_i = (x_1^i, x_2^i, ..., x_q^i)$, где x_j^i - количественное значение признака $p_i \in P$ для объекта данных $R_i \in R$. Для определенности будем полагать, что все значения x_j^i принимают некоторые действительные значения. Для определения векторов $x_i = (x_1^i, x_2^i, ..., x_q^i)$ применяется одна из следующих видов шкал измерений, описанных в [3]:

- шкала наименований;
- порядковая шкала;
- интервальная шкала;
- шкала отношений.

Векторы значений признаков $x_i = (x_1^i, x_2^i, ..., x_q^i)$ представим в виде так называемой матрицы данных D размерностью $(n \times q)$, каждая стока которой равна значению вектора x_i.

Далее для уточнения вида целевой функции $f(\Im(A))$ в рассмотрение вводятся некоторые дополнительные понятия. Прежде всего предполагается, что искомые нечеткие кластеры рисков представляют нечеткие множества A_k, образующие нечеткое покрытие исходного множества объектов кластеризации A=R (рисков), для которого условие нечеткого покрытия принимает следующий вид:

$$\sum_{i=1}^{c} \mu_{A_k}(R_i) = 1 \quad (\forall R_i \in R) \quad , \qquad (1)$$

где c – общее количество нечетких кластеров рисков $A_k (k \in \{2,...,c\})$, которое считается предварительно заданным.

На следующем шаге для каждого нечеткого кластера риска вводятся в рассмотрение так называемые *типичные представители* или *центры* v_k искомых нечетких кластеров рисков $A_k (k \in \{2,...,c\})$, которые рассчитываются для каждого из нечетких кластеров и по каждому из признаков по следующей формуле:

$$v_j^k = \frac{\sum_{i=1}^{n} (\mu_{A_k}(R_i))^m * x_j^i}{\sum_{i=1}^{n} (\mu_{A_k}(R_i))^m} \quad (\forall k \in \{2,...,c\}, \forall p_j \in P), \qquad (2)$$

где *m* – некоторый параметр, называемый экспоненциальным весом и равным некоторому действительному числу *(m>1)*. Каждый из центров кластеров представляет собой вектор $v_k = \{v_1^k, v_2^k, ..., v_q^k\}$ в не-

котором *q*-мерном нормированном пространстве, изоморфном R_q, т.е. $v_j^k \in R_q$, если признаки измерены в шкале отношений.

Наконец в качестве целевой функции будем рассматривать сумму квадратов взвешенных отклонений координат объектов кластеризации от центров искомых нечетких кластеров рисков:

$$f(A_k, v_j^k)\sum_{i=1}^{n}\sum_{k=1}^{c}(\mu_{A_k}(R_i))^m\sum_{j=1}^{q}(x_j^i - v_j^k)^2 \qquad (3)$$

где *m* – экспоненциальный вес нечеткой кластеризации рисков, значение которого задается в зависимости от количества элементов (мощности) множества рисков R. Чем больше рисков содержит множество *R*, тем меньше значение выбирается для m.

Задача нечеткой кластеризации рисков может быть сформулирована следующим образом: для заданных параметров матрицы данных D количества нечетких кластеров рисков $c(c \in N \text{ и } c > 1)$, параметра m определить матрицу U значений функций принадлежности рисков $R_i \in R$ нечетким кластерам рисков $A_k (k \in \{2,...,c\})$, которые доставляют минимум целевой функции (3) и удовлетворяют ограничениям (1), (2), а также дополнительным ограничениям (4) и (5):

$$\sum_{i=1}^{n}\mu_{A_k}(a_i) > 0 \quad (\forall k \in \{2,...,c\}) \qquad (4)$$

$$\mu_{A_k}(a_i) > 0 \quad (\forall k \in \{2,...,c\}, \ \forall a_i \in A) \qquad (5)$$

Условие (4) исключает появление пустых нечетких кластеров рисков в искомой нечеткой кластеризации. Последнее условие (5) имеет чисто формальные характер, поскольку непосредственно следует из определения функции принадлежности нечетких множеств. В этом случае минимизация целевой функции (3) минимизирует отклонение всех объектов кластеризации от центра нечетких кластеров пропорционально значениям функции принадлежности этих объектов соответствующим нечетким кластерам.

Поскольку целевая функция (2) не является выпуклой, а ограничения (1), (2), (4), (5) в своей совокупности формируют невыпуклое множество допустимых альтернатив, то в общем случае задача нечеткой кластеризации относится к многоэкстремальным задачам нелинейного программирования.

2. Выбор кластер-процедуры для решения задачи

Необходимые рекомендации по выбору метода нечеткого решения задачи автоматической классификации рисков программного проекта сформулированы здесь исходя из целей классификации и имеющихся содержательных установок о компактности выделяемых групп рисков по характеристикам качества программного проекта и важности рисков для этапа мониторинга [4,5]:

1) если у исследователя существуют содержательные представления об условиях объединения рисков в классы, следует выбрать группу эвристических методов нечеткого подхода в кластерном анализе;

2) если целью классификации является получение нечеткого разбиения на заранее известное число классов, исследуемой совокупности рисков, то следует выбрать группу оптимизационных методов нечеткого подхода в кластерном анализе;

3) если целью классификации является получение наглядного представления о нечеткой структуре классифицируемой совокупности рисков сравнительно небольшого объема, то следует выбрать иерархические методы нечеткого подхода в кластерном анализе.

На следующем этапе рассматриваемого способа требуемый алгоритм эвристического направления нечеткого подхода к решению задачи автоматической классификации рисков программного проекта может быть выбран в соответствии с рекомендациями [2,3,5]:

1) если число рисков исследуемой совокупности достаточно велико и на множестве рисков оказывается возможным определить нечеткое подмножество рисков, то следует обосновать выбор необходимой метрики классификации и использовать алгоритм Гитмана - Левина;

2) если целью классификации является предварительный анализ исследуемой совокупности рисков программного проекта, в процессе которого требуется получить разбиение множества рисков проекта на заданное число четких классов рисков, то следует использовать алгоритм Тамуры - Хигути - Танаки;

3) если допускается пересечение нечетких кластеров, а также имеются предположения о минимальном числе объектов (рисков) в каждом нечетком кластере, то следует обосновать выбор порога классификации и использовать алгоритм Кутюрье - Фьолео;

4) если число элементов множества рисков программного проекта сравнительно невелико, а также существуют предположения о сложной форме кластеров и требуется визуальное представление результатов классификации, то следует выбрать алгоритм классификации на нечетких графах Берштейна - Дзюбы;

5) если все риски проекта должны быть классифицированы т.е. распределены по нечетким кластерам и количество формируемых кла-

стеров неизвестно то следует выбрать горный алгоритм Ягера – Филева;

6) если же для решения задачи классификации рисков проекта выбрано оптимизационное направление, то главной проблемой оказывается обоснование вида функционала. Поскольку выбор функционала определяется, помимо формы матрицы исходных данных, вида шкалы, в которой измерены признаки, и типа признакового пространства, также спецификой конкретной задачи, то при выборе функционала качества разбиения целесообразно учитывать содержательную интерпретацию функционала [2].

С учетом представленных выше рекомендаций одним из рациональных методов решения поставленной задачи нечеткой кластеризации рисков проекта может рассматриваться алгоритм нечеткой кластеризации методом нечетких *c*-средних [2,4].

3. Алгоритм нечеткой кластеризации рисков методом нечетких *c*-средних

Алгоритм нечетких с-средних (FCM, Fuzzy C-Means) для решения задачи нечеткий кластеризации рисков в виде (1)-(5) имеет итеративный характер последовательного улучшения некоторого исходного нечеткого разбиения рисков $\Re(A) = \{A_k \mid A_k \subseteq A\}$, которое задается пользователем или формируется автоматически по некоторому эвристическому правилу. На каждой из итераций рекуррентно пересчитываются значения функций принадлежности нечетких кластеров рисков и их типичные представители [2-4].

