УДК 004.052

М. С. СОЛОДОВНИК, А. М. АСЛАНОВ

Одесская национальная академия пищевых технологий, Украина

СРАВНИТЕЛЬНЫЙ АНАЛИЗ ТРАДИЦИОННЫХ АЛГОРИТМОВ МАРШРУТИЗАЦИИ В КОМПЬЮТЕРНОЙ СЕТИ

В статье проанализированы действующие алгоритмы решения задач поиска оптимального маршрута в компьютерной сети. Рассмотрены алгоритмы Флойда, Дейкстры и алгоритм поиска оптимального маршрута путем возведения матрицы маршрутов в степень максимального ранга. При рассмотрении каждого из алгоритмов, приведен пример его реализации на примере графа, представляющего собой компьютерную сеть, а также выявлены преимущества и недостатки рассмотренных алгоритмов. Отмечено, что наряду с рассмотренными классическими алгоритмами, существуют и интеллектуальные алгоритмы, преимуществами которых являются – учет мнения эксперта и способность к самообучению.

***Ключевые слова**: алгоритм Флойда, алгоритм Дейкстры, оптимальный маршрут, самообучение, матрица расстояний, максимальный ранг, вершина графа, компьютерная сеть, метка вершины графа.*

Введение

Статья посвящена сравнительному анализу классических алгоритмов поиска оптимального маршрута в локальной компьютерной сети и определения наилучшего алгоритма. Данный вопрос является актуальным т.к. развитие компьютерных сетей влечёт за собой необходимость развития алгоритмов поиска оптимальных маршрутов передачи данных. С учетом большого количества действующих методов и алгоритмов маршрутизации становится актуальной задача сравнительного анализа с выявлением достоинств и недостатков типовых подходов. Известно, что передача информации по каналам связи происходит с учетом весовых характеристик линий связи. Весовой характеристикой линии связи могут выступать различные параметры: качество передачи, число транзитных участков, надежность передачи, время передачи, расстояние, запаздывание и т.д. Для передачи информации по сети от узла к узлу компьютерная сеть выбирает оптимальный маршрут по заданным критериям оптимальности и, соответственно, по определенному алгоритму. Рассмотрим несколько наиболее известных алгоритмов поиска оптимального маршрута от узла к узлу в компьютерной сети.

Постановка задачи

Цель работы состоит в анализе существующих алгоритмов поиска оптимального маршрута в компьютерной сети для повышения эффективности их функционирования и выбор наилучшего из них.

Алгоритм поиска оптимального маршрута путем возведения матрицы L в степень максимального ранга

Первым из рассмотренных нами алгоритмов, будет алгоритм поиска оптимального пути от узла к узлу (типовой подход) путем возведения матрицы L в степень максимального ранга. В качестве исходных данных, для примера решения задачи, примем значения, характеризующие задержку между узлами сети, измеряемые в миллисекундах (мс).

Есть компьютерная сеть, имеющая вид графа, изображенного на рис. 1 и матрица L, отображающая числовые значения непосредственных связей данной сети [2].

$$L = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 60 & \infty & 60 & \infty \\ 60 & 0 & 50 & 80 & 90 \\ \infty & 50 & 0 & \infty & 70 \\ 60 & 80 & \infty & 0 & 30 \\ \infty & 90 & 70 & 30 & 0 \end{bmatrix} \end{array} \quad (1)$$

Возведение матрицы L в степень максимального ранга R_{max} даст нам матрицу оптимальных путей от узла к узлу между всеми парами узлов графа, матрицу оптимальных путей $L_{опт} = L^{(Rmax)}$.

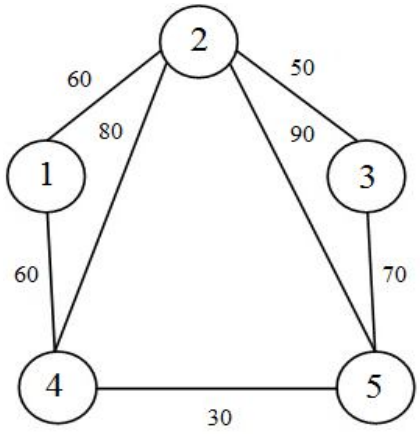

Рис. 1. Компьютерная сеть в виде графа

Если же при возведении матрицы L в некоторую степень q окажется, что

$$L^q = L^{(q-1)}, \qquad (2)$$

то вычислительный процесс следует прервать, так как тождество (2) влечет за собой тождество

$$L^q = L^{(q+1)}. \qquad (3)$$

Итак, вычислим для начала $L^{(2)}$ (4) – (6).

