Гринберг Я.Р., Курочкин И.И., Корх А.В.

Оптимизация трафика в сетях передачи данных методом линейного программирования. Предварительные данные.

ИППИ РАН, Москва, Россия.

1. Введение.

Задача маршрутизации информационных потоков в телекоммуникационных сетях с наименьшими издержками использования сетевых ресурсов и по сей день остается одной из ключевых при моделировании архитектур и топологий сетей передачи данных. Несмотря на то, что за последние пятнадцать лет скорость передачи данных для конечных пользователей возросла, в среднем, в 200 раз (с 52 кБит/сек до 10 Мбит/сек), рост объема пользовательского трафика был не меньшим. Более того, по прогнозам Cisco [1] до 2017 года предполагается существенный рост объема трафика от потокового видео и пиринговых сетей (см. рис. 1).

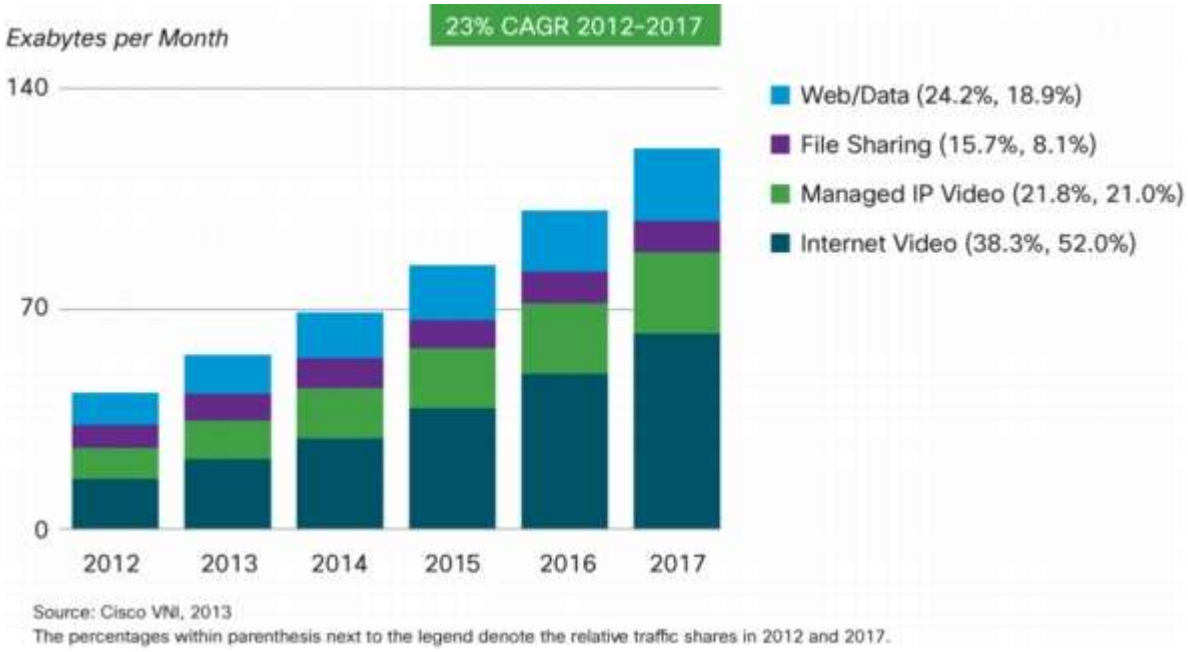

Рис. 1. Предполагаемая структура мирового пользовательского трафика [1].

В частности, специфика потокового видео предполагает гарантированное нахождение маршрута до конечного пользователя и резервирование доли пропускной способности каналов связи. В работах [2][3][4] была предложена модель телекоммуникационных сетей с заявками на передачу потока данных между узлами сети, возникающими последовательно во времени; и оригинальные алгоритмы маршрутизации этих потоков, максимизирующие суммарный объем трафика, и экономящие ресурс сети, такой как пропускная способность каналов связи. Вместе с этим, предложенные алгоритмы носили эвристический характер и не давали возможность оценить эффективность их функционирования. В настоящей работе будут предложены модификации данных алгоритмов, решающие точную задачу оптимизации прокладки потоков в сетях, и позволяющие дать верхнюю оценку величине суммарного потока, обслуживаемого эвристическими алгоритмами.

