Построение коммуникационных сетей: о применении алгоритма Краскала в задачах больших размерностей

Б. Ф. Мельников, Ю. Ю. Терентьева

Аннотация—**В статье рассматриваются вопросы проектирования топологии сверхбольших коммуникационных сетей, то есть сетей, содержащих несколько тысяч вершин. В большинстве рассматриваемых задач мы считаем, что координаты вершин неориентированного графа (узлы сети связи) каким-то способом заданы заранее, и для этого множества вершин должно быть построено множество рёбер. Как правило, целью является разработка топологии сети с минимально возможным значением суммы весов рёбер; однако в некоторых случаях (в некоторых постановках задач) мы применяем эвристические алгоритмы, при этом целью является описание алгоритмов, работающих за приемлемое время и дающих результаты, близкие к оптимальным. Кроме того, в рассматриваемых нами практических задачах (в наших предыдущих статьях были сформулированы две из них, но ими область применения не исчерпывается) подобный критерий минимальности часто не является единственной целевой функцией.**

Однако, несмотря на кажущуюся простоту алгоритма Краскала, применение которого возможно для решения большинства рассматриваемых нами задач, в действительности всё не так просто, и мы обычно не можем использовать простую версию этого алгоритма. Сложности его применения возникают в связи с представлением данных, например, в связи с невозможностью (или нежелательностью) использовать матрицу смежности. При этом в нашей ситуации, когда число рассматриваемых вершин составляет примерно от 5000 до 10000, работа простого варианта алгоритма (использующего обычное представление данных в виде матрицы смежности) занимает около получаса – что, конечно же, приемлемо для одноразового решения рассматриваемой задачи, но недопустимо в тех случаях, когда такие решения строятся многократно (в частности, итеративно).

Некоторые временны́е улучшения практической работы алгоритма получаются в результате различных вариантов использования сложных структур данных. Например, мы можем каким-то образом хранить некоторое количество неиспользуемых рёбер небольшой длины, а при необходимости сортировать эти ребра, добавлять к ним новые и т. д. Однако такой подход не является «панацеей», поскольку в худшем случае оценки сложности (и времени работы алгоритмов) совпадают. Всё это формулирует необходимость рассмотрения и реализации эвристических алгоритмов – вместо точных, «переборных».

Предмет данной статьи можно сформулировать следующим образом. Мы переходим от точных алгоритмов (в частности, от алгоритма Краскала) к некоторым эвристическим алгоритмам. Более того, для исходной задачи, которую мы рассматриваем, мы вообще не можем реализовать реально работающие программы без некоторых эвристик. При этом мы описываем два конкретных варианта простой реализации алгоритма Краскала для задач большой размерности – «1-й и 2-й алгоритмы с обычной реализацией». Мы также сформулировали и реализовали два варианта эвристик («3-й и 4-й алгоритмы»). На наш взгляд, один из этих алгоритмов оказался вполне приемлемым; это подтверждается приведёнными в статье практическими результатами вычислительных экспериментов. Также очень важно, что эти две эвристики будут полезны не только для такой задачи, но и для задач существенно более сложных, для которых мы применяем совершенно иные алгоритмы; постановки некоторых из них были приведены в наших предыдущих публикациях.

Ключевые слова—**коммуникационные сети, большая размерность, алгоритм Краскала, эвристический алгоритм, anytime-алгоритм.**

I. ВВЕДЕНИЕ

Настоящая статья посвящена разработке топологии сверхбольших сетей связи (т. е. сетей, содержащих по несколько тысяч вершин). При этом, как правило, координаты вершин всей сети (неориентированного графа) так или иначе предопределены (например, как некоторые точки единичного квадрата, определяемые парами координат), а набор рёбер должен быть построен. Основным моментом рассматриваемых нами вариантов построения топологии сети является минимизация суммарной длины рёбер (суммы их весов); однако отметим заранее, что в рассматриваемых нами практических задачах этот критерий минимальности часто не является единственной целевой функцией.

В наших предыдущих работах ([1], [2] и др.) уже были сформулированы две рассматривавшиеся нами практические задачи. Однако, вопреки кажущейся естественной последовательности публикаций, в настоящей работе подробно рассматривается *не* «одна из последующих задач» (в предыдущих публикациях мы кратко описали не менее 8 постановок, для которых мы впоследствии предлагаем реализовать соответствующие компьютерные программы и привести значительно более подробные описания в статьях), а «предыдущая». Эта предыдущая задача является реализацией алгоритма Краскала [4], [5], [6] и, как следует из названия статьи, рассматривается для неориентированных графов очень больших размерностей.

Однако «не всё так просто», и «сложности программной реализации» рассматриваемых алгоритмов связаны именно с оценкой сложности алгоритма (вычислительной сложностью). Как и для большинства алгоритмов теории графов, невозможно *однозначно* ответить на вопрос о сложности этого алгоритма – поскольку ответ зависит от

Борис Феликсович Мельников, Российский государственный социальный университет (bf-melnikov@yandex.ru).

Юлия Юрьевна Терентьева, Центр информационных технологий и систем органов исполнительной власти (terjul@mail.ru).

Статья получена 17 октября 2020 г.

представления данных (т. е. от используемых структур данных). В нашей ситуации необходимо использовать все потенциальные рёбра, число которых (в случае графа, содержащего N вершин) равно N^2. В связи с этим полный перебор всех рёбер (просмотр, brute-force method) занимает те же N^2 шагов, а таких просмотров необходимо N. Таким образом, в нашей ситуации (когда число рассматриваемых вершин составляет примерно от 5 000 до 10 000) работа простой версии алгоритма занимает около получаса – что, конечно, приемлемо для однократного решения рассматриваемой задачи, но неприемлемо в том случае, когда подобные решения строятся многократно (в частности, итеративно).

Некоторые временны́е улучшения практической работы алгоритма дают различные варианты использования сложных структур данных (и подобные вопросы, среди прочих, также рассматриваются в настоящей работе). Например, можно как-то хранить некоторое количество пока неиспользовавшихся рёбер малой длины – и при необходимости сортировать эти рёбра, добавлять к ним новые и т. п. Однако «панацеей» такой подход не является – поскольку в худшем случае оценки сложности (и время работы алгоритмов) получаются такими же. Всё это приводит к необходимости рассмотрения и реализации эвристических алгоритмов – вместо точных, переборных, т. е. использующих вышеупомянутый brute-force method.