$$l_{12}^{(2)} = \min(l_{11}^{(1)} + l_{12}^{(1)}, l_{12}^{(1)} + l_{22}^{(1)}, l_{13}^{(1)} + l_{32}^{(1)}, l_{14}^{(1)} + \\ + l_{42}^{(1)}, l_{15}^{(1)} + l_{52}^{(1)}) = \min(60, 60, \infty, 140, \infty) = 60, \qquad (4)$$

$$l_{13}^{(2)} = \min(l_{11}^{(1)} + l_{13}^{(1)}, l_{12}^{(1)} + l_{23}^{(1)}, l_{13}^{(1)} + l_{33}^{(1)}, l_{14}^{(1)} + \\ + l_{43}^{(1)}, l_{15}^{(1)} + l_{53}^{(1)}) = \min(\infty, 110, \infty, \infty, \infty) = 110, \qquad (5)$$

$$l_{14}^{(2)} = \min(l_{11}^{(1)} + l_{14}^{(1)}, l_{12}^{(1)} + l_{24}^{(1)}, l_{13}^{(1)} + l_{34}^{(1)}, l_{14}^{(1)} + \\ + l_{44}^{(1)}, l_{15}^{(1)} + l_{54}^{(1)}) = \min(60, 140, \infty, 60, \infty) = 60. \qquad (6)$$

Аналогично рассчитав все остальные элементы матрицы $L^{(2)}$, получим следующую матрицу

$$L^{(2)} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{pmatrix} 0 & 60 & 110 & 60 & 90 \\ 60 & 0 & 50 & 80 & 90 \\ 110 & 50 & 0 & 100 & 70 \\ 60 & 80 & 100 & 0 & 30 \\ 90 & 90 & 70 & 30 & 0 \end{pmatrix} \end{array} \qquad (7)$$

Теперь сверяем полученную матрицу с матрицей исходной и получаем $L^{(2)} \neq L$. Это означает, что поиск оптимальных путей следует продолжить. Для этого вычисляем $L^{(3)} = L^{(2)} * L$ (8) – (10)

$$l_{12}^{(3)} = \min(l_{11}^{(2)} + l_{12}^{(1)}, l_{12}^{(2)} + l_{22}^{(1)}, l_{13}^{(2)} + l_{32}^{(1)}, l_{14}^{(2)} + \\ + l_{42}^{(1)}, l_{15}^{(2)} + l_{52}^{(1)}) = \min(60, 60, 160, 140, 180) = 60, \qquad (8)$$

$$l_{13}^{(3)} = \min(l_{11}^{(2)} + l_{13}^{(1)}, l_{12}^{(2)} + l_{23}^{(1)}, l_{13}^{(2)} + l_{33}^{(1)}, l_{14}^{(2)} + \\ + l_{43}^{(1)}, l_{15}^{(2)} + l_{53}^{(1)}) = \min(110, 110, 110, 160, 160) = 110, \qquad (9)$$

$$l_{14}^{(3)} = \min(l_{11}^{(2)} + l_{14}^{(1)}, l_{12}^{(2)} + l_{24}^{(1)}, l_{13}^{(2)} + l_{34}^{(1)}, l_{14}^{(2)} + \\ + l_{44}^{(1)}, l_{15}^{(2)} + l_{54}^{(1)}) = \min(60, 140, 210, 60, 120) = 60. \qquad (10)$$

Аналогично рассчитываем и остальные элементы матрицы $L^{(3)}$. Полученная матрица $L^{(3)}$ будет иметь вид

$$L^{(3)} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{pmatrix} 0 & 60 & 110 & 60 & 90 \\ 60 & 0 & 50 & 80 & 90 \\ 110 & 50 & 0 & 100 & 70 \\ 60 & 80 & 100 & 0 & 30 \\ 90 & 90 & 70 & 30 & 0 \end{pmatrix} \end{array}. \qquad (11)$$

Теперь воспользуемся формулой (2) и получим $L^{(3)} = L^{(2)}$. Из этого делаем вывод, что $L_{опт} = L^{(3)} = L^{(2)}$, следовательно, оптимальные пути найдены, и расчёт матрицы 4 ранга можно не производить [6].

