OSTIS-2011
(Open Semantic Technologies for Intelligent Systems)

УДК 519.711.74

РЕКУРСИВНЫЙ ПОИСК В ДИНАМИЧЕСКОЙ АССОЦИАТИВНОЙ РЕСУРСНОЙ СЕТИ

Л.Ю. Жилякова (*zhilyakova.ludmila@gmail.com*)
Институт проблем управления им. В. А. Трапезникова РАН, г. Москва, Россия

В работе рассматривается модель памяти, основанная на двусторонней динамической ресурсной сети с переменной топологией. Ассоциативная близость понятий в такой сети прямо пропорциональна пропускным способностям связывающих их двусторонних пар; доступность вершины при поиске характеризуется также пропускной способностью ее петли. Ассоциативный поиск в модели реализуется с помощью рекурсивных процедур, позволяющих поддерживать длинные цепочки ассоциаций.

Ключевые слова: ресурсная сеть, пропускная способность, ассоциативная память, ассоциативный поиск, рекурсия.

Введение

В работе продолжаются исследования свойств модели ассоциативной памяти, предложенной в [Жилякова, 2009]. В основе этой модели лежит математический аппарат, разработанный в [Кузнецов, 2009; Кузнецов и др., 2009], – двусторонняя ресурсная сеть с петлями – потоковая модель, принципиально отличающаяся от классических потоковых моделей (см. например, [Форд, Фалкерсон, 1966]) и их динамических аналогов [Ahuja et al., 1993], и предназначенная для решения иных задач. Задача в модели Форда–Фалкерсона состоит в нахождении максимального потока от источника к стоку, в то время как в ресурсной сети нет источников и стоков, течение ресурса происходит во все стороны одновременно; задача оптимизации потока в указанном понимании не ставится.

Ассоциативная ресурсная сеть – семантическая модификация ресурсной сети. Однако в отличие от неоднородных семантических сетей, семантика присуща только вершинам: метки ребер отвечают за их пропускные способности и соответствуют силе ассоциаций между понятиями. Таким образом, в сети реализуется только одно отношение: отношение ассоциативной связи понятий.

Модель обладает переменной топологией – пропускная способность ребра увеличивается всякий раз, когда по нему течет ресурс. Благодаря этой особенности, наиболее часто используемые данные оказываются и наиболее доступными. Такая организация памяти позволяет значительно ускорить обработку запросов к часто используемым данным.

Быстрый доступ к часто используемым данным обеспечивается следующими особенностями сети: каждая вершина обладает способностью хранить неограниченное количество ресурса, отвечающего за ее яркость, которая повышает ее доступность при поиске; каждое ребро сети имеет пропускную способность, соответствующую силе ассоциативной связи между сущностями. Пропускная способность петли отвечает за способность вершины оставлять себе ресурс при его распределении, и тем самым, соответствует автоассоциации понятия.

Еще одна характеристика вершины – ее дивергенция: разность между суммарной входной и выходной пропускными способностями. Некоторые вершины с неотрицательной дивергенцией способны аккумулировать значительную часть яркости при выполнении запросов. Такие вершины в ресурсных сетях получили название потенциальных аттракторов.

При ненаправленном распространении ресурса по сети во время поиска зона яркости увеличивается равномерно во все стороны. Чтобы создать длинные ассоциативные цепочки без лишнего рассеяния ресурса, – задать направление движения пятна яркости по сети, – используется рекурсивный поиск, основанный на автоматически порождаемых рекурсивных запросах.

1. Двусторонняя ресурсная сеть с петлями

Ресурсной сетью называется двусторонний ориентированный граф, вершинам v_i которого приписаны неотрицательные числа $q_i(t)$, называемые *ресурсами*, и изменяющиеся в дискретном времени t, а ребрам (v_i, v_j) – неотрицательные числа r_{ij}, постоянные во времени и называемые *пропускными способностями*.