Алгоритм FCM закончит работу в случае, когда произойдет выполнение заданного априори некоторого конечного числа итераций, либо когда минимальная абсолютная разность между значениями функции принадлежности на двух последовательных итерациях не станет меньше некоторого априори заданного значения.

Формально алгоритм FCM определяется в форме итеративного выполнения следующей последовательности шагов:

1. Предварительно необходимо задать следующие значения: количество искомых нечетких кластеров рисков $c(c \in N \text{ и } c > 1)$, максимальное количество итераций алгоритма s $(s \in N)$, параметр сходимости алгоритма $\varepsilon(\varepsilon \in R+)$, а также экспоненциальный вес расчета целевой функции и центров кластеров рисков m (как правило, m=2). В качестве текущего нечеткого разбиения рисков на первой итерации алгоритма для матрицы данных D задать некоторое исходное разбиение $\Re(A) = \{A_k \mid A_k \subseteq A\}$ на с непустых нечетких кластеров рисков, которые описываются совокупностью функций принадлежности $\mu_k(R_i)(\forall k \in \{2,...,c\}, R_i \in R)$.

2. Для исходного текущего нечеткого разбиения рисков $\Re(A) = \{A_k \mid A_k \subseteq A\}$ по формуле (2) рассчитать центры нечетких кластеров рисков $v_j^k(\forall k \in \{2,...,c\}, P_j \in P)$ и значение целевой функции $f(A_k, v_j^k)$ по формуле (3). Количество выполненных итераций положить равным 1.

3. Сформировать новое нечеткое разбиение рисков $\Re'(A) = \{A_k \mid A_k \subseteq A\}$ исходного множества рисков R на c непустых нечетких кластеров рисков, характеризуемых совокупностью функций принадлежности $\mu_k'(R_i)(\forall k \in \{2,...,c\}, R_i \in R)$, которые определяются по формуле:

$$\mu_k'(R_i) = \left(\sum_{l=1}^{c} \left(\frac{(\sum_{j=1}^{q}(x_j^i - v_j^k)^2)^{\frac{1}{2}}}{(\sum_{j=1}^{q}(x_j^i - v_j^l)^2)^{\frac{1}{2}}} \right)^{\frac{2}{m-1}} \right)^{-1} \quad (\forall k \in \{2,...,c\}, R_i \in R).$$

4. При этом, если для некоторого $k \in \{2,...,c\}$ и некоторого $R_i \in R$ значение $\sum_{j=1}^{q}(x_j^i - v_j^k)^2 = 0$, то для соответствующего нечеткого кластера рисков A_k полагаем $\mu_k'(R_i) = 1$, а для остальных $A_l(\forall l \in \{2,...,c\}, l \neq k)$ полагаем $\mu_l'(R_i) = 0$. Если же таких $k \in \{2,...,c\}$ для некоторого $R_i \in R$ окажется несколько, т.е. для них значение $\sum_{j=1}^{q}(x_j^i - v_j^k)^2 = 0$, то эвристически для меньшего из k полагаем $\mu_k'(R_i) = 1$, а для остальных $l \in \{2,...,c\}, l \neq k$ полагаем $\mu_l'(R_i) = 0$.

5. Для нового нечеткого разбиения рисков $\Re'(A) = \{A_k \mid A_k \subseteq A\}$ по формуле (2) рассчитать центры нечетких кластеров рисков $v_j^k(\forall k \in \{2,...,c\}, P_j \in P)$ и значение целевой функции $f'(A_k, v_j^k)$ по формуле (3).

6. Если количество выполненных итераций превышает заданное число s или модуль разности $\left| f(A_k, v_j^k) - f'(A_k, v_j^k) \right| \leq \varepsilon$, т.е. не пре-

вышает значения параметра сходимости алгоритма ε, то в качестве искомого результата нечеткой кластеризации принять нечеткое разбиение рисков $\Re^{/}(A) = \{A_k \mid A_k \subseteq A\}$ и закончить выполнение алгоритма. В противном случае считать текущим нечетким разбиением рисков $\Re(A) = \Re^{/}(A)$ и перейти на шаг 3 алгоритма, увеличив на 1 количество выполненных итераций.

Алгоритм FCM по своему характеру относится к приближенным алгоритмам поиска экстремума целевой функции (3) при наличии ограничений (1), (2), (4), (5). Поэтому в результате выполнения данного алгоритма, определяется локально-оптимальное разбиение рисков $\Re^{*}(A)$, которое описывается совокупностью функций принадлежности $\mu_k(R_i)(\forall k \in \{2,...,c\}, R_i \in R)$, а также центры или типичные представители каждого из нечетких кластеров $v_j^k(\forall k \in \{2,...,c\}, P_j \in P)$.

Для получения более адекватных результатов необходимо многократно выполнять алгоритм FCM для различных исходных разбиений и, если неизвестно количество нечетких кластеров, для различных значений $c(c \in N \text{ и } c > 1)$. Полученные результаты для одинаковых значений *c* сравниваются по значениям целевой функции полученных нечетких разбиений с целью принятия окончательного решения об искомой нечеткой кластеризации.

Таким образом, предложенная постановка задачи нечеткой кластеризации рисков проекта и рассмотренный метод ее решения позволяют построить автоматизированную процедуру определения квазиоптимального состава рисков программного проекта для этапа мониторинга рисков проекта. Включение такой процедуры в состав автоматизированного процесса управления программными проектами значительно повышает эффективность процесса управления рисками по характеристикам качества [4].

5. Программная процедура определения квазиоптимального состава рисков проекта для этапа мониторинга

Программно реализованная процедура определения квазиоптимального состава рисков программного проекта, представленая на рисунке, и содержит два этапа анализа и автоматической классификации рисков проекта [5,6]:

1) ввод исходных данных, содержащих экспертные оценки влияния потенциальных рисков на характеристики проекта;

2) классификация рисков по характеристикам качества проекта: функциональность, надежность, удобство применения, эффективность, удобство сопровождения, переносимость;

3) классификация рисков проекта в пределах каждой группы по важности рисков;

4) с учетом ресурсных ограничений проекта формирование окончательного квазиоптимального состава рисков для этапа мониторинга рисков проекта.

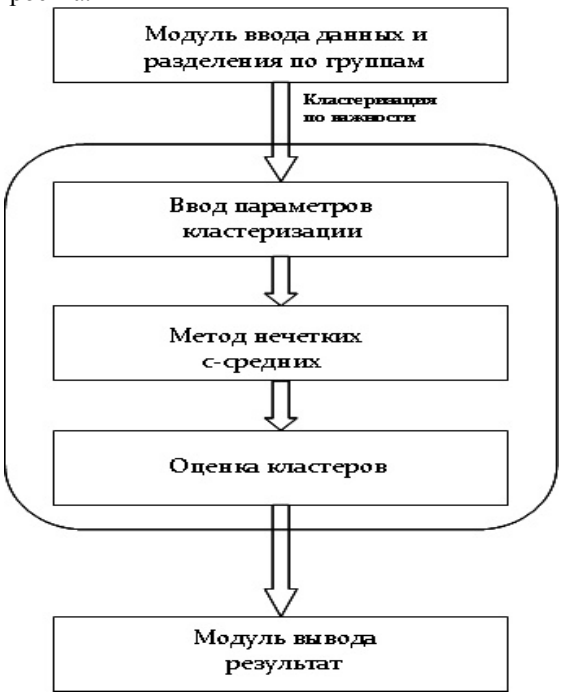

Рис.1. Процедура классификации рисков проекта

Заключение

Представленные в работе результаты исследований по формализации и решению задачи автоматической классификации и оптимизации состава контролируемых рисков проекта на этапе анализа рисков для этапа мониторинга, позволяют существенно повысить эффективность процесса управления рисками проекта в условиях нечеткости исходных проектных данных.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Корячко В.П., Таганов А.И. Программный метод управления рисками качества проекта информационной системы // Научно-технический журнал «Известия Белорусской инженерной академии». Выпуск 1(17)/4, 2004. - С. 168-179.