Предмет настоящей статьи можно сформулировать следующим образом. Как мы уже отметили, в реальных условиях для наших задач самая простая реализация алгоритма Краскала не очень приемлема[1]. Кроме того, мы предполагаем, что разрабатываемые нами алгоритмы будут выполняться не 1 раз, а многократно: *пользователь* работает с такой задачей именно для того, чтобы иметь возможность многократно изменять входные данные. Всё это – ещё один вариант объяснения того, что от точных алгоритмов (в частности, от алгоритма Краскала) мы *переходим к эвристикам*. Более того, для рассматриваемой нами «нулевой» (стартовой) задачи (а как мы уже отмечали, нами описано не менее 8 связанных с ней задач) мы без эвристических алгоритмов совсем не можем обойтись. Однако для начала мы описываем два конкретных варианта простой реализации алгоритма Краскала для задач больших размерностей (в наших терминах – 1-й и 2-й алгоритм).

Поэтому для нашей задачи мы пока сформулировали две эвристики (в наших терминах – 3-й и 4-й алгоритм). Один из этих алгоритмов оказался, по нашему мнению, вполне приемлемым. Кроме того, очень важно то обстоятельство, что эти две эвристики будут полезны не только для подобной задачи, но и для задач существенно более сложных, в других рассматриваемых нами алгоритмах. О двух из них, т.н. 1-й и 2-й задачах (надо эти номера не путать с номерами алгоритмов), мы уже писали в [1], [2], [3].

Настоящая статья имеет следующую структуру. Раздел II содержит не столько описание алгоритма Краскала (мы считаем, что это описание широко известно), сколько скорее мысли авторов *о возможности применения вариантов этого алгоритма* в рассматриваемых нами задачах. В разделе III мы рассматриваем два варианта обычной реализации алгоритма Краскала для данных большой размерности; также в нём приводятся некоторые соответствующие результаты вычислительных экспериментов.

В разделах IV и V мы рассматриваем две эвристики, дополняющих алгоритм Краскала. При этом, вопреки нашим ожиданиям, только одна из них дает приемлемые результаты. Для всех упомянутых разделов – III, IV и V – мы приводим практические результаты вычислительных экспериментов.

В заключении (раздел VI) мы формулируем несколько задач, являющихся развитием рассмотренных в настоящей статье, а также ранее в [1], [2], [3]. Мы предполагаем описать подходы к решению этих задач в наших следующих публикациях.

II. ПРЕДВАРИТЕЛЬНЫЕ СВЕ́ДЕНИЯ

Этот раздел содержит не столько описание алгоритма Краскала (мы считаем, что это описание широко известно), сколько взгляды авторов статьи о возможности применения конкретных вариантов этого алгоритма в рассматриваемых нами задачах.

Хорошо известно, что алгоритм Краскала является эффективным алгоритмом построения минимального остовного дерева взвешенного связного неориентированного графа. Он также используется для некоторых других связанных задач. Некоторые авторы считают, что этот алгоритм совпадает с т. н. алгоритмом Борувки – Соллина [5], предложенным ещё в 1926 году.

В стандартной версии алгоритма Краскала необходимо для заданного взвешенного ориентированного графа (веса приписаны к каждому его ребру) построить остовное дерево, имеющее минимально возможную стоимость[2]. Сам алгоритм предельно прост: мы последовательно включаем в строимое остовное дерево то из оставшихся рёбер, которое имеет минимальную стоимость; при этом, как обычно в алгоритмах построения некоторого дерева, мы дополнительно следим за тем, чтобы из выбранных рёбер не образовывались циклы. Легко доказать, что в результате реализации описанного таким образом алгоритма получается искомое дерево. Заметим также, что мы не будем различать между собственно алгоритмом Краскала и алгоритм Ярника – Прима – Дейкстры (см. [7], либо также [5]), поскольку в наших конструкциях порядок выбора рёбер значения не имеет.

Однако, как мы уже отмечали во введении, «все не так просто», и «сложности реализации» рассматриваемых нами алгоритмов связаны именно с оценкой сложности алгоритма; а ответ на вопрос о сложности зависит *от представления данных*. Покольку мы не различаем собственно алгоритм Краскала и алгоритм Прима, то любой вариант работы с матрицей смежности приводит к формальной оценке сложности N^3. И поэтому, как также было отмечено во введении, мы используем эвристические алгоритмы.

[1] Реально сложность алгоритма получается получается кубической, и при этом сами веса рёбер графа мы хранить в памяти компьютера не можем, поскольку их для 5 000 вершин будет более 12 миллионов. А если и можем хранить – то *многократно* (например, итеративно) выполнять алгоритмы сортировки для 12 миллионов весов рёбер вряд ли возможно.

[2] Или «минимального веса». В литературе используются обе эти формулировки.

Согласно одной из возможных классификаций, мы среди эвристических алгоритмов выделяем так называемые anytime-алгоритмы. Это – специальный класс алгоритмов реального времени, которые в каждый определённый момент работы хранят лучшее (на данный момент) решение; при этом пользователь (или какая-то управляющая программа) может просматривать эти псевдооптимальные решения в режиме реального времени, и последовательность таких решений в пределе обычно даёт оптимальное решение. Отметим, что такие алгоритмы на практике нужны, например, в связи с тем, что мы запускаем программу для нашего алгорима много раз, и окончательное решение принимает человек, эксперт. Всё это хорошо согласуется с одним из определений экспертной системы – как «предикативной системы, включающей в себя знания об определённой слабо структурированной и трудно формализуемой узкой предметной области и способной предлагать и объяснять пользователю разумные решения» ([8]); в нашем случае подобным псевдооптимальным решением является некоторое конкретное остовное дерево «небольшого» веса, и человек-эксперт принимает решение, является ли это дерево приемлемым.