Структура представленной работы следующая: во второй части будет дано описание модели телекоммуникационных сетей и даны краткие описания эвристических алгоритмов, использовавшихся для маршрутизации потоков в них. В третьей части будет сформулирована оптимизационная задача маршрутизации трафика и предложен составной алгоритм, позволяющий получить точную маршрутизацию потоков данных в сети. В четвертой части

будут приведены результаты численных экспериментов по прокладке трафика в сетях и, наконец, в пятой части будет дано некоторое заключение по проделанной работе.

2. Постановка задачи. Алгоритмы маршрутизации.

Определим математическую модель телекоммуникационной сети, которой будем пользоваться на протяжении данной работы. Сеть передачи данных может быть представлена связным графом $G = (V, E)$, где V - множество узлов графа, E - множество ребер графа, соединяющих узлы. Каждому ребру $e_{ij} \in E$, $\{i, j\} \in V$ графа G поставлено в соответствие неотрицательное число $c_{ij} \geq 0$ - пропускная способность ребра. Дополнительно введем следующее понятие потока f между вершинами s и t - неотрицательная функция на ребрах графа, обладающая следующими свойствами: поток неотрицателен для любого ребра $e_{ij} \in E$, то есть $f_{ij} \geq 0$; $\sum_k f_{ki} = \sum_j f_{ij}$, $\forall i \in V, i \neq \{s,t\}$ - поток не накапливается в промежуточных узлах между s и t; $f_{ij} \leq c_{ij}$, $e_{ij} \in E$ - поток по ребру не превышает ее пропускной способности. Остаточная пропускная способность ребра e_{ij} определяется как разность пропускной способности ребра и потока по ней, то есть $c_{ij}^f = c_{ij} - f_{ij}$. Остаточной сетью называется сеть $G^f = (V, E^f)$, получаемая из исходного графа G, в которой остаются ребра с положительной остаточной пропускной способностью. В каждый момент времени требуется проложить поток заданной величины (в дальнейшем эта ситуация будет называться *возникновением заявки в сети*) между одной из пар вершин $\{s_1, t_1\}, \ldots, \{s_n, t_n\}$, называемых полюсами. Время жизни заявок конечно – если заявка была удовлетворена, и маршрут между полюсами был проложен, то по истечении времени жизни заявка сбрасывается, освобождая соответствующую пропускную способность ребер сети. Множество потоков между каждой определенной парой полюсов $\{s_m, t_m\}$ называется продуктом ν_m.

Как уже было сказано, для решения задачи маршрутизации потоков в работах [2][3][4] предлагался ряд алгоритмов, основанных на принципе экономии дефицитного ресурса. В данном случае, в качестве дефицитного ресурса предполагалась пропускная способность каналов связи, представленных ребрами графа сети. Дополнительно, к существующей модели сети требовалось вычисление $\Re_m$ - множеств минимальных разрезов каждого продукта ν_m и R_m - пропускных способностей этих разрезов. Напомним, что разрезом называется [5] множество дуг, удаление которых разрывает сеть на несколько несвязных между собой частей, при этом полюса оказываются в разных частях. Минимальный разрез – множество ребер разреза с минимальной суммой их остаточных пропускных способностей. Основная идея алгоритмов последовательного заполнения сети потоками продуктов основывалась на следующей эвристике: поиск пути наименьшей стоимости, причем значения стоимости присваивались ребрам сети в зависимости от величины разделяемого ими дефицитного ресурса. В зависимости от способа определения стоимости дуг было выделено три типа алгоритмов последовательного заполнения сети потоками продуктов:

1. **Простой алгоритм**. Каждому ребру с ненулевой остаточной пропускной способностью назначается единичный вес. Таким образом, вычисляемый путь потока продукта между парой корреспондирующих вершин является кратчайшим путем между данными вершинами сети. Этот алгоритм позволяет дать оценку снизу величины суммарного потока в сети за все время имитационного моделирования.
2. **Дуговые алгоритмы.** Ребрам назначаются стоимости обратно пропорциональные их остаточной пропускной способности. Таким образом, найденные путь минимальной стоимости будет использовать ребра с большим запасом дефицитного ресурса – пропускной способности. В упомянутых работах это *дуговой* и *субоптимальный дуговой* алгоритмы.