Свойство двусторонности означает, что если существует ребро (v_i, v_j) с ненулевой пропускной способностью r_{ij}, то обязательно существует и противоположно ориентированное ребро (v_j, v_i) с ненулевой пропускной способностью r_{ji}, причем равенство $r_{ij} = r_{ji}$ в общем случае не выполняется.

Двустороннюю пару будем обозначать $\langle(v_i, v_j), (v_j, v_i)\rangle$.

Каждая вершина v_i обладает петлей с пропускной способностью, равной r_{ii}.

Состоянием $Q(t)$ сети в момент t будем считать вектор $Q(t) = (q_1(t), \ldots, q_n(t))$, состоящий из значений ресурсов в каждой вершине.

Матрицей пропускной способности будем называть матрицу $R = \|r_{ij}\|_{n \times n}$.

Если пары $\langle(v_i, v_j), (v_j, v_i)\rangle$ не существует, $r_{ij} = 0$ и $r_{ji} = 0$.

Суммарной пропускной способностью сети, r_{sum}, назовем сумму пропускных способностей всех ее ребер: $r_{sum} = \sum_{i=1}^{n}\sum_{j=1}^{n} r_{ij}$.

Суммарную пропускную способность входных ребер вершины с номером i будем называть ее *входной пропускной способностью* и обозначать через $r_i^{in} = \sum_{j=1}^{n} r_{ji}$; суммарную пропускную способность выходных ребер, соответственно, назовем *выходной пропускной способностью* и обозначим через $r_i^{out} = \sum_{j=1}^{n} r_{ij}$. Пропускная способность петли входит в обе суммы.

Распределение ресурса в сети происходит по одному из двух правил, выбор которых зависит от величины ресурса в вершинах. В момент t вершина v_i в ребро, соединяющее ее с вершиной v_k, отдаст:

правило 1: r_{ik} единиц ресурса, если $q_i(t) > r_i^{out}$;

правило 2: $\frac{r_{ik}}{r_i^{out}} q_i(t)$ в противном случае.

Ресурсная сеть называется *несимметричной*, если в ней существует хотя бы пара вершин, для которых $\left|r_i^{in} - r_i^{out}\right| > 0$.

Для произвольной вершины v_i обозначим эту разность через Δr_i: $\Delta r_i = r_i^{in} - r_i^{out}$. Тогда все вершины несимметричной сети можно разделить на три класса:

- *вершины-приемники*, для которых $\Delta r_i > 0$;
- *вершины-источники*, для которых $\Delta r_i < 0$;
- *нейтральные вершины*, для которых $\Delta r_i = 0$.

Распределение ресурса в сети представляет собой Марковский процесс. На основании свойств регулярных стохастических матриц [Кемени и др., 1970] доказана сходимость процесса распределения ресурса для любой конфигурации сети. Предельное состояние в двусторонней сети с петлями сети существует для любого значения ресурса, циркулирующего в ней. Обозначим его через Q^*.

2. Ассоциативная ресурсная сеть. Автоматическое изменение топологии сети

Динамическая ассоциативная ресурсная сеть представляет собой ресурсную сеть, каждая вершина которой обозначает некоторую сущность предметной области: объект, атрибут или класс, – и имеет имя v_i из множества имен V. Рёбра сети обозначают ассоциации между сущностями.

Двусторонность сети отвечает за существование как прямой, так и обратной ассоциации – пропускные способности ребер внутри каждой пары могут не совпадать.

Ресурс, находящийся в вершине, – в ассоциативной сети соответствует яркости понятия в памяти. В неактивном состоянии сеть не имеет яркости. Яркость поступает в сеть только на стадии обращения к ней с поисковым запросом.

Запросы к сети служат для выявления ассоциативных цепочек и понятий, ассоциированных с некоторым начальным набором. Запрос – это задание начального множества вершин с указанием количества ресурса (начальной яркости) в каждой из них.

Следуя правилам 1 и 2 из п.1, яркость начинает распространяться по сети от каждой вершины по всем инцидентным ребрам. Распространение яркости происходит в быстром времени t.