Построение подобных алгоритмов является одной из задач, которые могут быть объединены общей тематикой с примерным названием «Обучение нечётких систем». Авторы, среди прочего, надеются, что материал данной статьи формулирует *метод* построения anytime-алгоритмов для целого класса задач – а именно, *оптимизационных задач теории графов*. Также стоит отметить, что основная цель статьи – *практические* вопросы построения алгоритмов, а не создание соответствующей теории.

В наших задачах наиболее важны следующие два обстоятельства. Во-первых, подобные эвристические алгоритмы обычно вписываются в рамки общего мультиэвристического подхода к задачам дискретной оптимизации ([9], [10] и др.) и поэтому могут быть решены примерно такими же методами. Во-вторых, конечно же, сложность наших эвристических алгоритмов существенно меньше; в нашем случае она соответствует N^2 (или немногим превышает эту оценку), что вполне приемлемо; более того, при представлении графа в виде матрицы смежности нереально ожидать лучших оценок. Заранее отметим, что такой приемлемостью для рассматриваемых нами размерностей мы считаем время решения порядка нескольких секунд – а не нескольких десятков минут.

III. ДВА АЛГОРИТМА ДЛЯ ОБЫЧНОЙ РЕАЛИЗАЦИИ

В этом разделе мы рассматриваем два варианта *обычной* реализации алгоритма Краскала для данных большой размерности; также мы приводим некоторые соответствующие результаты вычислительных экспериментов.

Как мы уже отмечали, мы считаем, что алгоритм Краскала широко известен; поэтому мы не будем описывать простейший вариант его реализации[3]. Точки случайно (с помощью равномерного распределения) бросаются в единичный квадрат, а выбор очередного добавляемого ребра происходит после перебора всех рёбер, ещё не включённых в граф. Конечно, мы при этом следим за тем, чтобы не образовывался цикл; для этого мы хранили номера текущих компонент связности создаваемого графа.

Для исполнения (при проверке как этого алгоритма, так и всех последующих) использовался компьютер с тактовой частотой около 2.4 ГГц. При проверке мы старались, чтобы другие программы не выполнялись; понятно, что мы не можем гарантировать очень большую точность полученных значений времени вычисления, – но, по-видимому, такая очень большая точность и не нужна[4].

В качестве результатов работы мы приводим только время, усреднённое за несколько (10–15) запусков программы; точность – 3–4 значащих десятичных цифры.

Таблица I
Простейшая реализация алгоритма

Размерность	Усреднённое время выполнения (сек)
512	1.55
1024	12.4
1536	42.9
2048	101.3
3072	358.5
4096	1109

Из приведённых временны́х результатов понятно, что многократно использовать такой алгоритм для интересующих нас размерностей 5000–10000 нецелесообразно.

Модификация этого алгоритма очевидна[5]. В ней мы для каждой вершины запоминали текущий вариант вершины, ближайшей к ней (и при этом ещё не включённой в формируемый граф), а также соответствующее минимальное расстояние. При включении в граф некоторого ребра мы *сохраняли промежуточную информацию* – чтобы не осуществлять полный перебор всех возможных рёбер графа

Результаты вычислений приведены в таблице II, смысл столбцов совпадает тот же, что и в предыдущей таблице I.

Таблица II
Простейшая реализация с запоминанием вре́менных результатов

Размерность	Усреднённое время выполнения (сек)
512	0.277
1024	1.30
1536	3.75
2048	8.23
3072	28.7
4096	54.7
6144	140

Из результатов видно, что пропорциональности не наблюдается, а есть ожидаемое улучшение, связанное с *применением результатов, хранящихся в памяти*; более того, можно сказать, что время работы алгоритма не намного превышает «квадратичное». Однако, во-первых, несложные оценки в худшем дают то же самое, что и без применения описанного улучшения; во-вторых время

[3] При достаточных объёмах памяти сложность реализации этого варианта сравнима со студенческой лабораторной работой.

[4] Специально запускать программу без работающей операционной системы в нашей ситуации, по-видимому, ни к чему.

[5] Хотя реализация этой модификации несколько сложнее, она также по сложности не намного превышает обычную студенческую лабораторную работу.

работы всё равно велико[6]. Поэтому, как мы уже отмечали во введении, для нашей задачи необходимо применение *эвристических* алгоритмов.

IV. ПЕРВАЯ ЭВРИСТИКА (И ТРЕТИЙ АЛГОРИТМ)

В этом разделе мы рассматриваем первую из эвристик, дополняющих алгоритм Краскала. Эта эвристика заключается в следующем. Мы заранее формируем информацию не об одной вершине, ближайшей к рассматриваемой, а *о нескольких таких вершинах* (в программе для общего числа вершин от 512 до 8192 применялись количества ближайших вершин от 10 до 40). Более того, мы для уменьшения общего времени работы алгоритма пытались соединить некоторую рассматриваемую вершину с ближайшей возможной (с одной из тех, которые ещё не образуют цикл) – выбирая эту новую вершину из числа запомненных заранее (их число, как мы уже отмечали, от 10 до 40). В случае, если после работы алгоритма оставалась более одной компоненты связности, мы применяли окончание алгоритма, которое фактически совпадает с самой простой версией алгоритма Краскала (с первым алгоритмом). Ключевые моменты программы на Си++ приведены в следующем фрагменте программы:

```
void AllPoints::Run3begin(Graph& g) {
  for (;;) {
    bool b = false; // now, we find nothing
    for (int i=0; i<=KOL-1; i++) {
      for (int k=0; k<=NEAR-1; k++) {
        int J = points[i].near.Near[k];
        if (gruppa[i]==gruppa[J]) continue;
        b = true;
        Step1(i,J,g);
        break;
      }
    }
    if (!b) return;
  }
}
```

К тексту этого метода необходимы следующие комментарии.