3. **Минимально-разрезные алгоритмы**. Для каждого ребра с ненулевой остаточной пропускной способностью определяется ее принадлежность к минимальным разрезам между всеми парами корреспондирующих полюсов. Веса ребрам назначались обратно пропорционально величинам минимальных разрезов, которым они принадлежат. Таким образом, найденные пути наименьшей стоимости не проходили через «узкие места» сети, расходуя избыточную пропускную способность каналов связи. В упомянутых работах это *субоптимальный минимально-разрезный, аддитивный* и *гибридный* алгоритмы.

Наиболее оптимальным считался тот алгоритм, который удовлетворял наибольшее число поступающих в сеть заявок. По результатам проведенных численных экспериментов таковым являлся субоптимальный минимально-разрезный алгоритм. Тем не менее, при возникновении заявки, которая не может быть удовлетворена с помощью какого-либо из перечисленных эвристических алгоритмов, оставался вопрос: возможно ли при данном множестве существующих удовлетворенных заявок с неистекшим времени жизни перестроить потоки в сети так, чтобы можно было удовлетворить поступившую заявку?

3. Модель линейного программирования и составного алгоритма.

Для того, чтобы ответить на поставленный вопрос рассмотри задачу нахождения многопродуктового потока при заданном числе заявок с неистекшим временем жизни [6].

$$\text{minimize} \qquad \sum_{i=1}^{k}\sum_{e\in E} f_e^i$$

$$(3.1)\quad \sum_{(j,i)\in E} f_{ji}^l - \sum_{(i,j)\in E} f_{ij}^l = 0 \quad \forall i \in V, i \notin \{s_1, t_1, \ldots s_k, t_k\} \qquad \forall l \in [1\ldots k]$$

$$(3.2)\quad \sum_{l=1}^{k} f_{ij}^l + \sum_{l=1}^{k} f_{ji}^l \le c_{ij} \qquad \forall (i,j) \in E$$

$$(3.3)\quad \sum_{(s_i,j)\in E} f_{s_i j}^i = d_i \qquad \sum_{(j,s_i)\in E} f_{j s_i}^i = 0 \qquad \forall i \in [1\ldots k]$$

$$(3.4)\quad \sum_{(j,t_i)\in E} f_{j t_i}^i = d_i \qquad \sum_{(t_i,j)\in E} f_{t_i j}^i = 0 \qquad \forall i \in [1\ldots k]$$

$$(3.5)\quad f_{ij}^l \ge 0 \qquad \forall (i,j) \in E \qquad \forall l \in [1\ldots k]$$

Здесь f_{ij}^l - неотрицательный поток l-ого продукта по дуге $e_{ij} \in E$, c_{ij} - неотрицательная пропускная способность дуги e_{ij}; d_l - величина потока l-ого продукта, $1 \le l \le k$. Суммарный поток минимизируется, чтобы обеспечить равномерность его распределения по сети и максимизировать остаточные пропускные способности дуг. Первое ограничение равенства обеспечивает ненакопление потока в промежуточных узлах (аналогично первого закона Кирхгофа); второе ограничение отвечает за то, чтобы сумма потоков по любому из ребер сети не превосходила его пропускной способности. Третье и четвертое ограничения обеспечивают необходимые величины потоков различных продуктов между соответствующими полюсами. Необходимо заметить, что для терминальных вершин требуется дополнительно задать нулевые входящие потоки их продуктов, чтобы избежать возникновения несвязанных циклов потоков в полюсах. Последнее неравенство требует, чтобы потоки продуктов оставались неотрицательными величинами. Такую задачу еще называют задачей допустимости многопродуктового потока.

