ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА И ПРИКЛАДНАЯ МАТЕМАТИКА

УДК 629.7.06(082): 629.7.018

Е.В. Ларкин, Н.А. Рудианов

МАТЕМАТИЧЕСКАЯ МОДЕЛЬ ПОПЕРЕЧНЫХ КОЛЕБАНИЙ ПОДВИЖНОГО НАЗЕМНОГО ОБЪЕКТА

Разработана модель, определяющая положение равновесия корпуса многоопорного подвижного наземного объекта при его перемещении по Земной поверхности с произвольным рельефом для случаев жестких и вязкоупругих колес. С учетом воздействия дороги при движении сформирована математическая модель поперечных колебаний транспортного средства по линейной вертикальной координате, углу тангажа и углу крена. Приведены результаты интегрирования полученной системы дифференциальных уравнений, включающих чистое запаздывание по входному воздействию. Результаты рекомендовано использовать в математическом и программном обеспечении тренажеров.

Ключевые слова: *тренажер, подвижный наземный объект, положение равновесия, рельеф, поперечные колебания, вертикальная координата, угол тангажа, угол крена, динамическая модель.*

Введение. Тренажеры и тренажерные комплексы в настоящее время являются одним из основных средств обучения операторов управлению подвижными наземными объектами (ПНО). Современный тренажер представляет собой сложный аппаратно-программный комплекс, в котором оператор, воздействуя на имитаторы органов управления, получает визуальные, слуховые, тактильные и акселерационные ощущения, подобные тем, которые он получал бы в реальном транспортном средстве [1, 2]. Исполнительные системы современных тренажеров, создающие подобные воздействия на оператора, как правило, управляются от ЭВМ, программное обеспечение которых должно включать математические модели процессов, протекающих в реальных ПНО. *Целью данной статьи* является разработка математической модели пространственных движений транспортного средства, что является важной и актуальной задачей при создании программного обеспечения тренажерных систем [3, 4].

Обобщенная конструкция подвижного наземного объекта. Конструктивно моделируемый ПНО представляет собой корпус, связанный через 2*N* вязкоупругие подвески с колесами движителей, которые, в свою очередь, опираются на Земную поверхность. Корпус транспортного средства является абсолютно жестким телом, симметричным относительно продольной вертикальной плоскости. Центр масс корпуса лежит в вертикальной плоскости его симметрии (рисунок 1).

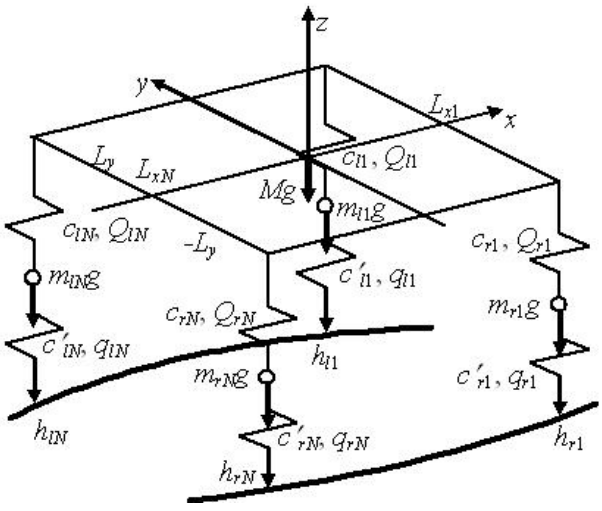


Рисунок 1 - К определению положения корпуса в установившемся режиме

Подвески располагаются вдоль левого (*l*) и

правого (r) бортов корпуса симметрично на расстоянии $\pm L_y$ от продольной вертикальной плоскости симметрии. Подвески перенумерованы от носа ПНО (номер 1) к его корме (номер N). Пара подвесок с разных бортов, имеющих n-й номер, расположена на расстоянии L_{xn} от центра масс корпуса транспортного средства.

Вязкоупругие подвески с номером n левого и правого бортов имеют начальную длину в недеформированном состоянии, соответственно Q_{ln0} и Q_{rn0} (рисунок 2). В процессе перемещения по дороге подвески сжимаются и растягиваются на величины Δ_{Qln} и Δ_{Qrn}, соответственно, таким образом, суммарные длины подвесок составляют

$$Q_{ln} = Q_{ln0} + \Delta_{Qln};\ Q_{rn} = Q_{rn0} + \Delta_{Qrn},\ 1 \le n \le N. \quad (1)$$

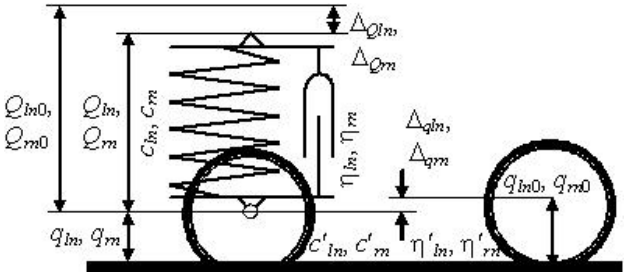


Рисунок 2 - Типовая подвеска транспортного средства

Упругие свойства подвесок определяются рессорами, которые имеют жесткость c_{ln}, c_{rn}. Диссипативные силы подвески считаются линейно пропорциональными скорости изменения ее длины, при этом η_{ln}, η_{rn} представляют собой коэффициент вязкого трения.

Колеса, покрытые шинами и имеющие радиусы q_{ln0}, q_{rn0}, считаются вязкоупругими демпферами. Расстояние оси n-го колеса от поверхности Земли составляет q_{ln} и q_{rn} соответственно за счет деформации шин на величины Δ_{qln} и Δ_{qrn} соответственно. Таким образом, действительные текущие радиусы колес составляют

$$q_{ln} = q_{ln0} + \Delta_{qln};\ q_{rn} = q_{rn0} + \Delta_{qrn},,\ 1 \le n \le N. \quad (2)$$

В первом приближении шины представляются в виде линейных пружин, которые имеют жесткость c'_{ln}, c'_{rn}. Диссипативные силы шин считаются линейно пропорциональными скоростям изменения их деформаций с коэффициентами вязкого трения η'_{ln} и η'_{rn}.

Точки подвеса лежат в одной плоскости, связанной с корпусом и с центром масс. Корпус не имеет продольной и поперечной линейной степени свободы относительно движителей. Таким образом, центры колес могут перемещаться только по прямой, перпендикулярной к плоскости расположения точек подвеса.

Если все подвески и шины одинаковы, то

$$\begin{gathered} Q_{l10} = Q_{r10} = ... = Q_{ln0} = Q_{rn0} = ... \\ = Q_{lN0} = Q_{rN0} = Q_0; \\ q_{l10} = q_{r10} = ... = q_{ln0} = q_{rn0} = ... = q_{lN0} = q_{rN0} = q_0; \\ c_{l1} = c_{r1} = ... = c_{ln} = c_{rn} = ... = c_{lN} = c_{rN} = c; \quad (3) \\ c'_{l1} = c'_{r1} = ... = c'_{ln} = c'_{rn} = ... = c'_{lN} = c'_{rN} = c'; \\ \eta_{l1} = \eta_{r1} = ... = \eta_{ln} = \eta_{rn} = ... = \eta_{lN} = \eta_{rN} = \eta; \\ \eta'_{l1} = \eta'_{r1} = ... = \eta'_{ln} = \eta'_{rn} = ... = \eta'_{lN} = \eta'_{rN} = \eta'. \end{gathered}$$

Положение равновесия корпуса относительно движителей. При моделировании поперечных колебаний ПНО относительно движителей рельеф представляется функцией высот в земной системе координат:

$$z = h(x, y). \quad (4)$$

Таким образом, все точки контакта колес ПНО с Земной поверхностью имеют разную высоту. Если принять допущение, что углы тангажа ϑ и крена γ корпуса транспортного средства являются малыми, транспортное средство располагается вдоль оси Ox, центр масс имеет координаты в горизонтальной плоскости $x_{\text{цм}}$ и $y_{\text{цм}}$, то высота рельефа под левыми и правыми колесами определяется по зависимости

$$h_{ln} = h(x_{\text{цм}} + L_{xn},\ y + L_y);\ h_{ln} = h(x_{\text{цм}} + L_{xn}, y + L_y),\ 1 \le n \le N. \quad (5)$$

Для малых углов ϑ и γ пространственное положение корпуса ПНО в Земной системе координат в положении равновесия описывается следующей системой алгебраических уравнений:

кинематики -

$$\begin{cases} z_{ln} = h_{ln} + \left(Q_{ln0} + \Delta_{Qln}\right) + \left(q_{ln0} + \Delta_{qln}\right); \\ z_{rn} = h_{rn} + \left(Q_{rn0} + \Delta_{Qrn}\right) + \left(q_{rn0} + \Delta_{qrn}\right); \\ 1 \le n \le N; \qquad \vdots \end{cases} \quad (6)$$

$$\begin{cases} z_{ln} = z + L_y\gamma + L_{xn}\vartheta; \\ z_{rn} = z - L_y\gamma + L_{xn}\vartheta; \\ 1 \le n \le N; \quad \vdots \end{cases} \quad (7)$$

и статики -

$$\sum_{n=1}^{N} \Delta_{Qln} c_{ln} + \sum_{n=1}^{N} \Delta_{Qrn} c_{rn} = Mg; \quad (8)$$

$$\sum_{n=1}^{N} \Delta_{Qln} c_{ln} = \sum_{n=1}^{N} \Delta_{Qrn} c_{rn}; \quad (9)$$

$$\sum_{n=1}^{N} \left(\Delta_{Qln} c_{ln} + \Delta_{Qrn} c_{rn}\right) L_{xn} = 0; \quad (10)$$

$$\begin{cases} \Delta_{Qln} c_{ln} + m_{ln} g = \Delta_{qln} c'_{ln}; \\ \Delta_{Qrn} c_{rn} + m_{rn} g = \Delta_{qrn} c'_{rn}; \\ 1 \le n \le N, \end{cases} \quad (11)$$

где M - масса корпуса; g - ускорение свободного падения; m_{ln} (m_{rn}) - масса n-го левого (правого) колеса.

Уравнения (6), (7) определяют высоты центра масс и точек подвеса опор на корпусе. Зависимость (8) представляет собой уравнение сил, а зависимости (9) и (10) представляют собой уравнения моментов по углам тангажа и крена соответственно. Уравнения (11) определяют высоты осей колес.

Если справедливы условия (3), то

$$\begin{cases} h_{ln} + \left(Q_0 + \Delta_{Qln}\right) + \left(q_0 + \Delta_{qln}\right) = z + L_y\gamma + L_{xn}\vartheta; \\ h_{rn} + \left(Q_0 + \Delta_{Qrn}\right) + \left(q_0 + \Delta_{qrn}\right) = z - L_y\gamma + L_{xn}\vartheta; \\ 1 \le n \le N; \end{cases}$$

$$\sum_{n=1}^{N}\Delta_{Qln} + \sum_{n=1}^{N}\Delta_{Qrn} = \frac{Mg}{c};$$

$$\sum_{n=1}^{N}\Delta_{Qln} = \sum_{n=1}^{N}\Delta_{Qrn}; \qquad (12)$$

$$\sum_{n=1}^{N}\left(\Delta_{Qln} + \Delta_{Qrn}\right)L_{xn} = 0;$$

$$\begin{cases} \Delta_{Qln}c + mg = \Delta_{qln}c'; \\ \Delta_{Qrn}c + mg = \Delta_{qrn}c'; \\ 1 \le n \le N. \end{cases}$$

В линейной системе (12) из (2N + 3)-х уравнений неизвестными являются: 2N деформаций подвесок Δ_{Qln}, Δ_{Qrn}, координата z центра плоскости расположения точек подвески, а также малые углы крена γ и тангажа ϑ.