2. Леоненков А. Нечеткое моделирование в среде MATLAB и fuzzyTECH. – СПб.: БХВ-Петербург, 2003. – 736 с.

3. Демидова Л.А., Кираковский В.В., Коняева Е.И. Классификация объектов жилого фонда на основе FCM-алгоритма и генетического алгоритма // Математическое и программное обеспечение вычислительных систем: межвуз. сб. науч. тр. / под ред. А.Н. Пылькина. – М.: Горячая линия – Телеком, 2008. – С. 21-32.

4. Таганов Р.А., Таганов А.И. Метод нечеткой кластеризации рисков для формализации анализа рисков программного проекта // Материалы III Международного научно-практического семинара «Интегрированные модели и мягкие вычисления в искусственном интеллекте». Коломна, 2005. - http:imscal.rk9.ru/2005.

5. Таганов А.И. Способ снижения размерности задачи анализа рисков программного проекта методами нечеткой классификации. // Современные проблемы информатизации в моделировании и социальных технологиях: Сб. трудов. Вып. 15/ Под ред. д.т.н., проф. О.Я.Кравца. – Воронеж: «Научная книга», 2010. - С. 290 – 291.

6. Таганов А.И., Таганов Р.А., Суворов А.В. Классификация рисков проекта методами нечеткого кластерного анализа // Материалы 15-й Международной науч. техн. конф. «Проблемы передачи и обработки информации в сетях и системах телекоммуникаций». Часть 2. Рязань: РГРТУ, 2008. - С. 22-24.

И.А.ТЕЛКОВ
Рязанский государственный радиотехнический университет

МАТЕМАТИЧЕСКАЯ МОДЕЛЬ МНОГОЯДЕРНЫХ ПРОЦЕССОРОВ В ПРОСТРАНСТВЕ СОСТОЯНИЙ

Предлагается тензорная модель современных многоядерных процессоров Intel и AMD, а также графических процессоров NVIDIA и ATI. Модели ориентированы на использование в процессе распараллеливания вычислений и его оптимизации.

Развитие современных технологий позволило значительно ускорить вычисления на современных компьютерах за счет выпуска многоядерных процессоров. С одной стороны, появились многоядерные процессоры фирм *Intel* и *AMD*, с другой стороны – новые графические процессоры (*GPU – Graphics Processing Unit*) фирм *NVIDIA* и *ATI*. Если в первом случае имеют место 2-4 процессорных ядра (планируется выпуск до 8-12 ядер), то во втором случае количество процессоров уже

достигает 240 (*GPU NVIDIA GeForce 280*) и 3200 (*GPU ATI Radeon 5970*) процессоров.

К сожалению, на настоящий момент отсутствуют универсальные средства распараллеливания вычислений для подобных систем. Имеются лишь специализированные (фирменные) системы параллельного программирования (например, система *CUDA* для программирования *GPU NVIDIA*). Поэтому задача создания универсальных средств параллельного программирования становится актуальной.

Для моделирования подобных сложных систем в [1] предложена тензорная модель, как наиболее универсальная и соответствующая возникающим в ходе проектирования задачам [2, 3]. Начальное проектирование при этом осуществляется в фазовом пространстве, определяющее множество состояний вычислительной системы, называемым *пространством состояний* (ПС) системы [1].

При описании функционирования системы в ПС учитываются внешние потоки данных и команд (внутренняя организация системы при этом не принимается в расчет). Формализация внешних информационных потоков в ПС соответствует классификации вычислительных систем (ВС) по Флинну [4]:

- ОКОД - ВС с одиночными потоками команд и данных;
- ОКМД - ВС с одиночным потоком команд и множественным потоком данных;
- МКОД - ВС с множественным потоком команд и одиночным потоком данных;
- МКМД - ВС с множественными потоками команд и данных.

Данная классификация не раскрывает внутреннюю организацию ВС, но представляет ценность для оценки потенциальных возможностей организации параллельного вычислительного процесса.

Представим описание информационных потоков в современных компьютерах в тензорной форме. Будем различать скалярные и многомерные (векторные, матричные и пр.) потоки данных, обрабатываемых унарными, бинарными и ***n***-арными командами, а также скалярные и многомерные потоки команд.

Информационный поток вида $<(x_i , t_i)>$, где x_i - порции информации, а t_i - моменты их поступления в систему, в тензорной форме записывается как ГО следующего вида:

$$e^{xt}.$$

Простейшие одномерные потоки команд $U = <(u_i , t_i)>$ и данных $D=<(d_i , t_i)>$ в тензорной форме могут быть представлены в виде суммарного потока - композиции потоков команд и данных (только в том

случае, если поступление данных в систему синхронизировано с поступлением команд управления):

$$e^{ut}\, e^{dt} = e^{udt}. \qquad (1)$$

При помощи подобной формы записи можно описывать потоки любой сложности. Например, поступление в очередной момент времени t_i информационного вектора ($x_{i1}, x_{i2}, \dots, x_{iq}$), то есть потока вида - $<((x_{i1}, x_{i2}, \dots, x_{iq}), t_i)>$, определяется тензором потока:

$$e^{t^x}.$$

Следующие тензоры:

$$e^{e^{t\,x}}, \quad e^{e^{t^x}}$$

описывают соответственно двумерный информационный поток, поступающий скалярными порциями (x_{ij}, t_{ij}), и трехмерный поток x_{ijk} поступающий векторами $((x_{ijk}, \dots, x_{ijp}), t_{ij})$.

Распараллеливание входного потока данных при многоядерных вычислениях опишем при помощи тензора распараллеливания π. Так для процессора *Intel Core 2 Duo*, содержащим два ядра, этот процесс описывается следующим тензорным уравнением:

$$e^{dt} \circ \pi_{dt}^{abt} = e^{at} e^{bt} = e^{abt},$$

где индексы a и b соответствуют параллельным потокам данных $<(a_i, t_i)>$ и $<(b_i, t_i)>$, направляемым на каждое из вычислительных ядер процессора *Intel Core 2 Duo*. Аналогичные уравнения можно составить и для четырехядерных процессоров (например, *Intel Core 2 Quad, Intel Core i5/i7, AMD Phenom II*).

Процесс обработки информационных потоков в ВС можно описать при помощи тензорного уравнения. Например, обработке простейшего потока (1) соответствует уравнение:

$$e^{udt} \circ A_{udt}^{xt} = e^{xt},$$

где A - тензор, определяющий преобразования, выполняемые ВС с архитектурой типа ОКОД. При этом ковариантные индексы тензора соответствуют входной, а контравариантные - выходной информации [1]. В правой части уравнения находится описание выходного информационного потока. В данном случае он представляет собой простейший поток вида $<(x_i, t_i)>$.

Архитектуры ОКМД, МКОД и МКМД по Флинну описываются следующими уравнениями:

$$e^{ut^d} \circ A_{ut_d}^{xt} = e^{xt};$$

$$e^{dt^u} \circ A_{dt_u}^{xt} = e^{xt};$$

$$e^{t^{ud}} \circ A_{t_{ud}}^{xt} = e^{xt}.$$

При помощи тензорных уравнений можно описывать архитектуры ВС, которые не вписываются в рамки существующих классификаций. Например, следующее тензорное уравнение:

$$e^{t^{u^d}} \circ A_{t_{u_d}}^{xt} = e^{xt}$$

описывает архитектуру МКМД, у которой входной поток команд представляет собой вектор управления (команды, входящие в вектор, поступают одновременно), а поток данных - двумерный (в каждый момент времени на вход системы поступает матрица данных. Выходные данные образуют поток скалярных величин $<(x_i, t_i)>$. Уравнение вида

$$e^{e^t\, u^d} \circ A_{e_t\, u_d}^{t^x} = e^{t^x}$$

также описывает архитектуру, относящуюся по Флинну к классу МКМД, но при этом входной поток команд - векторный, но поступление команд, входящих в вектор управления, не синхронизировано. Поток входных данных данной архитектуры - двумерный $<((x_{ijk}, \dots, x_{ijp}), t_{ij})>$, а выходных - векторный $<((x_{i1}, x_{i2}, \dots, x_{iq}), t_i)>$.