Соответственно можно предложить составной алгоритм маршрутизации потоков последовательных заявок в сетях, идея которого будет следующей: при поступлении очередной заявки, идет попытка удовлетворить ее с помощью эвристических алгоритмов, а при невозможности маршрутизации очередной заявки, нахождение точного решения с помощью решения задачи линейного программирования. Рассмотрим структуру работы составного алгоритма подробно по частям. Иллюстрация приведена ниже на схеме 2.

Начало работы:

На вход подается очередная *K* - ая заявка. Поскольку заранее неизвестно, будет ли она удовлетворена с помощью эвристического алгоритма, то формируется вектор действительных заявок. Для этого потребуется вектор оставшихся времен жизни всех заявок с *1-ой* по *K - ую*.

Формирование вектора действительных заявок:

Задачей этого блока, является определение того, какие заявки к *K - му* шагу будут еще действительны, т.е. время их оставшихся жизней будет не меньше, чем 1. Данная операция осуществляется в предположении, что в течение всего времени работы имитационной модели существует вектор оставшихся времен жизней удовлетворенных заявок, увеличивающийся на один элемент при каждой удовлетворенной заявке. После просмотра последнего элемента вектора времен жизней, запишем в вектор действительных заявок все те заявки, чье остаточное время жизни больше 1.

Выходные данные: вектор действительных заявок, вектор оставшихся времен жизни для оставшихся заявок. Переходим к следующему шагу.

Формирование ограничений равенства:

Считаем в векторе действительных заявок для каждого продукта, сколько заявок относится к тому или иному продукту. Их число определяет ограничения равенства для соответствующего продукта. После чего переходим к следующему шагу.

Выходные данные: вектор ограничений равенства для каждого продукта. Переходим к следующему шагу.

Алгоритм последовательного заполнения сети:

Пытаемся удовлетворить заявку с помощью одноуровневых последовательных алгоритмов заполнения сети (простой, дуговой, субоптимальный минимально-разрезный, аддитивный минимально-разрезный, гибридный, субоптимальный дуговой) или их двухуровневыми модификациями. Если удается удовлетворить *K - ую* заявку, то время ее жизни записывается в вектор оставшихся жизней заявок, все элементы вектора уменьшаются на 1. Уничтожаем потоки заявок, чье оставшееся время жизни стало равно нулю. Если в очереди остались еще заявки, то переходим в начало составного алгоритма. В случае невозможности удовлетворения *K - ой* заявки переходим к блоку решения задачи линейного программирования.

Выходные данные: матрица остаточных пропускных способностей дуг сети, вектор оставшегося времени жизни удовлетворенных заявок.

Блок решения задачи линейного программирования:

С помощью вектора действительных заявок и новой заявки формируются ограничения равенства на полюсах (3.3) – (3.4). С помощью матрицы исходных пропускных способностей описываются неравенства (3.2). Далее с помощью специализированного солвера (решателя) задач линейного программирования ищется решение задачи допустимости.

Если задача не имеет решения, то считается, что заявка не может быть удовлетворена при текущем наборе обслуживаемых заявок и сбрасывается из сети.

Если задача допустимости имеет решение, то считается, что заявка может быть удовлетворена и включается в вектор действительных заявок, а ее оставшееся время жизни включается в вектор времен жизней всех удовлетворенных заявок в сети. Затем все элементы этого вектора уменьшаются на 1. Уничтожаем потоки заявок, чье оставшееся время жизни стало равно нулю. Вторым важным шагом является модификация простых путей, по которым до *K - ой* заявки проходили потоки продуктов предыдущих заявок.

