АВТОМАТНОЕ ПРОГРАММИРОВАНИЕ. Часть 3

Непейвода А.Н.

1 Введение

Вот программист садится решать задачу. Хорошо, когда в его распоряжении есть мощный компьютер с терабайтами оперативной памяти. Хорошо, когда каждая операция на этом компьютере не занимает и наносекунды. Вообще замечательно, если при этом есть строгий математический аппарат, который позволяет решить задачу наилучшим образом при любых входных данных.

А если нет?

В жизни часто приходится принимать решения «на авось» — неважно, виновата ли в этом недостаточная мощность нашего мозга или нехватка данных для точного решения. А значит, и компьютеру порой бывает выгодно действовать с долей риска. Именно на этот случай учёные изобрели простой, но исключительно красивый метод — метод вероятностных автоматов.

2 Что такое вероятностные автоматы?

Как вы помните, (см. Потенциал №XXX) конечные автоматы — это алгоритмы, не имеющие памяти. Представим себе опытного игрока на футбольном поле, расчерченном на крупные квадраты. Когда спортсмен ударяет по мячу, то наверняка знает, в какой квадрат он попадёт — таким образом, координаты мяча зависят только от нынешнего действия футболиста и того квадрата, в котором мяч был в начале действия. Примерно так же работает детерминированный конечный автомат: аналог действия футболиста для него — текущий символ входной цепочки, аналог квадрата, с которого мяч начинает движение — текущее состояние.

А теперь заменим опытного игрока на новичка, недавно занявшегося футболом. В отличие от профи, его действия могут заставить мяч двигаться лишь в предсказуемом диапазоне, но не в заранее загаданное место. Такая структура будет соответствовать ещё одному типу автоматов — вероятностным. В

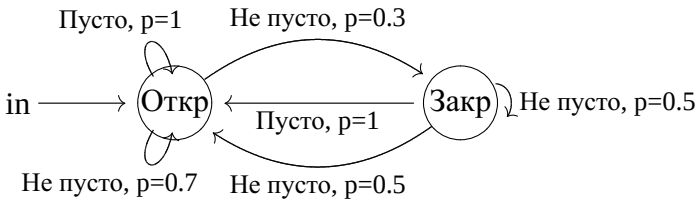
общем случае подобные автоматы допускают переход из любого состояния в любое другое, но с разными вероятностями (может быть, равными нулю). Вероятностные автоматы часто используются в моделировании как изящный и довольно эффективный приём.

Говоря более строго, вероятностный автомат можно рассматривать как детерминированный, на каждом переходе которого генерируется случайное число, в зависимости от которого будет выбрано следующее состояние.

Разберём пример: автоматический регулировщик движения на магистрали. Если из-за каждой машины на поперечной улице магистральное движение будет перекрываться, то в целом водители от такой регулировки только потеряют время (ведь на магистрали движение интенсивнее). Поэтому прибегнем к несложному вероятностному автомату.

Пусть входная лента представляет собой протокол загруженности поперечной улицы. На входной ленте могут быть два символа: на поперечной улице никого нет (**Пусто**) и на поперечной улице есть транспорт (**Не пусто**). Сам автомат может находиться в двух состояниях: движение по магистрали открыто (на поперечной улице — перекрыто), и закрыто (открыто на поперечной улице). Обозначим эти состояния за **Откр** и **Закр** соответственно. Логично, что, если поперечная улица пуста, то автомат обязательно переходит в состояние **Откр**. Если магистраль открыта, а на поперечной улице кто-то ждёт, то переходим в **Закр** с вероятностью 30%, а в 70% случаев остаёмся в **Откр**. Если магистраль перекрыта, а транспорт на поперечной всё ещё есть, то в половине случаев остаёмся в **Закр**, а в половине — перейдём в **Откр**.

Схема нашего автомата будет выглядеть так:



Посчитайте, сколько времени придётся ждать водителю на поперечной улице, чтобы проехать перекрёсток с вероятностью не меньше 80%, если показания снимаются раз в две минуты. Конечно, в редких случаях наш автомат может создавать длинные пробки, но в целом он работает удовлетворительно.