Для абсолютно жестких колес (гусеничные движители) система уравнений (12) принимает вид:

$$\begin{cases} h_{ln} + \left(Q_0 + \Delta_{Qln}\right) + q_0 = z + L_y\gamma + L_{xn}\vartheta; \\ h_{rn} + \left(Q_0 + \Delta_{Qrn}\right) + q_0 = z - L_y\gamma + L_{xn}\vartheta; \\ 1 \le n \le N; \end{cases}$$

$$\sum_{n=1}^{N}\Delta_{Qln} + \sum_{n=1}^{N}\Delta_{Qrn} = \frac{Mg}{c}; \qquad (13)$$

$$\sum_{n=1}^{N}\Delta_{Qln} = \sum_{n=1}^{N}\Delta_{Qrn};$$

$$\sum_{n=1}^{N}\left(\Delta_{Qln} + \Delta_{Qrn}\right)L_{xn} = 0.$$

Поперечные колебания корпуса относительно устойчивого положения описываются следующей системой дифференциальных уравнений [5]:

$$M\ddot{\delta}_z + 2\eta\sum_{n=1}^{N}\left(\dot{\delta}_z + \dot{\delta}_\vartheta L_{xn}\right) + 2c\sum_{n=1}^{N}\left(\delta_z + \delta_\vartheta L_{xn}\right) = \sum_{n=1}^{N}(\eta\dot{\delta}_{qln} + \eta\dot{\delta}_{qrn} + c\delta_{qln} + c\delta_{qrn});$$

$$J_y\ddot{\delta}_\vartheta + 2\eta\sum_{n=1}^{N}\left(\dot{\delta}_z + \dot{\delta}_\vartheta L_{xn}\right)L_{xn} + 2c\sum_{n=1}^{N}\left(\delta_z + \delta_\vartheta L_{xn}\right)L_{xn} = \sum_{n=1}^{N}(\eta\dot{\delta}_{qln} + \eta\dot{\delta}_{qrn} + c\delta_{qln} + c\delta_{qrn})L_{xn};$$

$$J_x\ddot{\delta}_\gamma + 2\eta N\dot{\delta}_\gamma L_y^2 + 2cN\delta_\gamma L_y^2 = L_y\sum_{n=1}^{N}(\eta\dot{\delta}_{qln} + c\delta_{qln} - \eta\dot{\delta}_{qrn} - c\delta_{qrn}); \qquad (14)$$

$$m\ddot{\delta}_{qln} + \left(\eta + \eta'\right)\dot{\delta}_{qln} + \left(c + c'\right)\delta_{qln} = \eta\left(\dot{\delta}_z + \dot{\delta}_\vartheta L_{xn} + \dot{\delta}_\gamma L_y\right) + c\left(\delta_z + \delta_\vartheta L_{xn} + \delta_\gamma L_y\right) + \eta'\dot{\delta}_{hln} + c'\delta_{hln};$$

$$m\ddot{\delta}_{qrn} + \left(\eta + \eta'\right)\dot{\delta}_{qrn} + \left(c + c'\right)\delta_{qrn} = \eta\left(\dot{\delta}_z + \dot{\delta}_\vartheta L_{xn} + \dot{\delta}_\gamma L_y\right) + c\left(\delta_z + \delta_\vartheta L_{xn} + \delta_\gamma L_y\right) + \eta'\dot{\delta}_{hrn} + c'\delta_{hrn};$$

$$1 \le n \le N,$$

где δ_z, δ_ϑ и δ_γ - отклонения от положения равновесия центра масс корпуса, угла тангажа и угла крена соответственно; δ_{qln} (δ_{qrn}) – отклонение от положения равновесия оси левого (правого) n-го колеса; δ_{hln} (δ_{hrn}) - приращение высоты точки касания левым (правым) n-м колесом поверхности Земли.

В том случае, если колеса можно представить в виде жестких несминаемых дисков, система (14) принимает вид:

$$M\ddot{\delta}_z + 2\eta\sum_{n=1}^{N}\left(\dot{\delta}_z + \dot{\delta}_\vartheta L_{xn}\right) + 2c\sum_{n=1}^{N}\left(\delta_z + \delta_\vartheta L_{xn}\right) = \sum_{n=1}^{N}(\eta\dot{\delta}_{hln} + \eta\dot{\delta}_{hrn} + c\delta_{hln} + c\delta_{hrn});$$

$$J_y\ddot{\delta}_\vartheta + 2\eta\sum_{n=1}^{N}\left(\dot{\delta}_z + \dot{\delta}_\vartheta L_{xn}\right)L_{xn} + 2\tilde{n}\sum_{n=1}^{N}\left(\delta_z + \delta_\vartheta L_{xn}\right)L_{xn} = \sum_{n=1}^{N}(\eta\dot{\delta}_{hln} + \eta\dot{\delta}_{hrn} + c\delta_{hln} + c\delta_{hrn})L_{xn}; \qquad (15)$$

$$J_x\ddot{\delta}_\gamma + 2\eta N\dot{\delta}_\gamma L_y^2 + 2\tilde{n}N\delta_\gamma L_y^2 = L_y\sum_{n=1}^{N}(\eta\dot{\delta}_{hln} + c\delta_{hln} - \eta\dot{\delta}_{hrn} - c\delta_{hrn}).$$

Пусть подвижный наземный объект перемещается вперед со скоростью $\dot{\xi}$, изменение которой на расстоянии, равном длине корпуса транспортного средства, незначительно. Любая точка траектории с соответствующей высотой рельефа, которая парой первых колес проходится в момент t = 0, парой вторых колес проходится через интервал времени $\tau_2 = \frac{L_{x1} - L_{x2}}{\dot{\xi}}$ после первых колес, парой n-х колес проходится через $\tau_n = \frac{L_{xn} - L_{x1}}{\dot{\xi}}$ после первых колес и парой N-х колес проходится через $\tau_n = \frac{L_{xN} - L_{x1}}{\dot{\xi}}$ после первых колес. С учетом этого факта последние 2N уравнений системы (14) и система (15)

принимают вид соответственно:

$$
\begin{gathered}
m\ddot{\delta}_{qln} + (\eta+\eta')\dot{\delta}_{qln} + (c+c')\delta_{qln} = \\
+\eta\left(\dot{\delta}_z + \dot{\delta}_\vartheta L_{xn} + \dot{\delta}_\gamma L_y\right) + \\
+c\left(\delta_z + \delta_\vartheta L_{xn} + \delta_\gamma L_y\right) + \\
+\eta'\dot{\delta}_{hl1}(t-\tau_n) + c'\delta_{hl1}(t-\tau_n);
\end{gathered}
$$

$$
\begin{gathered}
m\ddot{\delta}_{qrn} + (\eta+\eta')\dot{\delta}_{qrn} + (c+c')\delta_{qrn} = \\
= \eta\left(\dot{\delta}_z + \dot{\delta}_\vartheta L_{xn} + \dot{\delta}_\gamma L_y\right) + \\
+c\left(\delta_z + \delta_\vartheta L_{xn} + \delta_\gamma L_y\right) + \\
+\eta'\dot{\delta}_{hr1}(t-\tau_n) + c'\delta_{hr1}(t-\tau_n);
\end{gathered} \qquad (16)
$$

$$
1 \le n \le N;
$$

$$
\begin{gathered}
M\ddot{\delta}_z + 2\eta\sum_{n=1}^{N}\left(\dot{\delta}_z + \dot{\delta}_\vartheta L_{xn}\right) + 2c\sum_{n=1}^{N}\left(\delta_z + \delta_\vartheta L_{xn}\right) = \\
= \sum_{n=1}^{N}\left[\eta\dot{\delta}_{hl1}(t-\tau_n) + \eta\dot{\delta}_{hr1}(t-\tau_n) + \right. \\
\left. + c\delta_{hl1}(t-\tau_n) + c\delta_{hr1}(t-\tau_n)\right];
\end{gathered}
$$

$$
\begin{gathered}
J_y\ddot{\delta}_\vartheta + 2\eta\sum_{n=1}^{N}\left(\dot{\delta}_z + \dot{\delta}_\vartheta L_{xn}\right)L_{xn} + 2c\sum_{n=1}^{N}\left(\delta_z + \right. \\
\left. + \delta_\vartheta L_{xn}\right)L_{xn} = \sum_{n=1}^{N}\left[\eta\dot{\delta}_{hl1}(t-\tau_n) + \eta\dot{\delta}_{hr1}\cdot\right. \qquad (17) \\
\left.\cdot(t-\tau_n) + c\delta_{hl1}(t-\tau_n) + c\delta_{hl1}(t-\tau_n)\right]L_{xn};
\end{gathered}
$$

$$
\begin{gathered}
J_x\ddot{\delta}_\gamma + 2\eta N\dot{\delta}_\gamma L_y^2 + 2cN\delta_\gamma L_y^2 = \\
= L_y\sum_{n=1}^{N}\left[\eta\dot{\delta}_{hl1}(t-\tau_n) + c\delta_{hl1}(t-\tau_n) - .\right. \\
\left. - \eta\dot{\delta}_{hr1}(t-\tau_n) - c\delta_{hr1}(t-\tau_n)\right].
\end{gathered}
$$

В операторной форме (14), (16) и (17) имеют вид:

$$
\begin{gathered}
\delta_z(s)\cdot\left(Ms^2 + 2N\eta s + 2c\right) + \\
+\delta_\vartheta(s)\cdot\left(2\sum_{n=1}^{N}L_{xn}\right)\cdot(\eta s + c) = \ ; \\
= \sum_{n=1}^{N}\left[\delta_{qln}(s) + \delta_{qrn}(s)\right]\cdot(\eta s + c);
\end{gathered}
$$

$$
\begin{gathered}
\delta_\vartheta(s)\cdot\left(J_y s^2 + 2\eta\sum_{n=1}^{N}L_{xn}^2 s + 2c\sum_{n=1}^{N}L_{xn}^2\right) + \\
\_\delta_z(s)\cdot\left(2\eta\sum_{n=1}^{N}L_{xn}s + 2c\sum_{n=1}^{N}L_{xn}\right) = \\
= \sum_{n=1}^{N}L_{xn}\left[\delta_{qln}(s) + \delta_{qrn}(s)\right]\cdot(\eta s + c);
\end{gathered}
$$

$$
\begin{gathered}
\delta_\gamma(s)\cdot\left(J_x s^2 + 2\eta N L_y^2 s + 2cNL_y^2\right) = \\
L_y\sum_{n=1}^{N}\left[\delta_{qän}(s) - \delta_{qrn}(s)\right]\cdot(\eta s + c);
\end{gathered} \qquad (18)
$$

$$
\begin{gathered}
\delta_z(s)\cdot\left(Ms^2 + 2N\eta s + 2c\right) + \\
+\delta_\vartheta(s)\cdot\left(2\sum_{n=1}^{N}L_{xn}\right)\cdot(\eta s + c) = \\
= \sum_{n=1}^{N}\left[\delta_{hln}(s) + \delta_{hrn}(s)\right]\cdot(\eta s + c)\cdot\exp(-\tau_n s);
\end{gathered}
$$

$$
\begin{gathered}
\delta_{qln}(s)\cdot\left[ms^2 + (\eta+\eta')s + (c+c')\right] = \\
= \left[\dot{\delta}_z(s) + L_{xn}\dot{\delta}_\vartheta(s) + L_y\dot{\delta}_\gamma(s)\right]\cdot(\eta s + c) + \\
+\delta_{hl1}(s)\cdot\left[\eta' s + c'\right]\cdot\exp(-\tau_n s);
\end{gathered}
$$

$$
\begin{gathered}
\delta_{qrn}(s)\cdot\left[ms^2 + (\eta+\eta')s + (c+c')\right] = \\
= \left[\dot{\delta}_z(s) + L_{xn}\dot{\delta}_\vartheta(s) + L_y\dot{\delta}_\gamma(s)\right]\cdot(\eta s + c) + \\
+\delta_{hr1}(s)\cdot\left[\eta' s + c'\right]\cdot\exp(-\tau_n s);
\end{gathered}
$$

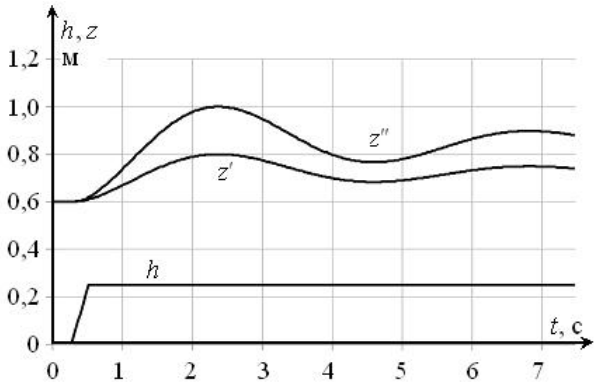
$$
1 \le n \le N;
$$

$$
\begin{gathered}
\delta_\vartheta(s)\cdot\left(J_y s^2 + 2\eta\sum_{n=1}^{N}L_{xn}^2 s + 2c\sum_{n=1}^{N}L_{xn}^2\right) + \\
+\delta_z(s)\cdot\left(2\eta\sum_{n=1}^{N}L_{xn}s + 2c\sum_{n=1}^{N}L_{xn}\right) = \\
= \sum_{n=1}^{N}L_{xn}\left[\delta_{hln}(s) + \delta_{hrn}(s)\right]\cdot(\eta s + c)\cdot \\
\cdot\exp(-\tau_n s);
\end{gathered} \qquad (19)
$$

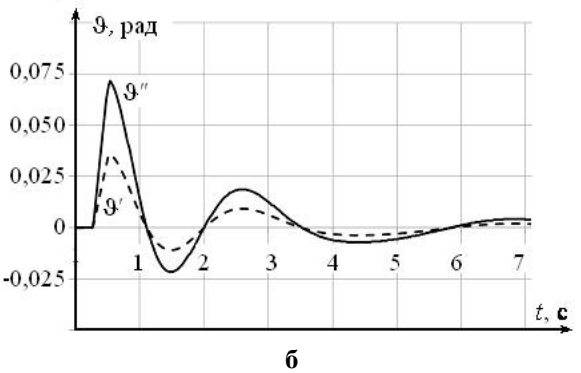
$$
\begin{gathered}
\delta_\gamma(s)\cdot\left(J_x s^2 + 2\eta N L_y^2 s + 2cNL_y^2\right) = \\
L_y\sum_{n=1}^{N}\left[\delta_{qln}(s) - \delta_{qrn}(s)\right]\cdot(\eta s + c)\cdot\exp(-\tau_n s).
\end{gathered}
$$

Таким образом, системы (18) и (19) показывают, что поперечные колебания корпуса ПНО относительно движителей формируются как колебания механической системы второго порядка. Вследствие наличия звеньев с чистым запаздыванием в данной механической системе присутствует эффект раскачивания корпуса [6, 7]. Этот эффект усиливается, если частота воздействий совпадает с частотой собственных колебаний системы.