- `AllPoints` – класс для всех заданных точек (вершин формируемого графа).
- `points` объект этого класса.
- `gruppa` текущий номер компоненты связности, в которую входит рассматриваемая вершина.
- Генерируемый граф `Graph& g` передаётся этой функции в качестве параметра; в этом же параметре возвращается ответ, т. е. сгенерированный граф.
- Значение переменной `b` мы устанавливаем `true` в том случае, когда находим возможность провести новое ребро (т. е. добавить ребро в генерируемый граф).
- `KOL` – общее число вершин графа; для некоторого упрощения мы считаем его константой.
- `NEAR` – число вершин, которые рассматриваются как ближайшие к рассматриваемой; их обычно (в наших разных реализациях) от 10 до 40; мы также счиаем это число константой.
- `Near` – массив номеров таких вершин.

[6] Напомним, что нам необходим алгоритм и соответствующая программа, которые будут многократно запускаться – либо пользователем, либо другой программой.

- Метод `Step1()` связывает две ужи имеющиеся компоненты связности. (Применено такое название метода, поскольку, как мы уже отмечали, он в точностью совпадает с рассмотренным нами ранее «первым алгоритмом».)

Однако, несмотря на первоначальные ожидания, эта эвристика не показала приемлемых результатов. При неплохом времени работы (что, конечно, ожидалось) соответствующая компьютерная программа дала плохие результаты сравнения целевой функции построенного графа с оптимальным вариантом. В приведённой далее таблице III мы, кроме предыдущих столбцов, указали ещё и среднее отношение (среднее ухудшение найденного ответа); в таблице мы назвали такое ухудшение “badness” (будем далее писать без кавычек). К этому названию вряд ли стоит пытаться подобрать соответствующий русский аналог; кроме того, отметим, что оно отражает связь с уже применявшимися нами одноимёнными терминами в областях, достаточно далёких от тематики настоящей статьи, см. [11], [12] и др.

Таблица III
Использование первой эвристики

Размерность	Усреднённое время выполнения (сек)	Badness
512	0.076	1.16
1024	0.269	1.13
1536	0.566	1.11
2048	0.979	1.12
3072	2.29	1.09
4096	4.03	1.11
6144	6.17	1.10

Итак, на интересующих нас размерностях мы видим уменьшение времени работы примерно в 20–25 раз, при том, что «ухудшение качества результата» (badness) составляет приемлемые значения $1.10-1.12$. Более того, с нашей точки зрения *стабильность* этих значений неявно свидетельствует о том, что описанная здесь эвристика весьма удачна – и, по-видимому, будет работать аналогично и на значительно бóльших размерностях, которые в настоящий момент не представляют интереса.

V. ВТОРАЯ ЭВРИСТИКА (И ЧЕТВЁРТЫЙ АЛГОРИТМ)

В этом разделе мы рассматриваем вторую эвристику, также дополняющую алгоритм Краскала. Заранее отметим, что эта эвристика, вопреки нашим ожиданиям, приемлемых результатов не дала – это будет видно из приведённых ниже результатов вычислительных экспериментов.

Эвристика, используемая в этом алгоритме, заключается в следующем. Мы делаем аналог одного из алгоритмов, применяемого для сортировки массива – а именно, метода последовательных обменов. При этом, в отличие от обычного алгоритма, мы фиксируем некоторый определённый шаг (который обычно равен 1, но иногда, если некоторая случайная величина не превышают 0.15, он может быть увеличен до 8, подробнее см. ниже в тексте программы); для этого шага (пусть M) мы проверяем, не станет ли сумма расстояний между двумя парами

точек[7]. (с номерами i и $i+M$, а также $i+2M$ и $i+3M$, для каждого возможного i) меньше, если мы поменяем местами 2-й и 3-й элементы этой четвёрки; и если станет – то действительно меняем их. После некоторого количества выполнений таких шагов (обменов) (в программах мы использовали число шагов от 5 000 до 30 000 для разных вариантов размерности массива точек) мы получаем «псевдоотсортированный» (по этому признаку) массив точек.

Заметим, что мы рассматриваем такую «близость» для всего рассматриваемого (заданного) множества точек. «Мотивация» такой эвристики состоит в том, что точки, имеющие соседние (или не очень далеко отстоящие) индексы в массиве, легче проверять на желательность включения в формируемое остовное дерево – именно на этой, «второй» эвристике основан применяемый нами алгоритм построения остовного дерева.

Также отметим, что если добавить «близость» последней и первой точки, то оптимальное размещение всех точек представляет собой оптимальное же решение задачи коммивояжёра (ЗКВ). Таким образом, оптимальным вариантом работы этой эвристики (после которого мы бы и применяли кратко описанную здесь модификацию алгоритма Краскала) было бы размещение точек согласно решению соответствующего частного случая ЗКВ – что, конечно же, в связи с труднорешаемостью этой проблемы неприемлемо. Именно поэтому мы предлагаем вариант эвристики, в которой для получения псевдооптимального решения частного случая ЗКВ необходимо относительно небольшое время, обычно порядка N^2, – что сравнимо со временем, необходимым для обычно рассматривамых нами эвристических алгоритмов, см. уже процитированные работы [11], [12], а также [13], [14].

Применяем всё сказанное выше в следующем методе Си++:

```
bool AllPoints::StepObmeny() {
  bool bRet = false;
  for (int i=0; i<=KOL-4; i++) {
    int kk = 1;
    if (DoubleRnd()<0.15) {
      int ll = (rand() % 7) + 2;
      if (i+3*ll<=KOL-1) kk = ll;
    }
    if (Delta(i,i+1*kk)+Delta(i+2*kk,i+3*kk)
    <= Delta(i,i+2*kk)+Delta(i+1*kk,i+3*kk))
      continue;
    Obmen(i+1*kk,i+2*kk);
    bRet = true;
  }
  return bRet;
}
```

Приведём небольшие комментарии к тексту метода.

