О применении нескольких предикторов в методе ветвей и границ (на примере случайного варианта задачи коммивояжёра)

Б. Ф. Мельников, Ю. Ю. Терентьева

***Аннотация*—Для т. н. случайного варианта задачи коммивояжёра к настоящему времени неизвестно быстрых алгоритмов, в том числе эвристических, дающих оптимальное решение (или близкое к нему) – по крайней мере, для размерности задачи, превышающей 300. Это принципиально отличает случайный вариант от т. н. геометрического варианта этой же задачи, для которого уже не менее 25 лет известны т. н. алгоритмы луковой шелухи (опоясывающие алгоритмы), дающие близкое к оптимальному решение для практически любой размерности. При этом случайный вариант задачи коммивояжёра – как и более общий вариант, псевдогеометрический – часто представляет собой удачную модель для прикладных задач разных классов; это утверждение вряд ли можно отнести к геометрическому варианту, поэтому алгоритмы решения дя случайного варианта продолжают оставаться актуальными.**

Также актуальность рассматриваемого в статье варианта задачи следует из её возможной применимости при формулировке вариантов подсчёта оценки т. н. живучести сети связи – интегрального показателя, являющегося усреднённым показателем оценок живучести всех возможных направлений связи, определяемых парой вершин графа сети связи. Обычно разработчики сети связи стремятся к тому, чтобы все направления имели резервный маршрут; предполагается, что мы должны это проверить – а такую проверку можно произвести с использованием алгоритма поиска оптимального Гамильтонова цикла. Итак, необходимо уметь решать случайный вариант задачи коммивояжёра – и, прежде всего, описывать т. н. anytime-алгоритмы для его решения.

Для случайного варианта задачи (причём как для точных, так и для anytime-алгоритмов) актуален классический подход к её решению, связанный с применением метода ветвей и границ. Мы пользуемся обычным описанием этого метода – но с многочисленными измененениями и добавлениями, описанными нами в предыдущих публикациях, а также в настоящей статье. По-видимому, для построения удачных конкретных вариантов алгоритмов, реализующих этот метод, наиболее важным является вспомогательный алгоритм выбора разделяющего элемента (причём последнее утверждение относится не только к задаче коммивояжёра, но и к любой задаче, решаемой с помощью метода ветвей и границ). Вспомогательная процедура выбора разделяющего элемента называется предиктором.

В настоящей статье авторы предлагают вариант использования для выбора разделяющего элемента интегральной оценки, применяющей два простейших предиктора; при этом подобный вариант легко обобщается на случай большего числа предикторов. В статье кратко описываются полученные результаты, подтверждающие улучшение алгоритма метода ветвей и границ при применении двух рассматриваемых предикторов.

Статья получена 26 ноября 2023 г.

Борис Феликсович Мельников, Университет МГУ – ППИ в Шэньчжэне (bormel@mail.ru).

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

***Ключевые слова*—эвристические алгоритмы, метод ветвей и границ, задача коммивояжёра, вариант проблемы, случайный вариант, предикторы.**

I. ВВЕДЕНИЕ

Сама общая формулировка задача коммивояжёра (ЗКВ), используемая в настоящей статье, согласована с [1], [2]. И, в отличие от нескольких предыдущих публикаций одного из авторов [3], [4], [5], [6], в которых рассматривалась т. н. псевдогеометрическая ЗКВ, – в настоящей работе рассматривается т. н. случайная ЗКВ. При этом ещё точнее было бы говорить о специальном варианте псевдогеометрической ЗКВ – что объясним далее во введении.

В примечании переводчика русского издания монографии [1] первый автор настоящей статьи привёл следующий комментарий – о применении в *хороших* изданиях на русском языке следующей терминологии, согласующейся с русским переводом классической монографии [7].

- «Вариант проблемы» определяется некоторым подмножеством всех возможных входов (аналог – знак включения множеств $\subseteq$). Иногда применяемые в русской литературе альтернативные термины для этого понятия – причём фактически являющиеся кальками неудачных английских названий – а именно «подпроблема» и «подзадача», представляются значительно менее удачными: в литературе на русском языке они обычно употребляются в другом смысле. Более того, этот термин («вариант проблемы», problem instance) в некоторых монографиях определяется строго – но, однако, нередко употребляется и до его строгого определения, см. [1] и др.
- А «частный случай» проблемы –это один конкретный возможный вход (аналог – знак принадлежности $\in$). При этом термин «экземпляр проблемы», по-видимому, тоже возможен (поскольку он не противоречит [7] – но менее удачен.

Добавим к сказанному в том примечании, что вся эта терминология относится к любой задаче дискретной оптимизации – а не только к ЗКВ.

Приведём содержание статьи по разделам. При этом сразу отметим, что существенную часть объёма статьи занимают различные варианты описаний результатов вычислительных экспериментов – но, по мнению авторов, без подобных описаний статья была бы «голословной».

Раздел II озаглавлен следующим образом: «Кратко о генерации исходных данных для рассматриваемых частных случаев задачи коммивояжёра». Конечно, подобные

алгоритмы генерации частных случаев проблемы очень важны для статистического анализа алгоритмов обработки, обычно являющихся основным предметом большинства статей – но мы считаем, что подобным алгоритмам генерации уделяется гораздо меньше внимания, чем они того заслуживают. Некоторые подробности на эту тему авторы настоящей работы привели в [8].

В статье, как следует из её названия, мы для решения ЗКВ методом ветвей и границ (МВГ) для выбора каждого разделяющего элемента применяем более одного предиктора – подробнее об этом в разделе III, озаглавленном соответствующим образом: «Применение нескольких предикторов …»

Некоторые полученные результаты вычислительных экспериментов мы приводим не в конце статьи, как делается обычно, а в разделе IV. Такой не совсем обычный порядок изложения можно объяснить тем, что далее, для краткого описания использованных алгоритмов (раздел V), мы ориентируемся прежде всего на описание тех использованных подалгоритмов (вспомогательных алгоритмов), которые нужны именно для получения удачных результатов вычислений.

В заключении (раздел VI) мы приводим краткое описание несколько возможных вариантов продолжения рассматриваемой нами темы.

II. КРАТКО О ГЕНЕРАЦИИ ИСХОДНЫХ ДАННЫХ ДЛЯ РАССМАТРИВАЕМЫХ ЧАСТНЫХ СЛУЧАЕВ ЗАДАЧИ КОММИВОЯЖЁРА

Далее приведём с небольшими изменениями несколько абзацев из статьи первого автора [9].

Случайный вариант ЗКВ [1] формируется путём применения равномерного распределения не к местоположению точек – а сразу к генерируемым элементам матрицы [2]. А основной алгоритм настоящей статьи, метод ветвей и границ, может быть применён, конечно, к любому частному случаю любого варианта ЗКВ – но часто очень удобно считать, что мы применяем его именно к случайному варианту ЗКВ.

Теперь рассмотрим *псевдогеометрический* вариант ЗКВ подробнее. Алгоритм генерации частных случаев задачи (как и ранее, совпадающий с её *определением*) для него заключается в следующем:

1) в качестве входных данных выбирается не только размерность матрицы, но и некоторое значение σ;
2) обычным образом генерируется «предварительная» матрица для геометрического варианта ЗКВ с выбранной размерностью [3]; отметим по этому поводу, что в научной литературе – причём, по крайней мере, с начала 1990-х годов, – именно такой геометрический вариант ЗКВ рассматривается в подавляющем большинстве публикаций, см. [10], [11], [12], [13] и мн. др. [4]
3) каждый элемент матрицы (или её верхнетреугольной части – при рассмотрении симметричного случая) умножается на случайную величину, выбираемую с нормальным распределением [5] с $\mu = 1$ и выбранным значением σ; при этом получающиеся с очень небольшой вероятностью отрицательные (либо неположительные) значения [6] – а также значения, соответствующие умножению на полученные случайные величины, большие (большие или равные) 2 – просто «не засчитываются» и пересчитываются; при этом возможен как симметричный случай (если заранее есть соответствующее условие), так и «несимметричный»;
4) возможно дополнительное требование – выполнение неравенства треугольника для всей получившейся матрицы (либо не очень большая «степень нарушения» этого неравенства); в этом случае должны быть применены какие-то дополнительные алгоритмы генерации, которые мы подробно обсуждать не будем – но отметим, что они связаны с тем, что во введении статьи [8] было названо предварительным алгоритмом для «хорошего расположения» последовательности точек.

Итак, про любой частный случай ЗКВ можно сказать, что он *был* сгенерирован с помощью алгоритма для псевдогеометрического варианта ЗКВ (как и для случайного варианта ЗКВ, мы это уже отмечали выше) – но, в отличие от случайного варианта ЗКВ мы для псевдогеометрического варианта можем сказать следующее: при соответствующем выборе значения σ могут быть получены не только любые частные случаи, но и *различные варианты* ЗКВ:

- при $\sigma = 0$ мы получаем геометрический вариант ЗКВ;
- наоборот, при $\sigma = \infty$ (т. е. реально – при достаточно большом значении) мы получаем случайный вариант ЗКВ.

Немного более подробно про генерацию частных случаев задачи коммивояжёра с разными значениями σ см. в [8].

Для уже сгенерированного частного случая ЗКВ часто бывает интересно решить обратную задачу – т. е. с помощью какого-либо из не связанных с генерацией входных данных алгоритма (см. [14, гл. 9, § 4] и мн. др.) определить, при каком именно конкретном значении σ некоторый заданный (рассматриваемый) частный случай ЗКВ мог быть сгенерирован с наибольшей вероятностью. Немного подробнее об этой обратной задаче см. в заключении.

Для многих наших предыдущих публикаций ([8], [15], [16], [17] и др.) представляет большой интерес такое

[1] Отметим, что его формулировка, как и для некоторых других вариантов ЗКВ, одновременно является и *алгоритмом генерации*.

[2] Вообще, про *любой* частный случай ЗКВ можно предполагать, что он был сгенерирован как случай случайного варианта ЗКВ. Более того, то же самое можно сказать и о псевдогеометрическом варианте ЗКВ.

[3] То есть такого варианта, в котором точки случайно бросаются в единичный квадрат (обе координаты при этом выбираются как случайные величины с *равномерным* законом распределения), а элемент матрицы коммивояжёра в точности равен расстоянию между соответствующими точками.

[4] В процитированных здесь отдельной статье, а также в разных статьях монографии, применяются сакмые разные методы решения ЗКВ. Однако их объединяет то, что рассматривается, в первую очередь, геометрический вариант.

[5] Возможный способ генерации таких случайных величин может быть построен, например, на основе изложенного на следующем сайте: `https://habr.com/ru/post/263993/`. Однако при этом авторы не нашли в Интернете описания *существенно более простого* способа, возможного для небольших значений σ; подробности такой генерации мы опубликуем в будущем.

[6] Конечно, такая приемлемая небольшая вероятность будет для малых значений σ.

возможное теоретическое приложение задачи коммивояжера – и, соответственно, её решения: приложение связано с *оценкой живучести сети связи*. Живучесть сети связи – это интегральный показатель, являющийся усреднённым показателем оценок живучести всех возможных направлений связи, определяемых парой вершин графа сети связи, см. [18], [19], [20] и др. Основополагающим фактором в обеспечении живучести направления связи является наличие географически разнесённых маршрутов, т. е. маршрутов, представленных вершинно-независимыми путями. Как правило, при проектировании и / или модернизации сети связи разработчики стремятся к тому, чтобы *все направления связи имели резервный маршрут*. В нашем случае мы предполагаем, что мы должны это проверить, – а такую проверку можно произвести с использованием алгоритма поиска оптимального Гамильтонова цикла.

Сразу отметим, что эта процедура в случае нахождения оптимального Гамильтонова цикла даёт достаточное условие того, что все направления связи сети связи имеют резервный маршрут. При этом доказательство достаточности очевидно, а строгое доказательство необходимости представляется интересным предметом дальнейшего исследования.

Говоря при этом о генерации данных для проверки предлагаемых алгоритмов нужно отметить следующий факт. Достаточно очевидно, что применительно к разработке и / или проверке живучести имеющейся сети связи мы рассматриваем не геометрический вариант ЗКВ – а какой-то более сложный. Поэтому и генерация данных для проверки использования некоторых конкретных алгоритмов, связанных с сетями связи, должна быть более сложной, чем та, которая даёт простой геометрический вариант ЗКВ. В настоящее время мы ещё не исследовали адекватность модели, связанной с использованием для генерации исходных данных псевдогеометрической версии некоторого конкретного значения σ – поэтому пока мы рассматриваем другой упрощённый вариант псевдогеометрической ЗКВ – а именно, случайный её вариант.