Аналогичным образом можно описать процесс обработки информационных потоков в двуядерных процессорах *Intel Core 2 Quad*:

$$e^{u\,dt} \circ \pi_{ut}^{wvt} \circ \pi_{dt}^{abt} \circ A_{w\,at}^{xt} \circ B_{vbt}^{yt} = e^{xyt}, \qquad (2)$$

где e^{udt} – входной поток команд $<(u_i, t_i)>$ и данных $<(d_i, t_i)>$; π_{ut}^{wvt} - тензор распараллеливания входного потока команд на два потока $<(w_i, t_i)>$ и $<(v_i, t_i)>$ для двух ядер; π_{dt}^{abt} - тензор распараллеливания входного потока данных на два потока $<(a_i, t_i)>$ и $<(b_i, t_i)>$ для двух ядер; соответственно два тензора A_{wat}^{xt} и B_{vbt}^{yt} описывают работу двух ядер; e^{xyt} – выходные потоки двух ядер $<(x_i, t_i)>$ и $<(y_i, t_i)>$.

Уравнение (2) описывает синхронизированную систему, так как в ней используется один поток времени $<t_i>$ для обоих ядер процессора. Если работа ядер не синхронизирована, то в модели необходимо

создать два временных потока, например, $< \tilde{t}_i >$ и $< \hat{t}_j >$. При этом уравнение (2) принимает вид:

$$e^{udt} \circ \pi_{ut}^{wv\tilde{t}} \circ \pi_{dt}^{ab\hat{t}} \circ A_{wat}^{x\tilde{t}} \circ B_{vbt}^{y\hat{t}} = e^{x\tilde{t}} \circ e^{y\hat{t}}, \qquad (3)$$

где e^{udt} – входной (внешний) общий для двух ядер поток команд и данных, а остальные тензоры привязаны к раздельным для ядер временным потокам $e^{\tilde{t}}$ и $e^{\hat{t}}$, включая и потоки выходных результатов работы ядер $e^{x\tilde{t}}$ и $e^{y\hat{t}}$.

Вычислительные процессы, происходящие в многошейдерных графических процессорах (*GPU NVIDIA GeForce* и *GPU ATI Radeon*), отличаются тем, что множество процессорных ядер (шейдеров) управляется единым потоком команд (архитектура ОКМД). При этом единый поток входных данных преобразуется в векторный поток, в котором каждый компонент вектора назначается на свой шейдер. В тензорной форме записи такие потоки команд и данных имеют следующий вид:

$$e^{ut} \circ e^{t^a} = e^{ut^a},$$

где e^{ut} – одномерный поток команд $<(u_i, t_i)>$, а e^{t^a} - векторный поток данных $<((a_{i1}, a_{i2}, \ldots, a_{in}), t_i)>$ для n шейдеров *GPU*, который получается путем распараллеливания одномерного потока входных данных $<(d_i, t_i)>$:

$$e^{dt} \circ \pi_{dt}^{t^a} = e^{t^a}.$$

Таким образом, тензорное описание работы современных *GPU* имеет следующий вид:

$$e^{ut} \circ e^{dt} \circ \pi_{dt}^{t^a} \circ A_{ut_a}^{t^x} = e^{t^x},$$

где тензор $A_{ut_a}^{t^x}$ описывает преобразования, выполняемые в многошейдерном устройстве *GPU*: векторный поток данных $<((a_{i1}, a_{i2}, \ldots, a_{in}), t_i)>$ под управлением одномерного потока команд $<(u_i, t_i)>$ формирует выходной векторный поток результатов $<((x_{i1}, x_{i2}, \ldots, x_{in}), t_i)>$.

Для более детального описания процесса вычислений в многоядерных процессорах, позволяющих учитывать, например, особенности реализации архитектур *Intel Nehalem* ядер *Bloomfield* и *Lynnfield*, использующихся в четырехядерных процессорах *Intel Core i5* и *Intel Core i7*, а также особенности «облачной» архитектуры *Intel SCC*, необходимо раскрыть геометрические преобразования потоков информации, скрытые в тензорах преобразований A и B. Для этого следует из пространства состояний перейти в пространство устройств [1].

Предложенная тензорная модель предназначена для использования при проектировании параллельных процессов в современных многоядерных процессорах и графических ускорителях, позволяющих использовать доступ к системе универсальных шейдеров.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Корячко В.П., Скворцов С.В., Телков И.А. Архитектура многопроцессорных систем и параллельные вычисления: Учеб. пособие.- М.: Высшая школа, 1999.- 200 с.

2. Арменский А.Е. Тензорные методы построения информационных систем. М.: Наука, 1989. 152 с.

3. Петров А.Е. Тензорная методология в теории систем. М.: Радио и связь, 1985. 152 с.

4. Майерс Г. Архитектура современных ЭВМ: В 2-х кн. М.: Мир, 1985. Кн.1. 364 с; Кн.2. 312 с.

А.П. ШИБАНОВ, Д.А. РЯБОВ
Рязанский государственный радиотехнический университет

МОДЕЛЬ КАНАЛОВ БАЗОВОЙ ОПОРНОЙ СЕТИ РЕГИОНА

Рассматривается GERT-модель телекоммуникационного тракта из агрегированных каналов для расчета показателей качества широкополосной базовой опорной сети региона.

Введение. При проектировании телекоммуникаций на стадии эскизного проекта решается задача предварительного определения наиболее важных параметров сети. Наилучшее решение ищется для множества вариантов структуры сети. Оптимальный вариант находится с использованием целевой функции и наиболее важных ограничений на переменные задачи. При поиске наилучшей структуры сети определяется число параллельно соединенных виртуальных каналов в каждом физическом канале между двумя коммуникационными устройствами при заданной нагрузке на сеть. При этом должны учитываться не только надежность оборудования, время восстановления после отказа, достоверность передачи информации, среднее время передачи, но и вариация (“джиттер”) среднего времени передачи, а также стоимостные характеристики. Например, в коммутирующих маршрутизаторах LSR (Label Switching Routers) высокоскоростных сетей на

основе коммутации меток MPLS (Multi Protocol Label Switching) при "конструировании" трафика применяют алгоритмы распределения потоков по виртуальным каналам, основанные на учете средней скорости передачи пакетов для каждого потока. Необходимость контроля "джиттера" вызывается тем, что при передаче синхронного трафика – речи, видео, аудио информации, информации с видеокамер и т.п. на этот параметр вводятся жесткие ограничения во избежание ухудшения качества передачи.

Весьма часто сети MPLS "надстраиваются" над сетями синхронной цифровой иерархии SDH (Synchronous Digital Hierarchy). При этом реализуется параллельная передача информационных пакетов в сети SDH. Один поток образуется путем создания "связки" из трибутарных и агрегативных виртуальных контейнеров. В каждой такой связке передается набор емкостей каналов T1/E1, T2/E2, T3/E3, T4/E4. В тракте SDH передается параллельно несколько потоков. Буферная память портов коммутаторов рассчитана в этом случае не на прием отдельных пакетов, а на хранение набора виртуальных контейнеров VC-11 (1,5 Мбит/с), VC-12 (2 Мбит/с), VC-2 (6 Мбит/с), VC-3 (34/45 Мбит/с) и VC-4 (140 Мбит/с). Виртуальные контейнеры являются единицей коммутации мультиплексоров SDH.

При проектировании как сетей MPLS, так и сетей SDH важно оценить число логических (виртуальных) соединений для создания агрегированных (транкинговых) каналов, необходимых как для передачи средних объемов передаваемого трафика пользователей, так и возможных его пульсаций.