- Функция `StepObmeny()` возвращает `true`, если мы сделали хотя бы один обмен
- `ll` – возможная величина шага (использование этого термина мы обсуждали выше); `kk` – реально сгенерированный шаг
- Вспомогательная функция `Obmen()` меняет два элемента, соответствующие индексам-параметрам
- Количество вызовов метода `StepObmeny()` также обсуждалось выше

После такого предварительного переформирования массива мы применяем алгоритм, который несколько похож на приведённый в предыдущем разделе. Однако мы не вычисляем (и, следовательно, не храним) ближайшие несколько точек, но пробуем соединить уже сформированные компоненты связности – прежде всего для двух точек, стоящих рядом в массиве. Все сказанное описывается следующей функцией:

```
void AllPoints::Run4begin(Graph& g) {
  for (int i=0; i<=KOL-NEAR-1; i++) {
    double rMin = 99;
    int J;
    for (int j=i+1; j<i+NEAR; j++) {
    if (gruppa[i]==gruppa[j]) continue;
    double r = Delta(i,j);
    if (r>rMin) continue;
    rMin = r;
    J = j;
    }
    if (rMin>2) continue;
    Step1(i,i+1,g);
  }
}
```

По-видимому, все используемые переменные и вспомогательные функции уже были прокомментированы ранее. При этом реализация «аналога алгоритма последовательных обмена» не занимает много времени и, более того, она не зависит от размерности задачи. Итак, как мы уже сказали, эта эвристика не показала приемлемых результатов – несмотря на наши первоначальные ожидания. При приемлемом времени выполнения (что естественно) соответствующая программа давала плохие результаты сравнения целевой функции построенного графа с оптимальным вариантом. Структура приведённой далее таблицы IV аналогична предыдущей таблице.

Таблица IV
Использование второй эвристики

Размерность	Усреднённое время выполнения (сек)	Badness
512	0.277	>4
1024	0.312	>4
1536	0.517	>4
2048	0.925	>4
3072	1.18	>4
4096	1.59	>4
6144	2.02	>4

Здесь было бы приемлемым значение, не превышающее 1.5, – но, как мы уже отмечали, таких значений не получается; однако и «слишком неудачным» его назвать нельзя. Поэтому повторим, что несмотря на достаточно большое значение величины badness, мы предполагаем в будущем описывать улучшения этого алгоритма. Приведённая мотивация даёт надежду, что при некоторой модификации этой второй эвристики получится алгоритм, который по совокупности критериев (время, badness) превзойдёт алгоритм, описаннный в предыдущем разделе (для которого получить дальнейшее улучшение, повидимому, не удастся). Именно поэтому мы и включили настоящий раздел в статью.

[7] С номерами *i* и *i + M*, а также *i + 2M* и *i + 3M*, для каждого возможного *i*.

VI. ЗАКЛЮЧЕНИЕ

Итак, мы считаем, что в результате описанного в настоящей статье мы получили приемлемый алгоритм, удовлетворяющий как временны́м, так и целевым критериям. Далее рассмотрим замечания к тексту статьи и сформулируем некоторые задачи, которые можно считать развитием рассматриваемой нами темы; мы предполагаем описать подходы к решению этих задач в последующих публикациях.

- Начнём с текста из [9] [8] об общности всех решаемых нами задач дискретной оптимизации; в том числе этот текст можно отнести и к проблеме, рассматриваемой в данной работе. «Каждая из этих задач может быть легко решена для малых размерностей, но при переходе к большим размерностям строго решить их в реальном времени невозможно – даже с помощью метода ветвей и границ и других подобных эвристик. Например, задача коммивояжёра (к которой можно *свести* практически все рассматриваемые нами задачи) даже в упрощённой постановке относится к классу NP-полных задач. Методы, которые мы применяем для решения этих задач, основаны на сочетании эвристик, взятых из нескольких различных областей теории искусственного интеллекта:
  - во-первых, используется незавершённый (truncated) метод ветвей и границ;
  - во-вторых, для выбора следующего шага этого метода при наличии нескольких эвристик используются динамические функции риска;
  - в-третьих, одновременно с функциями риска для выбора коэффициентов усреднения для различных эвристик используются и генетические алгоритмы;
  - в-четвёртых, наоборот, упрощённое самообучение с помощью тех же генетических алгоритмов используется для запуска незавершённого метода ветвей и границ.

  Таким образом, можно сказать, что указанные *комбинации* эвристик представляют собой особый подход к построению алгоритмов дискретной оптимизации для соответствующих задач». (Про динамические функции риска см. [15], [16] и др.) Итак, одной из задач ближайшей реализации явля- ется применение в рассматриваемой нами области метода ветвей и границ.
- Несколько задач для дальнейшего исследования связаны с функцией `StepObmeny()`, которую мы приводили выше. Например, среди них есть такая: всегда ли возможно за определённое число шагов [9] получить массив, являющийся по описанным критерием минимальным, – или же мы можем попасть либо в локальный максимум, либо в бесконечный цикл? И ещё одна связанная *практическая* задача (т.е. задача на описание эвристического алгоритма): как улучшить соответствующий алгоритм путём улучшения одной лишь функции `StepObmeny()`?
- В самой первой публикации об алгоритме Краскала [4] использовалось название «задача коммивояжёра»; однако в те годы соответствующая терминология еще не была устоявшейся. Действительно, по мнению авторов настоящей работы, многие вспомогательные алгоритмы, используемые при построении связного графа, аналогичны вспомогательным алгоритмам, используемым при эвристическом решении ЗКВ. За прошедшее время, по-видимому, гораздо большее количество публикаций было посвящено задаче коммивояжёра (чем задачам, в которых может быть применён алгоритм Краскала), и в связи с этим возникает следующий вопрос: какой из ранее описанных методов решения ЗКВ может быть применён в рассматриваемой нами предметной области?
- В нескольких наших предыдущих публикациях ([13], [14] и др.) мы рассматривали подходы к решению так называемой псевдогеометрической версии ЗКВ. Соответственно возникает вопрос о возможном применении алгоритмов из данной статьи к структурам данных, рассматриваемым в указанных публикациях[10].
- Ещё сто́ит отметить, что полиномиальные эвристические алгоритмы для ЗКВ[11] могут быть применены не только в рассматриваемой здесь задаче построения коммуникационных сетей для больших размерностей путём применения алгоритма Краскала – но и в следующих задачах:
  - в уже упоминавшейся нами задаче коммивояжёра, прежде всего – для её псевдогеометрического варианта, также уже упоминавшегося выше;
  - в рассматриваемом нами подходе к анализу близости ДНК-цепочек, см. [17], [18] и др.;
  - для оценки качества алгоритмов подсчёта парной корреляции (отметим, что в [19] описаны два таких алгоритма, но возможны и некоторые другие, также «вполне естественные»); эти алгоритмы, в свою очередь, можно применить как вспомогательные – например, для анализа оценок экспертов, см. [20], [21] [12], а также в подходе к анализу близости ДНК-цепочек, см. предыдущий подпункт.
  - и мн. др.
- Вопросы, рассматриваемые в настоящей статье, мы предполагаем применить и к другим задачам, связанным с разработкой сетей связи – в частности с получением эвристических оценок их надёжности, см., например, [22].