Мы увидели, что вероятностные автоматы — хороший способ решения практических задач, но у них есть серьёзный недостаток: нет никакой гарантии, что путь решения будет найден за количество шагов, меньшее заранее заданного **N**. Поэтому надо, чтобы шанс нахождения решения за приемлемое число шагов был достаточно велик — это следует помнить, назначая матрице переходов значения вероятностей.

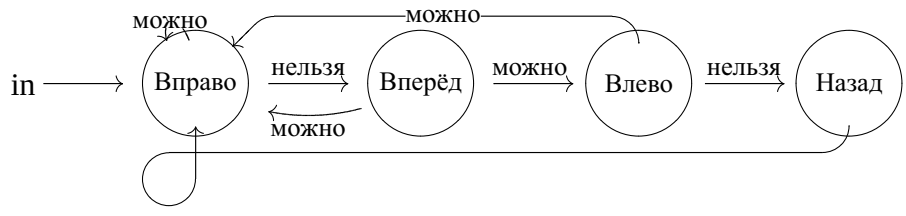
3 Применяем метод вероятностных автоматов к задаче поиска

Задача поиска ставится, в частности, почти в любой RPG или аркадной системе. К сожалению, достойным способом она разрешается далеко не всегда. Чаще всего используются очень простые алгоритмы, которые не справляются с мало-мальски сложными преградами (например, если «нюхач» находится в комнате, а цель — в коридоре за стенкой, противоположной входу, «нюхач» обязательно запутается). Разумеется, существуют алгоритмы поиска с полной информацией о карте, но для корректной работы они требуют огромных баз данных. Кроме того, такие алгоритмы поиска несправедливы по отношению к человеческому игроку.

А мы постараемся построить автоматный алгоритм поиска в ортогональном лабиринте. С первого взгляда задача кажется неподъёмной — ведь доказано, что поиск в графе автоматно не разрешим. Но когда говорят «не разрешим автоматно», подразумевают, что нельзя построить такой *детерминированный* автомат, который бы смог решить задачу в общем случае. Мы же будем пользоваться вероятностными алгоритмами.

Предположим, что преследователь не имеет никакой информации о карте, однако может оценить, в какой примерно стороне находится цель. Вообще говоря, преследователь даже не знает, находится ли перед ним стена или пустое пространство, пока не попробует пройти в этом направлении. Проблема сходна с той, что возникает перед человеком в полностью тёмном незнакомом подземелье, который идёт на шум, издаваемый целью, не зная пути и не имея возможности оставить метки. Правда, человек находится в более выгодном положении, чем автомат, — например, он может отличить песчаный пол от каменного на ощупь или ориентироваться по влажности и свежести воздуха в пещере. Но даже и в этом случае человек, как правило, будет придерживаться одной стены, дабы вести поиск более аккуратно. Мы будем руководствоваться этой же логикой.

Алгоритм хождения по стенке детерминирован и легко записывается в форме конечного автомата.



Разберёмся, что происходит в каждом состоянии.

Состояние Вправо. Производится поворот вправо и шаг. Если успешно, то переход в **Вправо**, иначе переход в **Вперёд**.
Состояние Вперёд. Производится поворот влево и шаг. Если успешно, то переход в **Вправо** , иначе переход в **Влево**.
Состояние Влево. Производится поворот влево и шаг. Если успешно, то переход в **Вправо**, иначе переход в **Назад**.
Состояние Назад. Производится поворот влево и шаг. Переход в **Вправо**.