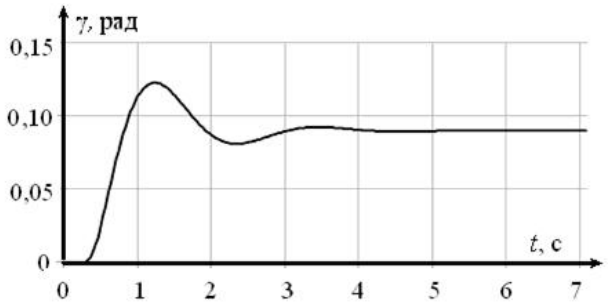
Результаты интегрирования системы (19) для гусеничного ПНО, имеющего по шесть опорных колес с каждой стороны, приведены на рисунке 3. Моделируется наезд на ступенчатое препятствие, график которого обозначен через *h* на рисунке 3, а.



а



б



в

Рисунок 3 - Результаты интегрирования системы (19) для гусеничного ПНО

Графики z' и z'' на рисунке 3, а показывают высоту центра масс при наезде на препятствие левым рядом колес и обоими рядами соответственно. Графики ϑ' и ϑ'' на рисунке 3, б показывают угол тангажа при наезде одним и обоими рядами колес, а график γ на рисунке 3, в показывает угол тангажа при наезде на препятствие левым рядом колес. Среднеквадратичное отклонение соответствующих реальных величин от их расчетных значений не превышает 5 %, что лежит в пределах точности определения исходных данных модели.

Заключение. В статье получена динамическая модель поперечных колебаний подвижного наземного объекта относительно положения равновесия, предназначенная для использования в математическом и программном обеспечении тренажерных систем.

Библиографический список

1. *Курочкин С.А., Ларкин Е.В.* Системный подход к проектированию тренажеров // Известия ТулГУ. Серия: Проблемы управления электротехническими объектами. Вып. 3. - Тула: ТулГУ, 2005. - С. 182 - 183.

2. *Веников В.А., Веников Г.В.* Теория подобия и моделирование. - М.: Высшая школа, 1984. - 440 с.

3. *Ларкин Е.В., Привалов А.Н.* Проектирование программного обеспечения вычислительных средств тренажерных систем. - Тула: ТулГУ, 2010. - 259 с.

4. *Ларкин Е.В., Привалов А.Н.* Создание программного обеспечения тренажерных систем на основе унифицированных программных модулей // Вестник компьютерных и информационных технологий.- 2010. - № 4. - С. 50 - 56.

5. *Силаев А.А.* Спектральная теория подрессоривания транспортных машин. - М.: Машиностроение, 1972. - 192 с.

6. *Курочкин С.А., Ларкин Е.В.* Моделирование движения наземного объекта в тренажере // Известия ТулГУ. Серия: Проблемы специального машиностроения. Вып. 6. Т. 2. - Тула: ТулГУ, 2003. - С. 190 - 197.

7. *Ларкин Е.В., Акименко Т.А., Лучанский О.А.* Оценка «смаза» изображения в системе технического зрения мобильного колесного робота // Вестник Рязанского радиотехнического университета. - Рязань: РИЦ РГРТУ, 2008. - С. 77 - 80.

УДК 004.05

А.Е. Балясов, В.В. Бухарин, В.И. Андрианов

ПРОВЕРКА СПЕЦИАЛЬНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ КОМПЛЕКСОВ РАДИОКОНТРОЛЯ НА НАЛИЧИЕ НЕДЕКЛАРИРОВАННЫХ ВОЗМОЖНОСТЕЙ И ОШИБОК

В работе рассмотрена последовательность действий при проверке специального программного обеспечения на наличие недекларированных возможностей и ошибок. Данный способ позволяет с высокой вероятностью определить отсутствие в специальном программном обеспечении недекларированных возможностей и ошибок, влияющих на точность и достоверность результатов измерений комплексов радиоконтроля специального назначения.

Ключевые слова: *верификация, специальное программное обеспечение, адаптация, недекларированные возможности, комплекс радиоконтроля.*

Введение. Современные комплексы радиоконтроля представляют собой средства измерений, включающие автоматизированные системы обработки измерительной информации (АСОИИ) [1].

Составной частью АСОИИ является специальное программное обеспечение (СПО). Оно определяет точность и достоверность метрологических характеристик АСОИИ, которые зависят от наличия в нем недекларированных возможностей и ошибок.

Цель работы. Разработать последовательность действий проверки СПО комплексов радиоконтроля, позволяющих с высокой вероят-

ностью определить отсутствие в нем недекларированных возможностей и ошибок $P_{\text{ндв тр}}$=0,95, влияющих на точность и достоверность результатов измерений.

Основные ограничения и допущения.

1. Ограниченность объема кода СПО (т.к. СПО решает строго определенные задачи).

2. Выявление недекларированных возможностей и ошибок производится статистическим контролем, что влечет пропуск ошибок, связанных с динамическим выполнением программы (например, обращение к неинициализированной области памяти).

3. В проверяемом исходном тексте СПО не используются управляющие структуры с параллельными вычислениями. Это ограничение принимается для упрощения методики и связано с тем, что в большинстве программ СПО параллельные управляющие структуры не применяются.

4. При разработке СПО используется определенный язык программирования. Это связано с тем, что при использовании другого языка программирования необходимо разработать соответствующие процедуры формирования условий корректности.

5. Имеющиеся во внешней спецификации функциональные спецификации и аксиоматические теории соответствуют предъявляемым к ним требованиям по правильности и полноте.

Решение. Основным свойством СПО является правильность функционирования, поэтому крайне необходимо, чтобы это качество соответствовало предъявляемым к нему требованиям [13]. Достижение данных требований в современных условиях возможно только при постоянном контроле СПО на наличие недекларированных возможностей, ошибок или изменение кода, причем как на этапе разработки, так и на этапе его функционирования.

Анализ применяемых и перспективных способов создания СПО, а также руководящих документов по защите информации показал, что имеются несоответствия при проверке (контроле) наличия недекларированных возможностей и ошибок, связанных с использованием только методов тестирования, не позволяющих дать однозначные результаты и не имеющих статистического анализа на строго формализованной основе.

В связи с этим возникает необходимость разработки и внедрения способов проверки (контроля) правильности СПО, позволяющих с высокой вероятностью определить отсутствие в нем недекларированных возможностей и наличие различных ошибок. Решение данной проблемы связано с применением верификации, основанной на анализе свойств всех допустимых выполнений программы с помощью формальных доказательств наличия требуемых свойств.

Известные способы, такие как дедуктивный анализ кода, построение управляющего графа программы, использование моделей программ и двоичных разрешающих таблиц не обеспечивают требуемой вероятности определения недекларированных возможностей и ошибок, так как представляют детерминированные процедуры. Повышение достоверности верификации предлагается реализовать в способе адаптивной верификации за счет накопления в базе знаний аксиоматических теорий предметных областей с возможностью ее постоянного дополнения. Это позволяет автоматизировать процесс верификации, что значительно уменьшит временные трудозатраты и повысит достоверность результатов проверки за счет использования интеллектуальной системы с безопасной (проверенной в предыдущих исследованиях и защищенной от вредоносных воздействий) базой знаний.

Предлагаемый способ заключается в выполнении следующих операций.

1. Ввод условий корректности программы, сформированных на этапе определения списка условий корректности.

2. Осуществление элементарных упрощений условий корректности программы путем приведения арифметических и логических выражений к канонической форме, подстановка равенств и истинностных значений (то есть логических констант *true* и *false*). Каноническая форма арифметического выражения – это упорядоченная сумма произведений. Условие корректности с помощью элементарных упрощений обычно приводится к виду: $\alpha_1 \wedge \ldots \wedge \alpha_m \Rightarrow \beta_1 \wedge \ldots \wedge \beta_n$, где формулы α_i β_j не содержат логических связок, кроме.

3. Проверка наличия в формулах условий корректности подформул общих теорий, используемых при решении широкого спектра задач. При наличии использования подформул общих теорий - осуществление доказательства подформул общих теорий, т.е. с использованием разрешающих процедур определяется выполнимость подформулы в определенной общей теории. К общим теориям можно отнести бескванторную теорию равенства неинтерпретированных функциональных символов, арифметику без умножения и ее расширения, а также комбинации бескванторных теорий.

Первая теория использует язык, состоящий из переменных, функциональных символов (все-

возможных арностей и, в частности, констант), единственного предиката "равенство" $(x = y)$ и обычных логических связок. Бескванторная теория равенства неинтерпретированных функциональных символов содержит аксиомы подстановочности равенства вида:

$$x = y \Rightarrow f(...x...) = f(...y...) \qquad (1)$$

для всех функциональных символов, а также обычные аксиомы рефлексивности, симметричности и транзитивности равенства.

При доказательстве условий корректности часто возникает задача вывода следствий из равенств в посылке условий. Для решения этой задачи полезна разрешающая процедура теории равенства, которая основана на процедуре построения конгруэнтного замыкания отношения на графе. Разрешающая процедура теории равенства состоит из трех этапов. Пусть задана формула α.

На первом этапе отрицание формулы α приводится к дизъюнктивной нормальной форме (ДНФ). Например, если $\alpha = (\alpha_1 \Rightarrow \alpha_2) = (\neg\alpha_1 \vee \alpha_2)$, причем α_1, α_2 не содержат дизъюнкции, то $\neg\alpha = (\alpha_1 \wedge \neg\alpha_2)$ – единственный дизъюнктивный член. В общем случае произвольный дизъюнктивный член формулы $\neg\alpha$ имеет следующий вид:

$$\beta = (t_1 = t_1' \wedge ... \wedge t_m = t_m' \wedge s_1 \neq s_1' \wedge ... \wedge s_n \neq s_n'), \quad (2)$$

где t_i, s_j – некоторые термы, образованные из переменных и констант с помощью функциональных символов.

На втором этапе по множеству T всех подтермов термов, входящих в формулу β, строится ориентированный граф G, в котором каждому терму из T соответствует вершина, помеченная функциональным символом или переменной. Построение графа G начинается с тех его вершин, с которыми сопоставляются символы переменных или констант. Пусть с термами $t_1,...,t_k$ сопоставлены вершины графа $v(t_1),...,v(t_k)$, причем с одинаковыми термами сопоставлена одна вершина. Тогда с термом $f(t_1,...,t_k)$ сопоставляется вершина графа, помеченная f, из которой выходят k дуг, где i-я дуга ведет в вершину $v(t_i)\ (1 \le i \le k)$. Формула β задает на графе G отношение R как множество пар вершин $\{(v(t_i), v(t_i') \mid i = 1,..., m\}$.

На третьем этапе определяется конгруэнтное замыкание R^+ отношения R на графе G. Операция конгруэнтности добавляет к отношению R_0 такие пары вершин, которые имеют одинаковые пометки, одинаковое число преемников, и их соответствующие преемники либо совпадают, либо связаны отношением R_0. Операция симметричности добавляет к отношению R_0, содержащему пару вершин (v_1, v_2), пару (v_2, v_1). Операция транзитивности добавляет к отношению R_0, содержащему пары вершин (v_1, v_2) и (v_2, v_3), пару (v_1, v_3). Конгруэнтное замыкание R^+ – это наименьшее отношение, содержащее R, замкнутое относительно операций конгруэнтности, симметричности и транзитивности. Известно, что такое отношение (типа эквивалентности) существует и единственно. Очевидный алгоритм построения отношения R^+ состоит в добавлении к R пар вершин посредством операций конгруэнтности, симметричности, транзитивности до тех пор, пока расширяется исходное отношение.

Для описания работы процедуры теории равенства на формуле α рассмотрим два случая.

1. Существует $i(1 \le i \le n)$ такое, что $(v(s_i), v(s_i')) \in R^+$. Тогда формула β невыполнима. Если все дизъюнктивные члены $\neg\alpha$ невыполнимы, то формула α истинна.

2. Для каждого $i(1 \le i \le n)$ имеем $(v(s_i), v(s_i')) \in R^+$. Тогда формула β выполнима, а значит, формула α ложна.

Во второй теории – арифметики Пресбургера язык состоит из символов констант, обозначающих натуральные числа, сложения (+), арифметические отношения $(=, \neq, \le, <)$. Бескванторные формулы арифметики Пресбургера строятся из этих символов и переменных с помощью обычных логических связок. При этом умножение константы на переменную означает сокращенную запись ограниченного числа сложений (например, $x = 3y$ эквивалентно $x = y + y + y$). Ограничимся двумя интерпретациями языка: в области целых и в области рациональных чисел. Эти интерпретации различаются бескванторными формулами.