Мы вернёмся к этим вопросам в наших следующих публикациях. Кроме того, мы собираемся рассмотреть

[8] Точнее, с его обратного перевода на русский язык.

[9] Достаточно большое число, по-видимому, факториальной сложности.

[10] Иными словами – о применении рассмотренных здесь алгоритмов к псевдогеометрическому варианту задачи коммивояжера. (Мы здесь используем понятие «вариант проблемы» – не совпадающее с понятием «частный случай проблемы».)

[11] Ещё точнее – эвристические алгоритмы, сложность которых не превышает N^2.

[12] В этих статьях мы, среди прочего, начали критику так называемого «метода анализа иерархий». Критика этого метода приводится и в нескольких статьях В. В. Подиновского с соавторами (мы не приводим конкретные ссылки: они мало связаны с предметом настоящей статьи), но отметим, что наша критика совсем иная. Мы собираемся вернуться в будущих публикациях и к этой тематике.

вопрос о значительном увеличении размерности рассматриваемых проблем: от примерно 5 000 – 10 000 (как в настоящей статье) до примерно 1 000 000.

Список литературы

[1] Melnikov B., Meshchanin V, Terentyeva Y. *Implementation of optimality criteria in the design of communication networks* // Journal of Physics: Conference Series, International Scientific Conference ICMSIT–2020: Metrological Support of Innovative Technological, ICMSIT-2020 (paper 3095). http://conf.domnit.ru/en/conferences/icmsit-2020-en/.

[2] Булынин А., Мельников Б., Мещанин В, Терентьева Ю. *Оптимизационные задачи, возникающие при проектировании сетей связи высокой размерности, и некоторые эвристические методы их решения* // Информатизация и связь. – 2020. – № 1. – P. 34–40. https://elibrary.ru/item.asp?id=42839357.

[3] Булынин А., Мельников Б., Мещанин В, Терентьева Ю. *Алгоритмы проектирования сетей связи с применением жадных эвристик разных типов* // Информационные технологии и нанотехнологии (ИТНТ-2020). Сборник трудов по материалам VI Международной конференции и молодежной школы. – 2020. – P. 856–860. https://www.elibrary.ru/item.asp?id=43576630.

[4] Kruskal J. *On the shortest spanning subtree of a graph and the traveling salesman problem* // Proceedings of the American Mathematical Society. – 1956. – Vol. 7, No. 1. – P. 48–50. doi: 10.1090/S0002-9939-1956-0078686-7.

[5] Cormen T., Leiserson C., Rivest R, Stein C. *Introduction to Algorithms, Second Edition* // Massachusetts: MIT Press and McGraw-Hill, 2001. ISBN 0-262-03293-7.

[6] Goodrich M., Tamassia R. *Data Structures and Algorithms in Java, Fourth Edition* // NJ: John Wiley & Sons, Inc., 2006. ISBN 0-471-73884-0.

[7] Prim R. *Shortest connection networks and some generalizations* // Bell System Technical Journal. – 1957. – Vol. 36, No. 6. – P. 1389–1401. doi: 10.1002/j.1538-7305.1957.tb01515.

[8] S. Russell S., Norvig P. *Artificial Intelligence: A Modern Approach* // NJ: Simon & Schuster, 1995. ISBN 978-0-13-103805-9.

[9] Melnikov B. *Discrete optimization problems – some new heuristic approaches* // Proceedings – Eighth International Conference on High-Performance Computing in Asia-Pacific Region, HPC Asia 2005, Beijing, 2005. – P. 73–80. doi: 10.1109/HPCASIA.2005.34.

[10] Melnikov B., Radionov A, Moseev A, Melnikova E. *Some specific heuristics for situation clustering problems* // Proceedings – 1st International Conference on Software and Data Technologies, ICSOFT 2006, Setubal, 2006. – P. 272–279. doi: 10.5220/0001317002720279 .

[11] Макаркин С., Мельников Б., Тренина М. *Подход к решению псевдогеометрической версии задачи коммивояжера* // Известия высших учебных заведений. Поволжский регион. Физико-математические науки. – 2015. – № 2 (34). – С. 135–147. https://elibrary.ru/item.asp?id=24254294.

[12] Мельников Б., Пивнева С., Трифонов М. *Мультиэвристический подход к сравнению качества определяемых метрик на множестве последовательностей ДНК* // Современные информационные технологии и ИТ-образование. – 2017. – Т. 13, № 2. – С. 89–96. https://elibrary.ru/item.asp?id=30258646.

[13] Мельников Б., Романов Н. *Ещё раз об эвристиках для задачи коммивояжёра* // Теоретические проблемы информатики и ее приложений. – 2001. – Т. 4. – С. 81–88. https://www.elibrary.ru/title_about.asp?id=38720.

[14] Макаркин С., Мельников Б. *Геометрические методы решения псевдогеометрической версии задачи коммивояжера* // Стохастическая оптимизация в информатике. – 2013.–Т. 9, № 2.– С. 54–72. https://www.elibrary.ru/item.asp?id=20960443.

[15] Мельников Б., Радионов А. *О выборе стратегии в недетерминированных антагонистических играх* // Программирование (РАН). – 1998. – № 5. – С. 55–63. https://elibrary.ru/title_about_new.asp?id=7966.