Однако сеть не является статической моделью некоторой предметной области: она изменяется с каждым новым запросом. Для того чтобы пропускные способности каждого ребра адекватно отображали ассоциативную связь между понятиями, построение сети (добавление новых вершин и связей) и обращение к ней с запросами происходит по одним и тем же правилам в медленном времени τ. Одному такту τ соответствует выполнение одного запроса или занесение новой единицы информации. Обращение к сети с каждым новым запросом, так же как и наполнение ее новой информацией, изменяет топологию сети и, соответственно, влияет на результаты будущих запросов [Жилякова, 2010].

Информация заносится в сеть минимальными структурными единицами. Они могут быть двух типов:

1) двусторонняя пара, связывающая две существующие вершины;

2) новая вершина с петлей и двусторонняя пара, связывающая эту вершину с уже имеющейся.

На каждом такте τ все ребра, которые участвовали в передаче яркости, включая петли вершин, увеличивают свою пропускную способность на некоторую величину $\rho(\tau)$. Таким образом, чем чаще вершины участвуют в запросах, тем больше пропускная способность их петель и тем больше яркости они могут удержать. Чем выше пропускная способность ребер, ведущих к вершине, тем больше яркости она получит из других областей сети. Вершины, способные вблизи предельного состояния удержать или накопить яркость, превышающую их выходную проводимость, называются *потенциальными аттракторами*.

Как правило, это некоторые «вершины холма» в распределении яркости – центры образов [Голицын, 1997], как, к примеру, «яблоко, курица, Пушкин».

3. Рекурсивные поисковые запросы

3.1. Реализация рекурсивных запросов

В больших сетях яркость может растекаться от каждой вершины неограниченно во все стороны. Чтобы локализовать некоторое пятно яркости в сети, процесс распространения ресурса останавливается по истечении конечного числа тактов быстрого времени t. Выходное множество вершин вместе с входным множеством преобразуются в соответствии с правилами, описанными ниже, и полученное новое множество вершин (с их яркостью) рассматривается как новое входное множество следующего запроса. Таким образом реализуются рекурсивные запросы. Глубина рекурсии может варьироваться от запроса к запросу.

Для каждой серии запросов можно дополнительно ввести порог яркости, с помощью которого, вершины, имеющие яркость ниже порога, по завершении каждого шага рекурсии автоматически становятся неактивными.

Входным множеством вершин очередного запроса в рекурсии не обязательно должно быть выходное множество от предыдущего запроса, то есть множество вершин, входящих в ответ на предыдущий запрос. Каждое новое входное множество состоит из предыдущего входного множества, подвергнутого определенным изменениям. Эти изменения могут быть двух видов.

I. Добавление одной или нескольких вершин из выходного множества предыдущего запроса. Обозначим количество добавляемых вершин через k^+: $k^+ = 1, 2, \ldots l - l^*$, где l – количество вершин в ответе, l^* – мощность пересечения входного и выходного множества. Этот тип изменений соответствует ситуации, когда самый ожидаемый (вероятный) ответ на поставленный запрос является удовлетворительным, но его нужно расширить и/или уточнить.

В таком случае из множества вершин, входящих в ответ, выбираются некоторые дополнительные вершины (вместе со своей яркостью) и добавляются в исходное множество вершин, участвующих в запросе. После чего снова происходит увеличение пропускных способностей соответствующих ребер, и весь процесс перераспределения ресурса повторяется заново – для нового начального состояния.

Количество новых запросов, которые возможно сгенерировать таким способом, составит: $N^+ = \sum_{i=0}^{l-l^*} C^i_{l-l^*} = 2^{l-l^*}$

II. Удаление одной или нескольких вершин из входного множества предыдущего запроса. Количество удаляемых вершин обозначим через k^-: $k^- = 1, 2, \ldots m$.

Этот вид изменений входного множества может преследовать различные цели.