Выделяя преимущества и недостатки данного алгоритма, следует отметить, что рассмотренный метод даёт возможность определить длины оптимальных путей (суммарных задержек передачи данных) между всеми узлами сети, но при этом не указывает, какие именно узлы входят в этот оптимальный маршрут. Таким образом, маршрутизатору необходимо прописывать наилучшие маршруты на основании итоговой матрицы, что занимает дополнительное время у сетевого администратора при проектировании и настройки маршрутизаторов.

Алгоритм Флойда

Как уже говорилось выше, информацию о маршруте будут содержать в себе матрица маршрутная и матрица дистанционная

$$L_{опт.} = D. \qquad (12)$$

Алгоритм Флойда состоит в следующем:

1. Для начала строятся матрицы D^0 и Γ^0

$$D^0 = L^{(1)},$$

где γ_{ij} – выбор пункта, который является 1-ым промежуточным из i в j.

2. Задается базовый узел (вершина)

$$k = 1, 2, \ldots, n$$

$$d_{ij}^{(q)} = \begin{cases} d_{ij}^{(q)}, \text{если} \quad d_{ij}^{(q-1)} \le d_{ik}^{(q-1)} + d_{kj}^{(q-1)}, \\ d_{ik}^{(q-1)} + d_{kj}^{(q-1)}, \text{ в противном случае,} \end{cases}$$

$$\gamma_{ij}^{(q)} = \begin{cases} \gamma_{ij}^{(q)}, \text{если } d_{ij}^{(q-1)} \le d_{ik}^{(q-1)} + d_{kj}^{(q-1)}, \\ k, \text{ в противном случае.} \end{cases}$$

Пункт 2 повторяется n раз, а именно все вершины должны быть выбраны в качестве базовых вершин

$$D^{(n)} = L_{опт}.$$

Для иллюстрации алгоритма Флойда, определим кратчайшие пути между всеми парами вершин графа, изображенного на рисунке 1.

Начальные значения элементов матрицы D^0 представлены формулой 13, а матрицы Γ^0 – формулой 14

$$D^0 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 60 & \infty & 60 & \infty \\ 60 & 0 & 50 & 80 & 90 \\ \infty & 50 & 0 & \infty & 70 \\ 60 & 80 & \infty & 0 & 30 \\ \infty & 90 & 70 & 30 & 0 \end{bmatrix} \end{array} \qquad (13)$$

$$\Gamma^0 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 \end{bmatrix} \end{array} \qquad (14)$$

На первом этапе вершина k = 1 определяется как базовая, а значит в матрице D^0 вычеркиваем первый столбец и первую строку. При этом все столбцы и все строки, в которых базовый элемент равен ∞, не рассматриваются. В нашем конкретном случае остались элементы 4.2 и 2.4. Рассчитываем их значение

$$d_{24}^{(1)} = \min(d_{24}^{(0)}, d_{21}^{(0)} + d_{14}^{0}) = \min(80, 60 + 60) = 80,$$

$$d_{42}^{(1)} = \min(d_{42}^{(0)}, d_{41}^{(0)} + d_{12}^{0}) = \min(80, 60 + 60) = 80.$$

После этого шага, нужно переписать полученные значения в матрицы D^0 и Γ^0. Однако полученные значения совпадают с исходными, поэтому необходимости переписывать матрицу ещё раз не имеет смысла. Полученный результат будет иметь вид формул 13 и 14 соответственно.

Далее определяем вершину 2 как базовую и повторяем процедуру вычислений для нужных элементов

$$d_{13}^{(2)} = \min(d_{13}^{(1)}, d_{12}^{(1)} + d_{23}^{1}) = \min(\infty, 60 + 50) = 110,$$

$$d_{15}^{(2)} = \min(d_{15}^{(1)}, d_{12}^{(1)} + d_{25}^{1}) = \min(\infty, 60 + 90) = 150,$$

$$d_{31}^{(2)} = \min(d_{31}^{(1)}, d_{32} + d_{21}^{1}) = \min(\infty, 50 + 60) = 110,$$

$$d_{51}^{(2)} = \min(d_{51}^{(1)}, d_{52}^{(1)} + d_{21}^{1}) = \min(\infty, 90 + 60) = 150,$$

$$d_{43}^{(2)} = \min(d_{43}^{(1)}, d_{42}^{(1)} + d_{23}^{1}) = \min(\infty, 80 + 50) = 130,$$

$$d_{34}^{(2)} = \min(d_{34}^{(1)}, d_{32}^{(1)} + d_{24}^{1}) = \min(\infty, 50 + 80) = 130.$$