Постановка задачи. В первичной сети выполняется передача сообщений пользователей, состоящих из заданного числа пакетов n. Перед началом передачи выполняется проверка условия готовности коммутационного устройства (КУ) к передаче. Вероятность готовности устройства равна p. Если устройство не готово, то оно восстанавливается за случайное время, распределенное по экспоненциальному закону. Пакеты в одном звене (КУ – линия связи – КУ) могут передаваться по одному или нескольким параллельно работающим, логически далее неделимым каналам. Назовем эти каналы первичными. Это могут быть как виртуальные каналы в MPLS, так и тракты, по которым передается информация групп административных блоков SDH. Время передачи пакета по виртуальному каналу принимается распределенным по экспоненциальному закону. На выход звена, состоящего из параллельных виртуальных каналов, пакеты приходят в разное время. Потом они синхронизируются в КУ и на вход следующего звена по-

ступают одновременно (задержка синхронизации является пренебрежимо малой).

Должны быть рассчитаны: среднее время передачи пакетов и его дисперсия, распределение времени передачи, как в отдельных звеньях, так и на всем тракте от одного пограничного КУ до другого. Проектировщик сети должен иметь возможность сравнения проектных вариантов, в которых используется разное число элементарных виртуальных каналов с одинаковыми или разными характеристиками.

Рассматриваются два основных варианта соединения КУ: 1) в виде цепи, 2) в виде кольца. Передача пакетов в этих случаях поясняется на рис. 1.

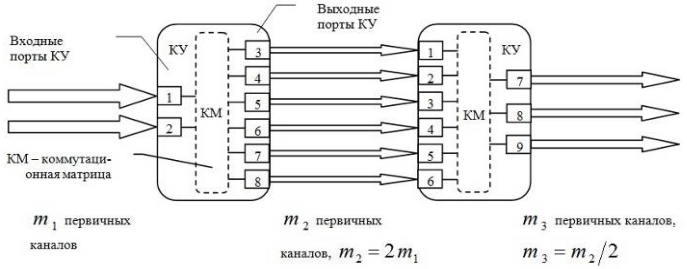

Рис. 1. Передача пакетов через КУ

Составление модели процесса передачи. GERT-сеть процесса передачи по одному звену канала связи изображена на рис. 2.

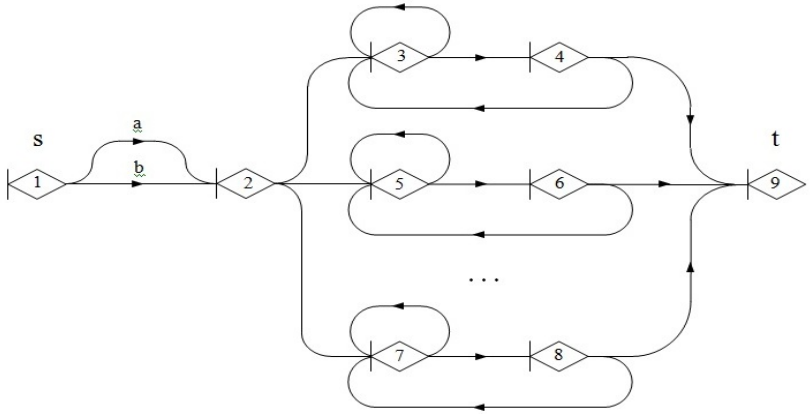

Рис. 2. Модель передачи пакетов через одно звено

Ветвь $(1,2)_b$ соответствует переходу на начало процесса передачи пакетов. Считаем, что время распознавания сигнала готовности КУ очень мало, и им можно пренебречь. Ветвь $(1,1)_a$ отражает время восстановления КУ в случае его неготовности. Предполагается, что непосредственно в процессе передачи сообщения устройство работоспособно. Ветви (2,3), (2,5), …, (2,7) отражают факт параллельной идентичной пересылки пакета по m_1 каналам первого звена. Поэтому вероятность выбора каждой из этих ветвей равна 1 (вероятности выбора этих ветвей не складываются). Производящие функции моментов времени прохождения ветвей (2,3), (2,5), … (2,7) равны 1. Эти ветви выполняют логическую функцию распараллеливания пакетов в КУ. Время выполнения данной операции ничтожно мало по сравнению со временем передачи пакета через одно звено. Ветви (3,4), (5,6), …, (7,8) отражают время передачи пакета и доставку положительного подтверждения правильности доставки пакета, а ветви (3,3), (5,5), …, (7,7) – время передачи пакета и доставку на передающую сторону отрицательной квитанции. Ветви (4,9), (6,9), …,(8,9), отражают выход из цикла передачи пакетов и логическую функцию сборки пакетов во входных портах следующего КУ. Фиксируется факт прихода последнего из всего множества передаваемых параллельно пакетов.

Вероятность выхода из цикла принята равной $1/n$. Данное предположение введено для упрощения анализа. Оно увеличивает погрешность расчетов распределения. Но вычисление математических ожиданий величин при последовательной свертке распределений выполняется точно [1]. Некоторое увеличение погрешности расчетов оправдано тем, что производится предварительная оценка основных параметров системы, в том числе и среднеквадратического отклонения времени передачи. При решении оптимизационной задачи итерационными методами приходится многократно использовать целевую функцию в виде несложных математических выражений (в частности, рассчитывать величину интервала “трех сигм”). Уточнение вида распределения возможно либо через вычеты, либо на основе численного метода с использованием формулы обращения [2]. При этом мы получаем более точные оценки вероятности попадания времени передачи пакетов в заданный интервал. Эти оценки производятся после нахождения решения близкого к оптимальному. Если найденное распределение удовлетворяет проектировщика сети (в первую очередь, по “джиттеру”), то проектное решение считается окончательным.

В ряде случаев введенное условие выхода из цикла согласуется с логикой работы протокола передачи. Например, среди группы передаваемых по одному виртуальному каналу пакетов могут быть такие, после успешного приема которых передача всей этой группы заканчивается. Это может быть при передаче однотипной сжатой информации, при обнаружении управляющего пакета среди информационных пакетов и т.п.

Характеристики ветвей модели представлены в табл. 1.

Табл.1. Характеристики ветвей модели

№ п/п	Ветвь	*W*-функция	*P*	*M*
1	$(1,2)_a$	W_{12a}	$1-q$	$\mu/(\mu-s)$
2	$(1,2)_b$	W_{12b}	q	1
3	(2,3), (2,5), … ,(2,7)	$W_{23}, W_{25}, \dots, W_{27}$	1	1
4	(3,4), (5,6), … ,(7,8)	$W_{34}, W_{56}, \dots, W_{78}$	p	$\lambda/(\lambda-s)$
5	(3,3), (5,5), … ,(7,7)	$W_{33}, W_{55}, \dots, W_{77}$	$1-p$	$\lambda/(\lambda-s)$
6	(4,3), (6,5), … ,(8,7)	$W_{43}, W_{65}, \dots, W_{87}$	$1-(1/n)$	1
7	(4,9), (6,9), … ,(8,9)	$W_{49}, W_{69}, \dots, W_{89}$	$1/n$	1

Эквивалентная W-функция времени передачи пакета по одному параллельному каналу равна:

$$W_E(s)=W_{\text{I}}\,W_{\text{II}}=(W_{12a}+W_{12b})\frac{W_{23}\,W_{34}W_{49}}{1-W_{33}-W_{34}W_{43}}=$$

$$=\left[(1-q)\frac{\mu}{\mu-s}+q\right]\cdot\frac{\frac{1}{n}\cdot p\frac{\lambda}{\lambda-s}}{1-(1-p)\frac{\lambda}{\lambda-s}-p\left(1-\frac{1}{n}\right)\frac{\lambda}{\lambda-s}}=\frac{\mu-qs}{\mu-s}\cdot\frac{\frac{p\lambda}{n}}{\frac{p\lambda}{n}-s}.$$

Величина $W_{\text{II}}=(p\lambda/n)\Big/\left(\frac{p\lambda}{n}-s\right)$ есть производящая функция моментов экспоненциального распределения с параметром $p\lambda/n$.

