ISBN 978-5-901795-13-2 ПРОГРАММНЫЕ СИСТЕМЫ: ТЕОРИЯ И ПРИЛОЖЕНИЯ. Переславль-Залесский, 2008

Д. Н. Степанов

Алгоритм назначения рецензентов как часть проведения научных конференций при поддержке информационной системы UPIS*)

Научный руководитель: д.ф.-м.н. С. В. Знаменский

Аннотация. Работа посвящена созданию новой версии алгоритма назначения рецензентов, который применяется во время проведения научных конференций в Университете города Переславля при поддержке информационной системы UPIS. На языке Perl написана программа, реализующая данный алгоритм.

1. Введение

Автоматизированная система управления Университетом города Переславля (УГП) UPIS [1] была запущена в эксплуатацию в 2007 году. Система совершенствуется и поныне, хотя уже доказала свою эффективность во время проведения научной конференции в прошлом году. Собственно, ее главным предназначением было обеспечить поддержку проведения конференций в УГП. В процессе реализации системы приходится решать задачи, не только относящиеся преимущественно к функционированию UPIS, но и алгоритмические, требующие довольно нетривиальных алгоритмов. Одна из таких задач — назначение рецензентов для статей, поданных на конференцию. Текущая реализация алгоритма для ее решения работоспособна, но не слишком эффективна в плане раздачи статей рецензентам (уже были попытки его модернизации [2]). Кроме того, начиная со следующего года, скорее всего, произойдут некоторые изменения в самой концепции объектов «рецензент» и «статья», о чем будет рассказано ниже. В данной работе предложен алгоритм, который, возможно, выигрывает в эффективности у существующего и отвечает новой концепции.

*) Представлено по тематике: *Математические основы программирования.*

2. Постановка задачи

Имеется некоторая конференция, для множества конференций в системе UPIS существует специальный класс «Конференция», объекты которого имеют довольно много атрибутов, в том числе список направлений или тем данного мероприятия. Одна из основных концепций системы сетевого администрирования Nadmin [3] и системы UPIS (вторая, собственно, и реализована с использованием ядра первой) состоит в том, что если у нас имеется некоторый класс объектов, то для доступа к его конкретному экземпляру можно указывать значения полей данного объекта (например, для доступа к *одному* конкретному объекту класса «Пользователь» указываем значение поля «Логин»), а можно использовать некоторое натуральное число (ID), которое является одним из полей объекта и является уникальным для всех объектов данного класса. Уникальность достигается с помощью так называемой сквозной нумерации. Такая нумерация с помощью ID существует у большинства классов систем Nadmin и UPIS, в том числе и для класса «Персона». Пусть общее количество направлений конференции равно N, и каждое из них имеет свой порядковый номер.

2.1. Статьи

Имеется множество статей, поданных на конференцию. Каждая статья имеет определенный набор атрибутов.

2.1.1. *ID статьи*

Подобно классу «Персона», доступ к ОДНОМУ конкретному объекту класса «Статья» также можно осуществить по некоторому натуральному числу.

2.1.2. *Список владельцев статьи*

Владельцами статьи являются объекты класса «Персона», следовательно, достаточно хранить список их ID. Под словами «владельцы статьи» подразумеваются, во-первых, автор(ы) статьи (допустим, студенты, писавшие статью), а во-вторых, научные руководители статьи. В соответствии с текущими правилами проведения научных конференций, если человек является автором статьи, то он просто обязан участвовать в конференции и в качестве рецензента. Ни один человек из числа владельцев данной статьи не может получить эту статью на рецензирование.

2.1.3. *Вектор направленности статьи*

Когда автор подает статью на конференцию, то он указывает, к каким направлениям конференции относится его статья. В текущей реализации алгоритма есть только две градации отношения статьи к каждому направлению: 0 и 1 (*«статья* не *относится к данному направлению»* и *«статья относится к данному направлению»*). Но как показывают жизненные реалии, в мире существует не так много вещей, для которых данная градация является достаточной. Поэтому в предлагаемом здесь новом подходе диапазон был расширен до трех значений: 0, 1 и 2 (*«статья не имеет ничего общего с данным направлением или отношение к данному направлению весьма и весьма условное»*, *«в статье, в общем-то, затрагиваются некоторые вопросы из данной научной сферы, но не слишком глубоко»* и *«определенно, в статье затрагиваются вопросы из данного направления, причем достаточно глубоко и не в паре строчек доклада»*). Все направления у нас пронумерованы. Список из N чисел из диапазона 0, 1, 2 и будет составлять вектор направленности статьи. В принципе, градация в данном случае может быть расширена еще больше (например от 0 до 9).