Аналогично рассчитываем элементы $d_{14}, d_{35}, d_{41}, d_{45}, d_{53}, d_{54}$. Переписываем полученные значения в матрицу D^0 и Γ^0. Полученный результат представлен формулами 15 и 16 соответственно

$$D^{(2)} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 60 & 110 & 60 & 150 \\ 60 & 0 & 50 & 80 & 90 \\ 110 & 50 & 0 & 130 & 70 \\ 60 & 80 & 130 & 0 & 30 \\ 150 & 90 & 70 & 30 & 0 \end{bmatrix} \end{array} \qquad (15)$$

$$\Gamma^{(2)} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 1 & 2 & 2 & 4 & 2 \\ 1 & 2 & 3 & 4 & 5 \\ 2 & 2 & 3 & 2 & 5 \\ 1 & 2 & 2 & 4 & 5 \\ 2 & 2 & 3 & 4 & 5 \end{bmatrix} \end{array} \qquad (16)$$

Выполняя аналогичные операции на следующих шагах для вершин 3 – 5, получим итоговые матрицы D и Г, имеющие следующий вид (17, 18).

$$D^{(5)} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 60 & 110 & 60 & 90 \\ 60 & 0 & 50 & 80 & 90 \\ 110 & 50 & 0 & 100 & 70 \\ 60 & 80 & 100 & 0 & 30 \\ 90 & 90 & 70 & 30 & 0 \end{bmatrix} \end{array} \quad (17)$$

$$\Gamma^{(5)} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 1 & 2 & 2 & 4 & 4 \\ 1 & 2 & 3 & 4 & 5 \\ 2 & 2 & 3 & 5 & 5 \\ 1 & 2 & 5 & 4 & 5 \\ 4 & 2 & 3 & 4 & 5 \end{bmatrix} \end{array} \quad (18)$$

Допустим, необходимо вычислить оптимальный маршрут из узла 1 к узлу 5. Для этого обратимся к матрице $D = D^{(5)}$

$$d_{15} = d_{15}^{5} = 90 .$$

Для определения последовательности вершин, составляющих данный маршрут, обратимся к матрице Г. Последовательно определяя каждую из вершин, получаем

$$\mu_{15} = \{ 1, 4, 5 \}.$$

Анализируя алгоритм, следует отметить, что алгоритм хорошо формализован, даёт возможность получить как весовую характеристику пути, так и сам путь в форме пунктов (узлов) его составляющих. Однако число итераций (шагов) не меньше числа n, его никак нельзя сократить [1, 3]. Таким образом, алгоритм является громоздким, многоэтапным и занимает большой объем времени при расчетах. Однако, при современном развитии микропроцессорной техники, данный недостаток не является существенным.

Алгоритм Дейкстры

Алгоритм, разработанный нидерландским ученым Эдсгером Вибе Дейкстрой в 1959 году. Данный алгоритм находит кратчайшее расстояние и длину маршрута от некоторой исходной вершины графа до всех остальных вершин. Сразу отметим, что главным недостатком данного алгоритма является невозможность работы алгоритма с отрицательными весами на линиях связи. Для нашего конкретного рассмотренного примера этот недостаток не является существенным, т. к. ни расстояние между узлами сети, ни временная задержка не принимает отрицательные значения. Рассмотрим пример применения алгоритма Дейкстры. Допустим, мы имеем компьютерную сеть, которая представлена в виде графа (для удобства опять рассмотрим рисунок 1).

Данный граф представим в виде матрицы С

$$C = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 60 & \infty & 60 & \infty \\ 60 & 0 & 50 & 80 & 90 \\ \infty & 50 & 0 & \infty & 70 \\ 60 & 80 & \infty & 0 & 30 \\ \infty & 90 & 70 & 30 & 0 \end{bmatrix} \end{array} . \quad (19)$$

Для примера в качестве исходной точки возьмем вершину 1. Это означает, что мы будем искать оптимальные маршруты из узла 1 ко всем остальным узлам данного графа. Смысл алгоритма Дейкстры лежит в том, что алгоритм пошагово перебирает все вершины графа и назначает им метки, которые являются известным минимальным расстоянием от вершины-источника до конкретной вершины.

Итак, присвоим 1-й вершине метку, которая равна 0 (источник), а остальным вершинам метку ∞ (рис. 2).