Если передача сообщения ведется по m параллельным каналам, то время передачи всего сообщения определяется моментом окончания передачи последнего пакета и характеризуется законом распределения максимальной из нескольких случайных величин

$$G(z) = P\{Z < z\} = \prod_{i=1}^{m} F_i(z),$$

где $F_i(z) = \int_{-\infty}^{\infty} f_i(t_i)dt_i, i = 1,2, \ldots, m;\quad f_i(t) = p_i \lambda_i e^{-p_i \lambda_i t}$.

Для виртуальных каналов, характеризующихся экспоненциальным распределением с одинаковыми параметрами выходной случайной величины, получаем функцию распределения $F(t) = \prod_{i=1}^{m}\left[1 - \exp\left(-\lambda_i p_i t / n_i\right)\right] = \left[1 - \exp\left(-\lambda_i p_i t / n_i\right)\right]^m$. Тогда плотность распределения времени передачи сообщения между двумя КУ равна

$$f(t) = \frac{dF(t)}{dt} = \left(m\lambda_i p_i / n_i\right)\left[1 - \exp\left(-\lambda_i p_i t / n_i\right)\right]^{m-1} \exp\left(-\lambda_i p_i t / n_i\right)$$

.

Структуры опорной сети типа “цепь” представлена на рис. 3.

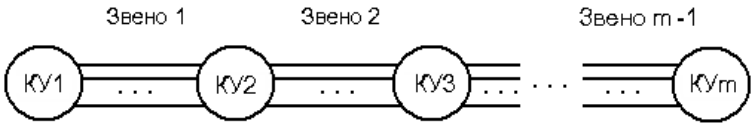

Рис. 3. Опорная сеть типа “цепь”

Когда все эти пакеты проходят первое звено LSP, они равномерно распределяются для передачи по виртуальным каналам второго звена и т.д.

Для построения более сложных GERT-сетей необходимо знание эквивалентных производящих функций моментов времени передачи сообщения по трактам КУ1 – КУ2, КУ2 – КУ3, … .

Зная $f(t)$, находим производящую функцию моментов $M_{\text{агр}}(s)$ случайного времени передачи сообщения по агрегированному каналу:

$$M_{\text{агр}}(s) = \int_0^{\infty} \exp(i\zeta t) f(t) dt =$$

$$= \left(m\lambda_i p_i / n_i\right) \int_0^{\infty} \exp(i\zeta t)\left[1 - \exp\left(-\lambda_i p_i t / n_i\right)\right]^{m-1} \exp\left(-\lambda_i p_i t / n_i\right) dt.$$

Для любого заданного значения m формула для $M_{\text{агр}}(s)$ находится путем разложения подинтегрального выражения в ряд с нахождением коэффициентов слагаемых по треугольнику Паскаля.

Пример. Рассмотрим пример, в котором в первом звене используются 2 виртуальных канала, а во втором – 4 виртуальных канала.

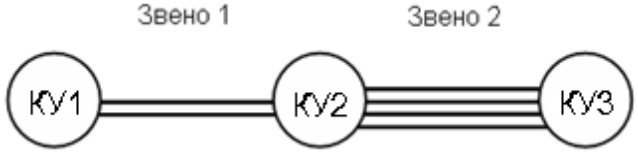

Рис. 4. Сеть из двух звеньев

Время опроса индикатора готовности канала с учетом возможного его восстановления в случае отказа характеризуется производящей функцией моментов $W_1 = \frac{q(\mu - s)}{q\mu - s}$. В тех случаях, когда рассматривается канал, составленный из нескольких звеньев, считаем, что по сравнению со временем передачи пакетов время опроса индикаторов готовности отдельных каналов пренебрежимо мало. Время восстановления относится ко всему сетевому тракту.

Агрегированный канал первого звена составлен из двух первичных каналов с одинаковыми параметрами распределения $\alpha = \lambda_1 p_1 / n_1$, а агрегированный канал второго звена – из четырех первичных каналов с параметрами распределения $\beta = \lambda_2 p_2 / n_2$ (рис. 4). Из принципа равномерного перераспределения пакетов по виртуальным каналам звена должно соблюдаться $n_1 = 2n_2$.

Время передачи пакетов через агрегированный канал КУ1-КУ2 характеризуется функцией распределения

$$F_{12}(t) = \left(1 - e^{-\alpha t}\right)^2 = 1 - 2e^{-\alpha t} + e^{-2\alpha t}.$$

Для канала LSR2-LSR3 получаем:

$$F_{23}(t)=\left(1-e^{-\beta t}\right)^4=1-4e^{-\beta t}+6e^{-2\beta t}-4e^{-3\beta t}+e^{-4\beta t}.$$

Плотность распределения вероятностей времени передачи пакетов через агрегированный канал КУ1-КУ2

$$f_{12}(t)=\frac{dF_{12}(t)}{dt}=2\alpha e^{-\alpha t}-2\alpha e^{-2\alpha t},$$

а через канал LSR2-LSR3

$$f_{23}(t)=\frac{dF_{23}(t)}{dt}=4\beta e^{-\alpha t}-12\beta e^{-2\alpha t}+12\beta e^{-3\alpha t}-4\beta e^{-4\alpha t}.$$

Производящая функция моментов времени передачи пакетов по агрегированному каналу КУ1-КУ2:

$$M_{12}(s)=\int_0^{\infty} e^{st}f_{12}(t)\,dt=\frac{2\alpha^2}{(\alpha-s)(2\alpha-s)}.$$

Производящая функция моментов времени передачи пакетов по агрегированному каналу КУ2-КУ3:

$$M_{23}(s)=\int_0^{x} e^{st}f_{23}(t)\,dt=\frac{24\beta^4}{(\beta-s)(2\beta-s)(3\beta-s)(4\beta-s)}.$$

Тогда эквивалентная производящая функция моментов времени передачи пакетов от КУ1 до КУ3 с учетом проверки готовности тракта

$$M_{13}(s)=\frac{48\alpha^2\beta^4(\mu-qs)}{(\mu-s)(\alpha-s)(2\alpha-s)(\beta-s)(2\beta-s)(3\beta-s)(4\beta-s)}.$$

Для оперативной оценки характеристик быстродействия канала нужно вычислить математическое ожидание и дисперсию времени передачи через составной канал. Математическое ожидание времени передачи пакетов по звену КУ1-КУ2 определяется по формуле

$\bar{t}_{12}(s)=\dfrac{d}{ds}\left[\dfrac{2\alpha^2}{(\alpha-s)(2\alpha-s)}\right]\Bigg|_{s=0}$. Данное выражение простое, и можно взять производную от дроби. Но в общем случае лучше предварительно прологарифмировать левую и правую часть равенства:

$$\ln M_{12}(s)=\ln\left[\frac{2\alpha^2}{(\alpha-s)(2\alpha-s)}\right]=\ln 2\alpha^2-\ln(\alpha-s)-\ln(2\alpha-s).$$

После дифференцирования имеем

$$\frac{\dfrac{dM_{12}(s)}{ds}}{M_{12}(s)}=-\ln(\alpha-s)-\ln(2\alpha-s)=\frac{1}{\alpha-s}+\frac{1}{2\alpha-s}.$$ Тогда

$$\bar{t}_{12}=\frac{dM_{12}(s)}{ds}\Bigg|_{s=0}=M_{12}(s)\Big|_{s=0}\left(\frac{1}{\alpha-s}+\frac{1}{2\alpha-s}\right)\Bigg|_{s=0}=\frac{3}{2\alpha}.$$

Аналогично канала КУ2-КУ3 $\bar{t}_{23}=25/12\beta$. Тогда математическое ожидание времени прохождения тракта КУ1-КУ3 равно $\bar{t}_{12}=\dfrac{3}{2\alpha}+\dfrac{25}{12\beta}$.