2.2. Рецензенты

Имеется множество персон, пожелавших принять участие в конференции в качестве рецензентов. Все авторы статей обязаны быть рецензентами (по крайней мере, в рамках студенческой конференции). О каждом рецензенте известно следующее.

2.2.1. *ID рецензента*

Так как рецензентом может быть только зарегистрированная в системе персона, то здесь мы имеем просто ID соответствующей персоны.

2.2.2. *Вектор предпочтений рецензента*

Здесь полная аналогия с вектором направленности статьи. Рецензент заполняет данные о том, какие направления конференции ему импонируют или он считает себя в них экспертом, к чему он относится нейтрально или где его познания не слишком велики, и в каких направлениях он считает себя дилетантом или ему это совсем неинтересно.

2.2.3. *Количество рецензируемых статей*

Рецензент указывает, сколько статей он хотел бы отрецензировать. В атрибутах конференции можно указывать минимальное количество работ для одного человека, рецензент не может попросить работ меньше, чем это число. В существующем алгоритме рецензент указывает не количество статей, а количество порций статей на рецензирование, причем он не имеет возможности узнать размер каждой порции, так как это значение вычисляется после того, как все статьи будут поданы на конференцию и все рецензенты заполнят данные о себе (точнее, все, кто успеет до определенной даты). В дальнейшем для обозначения этой величины (для одного конкретного рецензента) мы будем использовать выражение *possib*.

Итак, входные данные и их формат описаны. Сама же задача заключается в том, чтобы раздать статьи таким образом, что рецензентов, недовольных раздачей, будет как можно меньше.

3. Методы исследования

Для начала необходимо определить, что же такое «удачная раздача статей». Один из способов объективной оценки степени «удачности» вполне очевиден: мы имеем вектора направленности статей и вектора предпочтений рецензентов и можем вычислять углы между ними. Но возможна и такая ситуация: вектор направленности данной статьи и вектор предпочтений данного рецензента вроде как полностью совпадают, но у рецензента нет никакого желания работать с этой статьей просто потому, что человек — существо, обладающее эмоциями, симпатиями и обычно непостоянным настроением. Обратное тоже реально: мы ожидаем, что у рецензента с данной статьей будут большие проблемы, а он после конференции утверждает, что читал эту работу с упоением и всячески расхваливает автора. Увы, все это в алгоритме учесть никак не получится.

Для начала нужно определить, сколько рецензентов будет у каждой статьи. Складываем количества статей, которые хотят обработать все рецензенты (назовем эту величину sum_possib), и делим это на количество статей. В результате получим некоторое число (назовем его $aver\_float$), скорее всего, отличное от целого. Округлим его в большую сторону (получим новую величину $aver$), после этого нужно добиться того, чтобы при делении величины sum_possib на количество статей мы получили уже не $aver\_float$, а $aver$. Есть, по крайней

мере, два способа: или увеличивать значения величины *possib* у случайных рецензентов, или у тех, кто требует себе статей меньше всего. Теперь у каждой статьи появляется новый атрибут rev_of_pap, показывающий, скольким рецензентам осталось ее назначить, первоначально у всех статей он равен $aver$. В текущей версии алгоритма есть возможность для разных секций конференции (именно секций, а не направлений) назначать разное количество рецензий для одной статьи. Я же считаю, что у всех статей должно быть одинаковое число рецензентов. В противном случае трудно говорить о равноправии статей, в частности, при вычислении места, которое заняла статья по итогам конференции.