Итак, процедура решения задачи коммивояжёра может служить оценочной процедурой-индикатором для определения важнейших технических параметров сети связи. Учитывая квадратичную функцию количества направлений связи от вершин графа, представляющего собой такую сеть связи, а также предполагая большую размерность этого графа, можно сказать, что удачный эвристический алгоритм поиска оптимального Гамильтонова цикла в случае положительного его (алгоритма) завершения может конкурировать с традиционным подходом – заключающемся в последовательной обработке направлений связи с целью получения параметров живучести.

Специально отметим, что мы, как и во многих предыдущих рабоах, имеем в виду два варианта применения метода ветвей и границ:

- во-первых – для алгоритмов, предназначенных для получения точного решения задачи;
- и, во-вторых, для т. н. anytime-алгоритмов.

Согласно одной из возможных классификаций, 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 в режиме реального времени просматривать эти псевдооптимальные решения, а последовательность таких решений в пределе обычно даёт оптимальное решение. На практике такие алгоритмы нужны, например, в связи с тем, что мы запускаем программу для нашего алгоритма много раз, а окончательное решение принимает человек («эксперт»). Важно, что такой подход 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» ([21]); в нашем случае подобным псевдооптимальным решением является некоторый конкретный тур коммивояжёра, и человек-эксперт принимает решение, является ли этот тур приемлемым.

Немного подробнее про конкретные модификации стандартного варианта метода ветвей и границ, связанные с его оформлением как anytime-алгоритма, см. в конце следующего раздела.

III. ПРИМЕНЕНИЕ НЕСКОЛЬКИХ ПРЕДИКТОРОВ В МЕТОДЕ ВЕТВЕЙ И ГРАНИЦ

Слово «предиктор» применялось во многих наших предыдущих статьях (к приведённым выше ссылкам на работы авторов сто́ит добавить [22], [23], результаты которых будут обсуждаться и ниже). Этот термин обычно обозначает специальный (под) алгоритм для выбора разделяющего элемента на каждом шаге метода ветвей и границ.

По поводу последнего понятия (предиктора) приведём первое (из трёх) «критическое замечание» к известной монографии [24][7]. Цитата оттуда (стр. 95) такова: «В качестве перехода, используемого для разветвления, выбирается тот, у которого значение θ_{ij} максимально»; да, это в точности согласно классике (см. [25], а также ещё более ранние публикации), такой выбор возможен, он совсем неплохой – однако не сказано о том, что подобный вспомогательный алгоритм не является единственно возможным. Более того, по мнению авторов настоящей статьи, именно описание подобных алгоритмов является *определяющим фактором, влияющем на качество всего конкретного варианта метода ветвей и границ*[8].

Однако в вычислительных экспериментах, проведённых *для настоящей статьи*, мы реально применяли не «несколько», а *ровно два* предиктора; но всё-таки использованное нами название статьи вовсе не является «преувеличением» – и это можно объяснить следующими двумя обстоятельствами.

- Во-первых, всюду в ранее вышедших ранее монографиях [24], [25], а также в известных авторам статьях, предиктор был только один. Отметим по этому поводу, что возможность использования нескольких

[7] Про остальные два «критических замечания» см. далее. Отметим, что подобных «замечаний» можно привести ещё несколько.

[8] Понятие «качество» для разных вариантов МВГ мы здесь конкретизировать не будем, различные соответствующие его критерии были не раз использованы во многих наших предыдущих публикациях. Очень кратко скажем, что во-первых, могут быть как абсолютные критерии (разные варианты отличия найденного решения от оптимального), так и временны́е; и, во-вторых, возможны разные варианты усреднения получаемых значений.

предикторов следует из теории, изложенной в [2]; однако ни явного указания на такую возможность, ни соответствующих примеров в указанной монографии не приводится [9].

- Во-вторых, мы действительно описываем возможность применения нескольких предикторов – что делается совершенно таким же способом Отметим ещё, что впервые об этом было сказано у одного из авторов настоящей статьи ещё в 2005 г. – см. [26], [27].

К монографии [24] приведём второе «критическое замечание». Согласно другой цитате оттуда (стр. 95),

> «Если после применения процедуры приведения в каждой строке и в каждом столбце есть ровно один нуль, то мы решили задачу о назначении. Величина целевой функции равна d_0, а нули указывают назначения. Если нули образуют единственный цикл, то мы решили задачу о коммивояжере, и длина кратчайшего маршрута равна d_0».

Авторы настоящей статьи, однако, не видят большой связи между задачей о назначении и ЗКВ (несмотря на некоторую схожесть их формулировок): алгоритмы для решения первой из них довольно просты. Более того, реально ситуация «нули образуют единственный цикл» бывает с весьма малой вероятностью [10].

Также по связанному поводу стоит отметить, что рассматриваемые в [24] т. н. 1-деревья *не представляют собой альтернативу* построению т. н. последовательности правых (под) задач, рассматриваемой в нескольких наших предыдущих публикациях ([9], [28] и др.). Более того, они (1-деревья) применяются только для симметричного варианта ЗКВ, но при этом, вообще говоря, не для геометрического её варианта – а авторы настоящей статьи не видят для такого варианта ЗКВ приемлемых математических моделей (т. е. моделей, применяющихся в реальных задачах).

Теперь вернёмся к очень краткому описанию полученных нами результатов вычислительных экспериментов (более подробно – в следующем разделе). Также в наших предыдущих работах ([22], [23] и др.) исследовалась только сама возможность применения нескольких предикторов, были выполнены соответствующие компьютерные программы – однако *реальных* результатов, которые улучшали бы работу с одним предиктором, получено не было. Также в двух связанных публикациях [29], [30] [11] было показано, что для окончательного выбора разделяющего элемента возможно применение т. н. функций риска – вместо других алгоритмов многокритериальной оптимизации ([31], [32] и др.), алгоритмов существенно более сложных, и при том далеко не всегда дающих приемлемый практический выигрыш.

Итак, в отличие от предыдущих работ, в которых *только исследовалось* применение нескольких предикторов, в настоящей работе получены конкретные улучшения алгоритма МВГ. А именно, мы получили *уменьшение* количества первоначально сгенерированных вариантов проблемы (вариантов ЗКВ), в которых после применения достаточно большого числа итераций МВГ получаеся решение, которое ещё может быть улучшено (улучшено применением новых итераций). Это уменьшение приводится путём сравнения двух вариантов выполнения алгоритма:

- обычного варианта – с одним предиктором, вариант близок к [25];
- и варианта с двумя предикторами – вторым является размерность рассматриваемой подзадачи.

Однако мы должны оговориться, что предметом статьи является *самый простой вариант* употребления *двух* предикторов. Они используются не для собственно выбора разделяющего элемента (что можно было бы назвать «априорным» алгоритмом) – а для «апостериорного» размещения двух полученных подзадач в массиве подзадач; это, по-видимому, существенно проще.

Как мы отмечали ранее, с материалом этого раздела связаны понятия, относящиеся к anytime-алгоритмам; по этому поводу приведём с сокращениями два абзаца и рисунок из статьи первого автора [9]. Итак, возможная *причина* практического применения anytime-алгоритмов такова – она, кстати, выполняется для почти любой труднорешаемой задачи. Пусть мы оцениваем время выполнения всей задачи в 3 месяца – а заказчику (начальнику, пользователю) хочется получить *хоть какое-то* допустимое решение (конечно, желательно, чтобы оно было более-менее близко к оптимальному) гораздо скорее: скажем, после 1 часа работы программы. Если же полное время выполнения всей задачи существенно больше 1 часа, то какие-то близкие к оптимальным решения начинают получаться «почти» за 3 месяца, скажем, за 2 месяца 28 дней [12]. Что же делать? Вот для этого и используют дополнительные эвристики, дающие получение пусть даже «более далёкого» решения (чем если бы работал обычный МВГ) – зато очень быстро. Одной из таких эвристик и является использование последовательности правых задач (ППЗ).

Эта эвристика (см. её иллюстрацию на рис. 1) заключается в следующем. Каждый раз при выборе очередного разделяющего элемента в некоторой задаче (пусть это T; можно сказать – при получении очередной правой задачи) мы при её применении фактически строим такую последовательность:

- саму задачу T,
- правую (под)задачу задачи T,
- правую задачу правой задачи задачи T,
- и так далее.

Естественно, каждый раз строятся (и включаются в список задач для потенциального решения в последующем) и соответствующие левые задачи:

[9] По крайней мере, мы там таких примеров не нашли.

[10] Несложно показать, что для матриц коммивояжёра размерности N такая вероятность для случайной ЗКВ примерно равна $\frac{1}{N^2}$, также несложно получить примерно такое значение статистически. Подобная вероятность в реальных условиях (скажем, при $N > 50$) приводит к тому, что программе не надо тратить время на проверку, «образуют ли нули единственный цикл», – а надо продолжать выполнение обычных шагов метода ветвей и границ.

[11] Относящихся к несколько иной предметной области – но это непринципиально.

[12] Это реальные оценки. Действительно, в классическом описании МВГ первыми выполняются подзадачи, имеющие меньшие значения границ, – и вследствие этого, как правило, бóльшие размерности. При этом задачи с меньшими размерностями «отодвигаются в конец» – что и приводит к подобному сроку получения самых первых результатов.

- левая задача задачи T,
- левая задача правой задачи задачи T,
- и так далее.

Описанный процесс заканчивается:

- либо при получении тривиальной задачи (например, задачи нулевой размерности): в этом случае мы запоминаем её решение (границу, получаемый к моменту её постановки тур, и другие *характеристики*) в качестве *текущего на данный момент времени псевдо-оптимального решения* нашего anytime-алгоритма;
- либо при получении в какой-либо задаче достаточно большой границы – например, большей, чем имеющееся на данный момент времени псевдо-оптимальное решение [13].

Отметим, что на практике описанный процесс построения ППЗ *не занимает много лишнего времени* и, кроме того, *не приводит к большому увеличению списка задач*, предназначенных для потенциального решения в будущем.

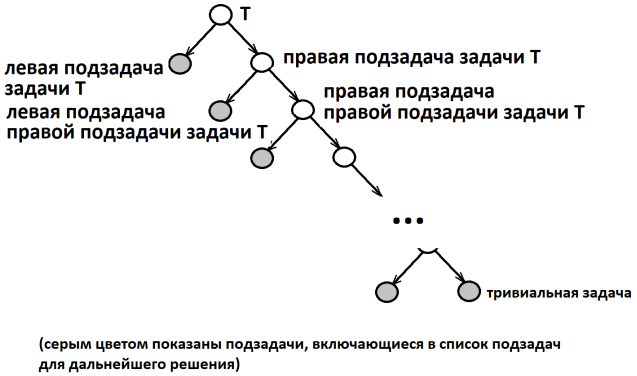

Рис. 1. Последовательность правых задач.

Итак, построение последовательности правых задач в процессе ветвления на шаге метода ветвей и границ можно назвать конкретной модификацией стандартного варианта этого метода.

Теперь приведём третье «критическое замечание». к монографии [24]. В ней на рис. 3.9 и 3.10 (стр. 95 и 97) приведены структуры вычисления (ветвления МВГ), можно сказать – «абстрактное» и «конкретное» ветвления; может показаться, что они аналогичны нашему рис. 1. Однако оба варианта описывают процесс ветвления неудачно, поскольку для этого ветвления на обоих рисунках каждый раз выбирается правая подзадача; конечно, такие варианты ветвления возможны – но «не очень информативны». А «более информативен» [24, рис. 3.11] (стр. 100) – но, по-видимому, его тоже нельзя считать хорошей иллюстрацией к исполнению алгоритма МВГ: во-первых, в том примере элементы матрицы – очень малые величины, и, во-вторых, оптимум получается как фактический конец ППЗ (что не очень интересно).

В этом смысле гораздо более интересными являются широко известные примеры, в которых сколь угодно

[13] В этом вопросе не всё так очевидно. Иногда процесс построения ППЗ может быть прерван ранее – для получения оптимального решения некоторой промежуточной задачи, например, методом перебора (при небольшой размерности задачи) или каким-либо иным методом. Этот вопрос связан с подробными исследованиями МВГ.