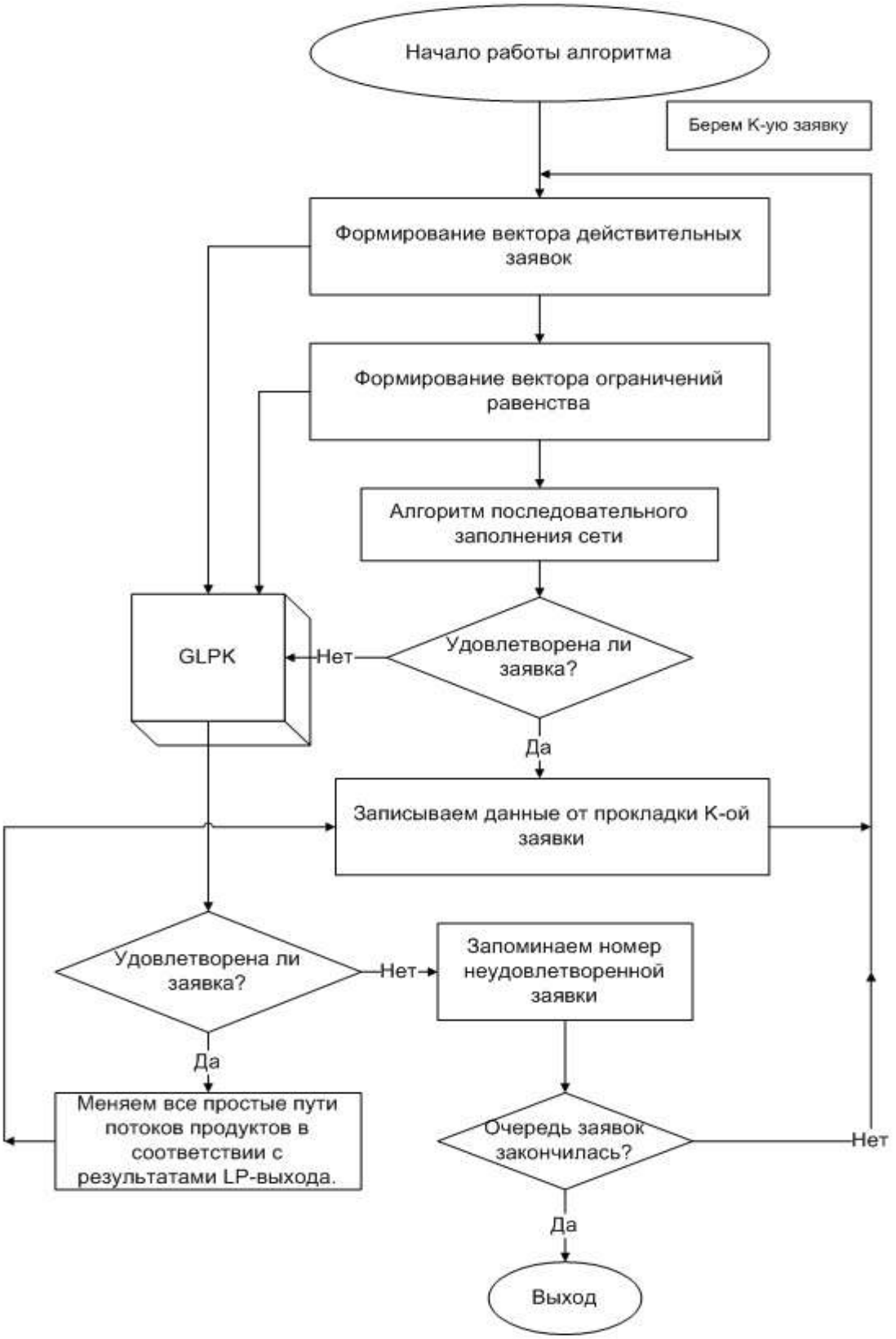

Рис. 2. Описание работы составного алгоритма.

Рассмотрим подробнее процедуру обновления простых путей потоков продуктов. Как было указано ранее, в приведенной формулировке задачи допустимости решением будет следующая структура данных: $\| \left(v_i, v_j\right) \| l \| f_{ij}^{l}$, где v_i и v_j - начальная и конечная вершины дуги графа $G = (V, E)$, l - номер продукта, эквивалентный номеру пары полюсов, между

которыми проходит данный продукт, f_{ij}^{l} - величина потока l-ого продукта, протекающего по ребру (v_i, v_j). Соответственно для каждого продукта можно выделить подмножество дуг, по которым протекает поток с суммарной интенсивностью равной количеству действительных заявок, удовлетворяемых на данной паре полюсов. Выделив это подмножество дуг, восстанавливаются все простые пути потоков, ведущих из источника в сток. По минимальной величине потока через ребро, определяется максимальная мощность потока, протекающего через каждый простой путь. Далее выбирается подмножество простых путей так, чтобы сумма потоков по этим простым путям была равна суммарной интенсивности потока между данной парой полюсов, т.е. пропорциональная числу заявок с неистекшим временем жизни между этими терминальными вершинами. Далее последовательно сравниваются пути, полученные с помощью алгоритма линейного программирования (LP) и пути, сформированные до поступления K - *ой* заявки. В этом случае возможно две альтернативы:

