Исследование алгоритмов для решения задачи маршрутизации пакетов в компьютерной сети

В. С. Мельник
Санкт-Петербургский государственный электротехнический университет
«ЛЭТИ» им. В.И. Ульянова (Ленина)
hack-me@list.ru, mazzahaker.vlad@gmail.com

***Аннотация*. В настоящей работе объект исследования – алгоритмы, вдохновленные живой природой, а также классические эвристические алгоритмы. Все алгоритмы используются для нахождения оптимального пути передачи пакета от устройства-источника к устройству-приемнику. Реализован программно-аппаратный комплекс для решения задачи авторизированного доступа в сеть Интернет, а также создан графический интерфейс для обеспечения возможности пользовательского вмешательства в систему. Произведено тестирование и сравнительный анализ выбранных алгоритмов как на физической инфраструктуре, так и на программных ее аналогах.**

Ключевые слова: программно-аппаратный комплекс; алгоритмы; вдохновленные живой природой; генетический алгоритм; муравьиный алгоритм; эвристические алгоритмы

I. Описание алгоритмов

В работе рассмотрены и подвергнуты сравнению классические и вероятностные алгоритмы, решающие задачу маршрутизации с целью выявления быстрейших.

В связи с быстрым развитием вычислительных сетей, возрастает и количество устройств, входящих в их состав. С увеличением численности пользовательского оборудования и маршрутизаторов возникает актуальная проблема быстрого поиска маршрутов от устройства-источника к устройству-приемнику.

В настоящей работе освещается альтернативный вариант решения задачи маршрутизации. В частности – поиск маршрута с минимальной метрикой или же частное решение задачи Коммивояжера.

Задача маршрутизации заключается в выборе маршрута для передачи пакета (пакетов) от отправителя к получателю. Вопрос об оптимальном маршруте целесообразно ставить лишь в случае наличия в сети большого количества маршрутизаторов, благодаря которым можно построить несколько маршрутов от отправителя к получателю.

Задача маршрутизации позволяет решить не только ряд проблем, она предполагает решение проблемы различными способами маршрутизациями, а также – видами. Каждый из них имеет свои достоинства и недостатки, но основной недостаток у них один и тот же: недостаточно быстрое время поиска маршрута с наименьшей метрикой. Наиболее явно он проявляется при использовании случайной и лавинной маршрутизации. В первом случае – время, потраченное на доставку пакета до адресата может быть большим, чем при использовании адаптивной маршрутизации, а в случае с лавинной маршрутизацией сеть наполняется большим количеством паразитных пакетов, замедляющих работу инфраструктуры в целом, что также замедляет ее быстродействие.

Ускорить поиск наилучшего маршрута можно с помощью эвристических алгоритмов поиска.

Для проверки эффективности различных алгоритмов были использованы несколько алгоритмов.

Алгоритм имитации отжига – позволяет решать задачу глобальной оптимизации. Основан на имитации физического процесса, происходящего при кристаллизации вещества.

Алгоритм относится к вероятностным методам решения. Ключевым моментом в таких подходах является случайный выбор одного из нескольких возможных решений вместо анализа каждого. Это позволяет сократить время счета, а время счета в некоторых задачах на графах имеет принципиальное значение.

В методе отжига очередной порядок следования по маршруту между городами выбирается случайно, небольшим изменением предыдущего решения, предположительно оптимального. Самый простой вариант изменения – перестановка двух случайно выбранных городов в маршруте следования. Если полученный маршрут лучше всех существовавших ранее, то этот маршрут берется за очередной. Если маршрут хуже, то самый простой вариант – это не брать его. Однако так решение может попасть в один из локальных минимумов, которыми изобилуют подобные задачи, поэтому плохому решению надо дать шанс. Шанс этот рассчитывается по формуле:

$$P = \exp(-\frac{\Delta L}{T})$$

где ΔL – положительная разность между качеством тестируемого и ранее полученного оптимального решений, T – некоторый постоянно уменьшающийся параметр (условно – температура).

Снижение температуры обычно производится по формуле:

$$T_{k+1} = \alpha \cdot T_k,$$
$$0 < \alpha < 1$$

Генетический алгоритм – основан на механизмах, аналогичных природному естественному отбору.

Любой организм может быть представлен своим фенотипом, который фактически определяет, чем является объект в реальном мире, и генотипом, который содержит всю информацию об объекте на уровне хромосомного набора. При этом каждый ген, то есть элемент информации генотипа, имеет свое отражение в фенотипе. Таким образом, для решения задач необходимо представить каждый признак объекта в форме, подходящей для использования в генетическом алгоритме.