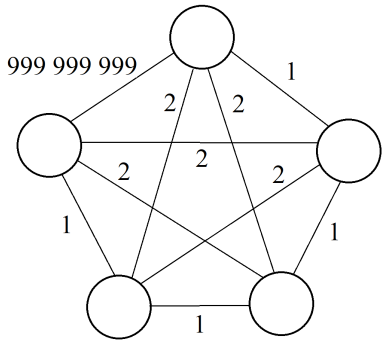

Рис. 2. Широко известный пример, для которого жадный алгоритм даёт ответ, далёкий от оптимума.

далёкими от оптимумов получаются решения, найденные без применения МВГ, а только с помощью жадных алгоритмов, т. е. фактически с помощью построения одной лишь ППЗ – однако такие примеры всё-таки надо назвать тривиальными (см. пример на рис. 2 и т. п.). По этому поводу см. следующий раздел настоящей статьи.

IV. НЕКОТОРЫЕ РЕЗУЛЬТАТЫ ВЫЧИСЛИТЕЛЬНЫХ ЭКСПЕРИМЕНТОВ

В первую очередь уточним, что матрицы для вычислительных экспериментов генерировались следующим образом: для каждого элемента матрицы выбиралось новая случайная величина – и эти случайные величины были независимы и равномерно распределены от 1 до 999. (Это небольшое описание генерации данных можно считать следствием материала раздела II – однако, по-видимому, сделанное уточнение полезно для дальнейшего изложения.)

Также сразу поясним, почему мы приводим результаты вычислительных экспериментов не в конце статьи, как это делается обычно, а в настоящем разделе. Как мы уже отмечали во введении, такой не совсем обычный порядок изложения можно объяснить тем, что далее, при кратком описании используемых алгоритмов (раздел V), мы ориентируемся прежде всего на описание тех применяемых подалгоритмов (вспомогательных алгоритмов), которые нужны именно для получения удачных результатов вычислений.

В связи с этим мы в первую очередь приведём интересный вариант матрицы ЗКВ размерности 99×99 (см. 9 квадратов матрицы в таблицах I – IX) [14] – об этом примере было кратко сказано в конце предыдущего раздела. Эта матрица – в качестве исходной для МВГ – интересна тем, что:

- после примерно 2 % времени работы, необходимого для полного выполнения стандартного варианта метода ветвей и границ, мы получаем близкое к оптимальному решение 1681; решение приведено в таблице X [15];

[14] Для проверки полученных нами результатов вычислений такой «большой» объём исходных данных (4.5 страницы текста) проблем вызывать не должен: из pdf-файла матрица копируется – например, в электронную таблицу csv-формата – с помощью обычных «копипастинга» и контекстной замены.

[15] Организация этой и следующей таблиц такова. В клетке, соответствующей числу N (оно изменяется от 1 до 99), записан номер города, куда в рассматриваемом туре осуществляется переезд из города с номером N.

Таблица I
Верхне-левый квадрат таблицы исходных данных: строки от 1 до 33, столбцы от 1 до 33.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
1	###	41	467	334	500	169	724	478	358	962	464	705	145	281	827	961	491	995	942	827	436	391	604	902	153	292	382	421	716	718	895	447	726
2	929	###	541	833	115	639	658	704	930	977	306	673	386	21	745	924	72	270	829	777	573	97	512	986	290	161	636	355	767	655	574	31	52
3	310	617	###	813	514	309	616	935	451	600	249	519	556	798	303	224	8	844	609	989	702	195	485	93	343	523	587	314	503	448	200	458	618
4	832	932	169	###	154	721	189	976	329	368	692	425	555	434	549	441	512	145	60	718	753	139	423	279	996	687	529	549	437	866	949	193	195
5	372	159	833	70	###	487	297	518	177	773	270	763	668	192	985	102	480	213	627	802	99	527	625	543	924	23	972	61	181	3	432	505	593
6	423	520	902	962	123	###	596	737	261	195	525	264	260	202	116	30	326	11	771	411	547	153	520	790	924	188	763	940	851	662	829	900	713
7	532	963	607	483	911	635	###	67	848	675	938	223	142	754	511	741	175	459	825	221	870	626	934	205	783	850	398	279	701	193	734	637	534
8	129	240	813	174	601	77	215	###	683	213	992	824	601	392	759	670	428	27	84	75	786	498	970	287	847	604	503	221	663	706	363	10	171
9	588	342	608	60	221	758	954	888	###	146	690	949	843	430	620	748	67	536	783	35	226	185	38	853	629	224	748	923	359	257	766	944	955
10	317	213	109	28	200	80	318	858	50	###	155	361	264	903	676	643	909	902	561	489	948	282	653	674	220	402	923	831	369	878	259	8	619
11	71	885	974	71	333	867	153	295	168	825	###	676	629	650	598	309	693	686	80	116	249	667	528	679	864	421	405	826	816	516	726	666	87
12	37	925	647	458	602	807	98	830	292	600	278	###	799	352	448	882	540	315	575	762	567	336	397	418	897	828	851	816	230	449	925	658	229
13	876	826	396	572	249	640	174	819	943	611	941	289	###	419	565	805	585	216	450	615	609	64	166	893	74	509	300	695	573	589	161	172	968
14	583	948	723	982	18	776	220	111	182	856	490	925	324	###	486	677	969	643	534	677	668	68	991	196	783	828	727	426	871	697	612	703	27
15	938	881	257	750	614	598	458	661	63	756	807	278	489	435	###	365	75	586	386	833	360	330	48	928	492	433	840	766	735	810	599	837	892
16	826	276	790	582	578	159	418	489	159	449	924	72	380	8	967	###	208	477	503	370	607	196	74	722	611	19	761	56	890	163	683	716	932
17	874	791	469	912	146	693	91	815	949	857	640	52	236	551	487	226	###	162	955	183	394	180	97	65	65	513	261	578	78	878	140	611	947
18	643	888	153	232	747	680	926	678	450	801	961	199	855	363	716	573	561	###	245	473	274	550	353	181	287	699	110	643	465	172	529	981	112
19	885	337	311	604	677	406	768	22	413	0	542	537	38	388	355	289	647	181	###	93	584	987	761	493	217	501	482	447	665	753	104	84	95
20	400	707	955	666	141	588	481	168	315	396	225	9	12	136	455	762	43	742	21	###	922	512	248	18	368	717	714	650	290	335	759	169	895
21	671	457	998	545	597	218	838	844	372	563	28	264	801	723	490	604	601	227	197	692	###	771	363	301	363	721	565	421	445	610	495	741	22
22	125	118	737	28	119	577	737	91	556	795	60	901	793	432	136	580	875	907	184	74	719	###	790	476	41	351	329	290	974	72	591	189	787
23	189	542	63	547	502	617	99	23	226	203	48	51	570	636	458	967	456	405	531	962	819	975	###	556	531	495	44	591	803	388	915	450	319
24	149	730	244	844	49	118	65	363	552	773	470	731	747	511	869	398	498	103	352	679	53	43	522	###	88	563	834	850	22	240	911	492	651
25	307	997	151	423	890	717	640	703	566	883	661	659	245	386	651	765	601	840	209	497	283	250	58	421	###	175	581	787	271	287	999	504	979
26	731	154	861	434	385	967	816	393	704	866	953	308	223	684	792	667	48	469	930	811	814	90	427	743	604	###	599	474	195	506	158	589	858
27	145	282	825	718	573	712	37	998	905	162	717	692	539	47	946	103	231	115	839	858	829	645	394	199	645	272	###	675	862	72	773	480	238
28	274	133	885	38	170	862	629	84	909	878	923	85	400	24	193	105	412	765	767	407	677	784	704	790	834	891	621	###	85	734	190	542	998
29	959	251	738	370	124	507	7	584	951	101	489	958	441	790	13	412	855	60	93	472	402	676	543	373	266	651	275	528	###	532	761	469	503
30	722	842	7	463	260	947	793	630	717	43	376	314	626	117	334	620	171	792	964	154	866	693	664	775	0	212	100	551	476	###	379	943	877
31	671	47	115	592	311	657	405	53	171	580	740	530	675	320	790	377	998	586	604	489	631	744	388	610	718	919	259	927	609	119	###	479	716
32	292	237	530	613	296	710	389	628	809	786	744	840	182	569	120	393	292	962	214	527	807	194	705	713	11	913	693	664	203	995	762	###	381
33	170	450	182	301	293	99	332	44	331	75	4	50	26	66	738	801	623	843	441	380	695	918	659	363	628	329	88	854	613	979	569	666	###

Таблица II
Верхне-средний квадрат таблицы исходных данных: строки от 1 до 33, столбцы от 34 до 66.

	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66
1	771	538	869	912	667	299	35	894	703	811	322	333	673	664	141	711	253	868	547	644	662	757	37	859	723	741	529	778	316	35	190	842	288
2	350	150	941	724	966	430	107	191	7	337	457	287	753	383	945	909	209	758	221	588	422	946	506	30	413	168	900	591	762	655	410	359	624
3	580	796	798	281	589	798	9	157	472	622	538	292	38	179	190	657	958	191	815	888	156	511	202	634	272	55	328	646	362	886	875	433	869
4	297	416	286	105	488	282	455	734	114	701	316	671	786	263	313	355	185	53	912	808	832	945	313	756	321	558	646	982	481	144	196	222	129
5	725	31	492	142	222	286	64	900	187	360	413	974	270	170	235	833	711	760	896	667	285	550	140	694	695	624	19	125	576	694	658	302	371
6	958	578	365	7	477	200	58	439	303	760	357	324	477	108	113	887	801	850	460	428	993	384	405	540	111	704	835	356	72	350	823	485	556
7	556	993	176	705	962	548	881	300	413	641	855	855	142	462	611	877	424	678	752	443	296	673	40	313	875	72	818	610	17	932	112	695	169
8	489	240	164	542	619	913	591	704	818	232	750	205	975	539	303	422	98	247	584	648	971	864	913	75	545	712	546	678	769	262	519	985	289
9	318	726	411	25	355	1	549	496	584	515	964	342	75	913	142	196	948	72	426	606	173	429	404	705	626	812	375	93	565	36	736	141	814
10	971	3	945	781	504	392	685	313	698	589	722	938	37	410	461	234	508	961	959	493	515	269	937	869	58	700	971	264	117	215	555	815	330
11	681	964	340	686	21	662	721	64	309	415	902	873	124	941	745	762	423	531	806	268	318	602	907	307	481	12	136	630	114	809	84	556	290
12	520	940	560	147	162	655	675	792	361	754	398	146	714	946	188	569	638	663	75	515	521	475	615	528	234	570	905	464	557	962	161	524	549
13	358	31	268	426	510	422	774	779	910	552	182	391	495	764	874	364	902	255	460	474	972	821	122	547	577	789	605	195	594	950	343	754	481
14	408	545	508	185	238	237	443	313	501	850	128	111	650	149	192	454	869	681	465	267	713	793	634	472	972	830	901	442	177	877	770	702	364
15	982	328	352	369	244	794	608	252	647	432	535	208	264	497	243	649	15	841	189	100	812	648	523	851	474	633	891	200	854	990	697	919	780
16	452	741	954	813	862	396	460	615	904	599	136	680	198	32	387	584	240	517	6	670	241	882	249	523	758	105	621	95	296	916	678	178	579
17	445	170	975	489	750	149	333	865	214	282	7	432	896	367	522	882	810	641	231	187	705	479	321	538	351	447	208	646	276	759	189	422	666
18	476	381	247	890	671	805	372	32	989	320	165	431	658	293	206	578	948	206	171	166	396	697	20	694	529	788	109	984	969	978	617	15	626
19	525	221	964	781	872	106	656	343	593	80	80	868	411	713	968	251	216	79	768	40	531	933	779	663	259	653	936	95	365	874	720	835	680
20	303	640	979	199	105	791	661	681	652	753	33	29	987	42	253	83	420	814	718	244	63	229	652	864	769	470	5	47	594	487	326	276	323
21	812	151	15	55	393	738	279	882	608	654	822	707	245	338	144	290	339	154	604	623	225	78	724	981	330	733	223	594	130	846	987	445	805
22	490	239	893	53	63	681	903	5	176	479	695	139	468	998	83	639	515	621	993	826	722	838	828	581	399	978	891	23	943	834	243	349	702
23	272	791	383	133	775	642	569	300	954	78	585	257	333	893	490	103	750	233	722	271	611	990	338	641	258	47	352	658	479	302	683	990	2
24	580	477	616	876	178	220	615	348	798	179	635	857	883	662	902	262	420	770	22	273	841	686	888	917	717	892	698	267	749	65	389	932	619
25	138	700	530	461	118	205	540	828	459	622	398	760	98	935	481	614	567	493	596	161	746	538	670	538	476	555	917	371	760	752	758	433	255
26	809	889	447	40	818	364	975	26	89	503	66	412	840	570	676	395	641	986	651	109	187	83	85	390	250	686	755	381	128	502	277	862	424
27	897	542	608	203	277	125	134	401	78	382	169	736	478	939	138	721	426	663	677	575	724	981	700	961	862	2	448	95	684	16	137	507	993
28	86	18	621	424	597	376	254	669	108	927	493	68	366	102	438	600	819	318	290	984	339	556	808	632	478	814	787	239	74	20	827	554	988
29	729	107	892	451	953	392	570	519	472	407	494	505	440	383	262	409	607	38	360	471	171	654	947	514	523	229	489	766	887	756	632	470	739
30	789	361	385	272	434	144	561	563	504	946	888	308	157	430	123	464	74	346	837	981	318	611	292	591	832	123	461	991	461	330	498	369	291
31	300	396	853	458	830	593	806	849	854	272	160	294	54	333	146	229	471	428	559	241	421	688	366	33	685	270	329	261	860	208	982	162	506
32	946	909	48	768	624	511	954	235	127	633	240	862	127	577	817	340	142	100	400	558	476	154	935	430	831	295	537	623	922	927	29	211	211
33	679	577	837	402	115	292	723	872	448	323	153	948	395	197	508	908	411	993	665	781	202	610	828	256	705	965	23	769	715	253	805	684	478