II a) Удаляются вершины из пересечения множеств вопрос-ответ, т.е. входного и выходного множеств предыдущего запроса. Такие изменения предназначены для отсекания самого очевидного ответа и поиска других, менее очевидных. То есть, чтобы получить заведомо «нетривиальный» ответ, сначала нужно узнать ответ тривиальный, и только затем его отсечь.

Количество удаляемых вершин может изменяться от 1 до l^*, где l^* – мощность пересечения множеств запроса и ответа.

Общее количество запросов, сгенерированных этим правилом, будет: $N_a^- = \sum_{i=0}^{l^*} C^i_{l^*} = 2^{l^*}$.

II b) Удаляются вершины из предыдущего входного множества, которых не оказалось в множестве выходном. Эти изменения производятся, если нужно создать длинные ассоциативные цепочки, – создать движение яркости сквозь сеть. Чем меньше вершин из предыдущего входного множества перейдет в следующее, тем быстрее будет передвигаться «пятно яркости» по сети, охватывая каждый раз новые участки. Если же основную массу из входного множества не трогать, пятно яркости будет блуждать около фиксированного центра.

С помощью этого правила можно составить $N_b^- = \sum_{i=0}^{m-l^*} C^i_{m-l^*} = 2^{m-l^*}$.

Итоговые формулы для N^+ и N^-: $N^+ = 2^{l-l^*}$, $N^- = 2^m$.

Комбинируя все возможные сочетания добавления и удаления вершин, получим $N = 2^{l-l^*+m}$ различных множеств, каждое из которых претендует на то, чтобы быть входным множеством запроса на следующем шаге рекурсии.

3.2. Операции над графами

При описании входных множеств рекурсивных запросов фрагменты сети рассматривались как множества вершин, имеющих яркость. Т.е. нигде не учитывалась структура графов, содержащих эти вершины.

Перейдем к графам и опишем последовательность действий, которую необходимо произвести, чтобы добавить новые вершины или удалить уже существующие из подграфов с известной топологией.

Ответом на единичный запрос к сети является некоторый фрагмент сети с распределением яркостей. Иными словами, это подграф, каждой вершине которого сопоставлено некоторое число, обозначающее находящееся в ней количество ресурса.

Эти числовые значения задают частичный порядок на множестве вершин. Пронумеруем вершины графа, независимо от его конфигурации, от более ярких к менее ярким. Если фрагмент содержит несколько вершин с одинаковой яркостью, упорядочим это подмножество в соответствии с некоторым заданным правилом, например, в лексикографическом порядке имен. Таким образом, получим линейный порядок на множествах вершин входного и выходного подграфов.

Входной граф запроса на шаге с номером i обозначим $G_{In}(i)$. Выходной граф этого же запроса обозначим $G_{Out}(i)$. Вершины этих графов представляют собой пары: имя+яркость.

Если в качестве ответа на запрос брать весь подграф, то будет выполняться следующее вложение: $G_{In}(i) \subseteq G_{Out}(i)$.

Если же отсекать вершины с самой низкой яркостью (порог яркости может быть относительной величиной и зависеть от суммарной яркости вершин фрагмента), то вложение уже, возможно, не будет иметь места. Более того, в таком случае множества вершин графов $G_{In}(i)$ и $G_{Out}(i)$ могут даже не пересекаться.

Введем оператор T, действующий на графах. $T(G)$ – транзитивное замыкание графа G. Действует он следующим образом: для любого графа G $T(G)$ – такой граф, что для любых двух вершин верно: если есть путь любой длины из вершины v_i в вершину v_j, то есть и двусторонняя пара $<(v_i, v_j),(v_j, v_i)>$, связывающая эти вершины напрямую.

Если граф G состоит из одной компоненты связности, то $T(G)$ – полный граф.

Как только на шаге i по входному графу $G_{In}(i)$ получается выходной граф $G_{Out}(i)$, к объединению графов $G_{In}(i) \cup G_{Out}(i)$ применяется оператор T.