При этом можно воспользоваться методом Блэдсоэ – Шостака ($sup-inf$-метод) разрешения бескванторных формул арифметики Пресбургера, истинных в области целых чисел. Этот метод состоит из трех этапов. На первом этапе отрицание формулы α приводится к дизъюнктивной нормальной форме. Предварительно из формулы α исключаются символы $=, \neq, <$ и

термы вида $t \le s$ с помощью замен:

$x = y$ на $x \le y \wedge x \ge y$;
$x \ne y$ на $x < y \vee x > y$;
$x < y$ на $x + 1 \le y$;
$\neg(x < y)$ на $y \le x$;
$\neg(x < y)$ на $y + 1 \le x$.

Кроме того, из формулы α устраняются члены, не содержащие переменных (вида $c_1 \le c_2$, где c_1, c_2 – константы), посредством замены их символами *true*, *false* и последующего исключения этих символов. В результате формула $\neg\alpha$ имеет вид: $\beta_1 \vee \beta_2 \vee ... \vee \beta_n$, где β_i – дизъюнктивные члены, представимые конъюнкциями формул вида $t \le t'$, а t, t' – линейные формы вида $c_1 x_1 + ... + c_m x_m + c_{m+1}$ (c_i – константы). Конечную совокупность неравенств вида $t \le t'$ можно рассматривать как задачу линейного целочисленного программирования. Значит, β_i выполнима тогда и только тогда, когда соответствующая задача имеет решение.

На втором этапе рассматривается произвольный дизъюнктивный член β формулы $\neg\alpha$, а также соответствующая β задача линейного целочисленного программирования. Для каждой переменной $\tilde{o}$ из β определяются функция $sup_\beta(x)$ как максимальное значение, которое принимает переменная $\tilde{o}$ в рациональном решении этой задачи, и функция $inf_\beta(x)$ как минимальное значение, которое принимает переменная $\tilde{o}$ в рациональном решении задачи. При этом если максимального значения $\tilde{o}$ не существует (т.е. $\tilde{o}$ принимает сколь угодно большое значение), то положим $sup_\beta(x) = \infty$, а если минимального значения $\tilde{o}$ не существует, то положим $inf_\beta(x) = -\infty$. В работе Шостака [1, 2, 3] даются алгоритмы вычисления функций $sup_\beta(x)$ и $inf_\beta(x)$. Если для некоторой переменной x из β замкнутый интервал $[inf_\beta(x), sup_\beta(x)]$ не содержит целых чисел, то задача не имеет целочисленного решения, а значит, формула β невыполнима (в области целых чисел). Понятно, что если каждый дизъюнктивный член формулы $\neg\alpha$ является невыполнимым, то формула α истинна.

Пусть для каждой переменной $\tilde{o}$ из β интервал $[inf_\beta(x), sup_\beta(x),]$ содержит целое число. На третьем этапа делается попытка установить выполнимость формулы β (т.е. ложность α) путем последовательного исключения переменных. Вначале в интервале $[inf_\beta(x), sup_\beta(x),]$ выбирается целое число $\tilde{o}_0$. Затем в формуле β переменная $\tilde{o}$ заменяется константой $\tilde{o}_0$, в результате чего получим (после упрощений) формулу β' и соответствующую ей задачу линейного целочисленного программирования. Задачи, соответствующие формулам β и β', одновременно, или разрешимы или неразрешимы в рациональных числах. Процесс исключения переменных применяется к формуле β' и т. д. Если удалось исключить все переменные из формулы β, подставив вместо каждой из них целое число, то набор этих целых чисел образует контрпример к формуле α (т.е. на этом наборе α ложна). Если же на некотором шаге исключения переменных в интервале $[inf_\beta(x), sup_\beta(x)]$ нет целых чисел, то отсюда нельзя вывести невыполнимость формулы β, а нужно вместо исключенных переменных взять другие константы.

При верификации реальных программ могут получаться условия корректности, которые являются бесквантворными формулами, выраженными в "смеси" теорий, т.е. содержат одновременно арифметику Пресбургера, операции над массивами, неинтерпретированные функциональные символы и т.д. Поэтому необходимо, чтобы по разрешающим процедурам для отдельных теорий имелась бы разрешающая процедура для их комбинаций, такая возможность реализуется в методе, предложенном в работах Нелсона и Оупена [4, 5].

Этот метод применим для теорий, которые не имеют общих функциональных и предикатных символов (как интерпретированных, так и неинтерпретированных), за исключением предиката равенства, причем все теории бесквантворные. Так, работа метода для формулы в объединении арифметики Пресбургера (в области рациональных чисел) и теории равенства неинтерпретированных функциональных символов $\alpha : x \le y \wedge y + z \le x \wedge z \ge 0 \wedge f(f(x) - f(y)) \ne f(z)$ состоит из двух этапов.

На первом этапе формула α преобразуется в равносильную формулу α', которая состоит из конъюнкции двух формул, таких, что одна является формулой арифметики, а другая – теории равенства. Для этой цели вводятся новые переменные. Пусть W_1 обозначает терм

$f(x)-f(y)$, W_2 – терм $f(x)$, W – терм $f(y)$. Тогда $\alpha' = \alpha \wedge \alpha$, где

$\alpha_1 : x \le y \wedge y+z \le x \wedge z \ge 0 \wedge W_1 = W_2 - W_3$ – формула арифметики Пресбургера, а $\alpha_2 : W_2 = f(x) \wedge W_3 = f(y) \wedge f(W_1) \ne f(z)$ – формула теории равенства неинтерпретированных функциональных символов.

Нетрудно видеть, что каждая из формул α_1, α_2 выполнима, в то время как формула α невыполнима. Следовательно, необходим обмен информацией между алгоритмами, разрешающими каждую из этих теорий. Обмен информацией между алгоритмами заключается в передаче равенств переменных.

На втором этапе делается попытка вывести равенства переменных. Из формулы α_2 равенства невыводимы, а из формулы α_1 выводимо равенство $x = y$. Поэтому формула α_2 заменяется формулой $\alpha_2' = (\alpha_2 \wedge x = y)$, где

$$\alpha_2' : W_2 = f(x) \wedge W_3 = f(y) \wedge f(W_1) \ne f(z) \wedge x = y . \quad (3)$$

Из формулы α_2' можно вывести равенство $W_2 = W_3$. После передачи этого равенства формула α_1 заменяется формулой $\alpha_1' = (\alpha_1 \wedge W_2 = W_3)$, где

$$\alpha_1' : \tilde{o} \le y \wedge y+z \le x \wedge z \ge 0 \wedge W_1 = W_2 - W_3 \wedge W_2 = W_3 . \quad (4)$$

Из формулы α_2' можно вывести равенство $W = z$. Передавая равенство $W_1 = z$, заменяем формулу α_2' формулой $\alpha_1' \wedge W_1 = z$, которая очевидно невыполнима. Значит, исходная формула α также невыполнима. В общем случае, как только обнаружится невыполнимая формула, процесс передачи равенств прекращается, и исходная формула также невыполнима. Если в процессе передачи равенств происходит насыщение (то есть новых равенств не генерируется) и все формулы выполнимы, то исходная формула выполнима. Заметим, что равенство ρ выводимо из выполнимой формулы β (переменные из ρ принадлежат β) тогда и только тогда, когда истинна формула $\beta \Rightarrow \rho$, а это устанавливается с помощью разрешающего алгоритма.

4. Если же использования подформул общих теорий не имеется, то выполняется ввод аксиоматических теорий проблемных областей имеющихся во внешней спецификации, поставляемой Разработчиком.

5. Осуществление доказательства подформул с использованием аксиоматических теорий проблемных областей.

При доказательстве подформул с использованием аксиоматических теорий проблемных областей можно воспользоваться теорией систем подстановки термов (для эквациональной теории) или поиском необходимых аксиом по образцу [6–10].

При завершении работы заданной системы аксиом и выдается каноническая система подстановок термов, последняя используется для доказательства нужных свойств в виде равенств посредством процедуры приведения обеих частей равенства к нормальной форме.

При применении аксиом с помощью поиска по образцу осуществляется преобразование промежуточных формул, возникающих в процессе доказательства условий корректности, в соответствии с данной аксиомой. Аксиомы бывают двух типов: безусловные и условные. Они содержат выражение, называемое образцом, в котором выделены специальные переменные, помеченные волнистой чертой.

Безусловная аксиома имеет следующий вид:

$$e(\tilde{x}_1,..., \tilde{x}_n) \leftrightarrow e_0(x_1,..., x_n) , \quad (5)$$

где $e(\tilde{x}_1,..., \tilde{x}_n)$ – образец с выделенными переменными $\tilde{x}_1,..., \tilde{x}_n$.

При применении данной аксиомы к формуле α вначале определяется самая внутренняя подформула β формулы α (т.е. $\alpha = P(\beta)$), которая отождествляется с выражением $\breve{a}$, когда переменные $\tilde{x}_i$ заменены выражениями $\mathring{a}_i$, (т.е. $\beta = e(e_1,..., e_n)$). Затем в формуле α подформула β заменяется на $e_0(e_1,..., e_n)$, т.е. получается формула $P(e_0(e_1,..., e_n))$.

Условная аксиома имеет следующий вид:
если $cond(x_1,..., x_n)$, то

$$e(\tilde{x},..., \tilde{x}) \leftrightarrow e(x,..., x) . \quad (6)$$

Эта аксиома применяется к формуле α аналогично описанному выше применению безусловной аксиомы с той разницей, что проверяется условие $cond(x_1,..., x_n)$, которое нужно вывести из посылок формулы α. Заметим, что, если условие $cond$ содержит специальные переменные, которые не входят в образец e, то при применении аксиомы с ними можно сопоставлять любые выражения.

Если же при применении аксиомы к формуле α условие $cond(e_1,..., e_n)$ не выводится из посылок формулы α, то определяется новая подформула β (не обязательно самая внутренняя), которая отождествляется с образцом e.

Таким образом, на выходе имеются дока-

занные подформулы условий корректности.

6. На следующем шаге - осуществление проверки на наличие в формулах условий корректности не доказанных подформул. При определении наличия недоказанных подформул проводится модификация программы при наличии в ней шибок или недекларированных возможностей. При этом производится формирование условий корректности для модифицированной программы. Новые условия корректности упрощаются и доказываются, что определяет интеракционный характер анализа программы (верификации)

Наличие недоказанных условий возможно по двум причинам:

- все условия корректности истинны, но блок доказательства не может доказать часть условий (часто из-за недостатка информации о свойствах предметной области).

- среди условий корректности имеются ложные. Тогда исходная аннотированная программа модифицируется пользователем и подается на блок синтаксического анализа. В этом случае ошибки могут быть как в инвариантах циклов, так и в операторах исходной программы, причем формально разделить эти ситуации нельзя.

7. Если определено, что все подформулы доказанные, то определеяется частичная корректность программы на основании доказанности имеющихся условий корректности. При анализе условий корректности участвует оператор, проводящий верификацию, так как возможна необходимость выбора стратегии доказательства или ввода дополнительной информации в блок доказательства.

8. Определение завершения анализируемой программы. Свойство завершения вычислений в программе, т.е. достижение в процессе выполнения программы оператора STOP (выходной точки программы), как уже отмечалось, – одно из важных свойств корректности программы. Особенность свойства завершения состоит в том, что оно не зависит от постусловия программы и для фиксированной программы полностью определяется предусловием. В общем случае формальный анализ свойства завершения представляет весьма сложную проблему.

Имеется, по меньшей мере, две причины, по которым программа с заданным предусловием P может не завершаться (не обладать свойством завершения):

1) обращение к частично определенной функции со значением аргумента вне его области определения;

2) выполнение бесконечной последовательности операторов в цикле (зацикливание).

Первая причина может приводить к аварийному завершению даже нециклических программ (например, при выполнении деления на нуль). Вторая – присуща программам с итеративными циклами. Для логического анализа завершения циклических вычислений два наиболее распространенных метода: метод Флойда и метод счетчиков.

Первый метод может рассматриваться как дальнейшее развитие изложенного выше метода индуктивных утверждений, предложенного Р. Флойдом [11, 12]. Основная идея метода состоит в том, чтобы с каждой контрольной точкой программы, лежащей на циклическом пути, связать некоторую частичную функцию, значения которой ограничены.

Идея второго метода состоит во введении в программу *Prgm`* новых переменных-счетчиков, добавляемых по одной в каждый цикл программы. Переменная-счетчик должна инициализироваться перед входом в цикл и увеличивать свое значение при каждом прохождении по циклу. В преобразованной таким образом программе *Prgm`* измененные инварианты циклов представляются в виде, содержащем условие ограниченности значений счетчиков функциями, зависящими только от входных переменных Это указывает на существование верхней грани значений счетчиков. Для преобразованной программы с указанными инвариантами циклов методом индуктивных утверждений доказывается частичная корректность, которая одновременно характеризует и завершение программы, т.е. ограниченность значений счетчиков.

В отличие от метода Флойда в методике счетчиков не требуется вводить вспомогательные ограничивающие функции в контрольных точках. Переменные-счетчики рассматриваются как органическая часть программы, позволяющая логически выразить свойство завершения в индуктивных утверждениях (инвариантах цикла). Преимущества перед методом Флойда состоят в том, что одни и те же инварианты используются при доказательстве частичной корректности и завершения. Однако метод Флойда обладает большей общностью.

9. Проверка выполнения условия завершения программы.

10. Формирование заключения о полной корректности программы на основании полученных результатов о ее частичной корректности и выполнении условия завершения. Принятие решения о полной корректности означает окончание верификации программы.