Таблица III

Верхне-правый квадрат таблицы исходных данных: строки от 1 до 33, столбцы от 67 до 99.

	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99
1	106	40	942	264	648	446	805	890	729	370	350	6	101	393	548	629	623	84	954	756	840	966	376	931	308	944	439	626	323	537	538	118	82
2	537	548	483	595	41	602	350	291	836	374	20	596	21	348	199	668	484	281	734	53	999	418	938	900	788	127	467	728	893	648	483	807	421
3	142	844	416	881	998	322	651	21	699	557	476	892	389	75	712	600	510	3	869	861	688	401	789	255	423	2	585	182	285	88	426	617	757
4	161	535	450	173	466	44	659	292	439	253	24	154	510	745	649	186	313	474	22	168	18	787	905	958	391	202	625	477	414	314	824	334	874
5	466	678	593	851	484	18	464	119	152	800	87	60	926	10	757	170	315	576	227	43	758	164	109	882	86	565	487	577	474	625	627	629	928
6	216	626	357	526	357	337	271	869	361	896	22	617	112	717	696	585	41	423	129	229	565	559	932	296	855	53	962	584	734	654	972	457	369
7	831	40	488	685	90	497	589	990	145	353	314	651	740	44	258	335	759	192	605	264	181	503	829	775	608	292	997	549	556	561	627	467	541
8	944	865	540	245	508	318	870	601	323	132	472	152	87	570	763	901	103	423	527	600	969	15	565	28	543	347	88	943	637	409	463	49	681
9	994	256	652	936	838	482	355	15	131	230	841	625	11	637	186	690	650	662	634	893	353	416	452	8	262	233	454	303	634	303	256	148	124
10	39	212	288	82	954	85	710	484	774	380	815	951	541	115	679	110	898	73	788	977	132	956	689	113	8	941	790	723	363	28	184	778	200
11	293	996	152	54	345	708	248	491	712	131	114	439	958	722	704	995	52	269	479	238	423	918	866	659	498	486	196	462	633	158	22	146	392
12	469	330	923	350	333	925	910	737	336	337	278	393	636	714	164	591	949	135	505	337	4	337	623	664	970	608	568	281	85	152	373	652	194
13	12	672	439	428	912	762	967	408	415	908	223	759	434	204	486	319	958	945	806	166	700	367	692	787	532	556	974	447	21	283	222	331	376
14	381	590	823	237	23	179	595	169	327	42	310	182	58	926	487	670	528	651	258	213	860	783	286	742	610	472	128	434	841	718	503	867	865
15	578	931	544	340	487	899	525	483	538	492	193	252	11	560	834	840	497	785	529	540	805	791	392	210	549	578	979	971	277	73	193	620	497
16	58	577	750	7	729	81	995	678	676	753	899	784	565	93	608	172	243	929	514	168	55	191	973	922	748	651	986	144	446	577	517	629	916
17	486	455	28	614	860	253	777	348	503	861	431	82	455	197	106	752	821	296	281	21	455	947	124	318	135	376	774	859	998	74	253	922	635
18	684	168	906	928	97	118	390	199	785	486	199	420	710	271	813	415	85	318	580	331	267	387	444	186	507	360	827	74	431	152	271	268	693
19	976	455	725	71	808	559	156	602	832	905	440	375	562	885	962	80	836	797	202	508	80	340	76	58	493	740	546	474	773	97	880	335	72
20	540	679	990	588	710	271	945	221	470	183	589	955	978	779	6	262	135	487	196	33	88	935	779	993	790	962	965	1	105	807	567	669	134
21	616	750	489	338	963	135	697	209	630	224	908	737	474	920	372	293	855	734	561	56	606	184	75	382	119	741	432	684	779	279	283	667	836
22	707	502	141	687	346	891	637	413	400	816	690	162	935	126	410	877	382	260	189	705	874	663	722	195	566	360	38	588	811	245	467	425	867
23	568	422	895	135	8	361	742	194	699	188	178	42	357	941	847	469	345	380	913	964	710	61	385	73	504	462	703	102	69	154	529	552	74
24	81	3	130	28	631	589	152	630	172	864	407	295	428	681	490	610	177	71	236	459	643	840	633	37	893	630	274	993	782	203	461	290	662
25	65	726	401	34	759	393	548	273	792	193	933	137	103	690	211	694	668	627	498	989	248	879	647	149	931	669	875	598	449	436	599	123	443
26	565	872	832	885	428	646	889	478	883	925	265	260	45	778	821	855	520	927	773	134	251	675	336	334	1	737	310	974	590	356	71	80	935
27	284	944	259	821	58	643	668	677	119	857	41	891	264	623	915	72	929	841	715	615	536	957	759	700	452	93	241	829	448	227	798	224	324
28	441	798	642	2	321	104	946	56	509	833	708	761	533	686	804	385	142	842	260	161	620	343	578	187	113	63	591	934	415	656	761	12	411
29	666	522	283	160	553	705	91	259	386	687	629	42	317	45	356	388	452	154	466	833	760	919	631	738	267	776	98	314	320	53	7	469	216
30	400	179	117	317	914	595	441	936	867	28	453	909	973	981	503	569	816	883	367	385	402	230	157	681	567	310	866	687	171	477	245	764	238
31	878	979	897	887	538	469	783	540	795	266	633	509	557	317	134	53	135	258	593	843	693	466	951	140	94	937	174	512	97	551	695	817	492
32	754	880	253	293	321	785	528	443	0	555	671	270	452	375	631	640	934	40	986	834	248	539	765	796	312	262	52	731	987	464	557	248	556
33	145	840	833	285	900	968	994	633	885	348	281	32	256	910	872	982	901	306	635	860	471	527	137	343	768	233	995	171	603	942	11	879	34

Таблица IV

Средне-левый квадрат таблицы исходных данных: строки от 34 до 66, столбцы от 1 до 33.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
34	512	593	530	161	234	258	926	906	191	443	188	227	719	484	224	93	435	224	94	582	867	627	5	146	686	589	540	237	850	759	578	956	677
35	293	117	717	403	943	893	639	11	925	28	978	748	850	892	444	654	623	353	36	547	346	826	61	904	238	571	886	405	299	56	728	267	751
36	484	943	495	811	293	892	798	990	299	890	493	74	549	774	314	173	69	218	865	833	238	911	252	463	508	149	88	245	231	920	106	271	368
37	85	652	751	397	751	1	435	693	72	916	716	328	511	125	838	296	939	544	56	826	902	758	588	21	586	108	777	628	979	947	549	280	245
38	968	710	968	708	641	732	437	684	499	701	869	583	256	692	640	87	12	810	820	835	663	615	897	677	774	134	553	533	878	286	624	734	75
39	712	896	909	841	130	90	34	915	286	523	666	344	736	37	416	893	949	971	844	777	285	200	590	285	325	118	668	6	729	8	541	225	536
40	200	643	856	61	647	939	808	290	452	979	618	419	7	284	547	977	675	613	709	371	642	339	336	621	501	34	129	358	436	469	897	504	798
41	15	974	252	846	956	524	966	581	144	240	334	296	876	27	205	18	291	641	465	191	471	259	700	217	16	943	624	64	505	157	502	666	275
42	345	883	865	696	833	390	632	189	839	183	817	129	753	679	613	600	252	145	861	151	318	963	813	907	554	476	617	211	429	264	966	531	942
43	681	879	108	207	259	706	849	75	26	124	607	774	884	128	546	145	481	714	149	168	338	307	793	504	105	697	217	955	641	343	133	97	679
44	656	318	201	142	276	506	702	684	23	270	312	250	475	311	850	195	488	520	101	183	798	892	848	784	245	187	462	343	258	859	871	16	172
45	257	370	920	654	359	80	298	391	960	510	591	867	33	675	136	499	775	790	376	256	272	703	86	967	515	542	495	566	456	186	437	198	626
46	934	102	587	417	463	153	126	595	495	213	406	370	859	305	496	833	223	852	879	519	126	793	585	465	371	444	107	271	87	99	519	596	715
47	987	94	983	364	264	714	746	390	425	255	163	467	792	231	619	880	718	128	948	720	79	91	766	587	808	126	825	159	518	220	35	314	456
48	665	552	759	755	417	844	554	443	824	999	422	121	733	413	917	793	873	60	175	74	985	501	622	208	395	26	720	291	988	929	72	977	193
49	691	588	538	664	104	877	353	942	4	623	852	625	611	842	724	149	135	590	146	328	855	804	741	909	381	779	102	83	692	532	312	99	161
50	423	347	396	566	230	665	152	0	617	173	786	668	597	579	224	728	260	158	425	267	312	734	452	169	562	570	211	235	890	711	64	750	960
51	366	885	439	718	107	268	554	50	432	799	20	735	98	792	916	182	29	316	59	758	583	399	135	357	70	271	342	780	877	406	295	686	403
52	469	519	341	197	545	654	158	58	794	791	341	837	900	232	675	736	438	958	580	85	13	284	253	727	285	426	670	536	281	370	549	348	77
53	717	928	590	99	580	870	990	412	0	64	275	84	682	911	106	868	644	826	436	366	592	988	838	865	212	947	446	659	52	771	256	110	57
54	39	527	394	490	648	54	517	192	530	618	148	56	990	282	743	439	301	493	927	269	393	33	431	592	135	26	842	80	616	961	717	907	920
55	71	775	9	430	794	622	736	890	15	452	754	782	933	188	855	960	945	923	542	659	487	943	347	584	389	668	788	198	280	479	707	813	860
56	120	810	128	970	621	596	4	514	799	210	269	552	235	471	344	830	382	261	395	151	666	83	710	758	372	295	906	14	452	284	905	812	21
57	511	67	265	840	379	955	28	661	432	227	160	139	666	237	48	848	404	48	565	221	133	175	241	514	147	362	954	676	454	658	721	784	688
58	565	599	251	318	81	558	103	358	238	178	464	457	572	221	922	756	867	260	298	343	234	841	19	352	62	411	811	793	405	881	651	760	898
59	742	748	103	815	47	73	965	749	137	169	331	970	53	768	52	178	156	981	695	586	929	731	278	859	752	89	958	238	54	972	208	656	171
60	437	599	35	179	654	652	399	350	132	945	293	871	649	590	204	177	705	319	750	528	228	722	668	785	774	184	651	884	37	915	189	681	74
61	762	237	272	792	200	343	519	481	370	991	145	984	714	85	79	399	282	386	119	650	193	185	516	608	888	499	425	896	769	331	814	502	970
62	907	551	110	557	77	9	403	576	540	23	574	454	369	906	485	999	48	930	573	658	816	884	373	31	604	783	791	422	10	987	364	568	994
63	239	768	589	683	209	987	795	245	912	639	935	248	479	207	230	954	601	940	43	718	412	377	623	472	983	885	686	720	257	702	439	606	276
64	343	446	532	568	684	361	472	286	266	89	151	604	233	594	76	598	604	533	468	184	417	964	616	14	66	996	873	148	510	396	941	791	866
65	508	878	338	723	5	411	193	117	943	227	514	290	416	801	339	544	236	12	274	433	740	401	183	135	828	633	437	134	577	671	722	263	659
66	74	324	344	863	749	276	597	334	647	426	43	244	592	761	269	175	139	696	481	471	268	713	666	30	806	109	290	59	684	546	759	103	606

Таблица V
Средне-средний квадрат таблицы исходных данных: строки от 34 до 66, столбцы от 34 до 66.