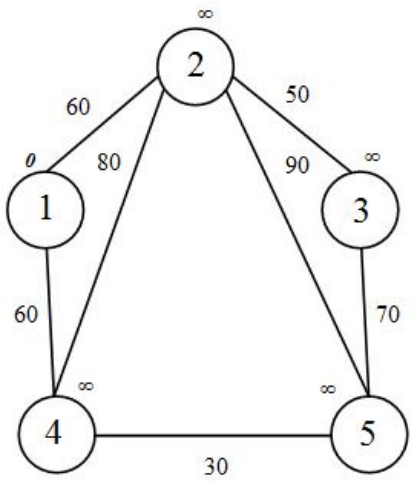

Рис. 2. Граф с присвоенными метками. Шаг 1

Следующим шагом алгоритма является выбор некой вершины Н, имеющей минимальную метку. Рассмотрим все вершины графа, в которые из Н есть прямой путь без промежуточных (транзитных) вершин. Для нашего случая выбираем 1-ю вершину с минимальной меткой равной нулю. Прямые пути

обозначим на рисунке сплошной линией, остальные же пути обозначим пунктирной линией (рис 3).

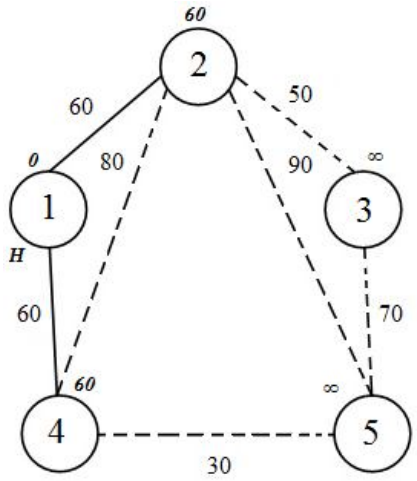

Рис. 3. Граф с присвоенными метками. Шаг 2

После того как мы рассмотрели все вершины, в которые есть прямой путь из заданной вершины H, мы отмечаем данную вершину как рассмотренную и выбираем из ещё непосещённых вершин такую вершину, которая имеет минимальное значение метки. В данном случае – это вершины 2 и 4 (в качестве H выбираем любую из них, допустим 2). При этом посещённые вершины закрасим (рисунок 4).

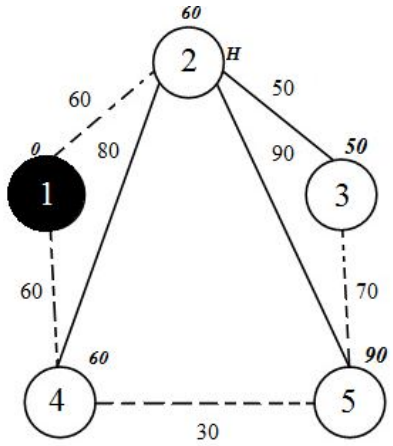

Рис. 4. Граф с присвоенными метками. Шаг 3

Отмечаем 2-ю вершину как рассмотренную и выбираем следующую вершину с минимальной меткой. Повторяем эти действия до тех пор, пока все вершины не будут отмечены как рассмотренные (рис. 5).

Из рисунка видно, некоторые метки стали меньше. Это означает, что был найден более короткий маршрут. Из преимуществ данного алгоритма можно отметить высокую скорость работы запрограммированного алгоритма, а также высокую точность результатов алгоритма [4, 5].

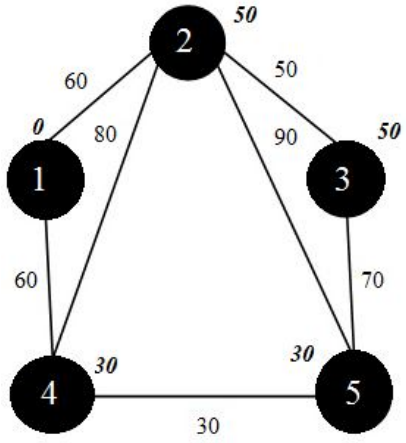

Рис. 5. Итоговый граф с оптимальными маршрутами

Заключение

Учитывая актуальность данной проблемной области, авторами рассмотрены несколько подходов для решения поставленной задачи – нахождения оптимального маршрута в компьютерной сети. Рассмотрены преимущества и недостатки каждого из методов, приведены наглядные примеры реализации алгоритмов.

Наилучшим из рассмотренных алгоритмов, по критериям быстродействия и простоты расчета, по результатам сравнительного анализа, является алгоритм Флойда. Именно этот алгоритм позволяет найти не только оптимальный маршрут между узлами сети, а указывает, какие именно узлы составляют данный маршрут, что существенно упрощает реализацию маршрутизатора.