Заключение. Таким образом, рассмотрен-

ный способ обеспечивает:

1) повышение вероятности определения недекларированных возможностей и ошибок $P_{\text{ндв}} = 0,98$ за счет использования аксиоматических теорий, позволяющих более полно формализовать предметную область и осуществить автоматизацию процесса верификации в отличие от известных аналогов;

2) повышение точности и достоверности получаемых результатов измерений комплексами радиоконтроля.

Библиографический список

1. *Shostak R.E.* Deciding linear inequalitities by computing loop residues // J. ACM. – 1981. – Vol. 28, №4. – P. 769 – 779.

2. *Shostak R.E., Schwartz R., Melliar-Smith P.M.* STP: A mechanized logic for specification and verification // Lect. Notes Comput. Sci. – 1984. – Vol. 170. – P. 1 – 42.

3. *Рабин М. О.* Разрешимые теории // Справочная книга по математической логике. – М.: Наука, 1982. – Т. 3. – С. 77 – 111.

4. *Nelson G., Oppen D.C.* Simplification by cooperating decision procedures // ACM Trans. Progr. Lang. Syst. – 1979. – Vol. 1, №2. – P. 245 – 257.

5. *Nelson G., Oppen D.C.* Fast desition procedures based on congruence closure // J. ACM. – 1980. – Vol. 27, № 2. – P. 356 – 364.

6. *Абрамов С.А.* Элементы анализа программ. – М.: Наука, 1986. – 128 с.

7. *Чень, Ли Р.* Математическая логика и автоматическое доказательство теорем. – М.: Наука, 1983. – 358 с.

8. *Ануреев И.С.* Интегрированные правила переписывания термов и их применение в автоматической верификации программ // Проблемы спецификации и верификации параллельных систем. – Новосибирск, 1995. – С. 185 – 213.

9. *Ющенко Е.Л., Касаткина И.В.* Современные методы доказательства правильности программ // Кибернетика, 1980. – № 6. – С. 37 – 62.

10. *Manna Z., Waldinger R.* The logic of computer programming // IEEE Trans. Software Eng. – 1978. – Vol. SE-4, № 3. –P. 199 – 229.

11. *Art K. R.* Ten Years of Hoars's Logic: a Survey Part 1//ACM Trans. Progr. Lang. Syst. – 1981. – Vol. 3, № 4. – P. 431 – 483.

12. *Хоггер К.* Введение в логическое программирование. – М.: Мир, 1988. – 348 с.

13. *Моисеев В.Л.* Метрологическое подтверждение пригодности программного обеспечения измерительных систем // Сборник материалов практической конференции: «Развитие менеджмента оборонных предприятий и повышение их конкурентоспособности на российском и зарубежном рынках товаров и услуг».

УДК 681.317.75:519.2

Д.А. Перепелкин

АЛГОРИТМ АДАПТИВНОЙ УСКОРЕННОЙ МАРШРУТИЗАЦИИ НА БАЗЕ ПРОТОКОЛА OSPF ПРИ ДИНАМИЧЕСКОМ ОТКАЗЕ ЭЛЕМЕНТОВ КОРПОРАТИВНОЙ СЕТИ

Предложен алгоритм адаптивной ускоренной маршрутизации на базе протокола OSPF при динамическом отказе элементов корпоративной сети, повышающий качество ее функционирования.

Ключевые слова: *адаптивная ускоренная маршрутизация, протокол OSPF, динамические изменения, алгоритмы маршрутизации, корпоративные сети.*

Введение. *Цель работы* – разработка нового эффективного алгоритма поиска оптимальных маршрутов на базе протокола OSPF при динамическом отказе элементов корпоративной сети, повышающего качество ее функционирования. Совершенствование сетевых технологий требует обеспечения качественного обслуживания передаваемого трафика. Загрузка и пропускная способность линий связи корпоративной сети динамически меняются, что, в свою очередь, может приводить к частой рассылке служебной информации об изменении маршрутов [1]. При сбоях или отказах маршрутизаторов и линий связи в корпоративной сети приходится полностью перестраивать таблицы маршрутизации. Одним из решений повышения качества функционирования корпоративных сетей является точное определение оптимальных маршрутов передачи данных и быстрое переключение более загруженных каналов связи на другие – сво-

бодные каналы, при динамическом отказе элементов корпоративной сети.

Теоретическая часть. Для повышения качества функционирования корпоративных сетей наиболее важной задачей является выбор эффективного алгоритма маршрутизации, который будет обеспечивать поиск оптимальных маршрутов с учетом различных свойств той или иной корпоративной сети. В настоящее время широкое применение получили алгоритмы адаптивной маршрутизации. Эти алгоритмы обеспечивают автоматическое обновление таблиц маршрутизации после изменения конфигурации сети.

Протокол OSPF (Open Shortest Path First) при решении задачи маршрутизации базируется на алгоритме состояния каналов. Характеристики и параметры качества обслуживания протокола OSPF подробно рассматриваются в работе [2]. Выбор оптимального маршрута в протоколе OSPF определяется по алгоритму Дейкстры. Трудоемкость построения таблиц маршрутизации с использованием этого алгоритма составляет порядка $O(N^2)$, где N – число маршрутизаторов в корпоративной сети.

В работе [3] предложен алгоритм парных переходов, позволяющий за счет сбора дополнительной информации учесть возможные изменения конфигурации корпоративной сети и не производить полный пересчет маршрутных таблиц. Это позволило снизить трудоемкость расчета таблиц маршрутизации до величины порядка $O(k \cdot N)$, где k – число фактически выполненных парных переходов.

При динамическом отказе элементов корпоративной сети использование данного алгоритма оказывается неэффективным, так как трудоемкость расчета дополнительной информации для осуществления парного перехода составляет $O(N^2 \log_2 N)$, что оказывается выше соответствующей оценки трудоемкости алгоритма Дейкстры.

Разработка новых, более эффективных алгоритмов адаптивной маршрутизации позволяет уменьшить трудоемкость построения таблиц маршрутизации в корпоративных сетях, использующих в своей работе протокол OSPF.

Разработка алгоритма. Для повышения качества функционирования корпоративных сетей на базе протокола OSPF предложен алгоритм адаптивной ускоренной маршрутизации, который позволяет уменьшить трудоемкость построения таблиц маршрутизации до величины $O(N)$ при динамическом отказе элементов корпоративной сети.

Представим корпоративную сеть в виде неориентированного взвешенного связного графа $G = (V,E,W)$, где V – множество вершин, $|V|=N$, E – множество ребер, $|E|=M$, W – множество весов ребер.

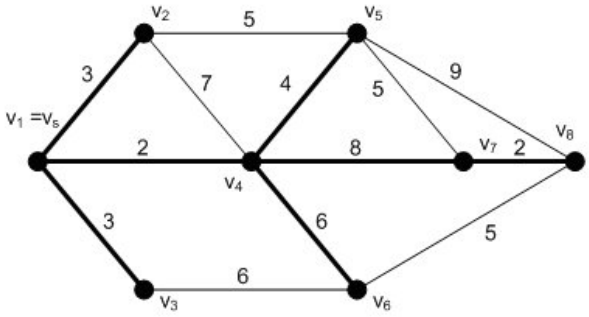


Рисунок 1 – Граф G корпоративной сети

Пусть на графе G в некоторый момент времени уже решена задача поиска кратчайших путей до всех вершин множества $V_{\dot{s}} = V \backslash \{v_s\}$ из начальной вершины v_s, т. е. построено дерево кратчайших путей с корнем в вершине v_s. Обозначим это дерево как T_g. На рисунке 1 жирными линиями обозначено построенное дерево кратчайших путей.

Рассмотрим множество ребер E графа G. По признаку вхождения ребер в дерево T_g можно разделить исходное множество E на два подмножества: $E_T \in T_g$ и $E_R \notin T_g$, $E_T \cup E_R = E$.

Множество ребер дерева E_T – множество ребер дерева T_g для графа G. Для заданного графа G, согласно свойству дерева, мощность множества E_T будет равняться мощности множества V минус единица $|E_T|=|V| - 1$.

Множество ребер замены для дерева E_R – множество ребер графа G, не вошедших в дерево T_g. При соответствующих условиях некоторое ребро $e_{i,j} \in E_R$, инцидентное вершинам v_i и v_j, может перейти во множество ребер дерева E_T, заменив собой некоторое ребро $e_{k,p} \in E_T$. При этом инцидентность ребра $e_{k,p}$ вершине v_i или v_j является обязательным условием. В свою очередь, ребро $e_{i,j}$ перейдет во множество E_R.

Будем называть такие переходы **парными переходами** и обозначать $e_{i,j} - e_{k,p}$.

Во множестве E_R можно выделить 2 подмножества.

Множество ребер замены E_S для дерева – это такое подмножество множества E_R, элементы-ребра которого участвуют, по крайней мере, в одном отношении парного перехода.

Множество непарных ребер E_P – это такое подмножество множества E_R, элементы-ребра которого не участвуют ни в одном отношении из множества R.

В общем случае множество E_P может быть пустым $|E_P| = 0$. Множество E_S будет пустым только при условии, что исходный связный граф G является деревом, и задача поиска кратчайших

путей в этом случае лишена смысла.

Для каждого ребра $e_{i,j} \in E$ на шкале значений весов определена точка вхождения в дерево $w_{i,j}^{t}$ и точка вхождения во множество замены $w_{i,j}^{s}$, причем $w_{i,j}^{t} \le w_{i,j}^{s}$.

Так, для графа G, представленного на рисунке 1, множество ребер дерева составляет $E_T = \{e_{1,2}, e_{1,3}, e_{1,4}, e_{4,5}, e_{4,6}, e_{4,7}, e_{7,8}\}$; множество ребер замены $E_S = \{e_{2,5}, e_{2,4}, e_{3,6}, e_{5,7}, e_{6,8}\}$; множество непарных ребер будет $E_P = \{e_{5,8}\}$. Если рассмотреть ребро $e_{4,5}$, то для него точка вхождения в дерево будет составлять 6, а точка вхождения во множество замены – 13. При этом данное ребро находится в отношении парного перехода с ребром $e_{2,5}$, а после попадания $e_{4,5}$ во множество непарных ребер эта парная перестановка примет вид $e_{2,5}$ - $e_{5,7}$.

Обозначим множество путей до вершины v_i из исходной вершины v_s через Π_i, где элемент множества $\pi_{i,k} \in \Pi_i$ будет множеством не повторяющихся ребер $e_{i,j} \in E$, образующих вместе путь, соединяющий v_s и v_i. Всем $\pi_{i,k} \in \Pi_i$ поставим в соответствие некоторое число, равное сумме весов входящих в него ребер, т.е. длину пути $d_{i,k} \in D_i$, где D_i представляет собой множество оценок кратчайших путей до вершины v_i из исходной вершины v_s. Кратчайший путь до вершины v_i будем обозначать π_i, а его оценку длины – d_i.

Для разработки алгоритма адаптивной ускоренной маршрутизации на базе протокола OSPF при динамическом отказе элементов корпоративной сети сформулируем следующие теоремы.

Теорема 1. При удалении некоторого ребра $e_{i,j}$, инцидентного вершинам V_i и V_j, входящего в дерево кратчайших путей, причем $d_i < d_j$, дерево кратчайших путей и их оценки до вершины V_i окажутся без изменения.

Доказательство. Вершины V_i и V_j входят в дерево кратчайших путей. Так как $d_i < d_j$, то $d_i + w_{i,j} = d_j$ и ребро $e_{i,j}$ не входит в путь π_i к вершине V_i. Поэтому при удалении ребра $e_{i,j}$ дерево кратчайших путей и их оценки до вершины V_i не изменятся. Теорема доказана.

Следствие. При удалении некоторого ребра $e_{i,j}$, инцидентного вершинам V_i и V_j, входящего в дерево кратчайших путей, причем $d_i < d_j$, необходимо определить новые кратчайшие пути для множества вершин, инцидентных вершине V_j.

Данному случаю соответствует граф, представленный на рисунке 2 при удалении ребра $e_{4,7}$.

На рисунке 2 жирными линиями обозначено дерево кратчайших путей, которое не требует изменения.

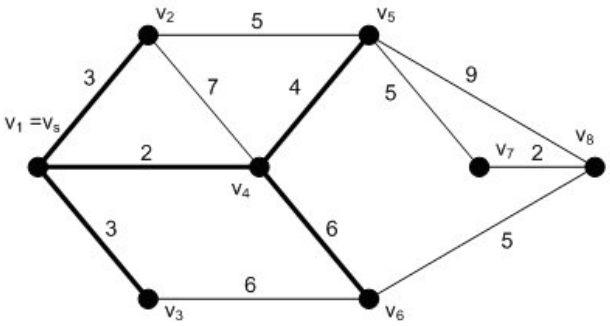


Рисунок 2 – Удаление ребра $e_{4,7}$

То есть при отказе ребра $e_{4,7}$ дерево кратчайших путей для вершин V_2, V_3, V_4, V_5, V_6 согласно теореме 1 не изменится. Для вершины V_7 и вершины V_8 необходимо построить новые кратчайшие пути с учетом отказа ребра $e_{4,7}$. Кратчайший путь до вершины V_7 изменится и составит $\pi_7 = \{e_{1,4}, e_{4,5}, e_{5,7}\}$, а его оценка $d_7 = 11$. Кратчайший путь до вершины V_8 составит $\pi_8 = \{e_{1,4}, e_{4,5}, e_{5,7}, e_{7,8}\}$, а его оценка $d_8 = 13$.