	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66
34	###	403	42	300	811	225	143	595	16	615	804	6	368	641	335	698	325	313	23	312	618	457	396	339	86	94	846	43	522	911	280	455	358
35	19	###	991	166	717	846	93	187	421	912	867	299	444	897	201	328	942	104	768	768	641	296	262	720	244	918	625	219	584	174	984	459	149
36	421	258	###	157	647	982	238	122	723	906	706	189	156	345	351	923	118	550	97	836	591	406	510	813	651	880	844	743	741	245	578	949	705
37	128	62	426	###	719	677	91	47	785	231	406	684	954	769	709	956	421	945	465	16	860	497	335	190	677	49	5	886	388	675	143	377	130
38	349	712	984	624	###	792	504	822	55	913	462	473	84	160	122	53	397	822	750	337	917	6	931	197	505	236	357	599	290	898	985	447	84
39	980	716	470	384	747	###	328	936	204	700	523	364	411	40	250	504	483	766	141	670	845	506	88	256	107	266	857	678	199	674	155	191	990
40	202	105	238	32	65	516	###	481	579	23	727	771	313	382	140	816	488	319	589	513	259	851	271	283	364	38	12	184	759	501	574	733	59
41	46	632	789	46	145	3	152	###	901	326	153	184	649	350	405	346	25	970	336	795	793	323	532	856	896	210	776	692	550	290	795	852	350
42	698	522	155	587	257	158	464	762	###	510	946	65	380	165	946	540	297	42	464	194	584	270	715	200	206	106	302	696	833	611	67	415	645
43	736	555	633	954	930	252	904	279	488	###	870	754	72	176	701	415	86	231	480	975	260	514	938	657	990	438	981	447	242	618	70	631	61
44	832	441	683	88	15	976	516	298	39	283	###	205	997	146	769	675	917	387	20	30	442	735	389	373	480	73	777	786	381	785	522	345	979
45	364	85	665	908	340	360	814	154	12	752	338	###	706	525	630	249	179	79	313	838	517	421	145	991	969	839	308	565	762	473	681	970	147
46	552	492	753	257	590	959	357	886	809	865	686	19	###	259	313	435	28	101	468	869	456	731	578	74	395	843	274	316	24	511	585	722	971
47	147	156	134	360	751	208	478	294	589	524	976	694	90	###	189	372	419	301	702	844	523	539	609	827	637	186	416	827	169	623	899	514	415
48	8	254	922	876	137	816	803	90	912	289	253	979	299	322	###	902	602	369	654	729	734	438	274	226	463	134	972	348	453	503	307	518	333
49	47	405	290	569	959	628	139	228	580	471	778	440	456	148	444	###	745	125	210	698	616	180	815	64	226	472	479	593	482	5	536	333	105
50	545	767	754	787	217	565	942	88	273	815	579	837	388	269	886	830	###	809	640	321	319	881	626	20	411	454	893	984	357	999	336	829	950
51	252	718	35	285	428	678	960	101	949	364	385	626	157	737	841	938	593	###	96	33	493	768	873	935	109	96	381	561	739	584	923	499	731
52	199	835	573	137	351	731	617	233	571	376	529	731	44	342	467	0	528	916	###	634	999	757	240	581	623	941	131	749	539	299	575	99	363
53	666	250	670	145	925	148	874	774	43	429	536	763	16	858	934	301	61	655	815	###	174	582	139	82	24	900	575	337	376	300	424	864	537
54	245	166	121	782	782	6	491	870	615	195	143	763	771	592	935	217	412	881	467	721	###	952	322	828	230	429	424	699	153	151	794	41	264
55	447	846	271	182	344	142	448	636	353	549	888	823	825	251	190	639	46	362	775	624	78	###	985	103	781	377	350	48	170	494	216	729	726
56	341	575	290	525	611	167	982	783	387	793	50	679	268	274	884	629	335	460	105	511	992	824	###	550	16	60	801	759	309	923	67	300	225
57	466	116	856	180	429	537	629	389	843	4	101	384	920	987	136	52	582	528	366	472	193	958	179	###	983	987	823	397	96	594	901	653	520
58	271	842	694	84	535	500	955	155	323	650	688	939	10	17	932	948	210	495	419	648	189	171	851	695	###	223	637	649	800	407	134	884	177
59	147	503	511	174	819	107	205	599	788	261	45	435	492	732	76	667	403	367	899	54	167	212	114	129	793	###	977	854	12	831	889	326	647
60	759	850	546	260	427	233	421	161	818	546	350	263	461	846	458	594	371	945	502	246	420	254	953	760	586	419	###	735	616	113	335	592	18
61	537	598	77	398	851	888	463	53	505	741	267	662	655	81	695	618	872	94	670	538	121	445	523	243	365	718	135	###	937	599	499	192	554
62	579	23	709	657	141	285	477	23	272	828	435	549	517	9	743	867	265	590	891	369	538	604	776	212	991	693	409	620	###	197	662	128	100
63	924	652	37	35	699	316	890	367	606	221	975	520	792	565	547	258	684	531	829	782	705	162	689	923	870	342	282	196	180	###	969	72	715
64	589	20	355	948	251	81	271	825	228	173	454	309	430	238	991	739	853	448	567	54	877	490	197	56	773	589	862	536	429	888	###	539	585
65	540	368	282	848	366	290	227	458	71	106	305	1	434	770	555	507	299	730	75	803	439	194	488	664	673	48	358	280	487	764	713	###	13
66	391	93	196	583	950	557	435	12	667	869	165	903	973	791	516	149	515	118	243	388	497	711	91	874	563	125	7	370	121	727	658	336	###

Таблица VI
Средне-правый квадрат таблицы исходных данных: строки от 34 до 66, столбцы от 67 до 99.

	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99
34	847	870	990	873	327	413	454	896	472	450	241	641	646	525	213	824	929	695	403	594	993	67	187	640	79	41	125	534	860	0	907	621	692
35	14	695	503	32	851	537	814	363	651	219	290	917	683	918	584	252	586	269	248	137	397	424	168	27	148	993	382	271	669	893	877	14	104
36	514	785	227	624	204	356	917	227	555	102	286	468	698	979	411	489	758	490	775	410	132	89	296	950	112	122	529	672	836	277	553	976	670
37	487	555	482	253	0	754	289	488	215	8	310	940	799	997	707	454	304	237	485	499	792	613	612	682	983	351	129	137	33	600	864	683	857
38	592	725	996	855	828	338	342	428	795	742	37	506	767	53	793	505	146	609	924	854	465	239	206	209	492	499	175	36	61	647	618	13	727
39	832	858	996	917	290	967	707	234	246	462	402	239	426	95	906	314	350	586	279	776	822	879	301	812	676	164	357	570	38	312	144	561	180
40	714	649	595	522	541	416	377	52	565	891	138	310	967	55	787	224	222	439	441	648	804	834	368	441	872	668	974	176	137	633	241	745	18
41	315	36	361	760	984	729	534	50	82	806	936	584	790	617	182	66	116	388	425	101	610	450	982	621	193	38	754	593	895	280	638	167	420
42	175	354	448	919	79	988	321	592	597	197	509	652	71	658	122	859	997	999	899	120	320	36	984	912	915	953	485	741	199	567	208	844	169
43	219	781	990	833	672	563	302	502	928	721	898	447	945	406	222	686	794	577	844	683	547	240	485	464	723	337	653	168	130	255	466	858	512
44	85	713	935	735	883	186	96	648	85	752	521	260	982	636	688	919	236	792	65	278	198	496	386	11	976	547	121	202	245	702	564	539	572
45	53	708	508	715	429	66	689	663	771	541	259	631	572	514	966	988	170	168	199	655	850	444	936	278	406	427	313	523	371	327	73	461	173
46	57	95	174	38	183	658	295	722	983	231	503	232	300	590	480	408	505	702	858	818	557	561	120	28	656	713	332	92	182	467	320	358	74
47	644	436	669	378	676	858	952	846	137	915	988	616	288	382	74	333	915	56	656	62	616	524	129	442	984	872	49	396	26	582	202	775	416
48	437	695	179	661	144	659	961	907	576	472	645	592	972	628	196	811	535	835	229	716	532	956	158	749	620	296	724	120	526	406	983	445	574
49	587	732	638	165	64	518	399	32	508	896	438	978	638	245	59	169	744	362	769	49	518	467	558	143	61	603	190	186	226	603	4	300	768
50	457	405	365	334	49	94	592	270	119	17	313	253	117	977	160	505	153	591	562	576	854	72	134	16	50	193	945	852	747	521	61	344	259
51	849	442	606	850	287	40	769	223	588	467	83	400	158	606	938	463	920	366	758	685	23	229	298	356	470	175	196	683	988	619	946	190	275
52	245	726	201	424	83	389	735	24	895	745	170	483	717	392	752	324	773	740	255	468	70	843	30	756	765	114	980	458	358	124	8	311	667
53	695	525	629	851	788	794	744	706	75	71	490	884	819	979	195	158	807	845	374	438	156	820	432	766	222	957	553	271	955	389	625	29	128
54	383	720	144	388	77	213	202	658	202	753	249	58	575	90	638	823	395	701	205	558	123	287	41	26	530	209	820	935	43	531	674	902	100
55	38	915	672	352	862	421	667	128	465	329	259	82	389	986	419	735	140	666	632	485	822	3	161	896	141	150	432	872	361	770	313	165	645
56	256	140	48	655	807	449	491	760	382	836	219	270	679	725	519	809	701	631	324	807	725	301	283	969	488	164	963	124	534	770	851	997	724
57	503	472	20	932	953	824	33	57	665	732	280	167	966	690	868	926	828	479	144	703	449	833	3	910	240	27	610	527	282	393	187	863	961
58	812	641	304	496	162	537	551	851	732	565	938	445	228	194	526	178	745	928	205	149	257	312	401	199	727	692	353	795	542	567	149	376	255
59	299	489	674	836	612	387	957	503	491	482	829	27	361	952	496	646	183	597	554	32	102	945	476	224	81	195	623	632	343	602	369	282	989
60	265	370	336	525	746	644	512	482	815	105	726	201	773	306	80	593	990	591	214	223	857	700	849	322	285	762	702	268	374	152	848	235	638
61	124	31	66	429	722	556	818	875	692	910	361	132	728	522	149	277	826	348	278	777	338	175	944	125	139	821	340	640	966	81	148	547	951
62	759	842	440	89	494	642	620	486	146	964	486	271	943	466	474	445	788	349	827	954	504	281	246	757	285	114	350	832	673	943	312	232	883
63	51	736	755	174	477	154	729	882	851	769	237	301	676	604	73	161	197	595	125	485	62	229	560	718	73	79	709	662	224	574	682	475	815
64	841	167	520	560	47	581	382	606	989	784	403	960	462	732	392	843	63	923	611	740	692	443	270	919	849	112	576	845	486	556	603	364	498
65	452	170	978	766	704	416	649	377	837	537	459	432	955	105	526	937	366	914	155	290	446	182	635	93	936	846	238	456	403	714	599	41	925
66	275	221	576	799	919	241	611	712	209	635	263	830	514	336	783	32	988	339	705	670	940	868	391	728	985	537	320	124	997	354	703	494	562