Получается граф $G_{InOut}(i) = T(G_{In}(i) \cup G_{Out}(i))$. Множество его вершин – это по-прежнему вершины графа $G_{In}(i) \cup G_{Out}(i)$.

Проводимость каждого вновь созданного ребра рассчитывается как среднее геометрическое проводимостей ребер, составляющих цепочку.

Если из вершины в вершину существует несколько путей, выбирается максимальное из средних геометрических их проводимостей.

При построении замыкания проводимости рассчитываются только по «опорным ребрам», т.е. тем, которые существовали в графе $G_{In}(i) \cup G_{Out}(i)$.

Введем еще два оператора, действующие на графах: A – добавление вершин к графу; E – удаление вершин из графа.

Добавление и удаление вершин, разумеется, производится со всеми инцидентными ребрами, соединяющими вершину со всеми смежными вершинами рассматриваемого графа.

Оператор A применяется к двум подграфам, – возможно, лежащим в одной компоненте связности. Запись: $A_{j_1,\ldots,j_k}(G_1, G_2)$ означает, что из графа G_2 в граф G_1 будет добавлено k вершин с номерами $j_1, \ldots, j_k$ вместе со всеми ребрами, соединяющими эти вершины с вершинами G_1.

Тогда на шаге $i + 1$ добавление во входной подграф вершин с номерами $j_1, \ldots, j_k$, где $k \le l - l^*$ ($l - l^*$ – мощность множества вершин графа $G_{Out}(i) \backslash G_{In}(i)$), запишется в следующем виде:

$G_{In}(i+1) = A_{j_1,\ldots,j_k}(G_{In}(i), G_{Out}(i))$.

Ребра в полученный граф добавляются из транзитивного замыкания $G_{InOut}(i) = T(G_{In}(i) \cup G_{Out}(i))$, чтобы не были потеряны ассоциации, если прервались их цепочки.

Оператор E применяется к одному графу.

Запись: $E_{j_1',\ldots,j_h'}(G)$ означает, что из графа G будет удалено h вершин с номерами $j_1', \ldots, j_h'$ вместе с их инцидентными ребрами.

Тогда на шаге $i + 1$ удаление из графа $G_{In}(i)$ вершин с номерами $j_1', \ldots, j_h'$, где $h \le m$ (m – мощность множества вершин $G_{In}(i)$), запишется в следующем виде:

$G_{In}(i+1) = E_{j_1',\ldots,j_h'}(G_{In}(i))$.

Будем считать, что сначала к графу $G_{In}(i)$ применяется оператор E, а затем к результату – оператор A.

Операторы не коммутируют, порядок их применения важен.

Заметим, что индексы $j_1', \ldots, j_h'$ – это номера вершин графа $G_{In}(i)$, а $j_1, \ldots, j_k$ – номера вершин $G_{Out}(i)$. Поэтому если сначала применить оператор A, то в $G_{In}(i)$ добавится k вершин, после чего всё множество вершин переупорядочится. Тогда набор индексов $j_1', \ldots, j_h'$ оператора E будет обозначать совсем не те вершины, что были на этих местах до применения оператора A.

Таким образом, на шаге $i + 1$ входной граф запроса находится по следующей формуле:

$G_{In}(i+1) = A_{j_1,\ldots,j_k}(E_{j_1',\ldots,j_h'}(G_{In}(i)))$.

Непосредственно из этой формулы вытекает, что каждый новый входной подграф однозначно определяется входным и выходным подграфами на предыдущем шаге и парой последовательностей натуральных чисел переменной длины: $(\{j_1', \ldots, j_h'\}; \{j_1, \ldots, j_k\})$.

Например, пара ({2, 5, 9}; {7, 11}) означает, что из графа, который был входным на предыдущем шаге, нужно удалить вершины с номерами 2, 5 и 9 и присоединить вершины с номерами: 7, 11 из графа выходного. Затем заново перенумеровать вершины полученного графа.