Если бы наш лабиринт не имел круговых путей и состоял из коридоров ширины, самое большее, 2, то такой стратегии было бы достаточно. Но мы решаем задачу в общем случае, поэтому посмотрим, как люди справляются с циклическими путями. Человек, долго идущий по стенке в темноте, рано или поздно решит, что надо менять тактику поиска, — здесь стоит выделить фразу «рано или поздно»: это прямое указание на то, что стоит использовать вероятностные автоматы. Здравые соображения подсказывают, что, повернув четыре раза на 90 градусов в одну и ту же сторону, ищущий может предположить зацикленность маршрута. Однако в спиральном лабиринте циклов нет, однако там наблюдается та же картина поворотов. Поскольку наш автомат не умеет оценивать расстояния, он просто не сможет различить спираль и круг. Поэтому, как и человек, автомат, повернув на 360 градусов, меняет тактику поиска не всегда, а лишь с некоторой вероятностью.

Есть ещё один тип «развилки», где можно поменять тактику. Она возникает на повороте в нужном направлении. Допустим, если цель далеко на востоке к северу и, идя на восток, мы наткнулись на стенку, то логично пойти по правой руке и, повернув один раз вправо, с некоторой вероятностью попытаться снова пойти прямо к цели. В ортогональном лабиринте учёт поворотов не мешает «автоматности» алгоритма, ведь достаточно различать их с точностью до 720 градусов, а значит, хотя и число состояний при поиске по стенке увеличивается в восемь раз, но наш поиск становится гораздо более осмысленным.

Ключевой момент в построении нашего алгоритма — подбор вероятностей на «развилках». Если сделать вероятности смены тактики слишком большими, то автомат будет проявлять «нетерпеливость», идя вдоль неровной стенки. Если сделать их слишком маленькими — возрастает опасность долго кружить по одному маршруту. Проще всего подбирать приемлемые значения «на глазок», проведя для разных вероятностей большое количество испытаний в нескольких лабиринтах и выбрав те параметры, для которых время поиска оказалось в целом меньше. В нашем случае эффективнее всего оказалась вероятность смены тактики, равная $\frac{1}{5}$.

Итак, мы построили автомат поиска пути в лабиринте. Испытания такого автомата показывают, что его выигрыш в количестве шагов по сравнению со стохастическим алгоритмом (движущимся за шаг случайным образом в лю-

бую из четырёх сторон) сильно (нелинейно) возрастает с увеличением размеров лабиринта.

4 Подытожим

Как видно, вероятностные автоматы хорошо подходят для создания моледей управления. Если задача плохо решается точно или наложены ограничения по используемой памяти, вполне возможно, это как раз тот случай, когда нужно обратиться к нестандартному автоматному программированию. Но нельзя забывать, что у вероятностных автоматов есть серьёзный недостаток — их непредсказуемость. Поэтому лучше потратить час или два на подбор оптимальных вероятностей на переходах, чем позволять непослушному автомату часами решать несложную задачу.

Особый «нрав» вероятностных автоматов стал среди математиков притчей во языцех: в иных философских статьях даже говорится, что человек — это очень сложный вероятностный автомат. Вряд ли к таким заявлениям стоит относиться серьёзно, но взятие аналогии с человеческим поведением (как в нашем поиске) — действительно хороший способ строить вероятностные алгоритмы!

Для закрепления материала ниже приводится несколько задач.

Упражнение 1. Предположим, что регулировщик, описанный в разделе про вероятностные автоматы, стоит на перекрёстке двух магистралей с одинаково интенсивным движением. Как надо изменить вероятности переходов, чтобы он справлялся с управлением наилучшим образом?

Упражнение 2. Подземелье — ортогональный лабиринт с разветвлённой системой коридоров, не образующей циклов, и множеством комнат, у каждой из которых ровно один выход в коридор. Измените вероятностный алгоритм поиска, чтобы он работал в таком подземелье быстрее, чем исходный. Комнаты и коридоры могут быть неправильной формы.

Список литературы

[1] Н. Н. Непейвода. *Стили и методы программирования*, М.: ИНТУИТ, 2004.

[2] Н. Н. Непейвода, И. Н. Скопин. *Основания программирования*, Москва-Ижевск: Институт компьютерных исследований, 2003.

[3] А. А. Шалыто *SWITCH-технология. Алгоритмизация и программирование задач логического управления*, СПб.: Наука, 2000.