Таблица VII

Нижне-левый квадрат таблицы исходных данных: строки от 67 до 99, столбцы от 1 до 33.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
67	865	554	700	302	714	629	794	26	463	528	748	679	351	199	3	442	280	16	290	98	858	730	553	551	262	444	649	485	55	572	889	858	552
68	320	188	821	264	3	754	722	316	854	586	425	524	818	302	120	659	488	682	341	574	500	566	222	754	552	850	516	226	486	602	737	901	901
69	912	141	209	450	180	569	535	562	365	111	817	11	763	321	606	345	443	850	593	664	442	444	833	736	502	25	559	953	7	6	543	129	751
70	164	58	511	383	846	943	737	831	655	251	351	290	52	295	420	61	360	767	273	34	133	191	301	378	387	879	500	615	709	72	61	952	970
71	162	685	115	321	502	541	804	573	733	452	811	229	224	159	983	665	845	950	826	591	897	213	371	986	662	190	584	906	418	622	719	666	218
72	638	80	620	545	929	168	141	762	834	633	270	82	0	560	364	141	99	336	665	833	29	827	254	28	319	984	110	168	417	397	125	406	410
73	139	898	973	638	815	2	41	485	63	211	66	827	583	930	565	82	982	831	235	290	812	256	802	448	513	126	497	64	215	293	907	437	418
74	304	990	843	557	857	319	875	149	190	946	798	467	346	509	80	479	981	207	719	138	425	699	770	14	510	355	232	765	618	720	237	264	709
75	552	874	977	697	370	941	267	341	617	968	65	321	655	727	914	283	137	932	549	162	882	35	592	135	930	534	520	728	688	724	634	233	603
76	297	433	140	622	545	540	727	250	82	247	99	50	730	396	304	331	805	51	82	139	767	370	676	143	556	973	116	957	286	366	433	843	672
77	472	670	161	15	825	729	494	37	195	830	549	300	701	794	561	303	992	959	948	206	613	969	639	455	561	651	410	677	576	539	222	502	224
78	833	574	688	884	668	468	425	204	844	921	960	911	922	951	700	342	630	425	430	451	720	198	527	480	405	382	425	614	612	204	192	9	938
79	401	796	172	109	731	987	670	525	184	802	727	746	688	352	197	989	895	398	951	18	373	257	719	873	915	68	883	610	859	125	596	839	583
80	847	14	914	212	507	622	327	492	320	999	898	271	386	192	552	217	524	988	693	89	688	414	296	268	346	358	737	745	568	797	825	647	221
81	773	894	419	526	883	533	782	512	436	825	52	812	24	845	296	202	773	758	305	677	358	111	281	650	570	980	253	327	359	716	797	118	718
82	591	28	661	612	696	876	100	958	171	191	248	122	698	550	758	728	145	742	438	860	473	477	493	695	404	213	32	525	904	295	221	848	168
83	477	714	846	475	82	831	839	619	169	647	362	406	560	752	905	561	720	380	441	550	339	738	124	951	5	940	989	625	482	104	653	380	430
84	592	413	808	52	41	791	576	924	657	708	946	958	632	263	627	831	441	679	637	100	637	909	750	654	540	597	827	979	46	139	838	81	898
85	93	444	809	507	40	523	222	647	150	225	262	382	969	740	1	855	305	526	287	863	979	966	478	345	209	886	906	720	643	606	442	361	983
86	133	511	976	305	665	206	98	418	619	791	24	467	850	934	495	975	75	694	787	465	762	639	508	460	682	586	495	122	935	948	559	119	138
87	807	809	330	91	111	544	344	75	689	205	756	724	217	856	48	760	280	650	88	966	67	778	173	843	48	932	576	763	598	335	616	857	980
88	834	101	205	900	732	456	636	210	917	593	477	792	910	990	408	775	889	69	391	139	835	306	39	173	69	573	444	827	195	336	395	102	316
89	92	693	199	32	849	873	343	933	444	856	486	25	364	0	332	22	145	53	709	895	747	508	319	634	136	123	244	982	490	375	546	439	48
90	978	306	624	115	473	160	678	336	719	608	188	782	321	616	223	712	186	170	766	42	268	458	876	662	946	981	753	704	838	597	322	595	911
91	32	223	787	178	739	503	137	890	463	177	801	203	913	830	314	99	998	103	598	117	790	868	374	692	448	106	184	848	926	567	235	407	977
92	734	452	275	541	925	201	855	304	378	391	433	260	725	826	125	690	189	707	935	847	719	883	614	428	313	48	26	640	727	709	466	565	965
93	469	924	386	972	802	652	471	580	925	380	708	599	137	867	728	812	831	684	293	658	140	887	688	77	559	959	530	630	676	140	638	391	860
94	872	578	289	772	693	658	999	478	366	610	642	995	567	577	854	454	992	602	226	994	45	374	548	883	444	881	975	670	847	318	98	967	594
95	380	585	702	73	653	594	227	388	954	802	866	168	147	371	137	633	312	862	320	101	555	685	86	117	741	763	648	932	667	953	146	213	594
96	585	529	584	447	868	939	999	615	522	242	712	707	631	848	454	971	490	209	458	925	727	868	964	555	905	289	207	355	396	323	341	49	634
97	415	741	56	609	19	827	862	248	779	343	731	751	251	96	498	979	138	306	991	301	413	781	451	469	286	277	142	541	355	435	658	799	961
98	143	916	28	661	790	424	741	447	347	719	81	382	489	105	403	749	390	991	168	545	317	345	167	270	538	986	925	202	182	95	628	255	396
99	354	942	643	500	150	866	209	534	398	500	182	916	685	693	253	924	621	248	111	275	43	103	623	116	723	285	203	712	117	84	469	807	966

Таблица VIII

Нижне-средний квадрат таблицы исходных данных: строки от 67 до 99, столбцы от 34 до 66.

	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66
67	129	640	860	46	627	44	422	936	473	154	333	756	51	391	534	931	322	79	202	176	580	269	388	429	885	2	610	327	274	760	99	42	55
68	749	967	102	265	881	843	710	508	489	858	184	121	106	586	987	371	490	972	518	73	353	281	662	996	213	451	115	804	785	333	564	504	837
69	70	395	228	541	59	220	715	944	178	658	384	738	383	638	737	635	30	782	315	80	21	425	752	204	446	412	4	45	825	244	549	994	529
70	823	64	537	50	610	791	978	443	161	821	718	298	183	933	695	709	72	468	703	864	845	849	50	646	142	444	155	459	768	885	301	768	847
71	991	959	566	103	279	498	867	345	103	430	20	25	534	821	833	235	287	393	827	348	624	692	918	800	244	431	86	695	508	753	597	192	121
72	463	48	690	969	344	599	643	155	464	537	558	206	911	414	947	293	129	587	80	85	764	234	76	15	118	384	847	648	547	555	832	781	837
73	338	424	70	183	345	328	59	483	221	537	560	349	923	883	905	847	313	786	984	409	234	500	970	839	157	668	856	485	592	22	733	735	383
74	703	544	50	541	301	344	427	141	910	696	824	683	98	640	131	570	754	535	903	445	609	414	32	917	630	318	432	413	90	461	132	635	517
75	125	122	782	760	65	930	264	423	322	486	908	638	631	565	426	362	563	378	419	851	741	315	76	168	521	98	587	531	239	339	552	830	474
76	814	298	471	937	261	809	920	586	327	441	53	323	528	491	597	99	881	823	310	671	578	993	90	653	234	266	247	113	422	745	974	488	22
77	196	849	23	912	723	714	371	808	620	981	156	298	149	145	497	466	443	448	392	531	204	695	294	450	149	951	581	598	845	675	245	584	433
78	5	866	666	171	80	964	541	26	886	935	733	251	724	76	292	159	951	596	527	807	287	27	920	805	127	837	710	583	803	137	483	341	635
79	518	159	518	61	167	603	166	938	226	193	882	876	760	738	557	780	554	78	61	868	521	673	98	757	258	419	345	901	189	216	956	239	725
80	237	643	907	670	549	546	39	252	12	20	431	476	691	525	845	99	416	774	56	677	872	736	8	896	309	132	358	788	744	4	930	727	649
81	777	381	65	152	713	371	176	586	874	242	241	509	97	662	248	242	148	592	536	620	617	229	624	781	638	706	561	350	547	458	39	905	981
82	84	620	91	838	232	119	337	566	432	262	534	831	637	694	102	11	514	642	269	834	547	730	503	41	606	411	297	737	102	15	63	117	317
83	611	606	58	787	51	327	430	622	747	407	296	714	205	949	406	810	515	935	588	785	334	913	23	476	963	177	726	922	707	529	266	627	425
84	792	221	697	293	813	206	929	693	351	816	194	238	988	675	971	421	980	683	966	99	756	609	466	668	359	395	184	928	94	332	866	458	266
85	938	817	795	709	799	497	795	742	252	367	140	103	855	755	52	856	445	262	997	650	789	65	705	127	447	427	503	276	265	675	246	781	803
86	466	373	750	60	341	93	548	406	918	268	212	65	619	127	571	114	42	750	749	728	261	977	736	25	606	465	783	953	405	626	153	165	545
87	166	661	491	933	929	85	250	891	693	288	288	786	57	559	827	910	226	290	241	264	666	74	692	148	412	892	216	158	827	159	42	839	163
88	7	22	764	948	8	772	8	790	157	47	831	86	696	906	467	814	190	467	605	511	383	653	439	826	942	929	440	527	540	886	443	47	153
89	565	962	168	719	469	982	113	287	340	766	633	290	321	636	677	995	906	463	249	620	56	133	566	955	57	421	461	501	660	916	163	738	378
90	115	387	221	237	604	578	640	864	566	694	931	365	756	184	65	106	265	0	984	389	402	411	957	807	407	227	686	943	35	112	231	677	49
91	471	205	30	920	845	458	759	598	42	808	277	66	633	505	278	290	311	440	17	402	869	543	848	226	760	57	863	646	490	668	526	386	610
92	887	473	976	750	653	953	229	509	343	600	771	611	778	763	188	568	2	960	968	373	666	360	418	385	1	485	619	946	613	696	389	894	804
93	276	60	94	328	437	53	385	139	927	728	196	304	368	396	975	813	183	391	148	646	333	559	127	773	601	207	495	552	835	460	652	171	595
94	572	67	138	991	789	268	460	512	300	812	434	677	575	617	890	47	990	340	705	484	643	814	564	500	298	682	875	370	923	498	51	864	677
95	707	75	266	729	482	508	802	494	571	506	698	955	144	444	224	778	101	31	740	303	925	528	559	589	270	849	912	788	563	400	541	514	505
96	784	426	786	611	822	869	258	411	800	804	985	503	362	608	922	40	401	937	12	347	919	91	148	160	405	425	90	330	281	888	795	638	411
97	380	374	584	538	113	656	593	373	671	31	252	253	399	623	100	36	600	366	192	546	164	303	281	952	809	933	722	482	89	242	704	869	940
98	720	341	813	546	278	27	903	724	686	986	365	397	868	616	218	344	604	163	879	15	372	843	807	217	326	17	180	57	869	536	458	700	622
99	104	584	381	65	372	630	560	87	715	945	993	621	842	465	249	158	148	766	70	496	693	703	967	683	460	714	444	415	714	924	961	725	200

Таблица IX

Нижне-правый квадрат таблицы исходных данных: строки от 34 до 66, столбцы от 34 до 66.