Выражения для дисперсий времени прохождения агрегированных каналов КУ1 и КУ2 равны $\sigma_{12}^2=13/4\alpha^2$, $\sigma_{23}^2=205/144\beta^2$. Так как случайные величины, характеризующие время прохождения агрегированных каналов КУ1-КУ2 и КУ2-КУ3, независимые, то для их суммы дисперсии складываются $\sigma_{13}^2=\sigma_{12}^2+\sigma_{23}^2=\dfrac{13}{4\alpha^2}+\dfrac{205}{144\beta^2}$.

Плотность распределения времени передачи пакетов через тракт КУ1-КУ3 можно найти через вычеты. После выполнения комплексного преобразования $z=-s$ имеем

$$\acute{r}i_{13}(z)=\frac{48\alpha^2\beta^4(\mu+qz)}{(\mu+z)(\alpha+z)(2\alpha+z)(\beta+z)(2\beta+z)(3\beta+z)(4\beta+z)}.$$

Плотность распределения времени передачи сообщения через тракт КУ1-КУ3 находится интегрированием по контуру Бромвича через сумму вычетов по всем особым точкам. Если функция $\acute{r}i(z)$ в полуплоскости $\mathrm{Re}\,z<0$ удовлетворяет условиям леммы Жордана, то интеграл, взятый вдоль контура Бромвича, равен сумме вычетов функции $\acute{r}i(z)$ относительно всех ее особенностей:

$$f(t)=\frac{1}{2\acute{r}\Phi}\int_{-i\infty}^{i\infty}e^{zt}\acute{r}i(z)dz=\sum_{k=1}^{n}\operatorname{Res}_{z=z_k}\left[e^{zt}\acute{r}i(z)\right].$$

Для того, чтобы выполнялись условия леммы Жордана, необходимо, чтобы в левой полуплоскости функция $\acute{r}i(z)$ была аналитической за исключением конечного числа полюсов, и равномерно относительно $\arg z$ стремилась к нулю при $|z|\to\infty$.

Функция $\acute{r}i_{13}(z)$ может иметь простые полюсы во всех особых точках $z_1 = \mu$, $z_2 = -\alpha$, $z_3 = -2\alpha$, $z_4 = -\beta$, $z_5 = -2\beta$, $z_6 = -3\beta$, $z_7 = -4\beta$ или. Для простых полюсов функции $\acute{r}i(z)$ вычет от функции $\exp(zx)\acute{r}i(z)$ можно представить в виде

$$\operatorname*{Res}_{z=z_i}\left[e^{zt}\Phi(z)\right]=\frac{\eta(z)}{\psi'(z)}.$$

Если в данной точке имеется полюс порядка r, то вычет c_{-1} от полюса z_k порядка r находятся по формуле

$$c_{-1} = \frac{1}{(r-1)!}\lim_{z\to z_k}\frac{d^{r-1}\left[(z-z_k)^r e^{zx}\Phi(z)\right]}{dz^{r-1}}.$$

При необходимости учет задержек в очередях системы производится в имитационной модели следующего (более высокого) уровня иерархии. Выборочные значения выходных случайных величин GERT-сетей интерпретируются как задержки в обслуживающих приборах системы массового обслуживания. Имеется специально разработанное программное обеспечение для моделирования GERT-сетей и функционально связанная с системой GERT визуальная программа имитации.

Области применения метода. Предложенный метод без существенных изменений может быть использован не только для моделирования однородных виртуальных каналов, но для и расчетов сетевых трактов с разными характеристиками элементарных каналов одного звена.

Применение рассмотренных моделей позволяет:

- оценить требующуюся полосу пропускания для как отдельных звеньев, так и для виртуальных каналов и путей между маршрутизаторами MPLS. При этом могут быть оценены задержки и их вариации в разных точках канала;
- предложенные модели могут быть использованы при решении задач многопутевой маршрутизации, для нахождения наилучших вариантов инжиниринга трафика в MPLS, а также для оптимальной балансировки загрузки коммутирующих маршрутизаторов в соответствии с минимаксным критерием;
- данная модель является базовой для проведения процесса оптимизации опорных сетей технологий SDH и MPLS. В модели связаны между собой важнейшие параметры – надежность коммуникационного оборудования, время восстановления в случае отказа, задержки передачи и их вариации, вероятность искажения пакета.

По данным характеристикам составляется целевая функция для проведения оптимизации. Предполагается выполнение структурной оптимизации с использованием генетических алгоритмов. При ее выполнении должно быть найдено число параллельных каналов в каждом звене, параметры надежности, времени восстановления, средних задержек передачи и их вариаций, допустимые вероятности искажения пакетов в виртуальных каналах. Целевая функция (функция полезности, фитнесс-функция) связана как со стоимостными характеристиками элементарных каналов, так и с указанными выше показателями качества сети.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Головкин Б.А. Расчет характеристик и планирование параллельных вычислительных процессов. М.: Радио и связь, 1983.

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

А.П. ШИБАНОВ, В.А. ШИБАНОВ, К.А. СЛАВНОВ
Рязанский государственный радиотехнический университет

МОДЕЛЬ ТЕЛЕКОММУНИКАЦИОНОГО КАНАЛА СО СТАРЕНИЕМ ИНФОРМАЦИИ

Рассматривается GERT-модель телекоммуникационного канала, состоящего из нескольких звеньев, с учетом эффекта старения информации.

Введение

В процессах сбора, передачи и обработки информации последняя может с течением времени терять свою ценность. Особенно часто информация устаревает в системах реального времени, например, при передаче аудио и видео информации, при проведении телеконференций. Есть объективные физические причины, вызывающие старение информации. Например, пакетирование двоичных кодовых комбинаций измерений речи перед передачей информации в сеть. Задержки вызывают перегруппировки байт информации для передачи в следующее звено сети по агрегированным каналам и т.п. В алгоритмах маршрутизации должны предусматриваться меры по исключению “заблудившихся” пакетов вследствие сбоев оборудования или скрытых ошибок в программах маршрутизации.

Степень потери ценности пакетов может быть различной. В телекоммуникационных системах устаревшие пакеты могут или сразу удаляться, или снабжаться метками для обработки по специальным алгоритмам. От того, какая стратегия обработки устаревших пакетов принята, существенно зависит быстродействие телекоммуникационной системы. Контроль на старение информации может выполняться не только на выходе системы, но и в ее промежуточных точках.

Простейшим случаем является фиксация событий, в которых случайная величина ξ превышает заданное фиксированное значение t, т.е. $\xi > t$. Такие информационные сообщения (пакеты) считаются потерявшими ценность, и удаляются из сетевых трактов. Если пакет "уложился" в заданное время, то он передается дальше по сети. Поверка на "старение" может выполняться, как по результатам выполнения отдельных операций, так и на выходе информационного тракта. Если бы "устаревшие" пакеты не уничтожались, то в модели время на их обработку необходимо было бы учитывать. При моделировании протоколов передачи пакетов с учетом "старения" информации исходное распределение должно быть заменено условным распределением с выполнением соотношения $P(\xi \le t)$.

Если определено время ξ прохождения заявки из некоторого начального узла i в заданный конечный узел j, то необходимо определить вероятность $P\{\xi \le t_1\}$ того, что оно не превысит значения t_1 и вероятность $P\{\xi \ge t_n\}$ того, что оно превысит значение t_n.

Реализация модели

Для моделирования телекоммуникационного тракта со старением информации применим метод GERT (*Graphical evaluation and review technique*) [1]. Для его использования необходимо знание производящих функций моментов случайных величин, характеризующих отдельные операции процесса.

Полагаем, что безусловные функции распределения ветвей, для которых проверяются условия старения, являются *экспоненциальными*. Найдем производящую функцию моментов случайной величины при условии, что заявки (пакеты, сообщения) *не устарели до момента* t_1, $P\{\xi \le t_1\}$. На рис. 1 показаны условные плотность и функция распределения времени передачи заявки по одному каналу со старением информации.