3.3. Способы задания рекурсивных запросов

Любой рекурсивный запрос будет однозначно идентифицироваться цепочкой, звеньями которой являются входное множество вершин с распределением яркостей и множества удаляемых и добавляемых вершин. Длина цепочки совпадает с глубиной рекурсии.

Как уже было показано, пару множеств удаляемых и добавляемых вершин можно сформировать $N = 2^{l \cdot l^* + m}$ способами, тогда количество способов для M пар будет равно N^M.

Используя вероятностные методы при ненаправленном ассоциативном поиске можно получить большое количество заведомо различных ассоциативных цепочек.

Наличие в сети яркости может позволить реализовать автоматическое управление направлением поиска по ассоциациям наибольшей силы или в сторону наиболее ярких аттракторов. Для этого на каждом последующем шаге нужно убирать k_1 самых тусклых вершин из множества $G_{In}(i)$ и добавлять k_2 самых ярких из $G_{Out}(i)$.

Еще один способ – вместо задания чисел k_1 и k_2 задавать порог яркости R_{bound}, отсеивать из $G_{In}(i)$ вершины с яркостью, меньшей пороговой, и добавлять из $G_{Out}(i)$ вершины, которые имеют яркость больше этого порога. Избрав этот способ, с одной стороны можно получить цепочки большой длины наиболее устойчивых и сильных ассоциаций, с другой же – возможно зацикливание на одной сильной ассоциации и стабилизация процесса, когда для некоторого номера I будет выполняться: $\forall i > I \; G_{In}(i) = G_{Out}(i)$.

Заключение

Ассоциативная ресурсная сеть – динамическая модель памяти, изменяющая свою топологию с каждым новым обращенным к ней запросом. Топология изменяется таким образом, что наиболее востребованная информация оказывается наиболее доступной.

Ассоциативность сети заключается не только в адресации по содержанию, но и в структуре взаимосвязей моделируемой предметной области, в которой близость понятий определяется не только и не столько семантикой, сколько самим функционированием сети, т.е. пользовательскими запросами и ответами на них.

Управление движением ресурса сквозь сеть осуществляется как самой топологией сети, которая направляет ресурс по ребрам с большей пропускной способностью, так и рекурсивным заданием нового входного множества и продолжением поиска в заданном направлении.

Введенные операции на графах позволяют сделать варьировать управление в соответствии с целями поиска.

Библиографический список

[Голицын, 1997] Голицын, Г.А. Образ как концентратор информации. Математика и искусство. Труды международной конференции. М., 1997, с.96-99.

[Жилякова, 2009] Жилякова, Л.Ю. Алгоритм построения ассоциативной ресурсной сети. // X международная научно-техническая мультиконференция. Актуальные проблемы информационно-компьютерных технологий, мехатроники и робототехники, 2009. с. 232-236.

[Жилякова, 2010] Жилякова, Л.Ю. Реализация рекурсивных запросов в динамической ассоциативной ресурсной сети. // Двенадцатая национальная конференция по искусственному интеллекту с международным участием КИИ'2010. Труды конференции, том 1. М. – Физматлит, 2010. с. 335-343.

[Кемени и др., 1970] Кемени, Дж., Снелл, Дж. Конечные цепи Маркова. М.: Наука. 1970.

[Кузнецов, 2009] Кузнецов, О.П. Однородные ресурсные сети. I. Полные графы. // Автоматика и телемеханика, 2009, № 11, с.136-147.

[Кузнецов и др., 2009] Кузнецов, О.П., Жилякова, Л.Ю. Двусторонние ресурсные сети – новая потоковая модель. // Доклады Академии Наук, 2010, – том 433, № 5, с. 1–4.

[Форд и др., 1966] Форд, Л.Р., Фалкерсон Д. Потоки в сетях М.: Мир, 1966. 276 с.

[Ahuja et al., 1993] Ahuja, R.K., Magnati, T.L., Orlin, J.B. Network Flows: Theory, Algorithms and Applications Prentice Hall, New Jersey, 1993.