	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99
67	###	393	65	928	572	284	726	798	523	201	587	88	997	652	175	750	678	455	893	630	406	531	489	165	521	127	964	440	113	389	710	488	763
68	453	###	784	333	972	921	544	370	257	88	129	297	340	923	901	343	876	731	500	648	613	253	311	693	332	451	814	599	216	212	834	849	252
69	132	9	###	316	429	761	133	930	746	532	283	873	439	880	835	987	570	589	629	560	521	597	189	163	444	621	705	815	713	334	999	137	615
70	606	187	535	###	309	364	633	351	140	625	724	346	200	838	31	221	217	624	584	815	344	352	19	70	367	919	174	487	30	345	716	44	699
71	819	907	71	22	###	561	991	52	251	523	965	956	980	61	465	523	300	828	235	953	612	682	876	157	289	292	199	776	461	842	131	660	700
72	427	896	859	324	799	###	653	701	841	404	729	215	565	650	532	83	719	296	439	778	238	241	803	6	620	670	849	205	524	78	787	984	629
73	827	499	61	90	135	100	###	479	573	906	885	245	209	836	981	939	699	101	407	195	219	84	427	355	560	133	452	800	603	688	398	875	569
74	645	321	818	805	333	721	246	###	970	788	394	8	227	835	577	675	205	50	642	254	318	390	724	979	365	754	171	22	844	267	509	410	872
75	781	742	118	900	700	234	606	680	###	790	923	399	168	485	634	455	544	444	708	800	219	518	349	947	580	299	448	684	119	132	340	888	770
76	377	519	83	966	805	794	646	324	441	###	89	98	797	66	460	391	573	465	605	806	438	396	518	262	944	726	15	893	323	569	124	218	369
77	100	399	567	297	9	941	231	99	898	974	###	387	759	262	145	729	659	324	767	966	968	927	94	849	796	542	923	396	417	411	368	48	781
78	621	277	278	80	434	711	229	853	930	408	730	###	448	632	141	670	416	63	818	815	984	509	974	741	885	408	960	748	85	778	101	171	505
79	125	532	178	421	716	762	287	208	738	530	605	490	###	799	569	407	253	413	656	785	583	348	593	799	362	606	666	735	219	671	664	599	608
80	519	868	174	901	361	255	966	927	313	362	780	818	728	###	289	713	941	223	796	731	840	231	135	632	582	382	250	644	852	457	601	268	33
81	468	414	601	316	294	967	45	167	39	163	600	774	444	999	###	525	80	217	824	449	86	674	257	185	215	58	275	230	582	211	37	270	962
82	999	151	643	538	782	454	887	12	912	922	837	174	91	808	718	###	851	277	623	852	211	230	589	1	15	487	728	29	921	987	863	327	488
83	900	248	975	530	757	413	96	16	92	905	251	807	730	703	547	452	###	255	684	535	594	146	790	36	47	505	607	278	958	281	746	299	422
84	463	913	431	286	360	166	449	122	260	883	173	966	706	946	474	708	346	###	670	820	266	69	378	927	38	884	359	767	978	30	921	191	890
85	815	263	760	316	581	777	533	828	784	703	444	476	374	108	535	402	935	441	###	786	393	320	414	944	857	912	304	496	691	706	826	216	583
86	127	780	431	380	183	347	723	81	63	544	249	426	440	662	291	29	698	598	978	###	727	825	460	269	180	223	347	402	165	82	423	55	187
87	448	396	250	896	196	678	108	486	437	303	323	1	41	845	630	94	400	993	54	394	###	204	542	124	569	581	643	827	201	502	23	38	561
88	470	610	605	709	971	471	242	332	842	420	225	881	718	588	351	252	857	865	975	430	351	###	102	750	788	747	72	290	219	717	993	760	756
89	397	334	588	499	470	199	115	366	465	232	874	78	337	800	848	993	958	757	671	58	682	617	###	65	706	820	312	204	602	197	136	643	116
90	206	401	435	546	959	576	811	708	90	164	597	725	107	883	415	377	94	380	355	168	934	383	611	###	316	403	948	200	41	660	273	723	189
91	649	339	813	162	358	220	437	815	875	372	876	390	429	555	10	77	784	275	435	976	840	48	529	190	###	633	367	637	323	445	3	994	86
92	119	219	627	88	461	295	652	643	90	747	965	124	284	525	953	792	110	197	497	805	112	856	256	730	231	###	34	501	171	438	656	265	236
93	720	273	88	467	375	163	612	474	649	930	628	395	583	833	127	45	0	985	990	772	638	771	793	696	373	61	###	908	716	300	739	827	497
94	23	260	764	117	467	770	919	855	297	727	665	300	888	969	813	398	244	359	850	456	605	810	428	913	420	882	32	###	266	387	631	464	501
95	44	711	375	284	106	258	397	394	159	123	415	96	822	258	873	876	396	706	105	795	387	143	141	139	388	839	277	372	###	752	396	682	771
96	984	647	655	371	502	100	761	717	965	539	501	423	815	409	53	2	431	585	89	807	944	535	264	812	627	984	755	95	885	###	449	238	755
97	136	65	805	612	672	826	955	625	210	156	388	646	868	614	413	796	726	65	85	181	829	256	188	716	838	998	915	499	222	733	###	468	629
98	274	835	828	914	958	899	427	592	921	340	89	536	43	493	146	21	828	94	218	802	943	496	27	217	85	403	672	776	685	711	66	###	254
99	667	772	249	817	218	813	358	379	469	86	475	724	246	187	517	282	71	340	501	235	625	273	947	6	388	917	243	338	184	957	639	94	###

```cpp
bool Task::AddSubTask(SubTask* st, bool bSimple) {
    // если bSimple==false, то идёт "более сложный" алгоритм вставки подзадачи; см. текст;
    // в этом варианте всегда подзадачу добавляем (не теряем) ⇒ возврат всегда true
    if (SolveSubTask(st)) return true;  // при этом переполнение массива подзадач невозможно
    int nNewGran = st->GetGran();       // граница только что решённой ("доре́шенной") подзадачи
    if (nNewGran >= nOpt) { delete st; return true; }
                                        // обычное сравнение гранцы с текущим оптимумом
    // далее будем добавлять рассматриваемую подзадачу в массив;
    // при этом считаем, что последняя подзадача в массиве заведомо не приведёт к оптимуму
    if (nKol>=NDimArrSub) {
        delete Zadachi[NDimArrSub-1];   // удаляем последнюю подзадачу массива
        nKol = NDimArrSub-1;
    }
    // далее с nNewGran не работаем; место неё nNewKolb, на основе нескольких предикторов
    int nNewKolbas = st->GetKolbas();
    int IndNew = 0;                     // индекс для вставки подзадач в массив
    if (!bSimple) for (int i=0; i<nKol; i++) {  // индекс IndNew, возможно, изменится
        if (nNewKolbas <= Zadachi[i]->GetKolbas()) break;
        IndNew = i+1;
    }
    for (int i=nKol; i>IndNew; i--) Zadachi[i] = Zadachi[i-1];
    nKol++;
    Zadachi[IndNew] = st;
    return true;
}
```

Рис. 3. Метод `Task::AddSubTask()` – текущая версия.

```cpp
void Task::Run() {
    for (long long nnn=1; nnn<=NKolSteps; nnn++) {
        SubTask* ST1 = ExtractFirst();
        int I,J; if (!ST1->BestNull(I,J)) { cout << endl << "неожиданный конец"; break; }
        SubTask* ST2 = ST1->MakeRight(I,J);
        if (!AddSubTask(ST1,(ST1->GetDim()<=NTurbo)) || !AddSubTask(ST2,true))
            cout << "не удалось сохранить подзадачу" << endl;
        rez.NewIter(nKol);
            // отметили, что прошла новая итерация с возможными новыми оптимумами
        cout << *this << endl;
        if (nKol>0) continue;
        cout << rez;
        cout << endl << "естественный конец решения" << endl; break;
    }
    cout << endl << "конец решения в связи с ограничением числа ветвлений" << endl;
}
```

Рис. 4. Метод `Task::Run()` – текущая версия.

```
SubTask* SubTask::MakeRight(int nXdel, int nYdel) {
    // 1) сначала "на пустом месте" делаем правую
    SubTask* Return = new SubTask(nDim-1);
    Return->SetGran(nGran);
    // 1a) установка №№ строк и столбцов
    for (int i=1; i<=nDim; i++) {
        if (i==nXdel) continue;
        int iNew = i<nXdel ? i : i-1;
        Return->Lin->Set(iNew,Lin->Get(i));
    }
    for (int j=1; j<=nDim; j++) {
        if (j==nYdel) continue;
        int jNew = j<nYdel ? j : j-1;
        Return->Col->Set(jNew,Col->Get(j));
    }
    // 1b) установка самих значений
    for (int i=1; i<=nDim; i++) {
        if (i==nXdel) continue;
        int iNew = i<nXdel ? i : i-1;
        for (int j=1; j<=nDim; j++) {
            if (j==nYdel) continue;
            int jNew = j<nYdel ? j : j-1;
            Return->Set(iNew,jNew,Get(i,j));
        }
    }
    // 1c) установка старых путей
    Return->Next->InitCopy(Next);
    Return->Prev->InitCopy(Prev);
    // 1d) установка в них нового движения
    int III = Lin->Get(nXdel), JJJ = Col->Get(nYdel);
    Return->Next->Set(III,JJJ);
    Return->Prev->Set(JJJ,III);
    // 1e) установка "обратных бесконечностей" (объединяем ниже со следующим пунктом)
    // 1f) если не все поля Next'а заполнены, то установка "дальнейших обратных бесконечностей"
    if (nDim>=3) { // т.е. обходим, если размерность подзадачи Return мала
        int JJJJJ = JJJ;
        for (;;) {
            if (!Return->SetInftyByNumbers(III,JJJJJ)) break;
            JJJJJ = Return->Next->Get(JJJJJ);
            if (JJJJJ<=0) break;
        }
        int IIIII = III;
        for (;;) {
            if (!Return->SetInftyByNumbers(IIIII,JJJ)) break;
            IIIII = Return->Prev->Get(IIIII);
            if (IIIII<=0) break;
        }
    }
    // 1g) редукция (здесь - полная)
    Return->Reduction();
    // 2) теперь "из себя" делаем левую подзадачу
    Set(nXdel,nYdel,NInfty);
    ReductionLin(nXdel);
    ReductionCol(nYdel);
    // 3) конец работы функции
    return Return;
}
```

Рис. 5. Метод `SubTask::MakeRight()` – текущая версия.

Таблица X
Тур для псевдооптимального решения (равного 1681)
для приведённой выше матрицы коммивояжёра

	0	1	2	3	4	5	6	7	8	9
0		78	77	84	85	80	37	62	18	39
10	35	97	87	95	33	79	14	69	65	10
20	12	36	61	71	52	70	91	59	98	7
30	25	2	75	11	23	8	17	60	55	30
40	99	1	88	9	38	42	50	31	34	63
50	76	72	49	46	26	3	28	92	47	86
60	66	68	29	19	24	45	41	15	5	54
70	89	44	13	6	94	22	93	4	32	20
80	56	73	27	74	96	48	57	64	40	16
90	51	81	58	83	21	67	82	43	53	90

- далее в течение длительного времени работы работы МВГ (до примерно 97 % от общего времени работы) это решение продолжает сохраняться в качестве текущего псевдооптимального;
- однако оптимальный ответ соответствует решению 1679; такое решение приведено в таблице XI [16];

Таблица XI
Тур для оптимального решения (равного 1679)
для приведённой выше матрицы коммивояжёра

	0	1	2	3	4	5	6	7	8	9
0		78	77	84	85	80	37	62	32	39
10	35	97	87	95	33	79	52	69	65	10
20	12	36	61	71	29	70	91	59	98	7
30	25	2	75	11	23	8	17	60	55	30
40	99	16	88	9	38	42	50	31	26	63
50	76	72	21	58	1	3	28	92	46	86
60	66	68	47	19	24	45	41	18	5	54
70	89	44	13	6	94	22	93	4	34	20
80	56	73	49	74	96	15	57	64	40	14
90	48	81	27	83	67	51	82	43	53	90

- как будет видно из приведённых далее результатов, даже при наличии только одного предиктора (т. е., немного упрощая ситуацию, «для старого варианта вычислений») подобные «плохие» исходные матрицы появляются весьма редко: в наших вычислениях не более чем в 2 % случаев.

По-видимому, всё сказанное про приведённый частный случай ЗКВ косвенно свидетельствует о том, что при практически любых вариантах применения МВГ оптимальное решение «спрятано глубоко в левых подзадачах».

Но, конечно, существенно более интересны – а также более связаны с основным материалом статьи – вычисления, относящиеся к применению двух предикторов. Повторим, что мы получили *только первые удачные результаты* такого применения – но даже они свидетельствуют о том, что необходимо продолжать работу в этом направлении.

[16] Отметим, немного забегая вперёд, что примерно одинаковые временны́е результаты – аналогичные описанным здесь – были получены для обоих рассматривавшихся вариантов применения МВГ: с одним предиктором и с двумя.

При этом нужно отметить следующие два факта.

- Этот второй предиктор на самом деле тривиален: как и следовало ожидать, он является размерностью (под) задачи.
- Мы *пока* совершенно не проводили самообучения – рассмотрев для второго предиктора только эмпирически взятый коэффициент.