1. если старый простой путь движения потока f^i и путь, найденный с помощью LP совпадают, то старый простой путь остается в сети без изменений;
2. если старый путь и путь LP различаются, то меняем его на решение, полученное с помощью линейного программирования.

При этом, совпадающие пути оставляются по мере их увеличения времени их попадания в сеть, т.е. если выбирать, какой совпадающий путь оставить в первую очередь, «старый» или «новый», то оставляем «старый».

Таким, образом, в результате работы составного алгоритма можно получить оценку сверху эффективности работы, как одноуровневых последовательных алгоритмов, так и их двухуровневых модификаций, по величине суммарного потока, прошедшего через сеть за все время работы имитационной модели.

4. Численные эксперименты и их результаты.

Для численных экспериментов использовались сети со стохастической и кластерной топологиями (количество вершин колебалось от 30 до 45, плотность заполнения сети ребрами колебалась от 10% до 20%). В каждый момент времени произвольная пара корреспондирующих вершин инициировала заявку - требование проложить по сети единичный поток между этими двумя полюсами, имеющий конечное время жизни. По истечении времени жизни, заявка снималась, соответствующий поток исчезал, высвобождая ресурс сети, в данном случае остаточную пропускную способность ребер сети. Длина очереди заявок равнялась 45000, время жизни заявок колебалось от 100 до 10000 единиц времени с математическим ожиданием в 3000 единиц времени.

На представленном ниже рисунке показано превышение доли удовлетворенных заявок в процентах от их общего числа составными алгоритмами над долей удовлетворенных заявок исходными алгоритмами последовательного заполнения сетей со стохастической топологией. В качестве одноуровневых алгоритмов и их составных модификаций (ЛП модификаций) были использованы *простой* (Simple), *субоптимальный минимально-разрезный* (EmpCut) и *субоптимальный дуговой* (EmpEdge), как наиболее эффективные представители существующих эвристик построения простого потока между двумя корреспондирующими вершинами.