Однако, помимо классических подходов поиска оптимального маршрута в компьютерной сети, существуют интеллектуальные подходы, которые позволяют учитывать мнение экспертов и способны к самообучению [6]. И используя интеллектуальные алгоритмы, появляется возможность повысить эффективность действующих методов маршрутизации.

Литература

1. Левитин, А. Алгоритмы: введение в разработку и анализ [Текст] / А. Левитин. – М. : Вильямс, 2006. – 577 с.

2. Князева, Н. А. Теория проектирования компьютерных систем и сетей [Текст] / Н. А. Князева. – О. : СПД Бровуын О. В., 2012. – 240 с.

3. Алгоритмы: построение и анализ [Текст] / Т. Х. Кормен, Ч. И. Лейзерсон, Р. Л. Ривест, К. Штайн. – 2-е изд. – М. : Вильямс, 2006. – 1296 с.

4. Алгоритм Дейкстры. Поиск оптимальных маршрутов на графе [Электронный ресурс] : habrahabr.ru, сайт. – Режим доступа: http://habrahabr.ru/post/111361/. – 07.01.2011.

5. Dijkstra, E. W. A note on two problems in connexion with graphs [Text] / E. W. Dijkstra // Numerische Mathematik. – 1959. – V. 1. – 271 p.

6. Солодовник, М. С. Применение нейронечеткого подхода для повышения надежности оптимальной работы компьютерной сети [Текст] / М. С. Солодовник // Холодильна техніка та технологія. – 2014. – № 1(147). – С. 68-75.

Поступила в редакцию 8.04.2014, рассмотрена на редколлегии 19.05.2014

Рецензент: д-р техн. наук, проф., зав. каф. информационных технологий и кибербезопасности В. М. Плотников, Одесская национальная академия пищевых технологий.

ПОРІВНЯЛЬНИЙ АНАЛІЗ ТРАДИЦІЙНИХ АЛГОРИТМІВ МАРШРУТИЗАЦІЇ В КОМП'ЮТЕРНІЙ МЕРЕЖІ

М. С. Солодовник, О. М. Асланов

У статті проаналізовано існуючі класичні алгоритми для вирішення завдання пошуку оптимального маршруту в комп'ютерній мережі. Розглянуто такі алгоритми, як алгоритм Флойда, алгоритм Дейкстри і алгоритм пошуку оптимального маршруту шляхом зведення матриці маршрутів у ступінь максимального рангу. При розгляді кожного з алгоритмів, наведено приклад його реалізації на прикладі графа, що представляє собою комп'ютерну мережу, а також виявлено переваги і недоліки кожного з розглянутих алгоритмів. Також зазначено, що поряд з розглянутими класичними алгоритмами, існують і інтелектуальні алгоритми, перевагами яких є - врахування думки експерта і здатність до самонавчання.

Ключові слова: алгоритм Флойда, алгоритм Дейкстри, оптимальний маршрут, самонавчання, матриця відстаней, максимальний ранг, вершина графа, комп'ютерна мережа, мітка вершини графа.

COMPARATIVE ANALYSIS OF TRADITIONAL ROUTING ALGORITHM IN COMPUTER NETWORKS

M. S. Solodovnik, A. M. Aslanov

The article analyzes the existing classical algorithms for solving the problem of finding an optimal route in a computer network. Considered algorithms such as Floyd's algorithm, Dijkstra algorithm and finding the optimal route by raising the matrix routes in the power of maximal rank's algorithm. In considering each of the algorithms, an example of its implementation on the example graph, which is a computer network were considered, as well as identifying the advantages and disadvantages of each of the considered algorithms . Also noted that along with the considered classical algorithms exist and intelligent algorithms, the advantages of which are - accounting expert opinions and the ability to self-learning.

Keywords: Floyd's algorithm, Dijkstra's algorithm, the optimal route, self-study, distance matrix, the highest rank, the top of the graph, the computer network, the label vertices of the graph.

Солодовник Михаил Сергеевич – аспирант кафедры информационных систем и сетей, Одесская национальная академия пищевых технологий, Институт холода, криотехнологий и экоэнергетики им. проф. В. С. Мартыновского, Одесса, Украина, e-mail: respect_all@ukr.net.

Асланов Алексей Михайлович – аспирант кафедры информационных технологий и кибербезопасности, Одесская национальная академия пищевых технологий, Институт холода, криотехнологий и экоэнергетики им. проф. В. С. Мартыновского, Одесса, Украина.