Теорема 2. При удалении некоторой вершины V_i для всех вершин, не инцидентных вершине i (то есть не имеющих ребра $e_{i,k}$), кратчайшие пути и их оценки окажутся без изменений.

Доказательство. Пусть вершина V_i не входит в кратчайший путь к некоторой вершине V_j. Тогда и в пути к этой вершине $\pi_{v,j}$ отсутствует ребро $d_{i,k}$. При удалении вершины V_i будут удалены все ребра $d_{i,k}$, которые не входят в путь $\pi_{v,j}$, а следовательно, кратчайший путь и его оценка окажется без изменения. Теорема доказана.

Следствие. При удалении некоторой вершины V_i необходимо определить новые кратчайшие пути для множества вершин, инцидентных вершине V_i, имеющих в своем кратчайшем пути ребро $e_{i,k}$.

Данному случаю соответствует граф, представленный на рисунке 3 при удалении вершины V_7.

На рисунке 3 жирными линиями обозначено дерево кратчайших путей, которое не требует изменения.

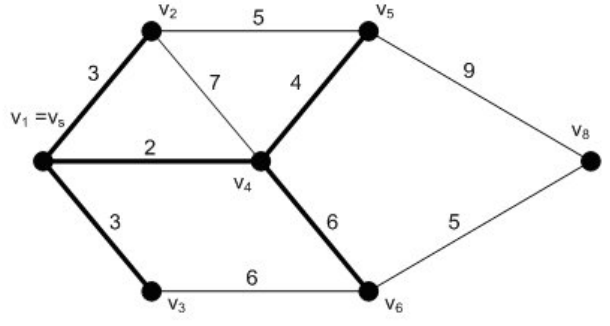


Рисунок 3 - Удаление вершины V_7

То есть при отказе вершины V_7 дерево кратчайших путей для вершин V_2, V_3, V_4, V_5, V_6 не изменится согласно теореме 2. Для вершины V_8 необходимо определить новый кратчайший путь с учетом отказа вершины V_7. Кратчайший путь до вершины V_8 составит $\pi_8 = \{e_{1,4}, e_{4,6}, e_{6,8}\}$, а его оценка $d_8 = 13$.

Использование доказанных выше теорем позволяет разработать алгоритм адаптивной ускоренной маршрутизации на базе протокола OSPF, уменьшающий размерность задачи поиска кратчайших путей при динамическом отказе элементов корпоративной сети.

Рассмотрим работу алгоритма адаптивной ускоренной маршрутизации на базе протокола OSPF в корпоративных сетях. Укрупненная схема алгоритма имеет следующий вид.

Шаг 1. Первоначальная инициализация исходных данных. Используя пакет HELLO протокола OSPF, определить веса линий связи $w_{i,j}$.

Шаг 2. Построить дерево оптимальных маршрутов корпоративной сети.

Шаг 3. Для вершины, являющейся листом дерева, производится поиск всех парных переходов без ограничений. Эти списки для удобства дальнейшей работы привязываются к вершине, инцидентной рассматриваемому ребру и расположенной ниже по иерархии.

Шаг 4. Если вершина не является листом дерева, то вычисляются парные переходы для этой вершины и выбираются лучшие значения потенциалов парных переходов для потомков вершины и собственных парных переходов. Подобная процедура выполняется для формирования списков парных переходов в случае динамического отказа элементов корпоративной сети.

Шаг 5. Для каждой вершины формируется полный список парных переходов. Число элементов в каждом из этих списков не превышает количества вершин графа. Такое решение позволяет отказаться от предварительной сортировки потенциалов или приращений для парных переходов без значительного усложнения алгоритма обработки изменения.

Шаг 6. Для каждого ребра графа корпоративной сети определить точку вхождения в дерево оптимальных маршрутов и точку вхождения во множество замены.

Шаг 7. Определить, есть ли пакеты на передачу:

а) если да, то перейти к шагу 8;

б) иначе – к шагу 16.

Шаг 8. Используя поля «Время жизни» и «Контрольная сумма заголовка» протокола IP, определить, требуется ли уничтожить (отбросить) данный пакет:

а) если да, то перейти к шагу 22;

б) иначе - к шагу 9.

Шаг 9. Используя поле «Тип сервиса», определить, требуется ли создание виртуального соединения:

а) если да, то перейти к шагу 10;

б) иначе – к шагу 12.

Шаг 10. Организовать виртуальное соединение, послав первый пакет «запрос вызова» адресату:

а) если пакет согласия на соединение от адресата пришел, то перейти к шагу 13;

б) если адресат отклонил вызов, то перейти к пункту 12.

Шаг 11. а) установить флаг передачи;

б) послать адресату пакет ликвидации соединения;

в) получить пакет подтверждения рассоединения от адресата.

Шаг 12. Проверка флага передачи:

а) если флаг установлен, то перейти к шагу 22;

б) иначе – к шагу 7.

Шаг 13. Используя поле «Тип сервиса», определить необходимую таблицу маршрутизации с учетом желаемого уровня качества обслуживания:

а) если таблица 1, то перейти к шагу 14;

б) иначе – к шагу 15.

Шаг 14. а) передать пакет, используя первую таблицу маршрутизации;

б) перейти к шагу 11.

Шаг 15. а) передать пакет, используя вторую таблицу маршрутизации;

б) перейти к шагу 11.

Шаг 16. Анализируя полученную протоколом OSPF информацию, определить, произошел ли отказ элементов корпоративной сети:

а) если да, то перейти к шагу 17;

б) иначе - к шагу 7.

Шаг 17. а) если удалилось ребро $e_{i,j}$ с весом $w_{i,j}$, причем вершина V_i располагается ниже по иерархии в дереве оптимальных маршрутов, чем вершина V_j, то возможны два случая:

1) ребро $e_{i,j}$ не входит в дерево кратчайших путей. Дерево кратчайших путей не изменится. Если ребро $e_{i,j}$ не входило во множество ребер замены, то оставить все без изменения. Ребро $e_{i,j}$ входило во множество ребер замены. Для вершины V_j определить новые ребра, входящие во множество ребер замены, и определить для них точки вхождения в дерево кратчайших путей. Исключить возможные пути замены, проходящие через ребро $e_{i,j}$, и добавить новые пути, проходящие через новые ребра замены для вершины V_j;

2) ребро $e_{i,j}$ входит в дерево кратчайших путей. Дерево кратчайших путей до вершины V_i и всех инцидентных ей вершин не изменится. Для вершины V_j из множества ребер замены выбрать ребро с минимальной оценкой и включить его в дерево кратчайших путей (выполнить парный переход). Исключить из множества путей замены все маршруты, проходящие через ребро $e_{i,j}$, и включить пути, проходящие через ребро замены. Для всех вершин, инцидентных вершине V_j (кроме вершины V_i), определить их новые оценки, выбрать ребро с минимальной длиной и включить его в дерево кратчайших путей. Определить инцидентные вершине V_j ребра, состоящие в отношении парного перехода, и включить их во множество замены. Для вершины V_j и всех инцидентных ей вершинам (кроме вершины V_i) определить новые точки вхождения ребер во множество замены и в дерево кратчайших путей;

б) если удалилась вершина V_k, имеющая связи с вершинами V_i и V_j, причем вершина V_i располагается ниже по иерархии в дереве оптимальных маршрутов, чем вершина V_j, то:

1) для всех вершин V_i, не инцидентных вершине V_k, т.е. не имеющих ребро $e_{i,k}$, дерево кратчайших путей не изменится;

2) для всех вершин V_j инцидентных вершине V_k, имеющих ребро $e_{k,j}$, не входящее в дерево кратчайших путей, дерево кратчайших путей не изменится. Исключить ребро $e_{k,j}$ из множества ребер замены. Исключить все возможные пути замены, проходящие через ребро $e_{k,j}$;

3) для всех вершин V_m, инцидентных вершине V_k, имеющих ребро $e_{k,m}$, входящее в дерево кратчайших путей, сделать парные переходы из множества замены для ребра $e_{k,m}$. Исключить ребро $e_{k,m}$ из множества ребер графа. Исключить все возможные пути замены, проходящие через ребро $e_{k,m}$.

Шаг 18. Используя список парных переходов, определить, требуется ли сделать парный переход:

а) если да, то перейти к шагу 19;

б) иначе – к шагу 20.

Шаг 19. Для каждой вершины, у которой в список возможных маршрутов входит ребро с изменившейся метрикой, определить путь минимальной длины и поместить его в дерево кратчайших путей.

Шаг 20. Построить новое дерево оптимальных маршрутов с учетом изменений.

Шаг 21. Сформировать таблицы маршрутизации.

Шаг 22. Проверка окончания работы маршрутизатора:

а) если да, то перейти к шагу 23;

б) иначе – сбросить флаг передачи и перейти к шагу 7.

Шаг 23. Завершение работы маршрутизатора.

Результаты моделирования. Для подтверждения правильности предложенного алгоритма адаптивной ускоренной маршрутизации на базе протокола OSPF при динамическом отказе элементов корпоративной сети разработана программа моделирования процессов маршрутизации.

При разработке основное внимание уделялось корректности предлагаемого алгоритма и размерности решаемой задачи.

Для каждого испытания на множестве обработанных изменений выбирались минимальное, максимальное и среднее значения размерности задачи, выраженные через количество вершин, для которых необходим поиск кратчайшего пути. Для каждого эксперимента были найдены значения оценок математического ожидания и среднего квадратичного отклонения числа изменений. Для алгоритма адаптивной ускоренной маршрутизации на базе протокола OSPF определялось число фактически выполненных парных переходов при динамическом отказе элементов корпоративной сети.

На рисунках 4 – 6 представлены результаты моделирования алгоритма адаптивной ускоренной маршрутизации на базе протокола OSPF.

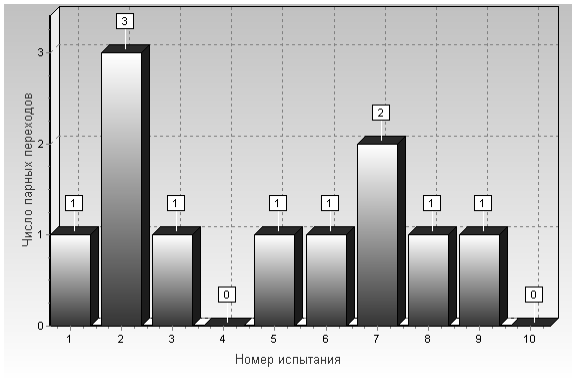


Рисунок 4 - Число изменений дерева в графе из 10 вершин

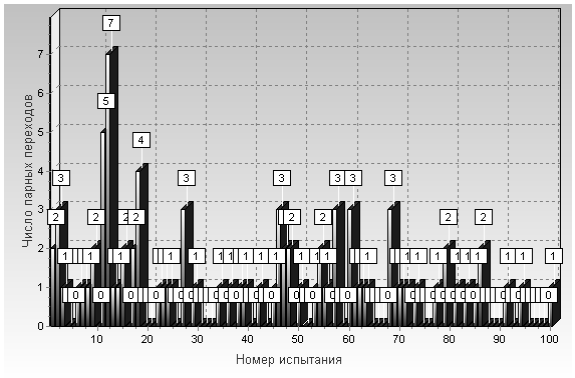


Рисунок 5 - Число изменений дерева в графе из 100 вершин

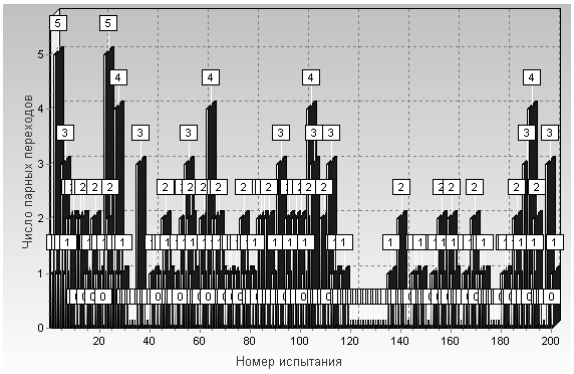


Рисунок 6 - Число изменений дерева в графе из 200 вершин

В таблице приведены обобщенные статистические характеристики для средней размерности задачи. В представленной таблице через СКО обозначено значение оценки среднего квадратичного отклонения.

Число вершин графа	Min значение	Max значение	Значение оценки МО	Значение оценки СКО
10	0	0,3	0,1375	0,0943
100	0	0,07	0,0531	0,1368
200	0	0,025	0,0155	0,1173

Были проведены исследования графов, состоящих из 10, 100 и 200 вершин. Исследование разработанного алгоритма адаптивной ускоренной маршрутизации на базе протокола OSPF показало, что максимальное значение числа изменений существенно ниже размерности для каждого из рассмотренных графов, а значение оценки математического ожидания не превышает единицы. Более того, обнаружена тенденция уменьшения значения оценки математического ожидания числа изменений дерева оптимальных маршрутов с увеличением количества вершин графа.