Особенность предложенного алгоритма состоит в том, что статья (на втором и третьем этапе раздачи) назначается рецензенту на основании угла между вектором направленности статьи и вектором предпочтений рецензента, между направлениями конференции не делается никакого различия. Поэтому возникает опасность, что какой-то рецензент получит совершенно неподходящую для него статью, даже если этот угол весьма мал. Назовем статью специфичной по i-тому направлению, если она удовлетворяет двум критериям:

(1) i-тая координата в векторе направленности равна двум;
(2) Длина вектора направленности статьи не превышает некоторого фиксированного числа $lim\_specific$.

Точно так же определяется понятие специфичного по i-тому направлению рецензента. Добавляются новые исходные данные: номера направлений конференции, которые являются специфичными (по мнению организаторов), а также число $lim\_specific$ (можно его задавать, а можно вычислить).

Собственно, с этого места начинается раздача статей. Она будет произведена в несколько этапов.

3.1. Раздача специфичным рецензентам специфичных статей

Для каждого из специфичных направлений делаем следующее. В цикле просматриваются все специфичные по данному направлению рецензенты x. Если какой-то x уже получил статей столько, сколько хотел, то переходим к следующему рецензенту. Для каждого x просматриваются все специфичные по данному направлению статьи p. И если x не является владельцем p, и p еще не роздана $aver$ число раз,

то p назначается рецензенту x. После того, как для x просмотрены все статьи, можно случайным образом переставить элементы в списке имеющихся статей (в качестве алгоритма перестановки использовался один из методов, изложенных в [4]). Тем самым мы исключаем ситуацию, когда люди, стоящие в начале списка рецензентов, получали статьи, стоящие преимущественно в начале списка работ.

В псевдокоде это выглядит примерно так:

```
procedure назначить ( paper $p, reviewer $x ) {
    %res{$x}{$p} = 1;
    // Глобальный хеш хешей, где храним информацию о том,
    // что персона такая-то ($x) -- рецензент статьи такой-то ($p)
    ($x->possib)--;
    ($p->rev_of_pap)--;
}

foreach $x from @all_reviewers {
    foreach $p from @all_papers {
        if ( ($x->possib) == 0 ) then break;
        if ( ($p->rev_of_pap) == 0 ) then continue;
        if ( isOwner( $p, $x ) ) then continue;
        if ( специфика($p) == специфика($x) ) then назначить( $p, $x );
    }
    случайная_перестановка(@all_papers);
}
```

Этот этап раздачи можно провести немного по другому: не для каждого рецензента просматриваются все статьи, а для каждой статьи будут просматриваться все рецензенты (со случайной перестановкой элементов в списке рецензентов). Преимущество этого способа в том, что практически все специфичные рецензенты получат хотя бы небольшое (а может, и достаточно приличное) количество работ, которые им точно подойдут. В первом же случае есть вероятность, что специфичным рецензентам, стоящим в начале списка всех рецензентов, достанутся почти все специфичные работы и,их потребности в статьях будут во многом удовлетворены, а стоящим в конце они могут совсем не достаться.

Вообще, по этому поводу можно сделать важное замечание: если мы хотим, чтобы рецензенты получили наиболее подходящие для себя статьи, то рецензенты будут брать себе статьи; если же мы хотим, чтобы статьи получили наиболее подходящих рецензентов, то статьи раздаем рецензентам.

Если же общее число направлений достаточно велико, то специфичные, но близкородственные направления (например, несколько направлений, объединенных общим словом «экономика») можно объединять в специфичные группы и определять теперь специфичность статьи или рецензента не по направлению, а по группе. Естественно, для каждой специфичной группы будет свое число $lim\_specific$.

3.2. Основная раздача

Сортируем по возрастанию список рецензентов по длине векторов их предпочтений. Тогда в первую очередь статьи будут получать самые «маломощные» рецензенты, наиболее же «универсальные» будут получать их в конце второго этапа. Теперь для каждого рецензента x делаем следующее. Просматриваются все статьи, и ему даются наиболее подходящие для него работы, то есть те, для которых угол между его вектором предпочтений и вектором направленности статьи будет каждый раз минимальным. Не даем рецензенту статьей, если:

- его потребности в количестве статей полностью удовлетворены;
- нет ни одной статьи, для которой выполнялись бы следующие требования:
  (1) x — не владельцец этой статьи;
  (2) эта статья еще не полностью роздана;
  (3) x еще не брал эту статью.