[16] Melnikov B., Melnikova E., Pivneva S., Churikova N., Dudnikov V, Prus M. *Multi-heuristic and game approaches in search problems of the graph theory* // Сборник трудов ИТНТ-2018. Самарский национальный исследовательский университет имени академика С.П.Королева. – 2018. – С. 2884–2892. https://elibrary.ru/item.asp?id=34895071, http://repo.ssau.ru/bitstream/Informacionnye-tehnologii-i-nanotehnologii/Multiheuristic-and-game-approaches-in-search-problems-of-the-graph-theory-69142/1/paper_389.pdf.

[17] Мельников Б., Тренина М. *Об одной задаче восстановления матриц расстояний между цепочками ДНК* // International Journal of Open Information Technologies. – 2018. – Т. 6, № 6. – С. 1–13. http://injoit.org/index.php/j1/article/view/571/562.

[18] Melnikov B., Trenina M., Melnikova E. *Different Approaches to Solving the Problem of Reconstructing the Distance Matrix Between DNA Chains* // Communications in Computer and Information Science, 1201 CCIS. – 2020. – С. 211–223. https://link.springer.com/chapter/10.1007/978-3-030-46895-8_17.

[19] Лагутин М. Наглядная математическая статистика // М.: БИНОМ. Лаборатория знаний, 2015. ISBN 978-5-9963-2955-7.

[20] Мельников Б., Зубова Т. *Математическое моделирование управления организацией по ценностным ориентирам: алгоритмы комплексной оценки и отбора псевдооптимальных воздействий* // International Journal of Open Information Technologies. – 2018. – Т. 6, № 3. – С. 1–8. http://injoit.org/index.php/j1/article/view/545/518.

[21] Зубова Т., Мельников Б., Солдатов А. *Математическое моделирование управления организацией по ценностным ориентирам: алгоритмы анализа иерархий на основе количественной значимости критериев* // International Journal of Open Information Technologies. – 2019. – Т. 7, № 8. – С. 1–8. http://injoit.org/index.php/j1/article/view/738/746.

[22] Терентьева Ю. *Метод получения оценки надежности крупномасштабной сети связи с учетом зависимых путей* // Информатизация и связь. – 2017. – № 1. – С. 122–128. https://www.elibrary.ru/item.asp?id=29076288.

Борис Феликсович МЕЛЬНИКОВ,
профессор Российского государственного социального университета (http://www.rgsu.net/),
профессор Университета МГУ – ППИ в Шэньчжэне (http://szmsubit.ru/),
email: bf-melnikov@yandex.ru,
mathnet.ru: personid=27967,
elibrary.ru: authorid=15715,
scopus.com: authorId=55954040300,
ORCID: orcidID=0000-0002-6765-6800.

Юлия Юрьевна ТЕРЕНТЬЕВА,
ведущий научный сотрудник Центра информационных технологий и систем органов исполнительной власти (https://citis.ru/),
email: terjul@mail.ru,
elibrary.ru: authorid=1577,
scopus.com: authorId=57216810316,
ORCID: orcidID=0000-0002-2418-003X.

Building communication networks: on the application of the Kruskal's algorithm in the problems of large dimensions

Boris Melnikov, Yulia Terentyeva

***Abstract*—The paper deals with the development of the topology of ultra-large communication networks, i.e. networks containing several thousand vertices. In this case, the coordinates of the vertices of the undirected graph are somehow predetermined and a set of edges must be constructed. The main point of the options we are considering for developing the network topology is the minimum of the sum of weights of the edges; however, we note in advance that this criterion of minimality is often not the only objective function in the practical problems we are considering. In our previous papers, two realistically considered tasks were formulated.**

However, everything is not so simple, and we cannot use the direct version of Kruskal's algorithm. The complexity of this algorithm depends on the representation of the data, i.e. the data structures used. In our situation (when the number of considered vertices is approximately 5000 to 10000), the operation of a simple version of the algorithm takes about a half an hour, which, of course, is acceptable for a one-time solution to the problem under consideration, but it is unacceptable in the case when such solutions are constructed repeatedly (in particular, iteratively).

Some temporary improvements to the practical operation of the algorithm provide different options for using complex data structures. For example, we can somehow store a certain number of unused edges of small length, and, if necessary, sort these edges, add new ones to them, etc. However, this approach is not a "panacea", since in the worst case the complexity estimates (and the running time of the algorithms) are the same. All this formulates the need to consider and implement heuristic algorithms, instead of exact, exhaustive ones.

The subject of this paper can be formulated as follows. We are moving from exact algorithms (in particular, Kruskal's algorithm) to some heuristics. Moreover, for the starting problem that we are considering, we cannot work without heuristic algorithm at all. However, we describe two specific variants of a simple implementation of Kruskal's algorithm for problems of large dimensions; in our titles, those are "the 1st and the 2nd algorithms of the usual implementation". We have formulated two heuristics ("the 3rd and 4th algorithms"). In our opinion, one of these algorithms turned out to be quite acceptable; we present some practical results of computational experiments. And it is very important that these two heuristics will be useful not only for such a problem, but also for much more complex problems.

***Keywords*—communication networks, large dimensions, Kruskal's algorithm, heuristic, anytime algorithm.**

References

[1] Melnikov B., Meshchanin V, Terentyeva Y. *Implementation of optimality criteria in the design of communication networks* // Journal of Physics: Conference Series, International Scientific Conference ICMSIT–2020: Metrological Support of Innovative Technological, ICMSIT-2020 (paper 3095), http://conf.domnit.ru/en/conferences/icmsit-2020-en/.

[2] Bulynin A., Melnikov B., Meshchanin V, Terentyeva Y. *Optimization problems arising in the design of high-dimensional communication networks, and some heuristic methods for their solution* // Informatization and Communication. – 2020. – No. 1. – P. 34–40, https://elibrary.ru/item.asp?id=42839357 (in Russian).

[3] Bulynin A., Melnikov B., Meshchanin V, Terentyeva Y. *Communication network design algorithms using various types of greedy heuristics* // Information technologies and nanotechnologies (ITNT-2020). Proceedings of the VI International Conference and Youth School. – 2020. – P. 856–860, https://www.elibrary.ru/item.asp?id=43576630 (in Russian).