При динамическом отказе элементов корпоративной сети разработанный алгоритм адаптивной ускоренной маршрутизации на базе протокола OSPF в отличие от алгоритма Дейкстры позволяет производить перестроение таблиц маршрутизации не полностью, а только той ее части, в которой произошли изменения. При этом трудоемкость изменения дерева кратчайших путей является линейной функцией от числа вершин и определяется выражением $O(N)$. То есть при отказе элементов корпоративной сети необходимо один раз просмотреть все вершины графа сети и на основе предварительно собранной информации о парных переходах выполнить изменения для вычисления новых оптимальных маршрутов. Если рассмотреть полученные результаты при моделировании предложенного алгоритма адаптивной ускоренной маршрутизации и сравнить их с алгоритмом Дейкстры, то видно:

1) для графа из 10 вершин максимальное число парных переходов составляет три изменения (= 3), при этом для алгоритма Дейкстры при каждом отказе элементов сети трудоемкость построения таблицы маршрутизации составляет $O(N^2) = 10^2 = 100$ элементарных операций;

2) для графа из 100 вершин максимальное число парных переходов составляет семь изменений (= 7), при этом для алгоритма Дейкстры при каждом отказе элементов сети трудоемкость построения таблицы маршрутизации составляет $O(N^2) = 100^2 = 10000$ элементарных операций;

3) для графа из 200 вершин максимальное число парных переходов составляет пять изменений (= 5), при этом для алгоритма Дейкстры при каждом отказе элементов сети трудоемкость построения таблицы маршрутизации составляет $O(N^2) = 200^2 = 40000$ элементарных операций.

На основе этого можно сделать вывод, что предложенный алгоритм адаптивной ускоренной маршрутизации на базе протокола OSPF является эффективным при поиске оптимальных маршрутов в условиях динамических изменений в структуре корпоративной сети и нагрузках на линиях связи за счет использования дополнительной информации о возможных парных переходах.

Заключение. Разработанный алгоритм адаптивной ускоренной маршрутизации на базе протокола OSPF позволяет повысить качество функционирования корпоративных сетей за счет уменьшения трудоемкости построения таблиц маршрутизации до величины порядка $O(N)$ при динамическом отказе элементов корпоративной сети.

Библиографический список

1. *Куракин Д.В.* Маршрутизаторы для глобальных телекоммуникационных сетей и реализуемые в них алгоритмы // Информационные технологии. 1996. № 2.

2. *Перепелкин Д.А.* Алгоритм адаптивной ускоренной маршрутизации на базе протокола OSPF при динамическом добавлении элементов корпоративной сети // Вестник Рязанского государственного радиотехнического университета. № 4 (выпуск 34). 2010. С. 65-71.

3. *Уваров Д.В., Перепелкин А.И.* Построение дерева кратчайших путей на основе данных о парных переходах // Системы управления и информационные технологии. № 4 (16), Москва-Воронеж. 2004. С. 93-96.

УДК 681.3

П.А. Баранчиков, Е.А. Баранчикова

ОПТИМИЗАЦИЯ ПОИСКА РЕГУЛЯРНЫХ ВЫРАЖЕНИЙ В БАЗЕ ДАННЫХ С ПОМОЩЬЮ ИНДЕКСНОЙ ТАБЛИЦЫ

Рассматривается вычислительная эффективность поиска регулярных выражений, содержащихся в таблице базы данных, по соответствию входному строковому значению. Предложен подход к оптимизации по времени такого поиска с использованием индексной таблицы, позволяющий заменить вычисление соответствия регулярного выражения строковой константе сравнением двух строковых значений. Рассчитана предполагаемая производительность предложенного подхода. Экспериментальные результаты показали увеличение скорости поиска до 10^3 раз.

Ключевые слова: *базы данных, регулярные выражения, поиск, оптимизация запросов.*

Введение. При обработке больших объемов данных часто бывает необходимо реализовать поиск регулярных выражений, которым соответствует заданная строка. Самым простым способом решения данной задачи можно было бы считать перебор всех регулярных выражений и опробование их на предмет соответствия входной строке, но при этом время поиска велико и соответствует единицам минут, что показано ниже.

Цель работы — оптимизация по времени поиска регулярного выражения в таблице, соответствующего входному слову.

Обозначим вычислительную сложность одной операции проверки регулярного выражения как t_{re}. Обозначим вычислительную сложность одной операции посимвольного сравнения двух строк как t_{sc}. Пусть в словаре регулярных выражений имеется n_{re} записей. Тогда выборка необходимых регулярных выражений будет иметь следующую трудоемкость:

$$O(n) = t_{re} \cdot n_{re}. \qquad (1)$$

Пусть в исходном алфавите имеется al_n различимых букв. Тогда можно создать индекс из некоторого количества начальных букв слова для последующего поиска по началу входного слова. Поиск будет осуществлен по следующему алгоритму.

1. Выборка всех регулярных выражений, у которых в индексной таблице есть начало слова, совпадающее с началом входного слова.

2. Поиск по выбранным регулярным выражениям подходящих путем перебора (поочередного применения регулярного выражения).

Возьмем для индексации l букв при количестве букв алфавита al_n. Тогда при выборке по индексам будет выбрано $n_{ост}$ потенциально подходящих регулярных выражений.

$$n_{ост} = \frac{n_{re}}{al_{n^l}}. \qquad (2)$$

При этом размер n_{ind} таблицы индексов будет превышать размер исходного словаря регулярных выражений n_{re} в b^l раз, где b — среднее количество вариантов букв, которые могут быть сопоставлены в одной позиции слова, соответствующего регулярному выражению.

$$n_{ind} = n_{re} \cdot b^l. \qquad (3)$$

Вычислительная сложность O_{re} выбора регулярных выражений путем сравнения с входным словом будет рассчитана так:

$$O_{re}(n) = n_{ост} \cdot t_{re}. \qquad (4)$$

Вычислительная сложность O_{ind} выборки таких индексов рассчитывается по формуле

$$O_{ind}(n) = n_{ind} \cdot t_{sc}. \qquad (5)$$

В таком случае рассчитаем общую вычислительную сложность выборки с использованием индексирования:

$$O(n) = n_{ind} \cdot t_{sc} + n_{ост} \cdot t_{re},$$

$$O(n) = n_{re} \cdot b^l \cdot t_{sc} + \frac{n_{re}}{al_{n^l}} \cdot t_{re}. \qquad (6)$$

Таким образом, получена формула, по которой можно определить вычислительную сложность алгоритма поиска регулярных выражений,

соответствующих входному слову. Проанализируем значения в формуле.

1. Количество регулярных выражений n_{re} — величина, задаваемая алгоритмом генерации регулярных выражений. Изменять ее не представляется возможным.

2. Количество букв в алфавите *al* неизменно.

3. Время сравнения двух строк t_{sc} и проверки соответствия слова регулярному выражению t_{re} определяются опытным путем и изменению не подвергаются.

4. Среднее количество комбинаций букв на одну позицию регулярного выражения *b* также определяется опытным путем.

Единственным изменяемым параметром является количество *l* букв, задействованных в индексации. При этом очевидно, что при увеличении *l* сильно растет первое слагаемое в формуле (6), а при уменьшении — второе. Следовательно, ставится задача минимизации вычислительной сложности алгоритма при варьировании количества букв индекса.

Создание индекса РВ. Для использования индексов регулярных выражений, общий принцип которых описан выше, необходимо создать сами индексы. Индексные значения состоят из набора букв, с которых может начинаться слово, соответствующее этому индексному значению.

Алгоритм создания индекса может быть описан следующим образом.

1. Создание списка слов, которые соответствуют данному РВ.

2. Для каждого слова из полученного списка определяется начальная часть (усечением строки по фиксированной длине) и записывается в индексную таблицу.

Этот алгоритм изображен на рисунке 1.

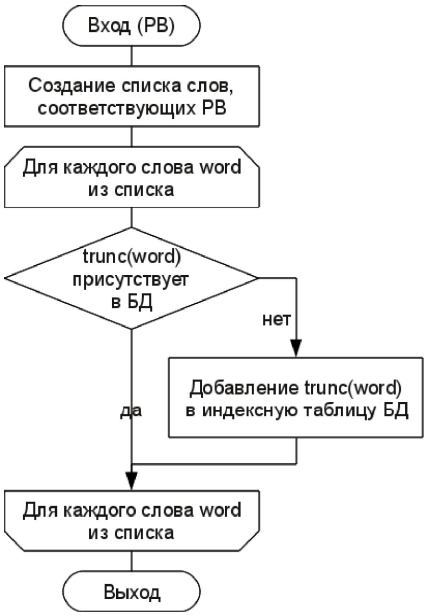


Рисунок 1 — Алгоритм добавления индексов в БД

Современные средства разработки предлагают программистам широкий функционал работы с различными списками, в том числе, удаление повторяющихся элементов. Таким образом, можно вынести операцию проверки на существование текущего индекса в таблице в виде операции удаления дублирующих значений списка перед добавлением в БД. Алгоритм изображен на рисунке 2. Такой подход позволит сократить сетевой трафик и сконцентрировать все вычисления на одном сетевом узле, в данном случае — клиентском рабочем месте.

Рисунок 2 — Алгоритм добавления индексов с очисткой дублирующих значений

В обоих случаях идет речь про индексацию одного РВ. Следовательно, уникальность индексных выражений понимается в рамках создания индекса для одного заданного РВ.

Приведем пример действия последнего алгоритма. Пусть необходимо добавить запись с регулярным выражением «(за|от)кры(ва)?ть». Словами, соответствующими этому РВ, будут:

1) закрывать;
2) закрыть;
3) открывать;
4) открыть.

Пусть индекс составляется по первым 4-м буквам слова. Тогда этот список далее преобразуется в список усеченных слов:

1) закр;
2) закр;
3) откр;
4) откр.

Как видно, имеются повторяющиеся слова (начала слов). После применения операции удаления дублирующих значений из списка слов получим:

1) закр;

2) откр.

После этого оба слова будут добавлены в индексную таблицу БД как соответствующие исходному регулярному выражению «(за|от)кры(ва)?ть».

Ввиду того, что РВ предполагается сгенерировать автоматически на основе синтаксиса русского языка, ставится задача именно генерации слов по РВ в данном алгоритме, а не наоборот.

Для того, чтобы обеспечить работу приведенных алгоритмов, необходимо иметь возможность создания списка слов, которые удовлетворяют исходному регулярному выражению.

Генерация слов по РВ. В литературе и документации есть множество статей по тому, как следует составлять РВ, однако нигде не поднимается вопрос генерации списка слов, которые удовлетворяют исходному РВ. Это связано с тем, что по синтаксису РВ могут соответствовать бесконечному количеству слов. Примером могут служить РВ с квантификаторами без ограничения количества повторений: «+», «*» и «{*n*,}».

Так как РВ в разрабатываемой системе созданы для отображения вполне конечного количества слов, существует возможность создать алгоритм генерации соответствующих слов, введя некоторые ограничения.

Количество повторений фрагментов РВ должно быть конечным. Если оно будет бесконечным, необходимо заменить максимальное количество повторений фрагмента каким-либо фиксированным значением.

В результате будет получена совокупность РВ, соответствующих конечному подмножеству слов русского языка.

В регулярные выражения Perl [1] можно представить как набор фрагментов с сопоставленными им квантификаторами, определяющими количество повторений фрагмента. Квантификаторы описаны явно в руководстве man perlre. К фрагментам следует отнести следующие:

1) группу (обозначенную круглыми скобками);

2) класс символа (обозначенный квадратными скобками);

3) простой одиночный символ (если не входит в предыдущие виды фрагментов).

Таким образом, выделив фрагмент и его квантификатор, можно повторить фрагмент необходимое количество раз и удалить квантификатор, а затем обработать сам фрагмент. Как при обработке квантификатора, так и при обработке фрагмента в результате будет получено несколько (множество) слов.

Разработанный алгоритм обработки РВ приведен на рисунке 3.

Рисунок 3 — Алгоритм генерации слов по РВ

Обработка подставленных фрагментов определяется типом фрагмента:

1) если используется класс символа (квадратные скобки) — подставляются все символы данного класса;

2) если используется группа с альтернативами (знаки «|» в круглых скобках), подставляются все возможные альтернативы;

3) если фрагмент является одиночным символом, подставляется этот символ.

Особо следует описать группу (круглые скобки). Так как в группе может находиться внутри любая другая группа, класс и/или квантификатор, можно утверждать, что группа позволяет осуществлять инкапсуляцию. Потому все подставляемые альтернативы необходимо также обработать как регулярное выражение. То есть для каждой альтернативы надо опять вызвать алгоритм генерации слов по РВ. Таким образом, разработанный алгоритм является косвенно-рекурсивным (через другой алгоритм).

Рассмотрим пример генерации слов на примере регулярного выражения «(за|от)кры(ва)?ть».

1. Выделяем первую группу: «(за|от)».

2. Выделяем ее квантификатор. Он отсутствует, следовательно, количество повторений группы — 1.

3. Осуществляем подстановку группы в соответствии с квантификатором. Получается исходное выражение.