Опять же, и на этом этапе можно статьи раздавать рецензентам, если для нас главное — дать статьям наиболее компетентных рецензентов.

Второй этап раздачи может закончится тем, что все статьи полностью розданы и потребности всех рецензентов полностью удовлетворены, и на этом алгоритм заканчивает свою работу. Но тесты показывают, что такое случается крайне редко. Остаются рецензенты, которые недополучили статьи. То есть свободные работы вроде как и есть, но дать их мы им не можем, так как они или являются их владельцами, или уже брали эти статьи. Тогда мы делаем так: увеличиваем на единицу значение $aver$, у каждой работы увеличиваем на единицу значение ее поля rev_of_pap (скольким еще рецензентам осталось раздать данную статью) и опять выполняем второй этап

раздачи. Если потребности всех рецензентов полностью удовлетворены, а часть работ осталась не розданной (мы же увеличивали у всех статей значение поля rev_of_pap), то переходим к третьему этапу раздачи.

Псевдокод этого этапа:

```
$all_is_good = False;

while ( ! $all_is_good ) {
    foreach $x from @all_reviewers {
        sort_1 ( @all_papers, $x);
        // Сортируем так, что в начало списка всех работ попадут статьи,
        // наиболее подходящие для $x, а в конец -- самые неудобные для $x.
        // В процессе сортировки используем углы между векторами.
        foreach $p from @all_papers {
            if ( ($x->possib) == 0 ) then break;
            if ( ($p->rev_of_pap) == 0 ) then continue;
            if ( isOwner( $p, $x ) ) then continue;
            if ( %res{$x}{$p} != 1 ) then
            // если ему еще не дали эту статью, то
            назначить( $p, $x );
        }
    }
    if ( все_рецензенты удовлетворены() ) then
        $all_is_good = True;
    }
    else {
        foreach $p from @all_papers {
            ($p->rev_of_pap)++;
        }
        continue;
    }
}
```

3.3. Раздача оставшихся статей

Итак, все рецензенты получили статьи, но часть работ осталась не до конца розданной. Но так как мы условились, что у всех работ должно быть одинаковое количество рецензентов, то их нужно обязательно раздать. Конечно, при этом часть рецензентов получит статей немного больше, чем хотела, но мы постараемся сделать так, чтобы эти дополнительные работы были как можно ближе им по тематике. На этот момент значение поля *possib* у всех рецензентов равно нулю.

Здесь уже опасно применять тот вариант, когда для каждого рецензента просматриваются все статьи. Непонятно, что будет делать каждый рецензент: он обязательно должен получить хотя бы одну статью, но весьма высока вероятность того, что он получит совершенно неподходящую для себя статью, так как к этому моменту свободных работ осталось немного.

В атрибутах конференции можно задать величину *quota*, которая будет показывать, на сколько больше можно дать статей рецензенту, чем он хотел. Конечно же, не принимается во внимание тот факт, что мы увеличиваем значение поля *possib* у некоторых рецензентов, когда добиваемся того, чтобы величины $aver\_float$ и *aver* сравнялись между собой. Если же для нас важна не абсолютная, а относительная погрешность, то тогда *quota* — некий коэффициент, больший единицы. Пусть она равна, например, 1.333. Пусть некто X попросил себе 5 статей. На начальном этапе работы алгоритма это число увеличилось до 7 (из-за того что величина *aver* должна быть натуральной). В итоге X получит не более чем 10 ($1.333 * 7 = 9.333$) статей.

Для каждой статьи выполняем следующее. Вначале сортируем список рецензентов по тому, сколько еще статей им нужно дать (сортируем по убыванию, в процессе работы алгоритма здесь будут появляться и отрицательные числа). Это частично способствует тому, что дополнительную нагрузку получит прежде всего тот, кто до этого получил этой дополнительной нагрузки меньше всех. Потом для статьи просматриваются все рецензенты, и выбираются наиболее подходящие (опять же, используем минимальный угол между векторами). Статью не даем рецензенту, если:

- этот рецензент — владелец данной статьи;
- этот рецензент уже получил данную статью;
- дополнительно этот рецензент получил работ больше, чем *quota* (или же $possib * (quota - 1)$).