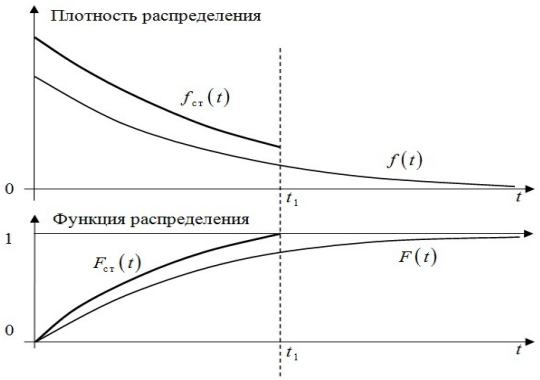

Рис. 1. Плотность и функция распределения времени передачи через канал со старением заявок, $P\{\xi \le t_1\}$

Плотность распределения времени передачи заявки по одному каналу со старением равна

$$f_{\tilde{n}\grave{o}}(t) = \frac{\lambda e^{-\lambda t}}{\int_0^{t_1} \lambda e^{-\lambda t} dt} = \frac{\lambda e^{-\lambda t}}{1 - e^{-\lambda t_1}} =, t \le t_1.$$

Функция распределения

$$F_{\tilde{n}\grave{o}}(t) = \int_0^t \frac{\lambda e^{-\lambda t}}{1 - e^{-\lambda t_1}} dt = \frac{1 - e^{-\lambda t}}{1 - e^{-\lambda t_1}}, t \le t_1.$$

По определению характеристическая функция $\chi^{(t)}(\zeta)$ случайной величины ξ с учетом старения заявок равна

$$\chi^{(t)}(\zeta) = \int_{-\infty}^{\infty} e^{i\zeta t} \frac{p(t)}{1 - e^{-t_1 \lambda}} dt = \frac{\lambda}{1 - e^{-t_1 \lambda}} \int_0^{t_1} e^{i\zeta t} e^{-\lambda t} dt =$$

$$= \frac{\lambda}{\lambda - i\zeta} \cdot \frac{1 - e^{-t_1(\lambda - i\zeta)}}{1 - e^{-t_1 \lambda}}.$$

Тогда производящая функция моментов случайной величины с экспоненциальным распределением с учетом старения

$$M^{(t)}(s) = \frac{\lambda}{\lambda - s} \cdot \frac{1 - e^{-t_1(\lambda - s)}}{1 - e^{-t_1 \lambda}}. \qquad (1)$$

Для любой GERT-сети выполняются соотношения: $W_E(s) = p_E M_E(s)$; при $s = 0$ $M_E(s) = 1$; $p_E = W_E(0)$; $M_E(s) = W_E(s)/W_E(0)$. Для последовательных ветвей со старением заявок $W_E^{(c)}(s) = W_E^{(c)}(0)\, M_E^{(c)}(s) = M_E^{(c)}(s)$, так как $W_E^{(t)}(0) = 1$.

Моделирование последовательно соединенных каналов со старением

Время передачи сообщения через канал связи принимаем распределенным по экспоненциальному закону. Длина сообщения может быть различной. Найдем математическое ожидание и дисперсию времени выполнения операции со старением заявки по производящей функции моментов (1).

$$\mu_1 = \left.\frac{\partial M^{(t_1)}(s)}{\partial s}\right|_{s=0} = \frac{1}{\lambda} - \frac{t_1 e^{-\lambda t_1}}{1 - e^{-\lambda t_1}}.$$

$$\sigma^2 = \left.\frac{\partial^2 M^{(t_1)}(s)}{\partial s^2}\right|_{s=0} - \left[\left.\frac{\partial M^{(t_1)}(s)}{\partial s}\right|_{s=0}\right]^2 = \frac{1}{\lambda^2} - \frac{t_1^2 e^{-\lambda t_1}}{\left(1 - e^{-\lambda t_1}\right)^2}.$$

Математическое ожидание времени передачи через m последовательно соединенных каналов с отбрасыванием устаревших пакетов равно сумме математических ожиданий времен передачи через каждый канал

$$\mu_\Sigma = \sum_{i=1}^{m} \left(\frac{1}{\lambda_i} - \frac{t_{1i} e^{-\lambda_i t_{1i}}}{1 - e^{-\lambda_i t_{1i}}} \right).$$

В том случае, когда все каналы однородные

$$\mu_{îäí\ \Sigma} = \frac{m}{\lambda} - \frac{m t_1 e^{-\lambda t_1}}{1 - e^{-\lambda t_1}}.$$

Так как длительности передачи пакетов через последовательно соединенные каналы характеризуются независимыми случайными величинами, то дисперсия времени передачи из конца в конец тракта определяется выражением

$$\sigma_\Sigma^2 = \sum_{i=1}^{m} \left[\frac{1}{\lambda_i^2} - \frac{t_{1i}^2 e^{-\lambda_i t_{1i}}}{\left(1 - e^{-\lambda_i t_{1i}}\right)^2} \right].$$

Если каналы однородные, то

$$\sigma^2_{îäí\ \Sigma} = \frac{m}{\lambda^2} - \frac{m t_1^2 e^{-\lambda t_1}}{\left(1 - e^{-\lambda t_1}\right)^2}.$$

Перед началом передачи пакетов выполняется проверка готовности тракта передачи. Считаем, что время распознавания сигнала готовности очень мало, и им можно пренебречь. Вероятность готовности тракта передачи равна *p*. В случае неготовности тракта он восстанавливается за время, распределенное по экспоненциальному закону с параметром μ. Производящая функция моментов времени проверки готовности тракта к работе с учетом возможного восстановления в случае отказа

$$M_{âîñò}(s) = p + (1-p)\frac{\mu}{\mu - s} = \frac{\mu - ps}{\mu - s}.$$

Найдем среднее время проверки с учетом возможного восстановления работоспособности тракта и дисперсия этого времени.

$\frac{\partial M_{âîñò}(s)}{\partial s} = M_{âîñò}(s)\frac{(1-p)\mu}{(\mu - s)(\mu - ps)}$, тогда

$$\bar{t}_{âîñò} = \left.\frac{\partial M_{âîñò}(s)}{\partial s}\right|_{s=0} = \frac{1-p}{\mu},$$

$$\sigma^2_{âîñò} = \frac{\partial^2 M_{âîñò}(s)}{\partial s^2} - t^2_{âîñò} = \frac{1-p^2}{\mu^2}.$$

Тогда среднее время передачи для тракта с восстановлением в случае отказа $\mu_{\Sigma\text{вост}}$ и его дисперсия $\sigma^2_{\Sigma\text{вост}}$ определяются выражениями

$$\mu_{\Sigma âîñò} = \frac{m}{\lambda} - \frac{m t_1 e^{-\lambda t_1}}{1 - e^{-\lambda t_1}} + \frac{1-p}{\mu},$$

$$\sigma^2_{\Sigma âîñò} = \frac{m}{\lambda^2} - \frac{m t_1^2 e^{-\lambda t_1}}{\left(1 - e^{-\lambda t_1}\right)^2} + \frac{1-p^2}{\mu^2}.$$

Эти формулы можно использовать для оперативной предварительной оценки вариантов проектов сети. При этом явно худшие из них отбрасываются. Эти выражения могут служить основой для формирования фитнесс функции при оптимизации сети со старением информации с использованием генетических алгоритмов.

Заключение

Получены выражения для математического ожидания и дисперсии времени прохождения канала, состоящего из последовательно соединенных агрегированных звеньев. Эти выражения могут служить для расчета временных характеристик и предварительного проектирования трактов физических и виртуальных каналов.

При необходимости модель может быть уточнена с использованием имитационного моделирования. При этом учитывается влияние очередей на состояние системы. Очереди могут возникать во входных и выходных портах КУ. При имитации системы массового обслуживания задержки в обслуживающем приборе реализуются как выборочные значения выходных величин GERT-сетей. В данной модели – это время прохождения канала одного звена.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

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