4. Обрабатываем саму группу «(за|от)». Получаем список слов:

1) «закры(ва)?ть»;

2) «откры(ва)?ть».

5. Ищем следующую группу или квантификатор. Это группа «(ва)» с квантификатором «?».

6. Осуществляем подстановку группы в количестве, указанном в квантификаторе: 0 или 1.

1. «закрыть».

2. «закры(ва)ть».

3. «открыть».

4. «откры(ва)ть».

7. Последним шагом осуществляется обработка группы в словах 2 и 4, так как в 1 и 3 ее уже нет. Группа является простой последовательностью, следовательно, обработка заключается в удалении скобок. В результате получена ожидаемая последовательность слов.

1. «закрыть».

2. «закрывать».

3. «открыть».

4. «открывать».

Потоковая версия генерации слов по РВ. Алгоритм, приведенный выше, имеет ряд недостатков, осложняющих программирование. Например, он подразумевает создание списка всех возможных вариантов альтернатив и подстановку их в результаты. В таком случае потребуется опять создавать список слов, заменяя в каждом из них выделенный фрагмент на очередную альтернативу из списка запрошенных альтернатив.

Суть потокового подхода обработки заключается в обработке лишь конечной части слова («хвоста»). Изначально «хвостом» принимается все исходное регулярное выражение. Далее идет его посимвольная обработка. После обработки первого символа или фрагмента с его квантификатором вызывается этот же алгоритм, но ему передается лишь «хвост» — не обработанная часть РВ. Такой алгоритм отображен на рисунке 4.

В приведенном алгоритме под тривиальностью квантификатора *q* подразумевается, что либо он явно не задан, либо он задает количество повторений 1. Под тривиальностью фрагмента *f* подразумевается, что он представляет собой одиночный символ.

Алгоритмы, которые приведены ниже, построены однотипно. На входе каждого алгоритма одно регулярное выражение, а на выходе — список возможных слов, соответствующих этому регулярному выражению. То есть алгоритмы возвращают уже обработанные РВ, в которых далее подставлять ничего не следует: возвращают слова (части слов).

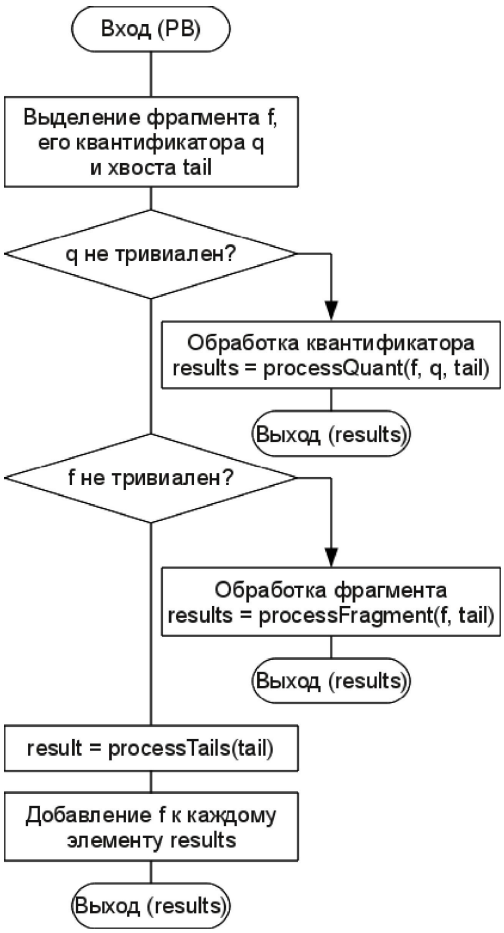


Рисунок 4 — Алгоритм потоковой генерации слов по РВ processTails

1. В начале алгоритма создается пустой список результатов.

2. Определяется множество вариантов последовательностей символов, которые получены из входного фрагмента.

3. Каждая из полученных последовательностей символов сцепляется с «хвостом» *tail*.

4. Для каждой из сцепленных последовательностей вызывается алгоритм *processTails*.

5. Результаты аккумулируются в списке результатов, который в конце алгоритма возвращается.

Таким образом, каждый из разработанных алгоритмов обработки фрагмента или квантификатора рекурсивно вызывает алгоритм *processTails*, который вызывает, в свою очередь, один из трех этих алгоритмов. То есть, предложенный алгоритм является косвенно-рекурсивным. Рассмотрим эти алгоритмы подробнее.

Алгоритм обработки квантификатора, изображенный на рисунке 5, осуществляет подстановку фрагмента f в поток символов количество раз, предусмотренное данным квантификатором.

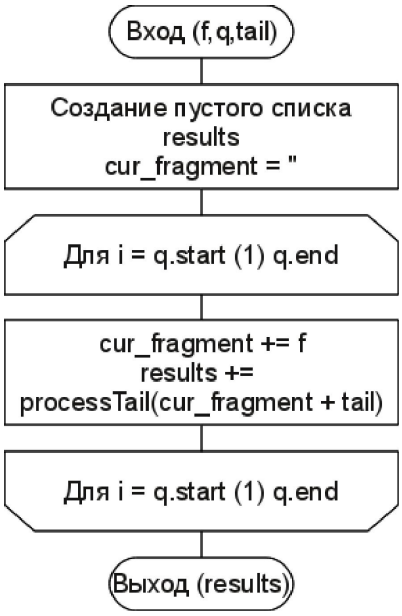


Рисунок 5 — Алгоритм обработки квантификатора processQuant

Алгоритм *processFragment* в зависимости от типа фрагмента (начинается ли он с открытой квадратной или круглой скобки) обрабатывает фрагмент необходимым образом. Обработка фрагмента с квадратной скобкой осуществляется с помощью вызова алгоритм *processSquareBrackets*, отображенного на рисунке 6. Алгоритм просто подставляет в различные элементы выходного списка символы, содержащиеся в классе символа, описанном квадратными скобками.

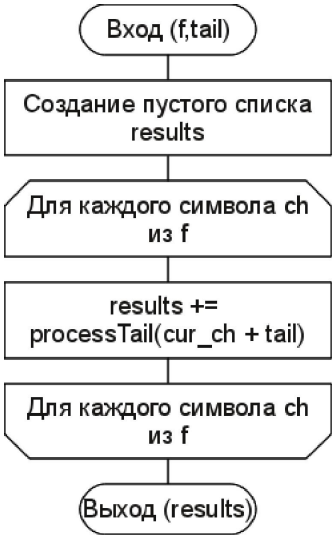


Рисунок 6 — Алгоритм обработки фрагмента в квадратных скобках processSquareBrackets

Алгоритм обработки круглых скобок приведен на рисунке 7. Алгоритм сначала выделяет возможные альтернативы и осуществляет их подстановку в списке выходных слов.

Рассмотрим работу разработанного алгоритма на вышеприведенном примере регулярного выражения «(за|от)кры(ва)?ть». В алгоритме указаны вызовы алгоритмов с указанием номера вызова в верхнем индексе.

Рисунок 7 — Алгоритм обработки круглых скобок processRoundBrackets

1. processTail[1] («(за|от)кры(ва)?ть»).
2. Выделение фрагмента и квантификатора, стоящих с начала РВ: *f* = «(за|от)», *q* = «», *tail* = «кры(ва)?ть».
3. q тривиален. Следует вызов *processFragment*[2] («(за|от)», «кры(ва)?ть»).
4. Выделяются альтернативы: «за» и «от».
5. Альтернативы дописываются к «хвосту»: «закры(ва)?ть» и «откры(ва)?ть».
6. *processTail*[3] («закры(ва)?ть»).
7. *q* и *f* тривиальны, сохраняется f^3 = «з». Вызов *processTail*[4](«акры(ва)?ть»).
8. *q* и *f* тривиальны, сохраняется f^4 = «а». Вызов *processTail*[5](«кры(ва)?ть»).
9. *q* и *f* тривиальны, сохраняется f^5 = «к». Вызов *processTail*[6](«ры(ва)?ть»).
10. *q* и *f* тривиальны, сохраняется f^6 = «р». Вызов *processTail*[7](«ы(ва)?ть»).
11. *q* и *f* тривиальны, сохраняется f^7 = «ы». Вызов *processTail*[8](«(ва)?ть»).
12. *q* не тривиален. Вызов *processQuantifier*[9] («(ва)», «?», «ть»).
13. Квантификатор определяет повторение последовательности 0 или 1 раз.
14. Варианты дописываются к «хвосту»: «(ва)ть», «ть».
15. Вызов *processTail*[10](«(ва)ть»). Квантификатор тривиален, фрагмент не тривиален.
16. Вызов *processFragment*[11](«(ва)», «ть»).
17. Выделяется одна альтернатива: «ва». Дописывается к «хвосту».
18. Вызов *processTail*[12](«вать»).
19. *q* и *f* тривиальны, сохраняется f^{12} = «в».

Вызов $processTail^{13}$(«ать»).

20. q и f тривиальны, сохраняется f^{13} = «а». Вызов $processTail^{14}$(«ть»).

21. q и f тривиальны, сохраняется f^{14} = «т». Вызов $processTail^{15}$(«ь»).

22. q и f тривиальны. Строка завершилась. Возвращается список из одного элемента «ь».

23. $processTail^{14}$ — добавляется «т». Возвращается «ть».

24. $processTail^{13}$ — добавляется «а». Возвращается «ать».

25. $processTail^{12}$ — добавляется «в». Возвращается «вать».

26. $processFragment^{11}$: возвращает «вать».

27. $processTail^{10}$ возвращает «вать».

28. Аналогично обрабатывается и второй вариант из $processQuantifier^{9}$, возвращая «ть».

29. $processQuantifier^{9}$ возвращает {«вать», «ть»}.

30. $processTail^{7}$ — добавляется «ы». Возвращается {«ывать», «ыть»}.

31. $processTail^{6}$ — добавляется «р». Возвращается {«рывать», «рыть»}.

32. $processTail^{5}$ — добавляется «к». Возвращается {«крывать», «крыть»}.

33. $processTail^{4}$ — добавляется «а». Возвращается {«акрывать», «акрыть»}.

34. $processTail^{3}$ — добавляется «з». Возвращается {«закрывать», «закрыть»}.

35. Аналогичным образом обрабатывается второй элемент в $processFragment^{2}$ и возвращается {«открывать», «открыть»}.

36. $processFragment^{2}$ возвращает {«закрывать», «закрыть», «открывать», «открыть»}.

37. $processTail^{1}$ возвращает {«закрывать», «закрыть», «открывать», «открыть»}.

Как видно, алгоритм вернул ожидаемые 4 слова.

Практическая реализация. На основе предложенного подхода было реализовано программное обеспечение, идентифицирующее регулярные выражения в письмах электронной почты [2, 3]. Проведенные тесты разработанного программного обеспечения показали существенное увеличение производительности поиска РВ при введении рассмотренного индексирования.

На рисунке 8 представлена производительность поиска регулярных выражений t в БД при индексации с различной длиной индекса n. Значение индекса 0 соответствует поиску регулярных выражений без индекса (простым перебором).

Как видно из графика, использование индексной таблицы с длиной индекса n = 5 дает увеличение производительности при обработке текстовой информации до 1000 раз по сравнению с простым перебором. При длине индекса n = 0 время обработки одного письма электронной почты составляло около 1,5 — 2 минуты на ПК Intel Core 2 Duo 2,4 MHz, 4 Гб ОЗУ, что являлось неприемлемым в рамках поставленной задачи. При использовании индекса, равного 5, время обработки на том же ПК было уменьшено до величины около 0,5 секунды.

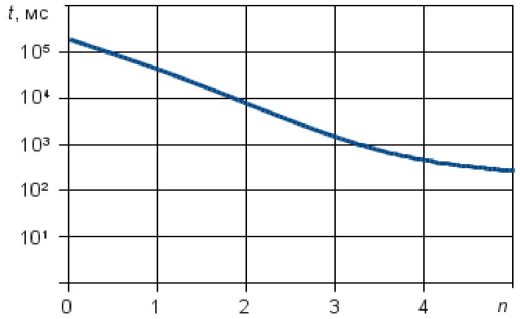


Рисунок 8 — Зависимость производительности обучения фильтра от длины индекса

Выводы. Разработанный подход позволяет ускорить поиск регулярных выражений в БД до 10^3 раз. Разработаны алгоритмы, осуществляющие генерацию слов по регулярному выражению для создания индексов. Практические разработки показали высокую эффективность предложенного подхода.

Библиографический список

1. *Randal L. Shwartz, Tom Phoenix* Leaning Perl, 3rd edition. - O'Relly&Associates, 2001. - 330 pp.

2. *Баранчикова Е.А.* Метод фильтрации электронной почты на основе теоремы Байеса с применением регулярных выражений//Новые информационные технологии в научных исследованиях и образовании: материалы 14 Всероссийской научно-технической конференции студентов, молодых ученых и специалистов. Рязанский государственный радиотехнический университет. - 2009. - С. 154 - 156.

3. *Баранчикова Е.А.* Алгоритм автоматической генерации регулярных выражений (РВ) для спам-фильтра на основе обучающей выборки // Информационные и телекоммуникационные технологии. Материалы 34-й Всероссийской научно-технической конференции. Часть 1. Рязань: РВВКУС. - 2009. - С. 380-381.