Впрочем, сначала на финальной стадии раздачи можно применить первый этап алгоритма. Раздав специфичные статьи специфичным рецензентам (или же дав возможность специфичным рецензентам получить подходящие для себя статьи), мы добьемся того, что часть рецензентов хоть и получит дополнительную нагрузку, но эта нагрузка будет полностью соответствовать их «специализации».

Можно сделать так, чтобы величина *quota* не задавалась заранее, а использовалась немного по другому. Пусть первоначально она

равна единице. Условимся, что каждому рецензенту можно дать количество дополнительных статьей, не более чем *quota* (естественно, не забываем о том, что статьи должны получить наиболее подходящие для них рецензенты). Статьи закончились — программа завершает свою работу. Если же каждый рецензент получил очередную дополнительную статью, а свободные работы еще остались, то *quota* увеличиваем на единицу и продолжаем раздачу. Преимущество такого подхода — дополнительная нагрузка будет равномерно распределена между всеми рецензентами и для каждого из них будет весьма незначительна. Недостаток — хоть каждый в отдельности и получает совсем чуть-чуть дополнительных работ, есть вероятность того, что части рецензентов достанутся статьи, совершенно им неподходящие.

Псевдокод (когда *quota* не задается заранее):

```
$limit = -1;
foreach $p from @all_papers {
    sort_2 ( @all_reviewers, $p);
    // Сортируем так, что в начало списка всех рецензентов попали
    //  наиболее подходящие для $p, а в конец -- наименее подходящие.
    // В процессе сортировки используем углы между векторами.
    while( True ) {
        $is = False;
        foreach $p from @all_papers {
            if ( ($p->rev_of_pap == 0 ) then break;
            if ( %res{$x}{$p} == 1 ) then continue;
            if ( isOwner( $p, $x ) ) then continue;
            if ( ($x->possib) > $limit) then назначить( $p, $x );
            $is = True;
        }
        if ( ! $is ) then {
            if ( ($p->rev_of_pap) == 0 ) then break;
            else {
                $limit--;
                continue;
            }
        }
    }
}
```

3.4. Дополнение: работа с преподавателями и научными руководителями

Так как в рамках студенческой конференции все рецензенты делятся на не слишком опытных (основная масса студентов) и достаточно квалифицированных (преподаватели, научные руководители

статей), то неплохо бы иметь ситуацию, когда каждая статья имеет хотя бы небольшое число наиболее компетентных рецензентов. В атрибутах конференции задаем число pap_for_expert, показывающее, сколько мы хотим иметь таких рецензентов у каждой статьи (хотя не факт, что наше требование будет выполнено). Теперь нужно получить список рецензентов с достаточным уровнем квалификации. Это те, кто является научным руководителем хотя бы одной статьи, а также те, кто сам попросил о занесении его в этот список. Данные об остальных рецензентах пока что находятся отдельно.

Естественно, раздачу статей «профессиональным» рецензентам нужно выполнять в самом начале работы алгоритма. Мне кажется, здесь не имеет смысла искать среди научных руководителей и преподавателей специфичных рецензентов, поскольку они по определению должны быть достаточно универсальными рецензентами.

Для каждой статьи добавляем сначала одного наиболее подходящего рецензента из числа компетентных (для определения наиболее подходящего используем углы между векторами), затем для каждой статьи добавляем еще по одному компетентному рецензенту, и так делаем pap_for_expert раз. Естественно, при раздаче нас преследуют ограничивающие факторы:

- большинство компетентных рецензентов — научные руководители, причем не одной статьи, а двух, пяти и даже более, и следовательно, являются их владельцами;
- нельзя дать рецензенту одну и ту же статью два раза;
- статью нельзя дать рецензенту, если он уже взял статей, сколько хотел (впрочем, здесь это не столь критично, так как наверняка «профессиональные» рецензенты в среднем будут просить себе ощутимо больше статей, чем не слишком квалифицированные).