В данном алгоритме, помимо обычного математического аппарата, присутствует и вероятностная составляющая, которая делает алгоритм удобным, гибким и способным подстраиваться под конкретную ситуацию. Схема алгоритма довольно проста:

Изначально происходит инициализация: случайным образом формируется начальная популяция, параметры которой полностью настраиваются пользователем. После создания популяции, вычисляется приспособленность каждой особи и популяции в целом. Это определяется рядом формул:

$$F_{A_i} = fit(A_i), \mathrm{i} = 1, ..., \mathrm{k}$$
$$F_t = fit(\mathrm{B}_i), \mathrm{i} = 1, ..., \mathrm{k}$$

В случае, если популяция будет неудачной, в целом, она заменяется. Дальнейшая работа алгоритма строится на проверки особей и приближении к решению поставленной задачи. Для этого производится сравнение особей между собой, скрещивание особей и мутация. В итоге получается новых особей и новую популяцию, которая превосходит старую за счет лучшей приспособленности.

Алгоритм Дейкстры – алгоритм, позволяющий находить кратчайшие пути от одной из вершин графа до остальных.

Алгоритм позволяет находить в графе кратчайший путь между двумя выделенными вершинами *s* и *t* при положительных длинах дуг. Главная идея, лежащая в основе алгоритма Дейкстры, предельно проста.

Каждой вершине в ходе алгоритма присваивается число $d(x)$, равное длине кратчайшего пути из вершины *S* в вершину *X* и включающем только окрашенные вершины. Положить $d(\mathrm{S}) = 0$ и $d(\mathrm{X}) = \infty$ для всех остальных вершин графа. Окрашиваем вершину и полагаем *y*=*S*, где *y* – последняя окрашенная вершина. Нахождение пути у данного алгоритма довольно простое.

Для каждой неокрашенной вершины *X* пересчитывается величина $d(x)$ по следующей формуле:

$$d(x) = \min\{\mathrm{d(x)}; \mathrm{d(y)} + \mathrm{a}_{y,x}\}$$

Если $d(\mathrm{X}) = \infty$ для всех неокрашенных вершин, то алгоритм заканчивается т. к. отсутствуют пути из вершины *S* в неокрашенные вершины. Иначе окрашивается та вершина, для которой величина $d(x)$ является минимальной. Окрашивается и дуга, ведущая в эту вершину в соответствии с выражением и полагаем *y*=*X*. Если *y*=*t*, то кратчайший путь из *S* в *t* найден.

Алгоритм ближайшего соседа – наиболее простейший эвристический алгоритм, который заключается в принятии локально оптимальных решений на каждом этапе решения задач.

Пункты обхода плана последовательно включаются в маршрут, причем, каждый очередной включаемый пункт должен быть ближайшим к последнему выбранному пункту среди всех остальных, ещё не включенных в состав маршрута.

Алгоритм относится к вероятностным методам решения. Ключевым моментом в таких подходах является случайный выбор одного из нескольких возможных решений вместо анализа каждого. Это позволяет сократить время счета, а время счета в некоторых задачах на графах имеет принципиальное значение. Простой перебор для задач, сложность которых (время счета) растет по показательному или факториальному закону в зависимости от порядка графа, может даже для небольших графов оказаться недопустимо длительным. Оценка этой характеристики в каждом приложении своя.

Простейший вариант записи алгоритма выглядит следующим образом:

$$\omega(i, u) = [i = 1]$$
$$\alpha(u) = \arg\max \sum_{i=1}^{m} [x_{i,u} = y] \omega(\mathrm{i}, \mathrm{u})$$

Муравьиный алгоритм – как и генетический, основан на природных процессах. В частности – на модели поведения муравьев, ищущих пути к источнику питания.

В основе данного алгоритма лежит поведение колонии муравьев при поиске чего-либо: маркировка наиболее удачных путей (критерии могут быть различны) большим количеством феромона.

Как и генетический, этот алгоритм также имеет и вероятностную составляющую, которая определяется формулой:

$$P_i = \frac{l_i^q \cdot f_i^p}{\sum_{k=0}^{N} l_k^q \cdot f_k^p},$$

где: P_i – вероятность перехода по данному пути; l_i – величина, обратная весу перехода; f_i – количество феромона на данном переходе; q – величина, определяющая «жадность» алгоритма; p – величина, определяющая «стадность» алгоритма; $p + q = 1$.

Изначально работает вероятностная составляющая: отправляется лишь один муравей по выбранному пути. В случае, если он находит источник пищи, он оставляет за собой след из феромона, привлекая по маршруту остальных муравьев. Таким образом вся колония будет двигаться по наиболее оптимальному маршруту, укрепляя дорожку из феромонов, а остальные маршруты в итоге просто исчезнут из-за испарения феромонов.

Помимо быстродействия, в ходе решения задачи маршрутизации также необходимо учитывать и количество ресурсов, потребляемых алгоритмом. Этот фактор также немаловажен т.к. большинство маршрутизаторов не имеют больших аппаратных ресурсов и попросту неспособны будут быстро найти наилучший маршрут тяжелым алгоритмом.