[4] Kruskal J. *On the shortest spanning subtree of a graph and the traveling salesman problem* // Proceedings of the American Mathematical Society. – 1956. – Vol. 7, No. 1. – P. 48–50, doi: 10.1090/S0002-9939-1956-0078686-7.

[5] Cormen T., Leiserson C., Rivest R, Stein C. *Introduction to Algorithms, Second Edition* // Massachusetts: MIT Press and McGraw-Hill, 2001, ISBN 0-262-03293-7.

[6] Goodrich M., Tamassia R. *Data Structures and Algorithms in Java, Fourth Edition* // NJ: John Wiley & Sons, Inc., 2006, ISBN 0-471-73884-0.

[7] Prim R. *Shortest connection networks and some generalizations* // Bell System Technical Journal. – 1957. – Vol. 36, No. 6. – P. 1389–1401, doi: 10.1002/j.1538-7305.1957.tb01515.

[8] S. Russell S., Norvig P. *Artificial Intelligence: A Modern Approach* // NJ: Simon & Schuster, 1995, ISBN 978-0-13-103805-9.

[9] Melnikov B. *Discrete optimization problems – some new heuristic approaches* // Proceedings – Eighth International Conference on High-Performance Computing in Asia-Pacific Region, HPC Asia 2005, Beijing, 2005. – P. 73–80. doi: 10.1109/HPCASIA.2005.34.

[10] Melnikov B., Radionov A, Moseev A, Melnikova E. *Some specific heuristics for situation clustering problems* // Proceedings – 1st International Conference on Software and Data Technologies, ICSOFT 2006, Setubal, 2006. – P. 272–279. doi: 10.5220/0001317002720279 .

[11] Makarkin S., Melnikov B., Trenina M. *An approach to solving the pseudo-geometric version of the traveling salesman problem* // News of higher educational institutions. Volga region. Physical and mathematical Sciences. – 2015. – No. 2 (34). – P. 135–147. https://elibrary.ru/item.asp?id=24254294 (in Russian).

[12] Melnikov B., Pivneva S., Trifonov M. *Multi-heuristic approach to comparing the quality of defined metrics on a set of DNA sequences* // Modern Information Technologies and IT Education. – 2017. – Vol. 13, No 2. – P. 89–96. https://elibrary.ru/item.asp?id=30258646 (in Russian).

[13] Melnikov B., Romanov N. *Once again on the heuristics for the traveling salesman problem* // Theoretical problems

of computer science and its applications. – 2001. – Vol. 4. – P. 81–88. https://www.elibrary.ru/title_about.asp?id=38720 (in Russian).

[14] Makarkin S., Melnikov B. *Geometric methods for solving the pseudo-geometric version of the traveling salesman problem* // Stochastic Optimization in Computer Science. – 2013. – Vol. 9, No 2. – P. 54–72. https://www.elibrary.ru/item.asp?id=20960443 (in Russian).

[15] Melnikov B., Radionov A. *A choice of strategy in nondeterministic antagonistic games* // Programming and Computer Software. – 1998. – Vol. 24, No. 5. – P. 247–252. https://elibrary.ru/item.asp?id=13280793.

[16] Melnikov B., Melnikova E., Pivneva S., Churikova N., Dudnikov V, Prus M. *Multi-heuristic and game approaches in search problems of the graph theory* // Proceedings of the ITNT-2018. Samara National Research University named after academician S. P. Korolev. – 2018. – P. 2884–2892. https://elibrary.ru/item.asp?id=34895071, http://repo.ssau.ru/bitstream/Informacionnye-tehnologii-i-nanotehnologii/Multiheuristic-and-game-approaches-in-search-problems-of-the-graph-theory-69142/1/paper_389.pdf.

[17] Melnikov B., Trenina M. *On a problem of reconstructing distance matrices between DNA chains* // International Journal of Open Information Technologies. – 2018. – Vol. 6, No. 6. – P. 1–13. http://injoit.org/index.php/j1/article/view/571/562 (in Russian).

[18] Melnikov B., Trenina M., Melnikova E. *Different Approaches to Solving the Problem of Reconstructing the Distance Matrix Between DNA Chains* // Communications in Computer and Information Science, 1201 CCIS. – 2020. – P. 211–223. https://link.springer.com/chapter/10.1007/978-3-030-46895-8_17.

[19] Lagutin M. Visual mathematical statistics // Moscow: BINOM. Knowledge lab, 2015. ISBN 978-5-9963-2955-7 (in Russian).

[20] Melnikov B., Zubova T. *Mathematical modeling of organization management by value guidelines: algorithms for complex estimation and selection of pseudo-optimal actions* // International Journal of Open Information Technologies. – 2018. – Vol. 6, No. 3. – P. 1–8. http://injoit.org/index.php/j1/article/view/545/518 (in Russian).

[21] Zubova T., Melnikov B., Soldatov A. *Mathematical modeling of organization management by value guidelines: algorithms for analytic hierarchy process on the basis of quantitative significance criteria* // International Journal of Open Information Technologies. – 2019. – Vol. 7, No. 8. – P. 1–8. http://injoit.org/index.php/j1/article/view/738/746 (in Russian).

[22] Terentyeva Y. *A method for estimating the reliability of a large-scale communication network based on dependent paths* // Informatization and Communication. – 2017. – No. 1. – P. 122–128. https://www.elibrary.ru/item.asp?id=29076288 (in Russian).

Boris MELNIKOV,
Professor of Russian State Social University
(http://www.rgsu.net/),
Professor of Shenzhen MSU – BIT University, China
(http://szmsubit.ru/),
email: bf-melnikov@yandex.ru,
mathnet.ru: personid=27967,
elibrary.ru: authorid=15715,
scopus.com: authorId=55954040300,
ORCID: orcidID=0000-0002-6765-6800.

Yulia TERENTYEVA,
Leading Researcher of The Center for Information Technologies and Systems for Executive Authorities
(https://citis.ru/),
email: terjul@mail.ru,
elibrary.ru: authorid=1577,
scopus.com: authorId=57216810316,
ORCID: orcidID=0000-0002-2418-003X.