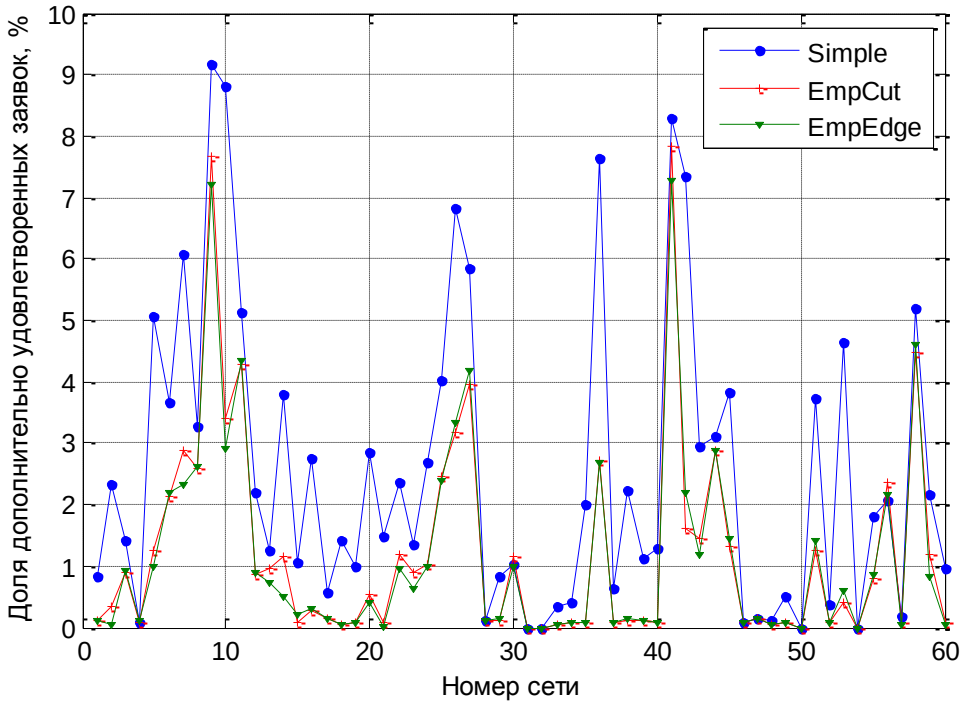

Рис. 3. Превышение доли удовлетворенных заявок в процентах от их общего числа составными алгоритмами над долей удовлетворенных заявок исходными алгоритмами последовательного заполнения сетей со стохастической топологией.

На рис.3 линии, соединяющие соседние точки оставлены для наглядности, несмотря на то, что результаты на соседних сетях никак не связаны. Данные имитационного моделирования показывают увеличение суммарного потока при использовании составного алгоритма (особенно явно это проявляется для простого алгоритма и его составной модификации). Однако, результат, полученный с помощью эвристических алгоритмов заполнения сетей потоками продуктов последовательных заявок, можно считать оптимальным, так как на достаточно большом числе сетей доля суммарного потока не на много отличается от доли суммарного потока составного алгоритма. Подробности в таблице 1.

Таблица 1. Число сетей, где доля заявок, удовлетворенных эвристическими алгоритмами отличается от доли заявок, удовлетворенных составным алгоритмом на заданное число процентов.

Отличие в доле удовлетворенных заявок	<5%	<4%	<3%	<2%	<1%
Simple	49	47	41	30	20
EmpCut	58	56	53	46	36
EmpEdge	58	55	54	45	40

На следующем рисунке показано превышение доли удовлетворенных заявок в процентах от их общего числа составными алгоритмами над долей удовлетворенных заявок исходными алгоритмами последовательного заполнения сетей со стохастической топологией.

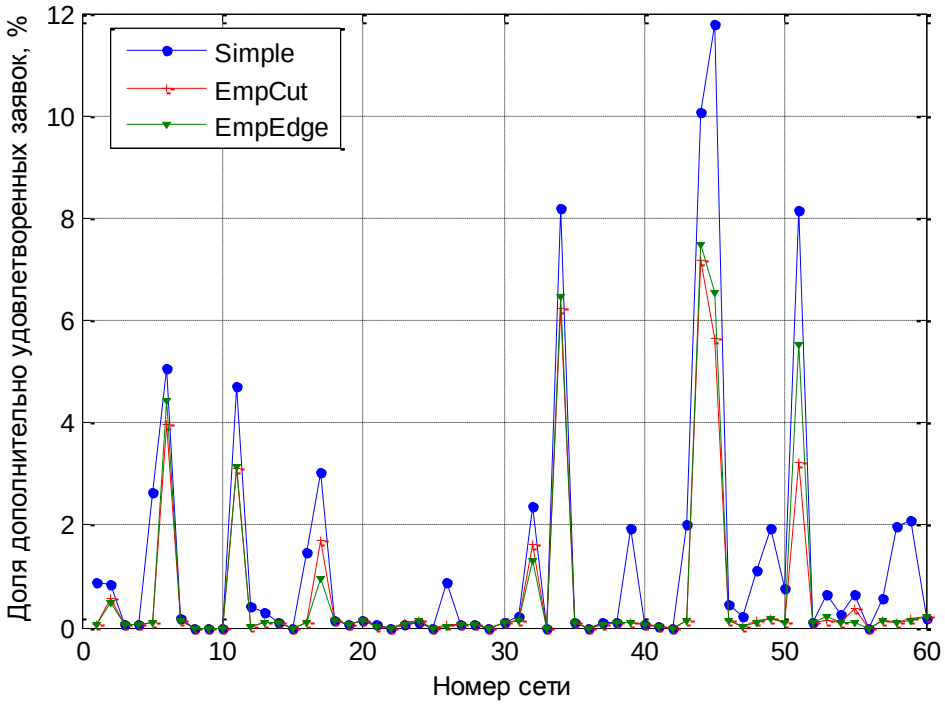

Рис. 4. Превышение доли удовлетворенных заявок в процентах от их общего числа составными алгоритмами над долей удовлетворенных заявок исходными алгоритмами последовательного заполнения сетей с кластерной топологией.