В связи с этим, было принято решение изначально сравнить алгоритмы между собой в быстроте и точности поиска наилучшего маршрута, а только потом дополнительно наложить аппаратное ограничение, базирующееся на мощности маршрутизаторов применяемых в настоящее время.

Методика эксперимента заключается в том, что на каждом из маршрутизаторов одновременно происходит замена алгоритма поиска маршрута одним из перечисленных и производится тестирование. Логически принцип работы должен будет быть идентичен принципу работы OSFP. То есть каждый маршрутизатор должен обмениваться таблицей маршрутизации и за счет этого просчитывать лучший маршрут в условиях избыточности подключений.

На данный момент сделана программная реализация на языке C#, позволяющая оценить скорость и точность работы вышеперечисленных алгоритмов. В данном случае на вход программы подается таблица – связь между промежуточными узлами. На выходе получаем список задействованных в процессе узлов и затраченное время.

Было проведено два глобальных тестирования: первое – без аппаратного ограничения, второе – с ограничением. В первом случае тестирование показывало в принципе вычислительную мощность алгоритмов, а во втором – реальную разницу при работе на реальном оборудовании.

Сравнительная таблица работы алгоритмов без аппаратного ограничения отображена ниже (таблица).

В таблице указаны лучшие значения, полученные при использовании указанных алгоритмов. Если при использовании стандартных алгоритмов вопросов настройке не возникает, то, в случае с более сложными (алгоритм имитации отжига, муравьиный и генетический алгоритм), есть масса вопросов по конфигурации алгоритмов. В первую очередь встает вопрос производительности. Для быстрейшего нахождения пути необходимо использовать большее количество особей (генетический и муравьиный алгоритм). Это – один из основных параметров, влияющих на быстроту и точность поиска нужного маршрута. Проблема заключается в том, что при увеличении количества популяции, увеличивается и количество ресурсов, потребляемое алгоритмом. Это может привести к тому, что устройства, с малыми вычислительными возможностями попросту не смогут отработать этот алгоритм.

ТАБЛИЦА I СРАВНИТЕЛЬНЫЕ РЕЗУЛЬТАТЫ

Название алгоритма	**Количество вершин**	**Время, затраченное на поиск (сек)**
Алгоритм имитации отжига	10	0,1
	50	0,3
	75	0,3
	100	0,4
	125	0,5
	150	0,7
Генетический алгоритм	10	0,1
	50	0,2
	75	0,2
	100	0,4
	125	0,5
	150	0,6
Муравьиный алгоритм	10	0,1
	50	0,25
	75	0,3
	100	0,4
	125	0,4
	150	0,6
Алгоритм Дейкстры	10	0,4
	50	0,6
	75	0,6
	100	0,8
	125	1
	150	1,2
Алгоритм ближайшего соседа	10	0,5
	50	0,6
	75	0,7
	100	0,9
	125	1
	150	1,4

При программировании алгоритмов обязательно необходимо учитывать определенный баланс: скорость работы и количество потребляемых ресурсов. В случае с маршрутизаторами этот вопрос стоит острее т.к. поиск наилучшего маршрута не является основной задачей устройства. Если учесть, что сеть может перестраиваться довольно часто из-за отключения какого-либо маршрутизатора (перебой с электроэнергией и прочие причины), то задача поиска маршрута для каждого устройства – ресурсозатратная задача. Можно попытаться улучшить временные показатели, которые указаны в таблице. Для этого необходимо произвести настройку работы алгоритма: указать большее количество особей и изменить прочие параметры в сторону большей ресурсозатратности, но, в таком случае их нельзя будет применить даже на маршрутизаторах с мощной аппаратной платформой.

Казалось бы, доли секунды, однако, представим сложную топологию и вспомним, что чаще всего в сети используется более десятка маршрутизаторов (это в идеальном случае).

В ходе поиска оптимального решения поставленной задачи было выявлено, что наилучшим алгоритмом поиска кратчайшего маршрута от отправителя к получателю является генетический алгоритм, однако, использовать его имеет смысл только на сетях с количеством маршрутизаторов большим либо равным 100. Алгоритм показал не только наилучшее время поиска наилучшего маршрута, но и является гибким в плане настройки: можно произвести конфигурацию под любой маршрутизатор. Для более мощных устройств – увеличить количество особей, для маломощных – оставить прежним либо вовсе уменьшить.

Список литературы

[1] E. Bonabeau, M. Dorigo et G. Theraulaz, 1999. Swarm Intelligence: From Natural to Artificial Systems, Oxford University Press. ISBN 0-19-513159-2

[2] Аллан Леинванд, Брюс Пински. Конфигурирование маршрутизаторов Cisco = Cisco Router Configuration. 2-е изд. М.: «Вильямс», 2001. С. 368. ISBN 1-57870-241-0.