Может такое случиться, что в процессе раздачи статей компетентным рецензентам какая-то работа попала в такую ситуацию, когда отдать ее на рецензирование попросту больше некому (здесь роль играют ограничивающие факторы, особенно первый), но алгоритм на этом шаге не имеет права завершиться. Тогда мы эту статью помечаем, выдаем сообщение, что ее не удалось раздать pap_for_expert компетентным рецензентам и в дальнейшем на этом этапе раздачи такую статью игнорируем. По окончанию этого этапа алгоритма компетентных рецензентов и всех остальных объединяем в общий список и на последующих этапах между ними никаких различий не делаем.

4. Результаты

Предложенный алгоритм запрограммирован на языке Perl. Выбор обусловлен тем, что именно этот язык использовался (и поныне используется) при разработке и совершенствовании систем Nadmin и UPIS. Кроме того, в этом языке изначально заложена поддержка таких структур данных, как списки и ассоциативные массивы (хеши), они сильно упрощают процесс написания даже весьма сложных программ, и с ними легко работать. В настоящий момент ввод и вывод данных — файловый. В принципе, внедрить в систему UPIS написанную программу не составляет труда.

5. Выводы

Самая главная проблема, возникшая при создании алгоритма — проблема входных данных. С одной стороны, «правильный» алгоритм должен работать «правильно» при любом количестве статей, рецензентов, любых векторах направленности статей, предпочтений рецензентов и т. д. Но мы решаем не какую-то абстрактную задачу, для решения которой, возможно, еще нужно найти применение. И поэтому очень хочется иметь входные данные, максимально приближенные к реалиям Университета города Переславля. Как мне кажется, эти реалии удовлетворяют следующим условиям:

- рецензентов будет ощутимо больше, чем статей (мой прогноз — примерно в два раза);
- будет некоторое количество преподавателей, которые являются научными руководителями не одной, а двух, пяти и даже более статей;
- универсальных рецензентов (тех, у кого вектор предпочтений по своей длине достаточно велик) среди студентов (основных авторов статей) и преподавателей будет немного. Скорее всего, будут присутствовать рецензенты, универсальные по некоторой совокупности направлений (например, по направлениям, объединенных общим словом «экономика»);
- статей, посвященных большинству направлений, практически не будет;
- скорее всего, будет несколько рецензентов, которые затребуют очень большое количество статей (сравнимое с числом всех присланных работ).

И все же вряд ли стоит надеяться, что каждая конференция будет удовлетворять таким условиям. Вполне может случится, что уже на следующей научной конференции все кардинально изменится.

И вторая проблема — нужно все же определиться с тем, что мы хотим получить в результате работы алгоритма: ситуацию, когда статьи получили наиболее компетентных для них рецензентов, или же ситуацию, когда рецензенты получили наиболее подходящие для себя работы.

Список литературы

[1] Коряка Ф. А. *Автоматизированная система управления ВУЗом — UPIS*, 2007.

[2] Предыдущая версия модернизации алгоритма назначения рецензентов. — `http://wiki.botik.ru/IS4UGP/SpecReviewer`.

[3] Е. Ермилова; А. Карлаш; А. Нестеров; П. Жбанов; Ю. Шевчук *Nadmin — система администрирования для региональных сетей*, 2004.

[4] Т. Кристиансен; Н. Торкингтон Perl Библиотека программиста: Издательство «Питер», 2000. — 739 с.

D. N. Stepanov. *Algorithm of assigning the reviewers as a part of scientific conference holding at the University of Pereslavl supported with the UPIS* // Proceedings of Programn Systems institute scientific-practical conference “Program systems: Theory and applications”, devoted to the 15^{th} anniversary of Pereslavl University named A. K. Ailamazyan. — Pereslavl-Zalesskij, 2008. — p. 221 — 233. — ISBN 978-5-901795-13-2 *(in Russian).*

Abstract. The work is devoted to creation of a new version of the algorithm of assigning the reviewers, which is used during scientific conferences supported with the UPIS. Perl is used for writing the program implementing the given algorithm.

Перевод проверен: Л. Н. Валеева