Как и в предыдущем случае (см. рис. 3) линии на графике между соседними значениями различных сетей оставлены для наглядности и лучшей иллюстративности. Оценим количество сетей, чей суммарный поток, полученный от одноуровневых последовательных алгоритмов заполнения, ненамного отличается от суммарного потока, полученного составными алгоритмами построения простых потоков.

Таблица 2. Число сетей, где доля заявок, удовлетворенных эвристическими алгоритмами отличается от доли заявок, удовлетворенных составным алгоритмом на заданное число процентов.

Отличие в доле удовлетворенных заявок	<5%	<4%	<3%	<2%	<1%
Simple	55	54	53	49	44
EmpCut	57	57	54	54	52
EmpEdge	56	55	54	54	52

Таким образом, в случае рассмотренных сетей, численные эксперименты показали, что в подавляющем большинстве случаев применение составного алгоритма дает увеличение суммарного потока через сеть, вместе с этим в субоптимальном минимально-разрезном и субоптимальном дуговом алгоритмах в большинстве сетей применение линейного программирования дает улучшение результата не более чем на 5% дополнительно удовлетворенных заявок. С другой стороны при моделировании сетей больших размеров (несколько сотен узлов) применение субоптимального минимально-разрезного алгоритма становится неприемлемым вследствие его большой вычислительной сложности - $O(|V||E|^2 P)$, где V - число узлов сети, E - число ребер графа, описывающего телекоммуникационную сеть, P - число пар полюсов сети. В этом случае становится целесообразным использование менее

точных, но более быстрых дугового и простого алгоритмов, и их составные модификации позволяют получить достаточно хороший результат.

5. Заключение.

В данной работе была рассмотрена задача оптимизации маршрутизации трафика в сетях передачи данных и дано ее описание в качестве задачи линейного программирования. Были рассмотрены алгоритмы, минимизирующие расход ресурса сети, а именно пропускной способности каналов, а также их модификации, позволяющие получить точное решение задачи маршрутизации многопродуктового потока. Численные эксперименты показали, что эвристика экономии дефицитного ресурса сети позволяет получить величину суммарного потока, близкую к максимально возможной.

Литература

1. Cisco Visual networking Index: Forecast and Methology, 2012-2017. / Cisco public White Paper, 2013.
2. Афанасьев А. П., Гринберг Я. Р., Курочкин И. И. Равномерное заполнение телекоммуникационной сети каналами связи / В кн. Прикладные проблемы управления макросистемами, Труды ИСА РАН, том 8, изд-во УРСС, 2004. стр. 118-123.
3. Гринберг Я. Р., Курочкин И. И. Анализ результатов численного эксперимента по последовательному заполнению сетей со стохастической топологией / Проблемы вычислений в распределенной среде: распределенные приложения, коммуникационные системы, математические модели и оптимизация: Сборник трудов ИСА РАН / Под ред. А.П. Афанасьева – Т.25 - М.: КомКнига, 2006, с.99-128.
4. Гринберг Я.Р., Курочкин И.И. Исследование результатов математического моделирования последовательного заполнения сетей с кластерной топологией / Проблемы вычислений в распределенной среде: Труды ИСА РАН / Под ред. С.В. Емельянова, А.П. Афанасьева – Т.46 - М.: КРАСАНД, 2009, с.198-232.
5. Т. Ху. Целочисленное программирование и потоки в сетях / перевод с англ. П. Л. Бузыцкого, Е. В. Левнера, Б. Г. Литвака; под ред. А. А. Фридмана - М.: Мир, 1974 - 520 с. - Перевод изд.: Мир, 1974.
6. Naveen Garg, Vijay V. Vazirani, Mihalis Yannakakis. Approximate max-flow min-(multi) cut theorems and their applications / Proceedings ICALP, 1994.