Поэтому, поскольку

- итоговые решения ЗКВ для рассматриваемой размерности 99 обычно получаются несколько более 1600,
- а «наиболее интересные вычисления» идут для подзадач, размерность которых примерно равна 50, –

мы для примерного равенства этих показателей и заданного коэффициента 1.0 для основного предиктора выбрали для второго предиктора коэффициент 25.0.

Вычисления были организованы следующим образом. Выбирались:

- *размерность массива подзадач* (точнее – указателей на подзадачи): 100 000, 200 000 или 300 000 (при переполнении этого массива (под) задача «теряется» – но, поскольку она «худшая на данный момент», то мы считаем, что её «потеря» приведёт к потере оптимального решения с приемлемо малой вероятностью);
- *ограничение на общее число шагов (ветвлений)* метода ветвей и границ: 1 000 000, 2 000 000 или 3 000 000.

Для минимальной пары значений – 100 000 и 1 000 000 – мы, аналогично [33], [34] получили, что среднее время вычисления подзадачи на среднем современном компьютере [17] составляло около 30 — 40 минут – что существенно меньше приведённого в [34] медианного значения [18].

Однако при достижении ограничения на число шагов и *неполучении* при этом оптимального решения (т. е. незавершении задачи) мы вычисления продолжали – всё-таки ограничивая *окончательное* общее число шагов (ветвлений) значением 10 000 000; это значение числа шагов в вычислениях ни разу достигнуто не было. Мы фиксировали случаи, когда после выбранного нами *предварительного* ограничения на число шагов впоследствии *было получено улучшение* решения задачи коммивояжёра; количества этих улучшений приведены в следующей таблице – которую можно назвать *основным результатом настоящей статьи*.

Таблица XII
Число матриц (из 100 случайно сгенерированных), для которых *не было* получено оптимального решения (слева – 1 предиктор, справа – 2 предиктора). Подробности в тексте статьи.

	1000	2000	3000
100	2	2	2
200	2	2	1
300	2	1	1

	1000	2000	3000
100	2	2	2
200	2	2	0
300	1	0	0

Организация таблицы следующая.

[17] `Intel(R) Core(TM) i7-8700 CPU @ 3.20GHz`.

[18] Там, однако, использовалось большее значение размерности массива подзадач.

- Левая часть таблицы XII относится к варианту с применением одного предиктора, правая – двух; при этом, как было сказано выше, в вычислениях значение второго предиктора шло с коэффициентом 25 [19].
- Все значения – пометки строк (размерность массива подзадач) и столбцов (ограничение на общее число шагов) приведены «в тысячах» (т. е. с коэффициентом 1 000, например, 2 000 означает 2 000 000.
- В клетках таблицы (конечно, без коэффициента) – число матриц (из 100 случайно сгенерированных, причём одинаковых для обоих вариантов вычисления), для которых после рассматриваемого числа шагов *не было* получено оптимального решения [20].

Итак, даже для столь простого варианта применения двух предикторов и столь простой организации вычислений – мы для варианта с применением второго предиктора получили относительное улучшение работы алгоритма.

V. КРАТКОЕ ОПИСАНИЕ ПРОГРАММ ДЛЯ ИСПОЛЬЗОВАННЫХ АЛГОРИТМОВ

Как было отмечено во введении, в этом разделе мы ориентируемся прежде всего на описание тех использованных подалгоритмов (вспомогательных алгоритмов), которые нужны именно для получения удачных результатов вычислений. Конкретно – мы привели выше, на рисунках 3 – 5, текущие варианты тех функций, которые подверглись изменениям по сравнению с вариантами, приведёнными в [34]. Однако сами пояснения (дополнительно к комментариям в текстах функций) мы приводим очень близко к [34]: иначе программы понять вряд ли возможно.

Перед описаниями трёх самых важных методов отметим, что заголовки содержащих их классов (класса-задачи `Task()` и класса-подзадачи `SubTask()`) полностью совпадают с приведёнными в [34] [21] – и поэтому мы здесь повторять их не будем.

Итак, во-первых – метод для добавления подзадачи в массив (рис. 3), `MakeRight()`. Здесь нужны такие комментарии.

- Решаемая задача передаётся «на указателе» первым параметром.
- В случае, когда после окончания работы метода нам рассматриваемая подзадача больше не нужна – мы её удаляем деструктором. А иначе – включаем её обратно в массив подзадач, т. е. соответствующий ей указатель «не пропадает». Таким образом, при любом варианте развития событий «дыр» в динамической памяти не будет.
- Рассматриваемый метод `AddSubTask()` возвращает `true` тогда и только тогда, когда после его выполнения не произошло переполнение массива подзадач (т. е. он завершился успешно). В частности, в случае дорешивания подзадачи такого переполнения быть не может.
- Цикл, не выполняющийся в случае соответствующего ответа на проверку условия `if (!bSimple)` (значение `bSimple` передаётся вторым параметром метода) – возможно, самая важная эвристика, «относящаяся к первому (основному) предиктору»: как показывают вычислительные эксперименты, она существенно улучшает общее время решения по сравнению с [25]. Точнее, таковой является не сама эвристика, а её применение, зависящее от варианта вызова в методе `Run()` (который мы кратко рассмотрим далее).
- Практически такой же вариант работы (такой же предиктор) применяется и для построения ППЗ.

Во-вторых – основной метод класса-подзадачи («интерфейсный»), `Run()`, рис. 4. Именно в нём вызывается `MakeRight()`, после чего идёт обработка получившихся в результате подзадач. Отметим, что всегда равный значению `true` второй параметр при вызове функции обработки получившейся правой подзадачи отражает принятый нами вариант построения ППЗ.

В-третьих – самый важный метод; причём мы считаем, что он является самым важным не только для рассматриваемого класса-подзадачи `SubTask`, но и для всего описываемого проекта. Более того, можно сказать, что подобный метод – самый важный (ключевой) для всего подхода к реализации МВГ, т. е. для любой задачи дискретной оптимизации, а не только для ЗКВ.

Текст этого метода `MakeRight()` мало отличается от соответствующего из [34] – однако важные отличия всё-таки есть. Поэтому мы приводим модифицированный текст метода (рис. 5), повторяя с небольшими изменениями и добавлениями некоторые комментарии из предыдущей статьи.

- Параметры метода вычислены заранее – именно предиктором (или «линейной комбинацией» двух предикторов). Они указывают на применяемый разделяющий элемент.
- Возвращать построенную подзадачу будем по указателю `Return` – а поэтому для него сразу применяем оператор `new`.
- Пункт 1 – так в комментариях обозначено создание («рождение») правой подзадачи, он состоит из нескольких подпунктов – вспомогательных алгоритмов, которые в комментариях озаглавлены 1a, …, 1f.
- Подпункты 1a, 1b и 1c, по-видимому, дополнительных комментариев не требуют.
- Подпункт 1d – это запись того, что в правой подзадаче становится обязательной поездка между выбранной парой городов. Но ведь города выбирались по номерам строк / столбцов текущей матрицы (также см. [34]) – поэтому в этом методе мы выбираем эти номера из соответствующих массивов (`Lin` и `Col`). Далее проставляем в массивах `Next` и `Prev` следующий город для номера `III` и предыдущий город для номера `JJJ`.
- Подпункт 1e – это вызов установки замены т. н. «серого нуля» ([9], [34]) на бесконечность.
- А подпункт 1f – это поиск таких возможных «серых нулей», которые получаются как возможное замы-

[19] В связи с неприменением самообучения пока мы использовали целое значение; немного забегая вперёд отметим, что в приведённой компьютерной программе это значение вычисляется в методе `GetKolbas()`, причём этот метод относится к классу-подзадаче.

[20] Как можно понять из сказанного выше, мы условно считали, что для рассматриваемой размерности исходной задачи такое оптимальное решение должно обязательно получиться до достижения общего числа ветвлений (шагов МВГ), равного 10 000 000.

[21] По крайней мере, они полностью совпадают для интересующих нас методов.

кание «малых циклов» (вспомогательный алгоритм, кратко упомянутый в [9], но при этом отсутствовавший в [25]); впрочем, без этого вспомогательного алгоритма можно и обойтись – поскольку отсутствие подобных «малых циклов» автоматически проверяется и перед рассмотрением решения как допустимого и проверки его на псевдооптимальность, см. более подробное описание предыдущего варианта класса `Task` в [34].
- Подпункт 1g – редукция полученной правой подзадачи.
- Пункт 2 – окончательное оформление «себя» в качестве левой подзадачи (новой задачи); также в пункте 2 выполняется редукция полученной левой подзадачи по строке и столбце.

VI. ЗАКЛЮЧЕНИЕ

Ещё раз повторим, что мы получили только первые удачные результаты применения нескольких предикторов – но даже такие результаты свидетельствуют о том, что необходимо продолжать работу в этом направлении. Вообще, настоящая статья предполагает несколько возможных вариантов продолжения – причём некоторые из описываемых в этом разделе вариантов уже были кратко упомянуты выше по тексту статьи.

- «Полностью теоретическое» продолжение, по-видимому, только одно[22]: как было сказано выше, в случае успеха процедуры нахождения оптимального Гамильтонова цикла мы получаем достаточное условие того, что в рассматриваемой нами модели все направления связи сети связи имеют резервный маршрут. Возможное строгое доказательство необходимости представляется важным предметом дальнейшего исследования.

В приведённом пункте было описано направление, связанное со строгим доказательством; а все остальные направления возможной дальнейшей работы (приведённые ниже) связаны с возможным применением *эвристик*, а также со *статистическими* исследованиями работы рассматриваемых алгоритмов – а не с их математическим обоснованием.

- Как было отмечено выше, для уже сгенерированного частного случая ЗКВ часто бывает интересно решить задачу, которую можно назвать «обратной»: определить, *при каком именно* конкретном значении σ некоторый заданный (рассматриваемый) частный случай ЗКВ *мог быть сгенерирован с наибольшей вероятностью*, при этом соответствующую процедуру нужно выполнить с помощью какого-либо из не связанных с генерацией входных данных алгоритма. Возможный (но не единственный) вариант – метод максимального правдоподобия, [14, гл. 9, § 4] и мн. др.
- Отдельным пунктом приведём *возможность применения* результатов, полученных при исследовании пункта предыдущего. Это связано с применением разных алгоритмов для разных вариантов генерации входных данных – что можно найти в нескольких разных разделах главы 4 монографии [2]. Однако, конечно, исследования в этом направлении никогда не станут «завершёнными»: они свои для каждой новой рассматриваемой оптимизационной задачи.
- С двумя последними пунктами связана и такая задача. В настоящее время мы пока ещё не исследовали *адекватность модели*, связанной с использованием для генерации исходных данных псевдогеометрической версии некоторого конкретного значения σ – точнее сказать, её (модели) *адекватность для некоторой конкретной предметной области*. При этом понятно, что в первую очередь нас интересует такая адекватность для алгоритмов *генерации случайного графа*, описывающего некоторую сеть связи. Вероятно, здесь также можно применять метод максимального правдоподобия, но, конечно, возможны и другие варианты; некоторые из возможных подходов мы кратко описали в разделе с соответствующим названием («Возможный подход к генерации входных данных …») статьи [8], а также полностью в [35].
- Мы в статье употребляли *только два* предиктора. Поэтому в будущих работах желательно:
  - применить какой-либо другой предиктор – новый, не описанный в настоящей статье;
  - рассмотреть ситуации, когда число предикторов превышает 2;
- Также повторим, что предметом статьи является *самый простой вариант* употребления предикторов. Эта «простота» заключается в том, что мы пока используем их не для собственно выбора разделяющего элемента (что можно было бы назвать «априорным» алгоритмом) – а для «апостериорного» размещения двух полученных подзадач в массиве подзадач; это, по-видимому, существенно проще. Поэтому в будущих работах желательно осуществить такой упомянутый кратко «априорный» подход.
- Для каждого варианта алгоритма, кратко описанного в двух последних пунктах (в частности, в подпунктах), – организовать дополнительно какое-либо *самообучение*.
- Рассмотреть возможности применения нескольких предикторов – причём, конечно, не только «апостериорного», но и «априорного» подходов – и в других задачах дискретной оптимизации.
- Кроме того, в будущем мы предполагаем продолжить работы, в которых на каждом шаге МВГ окончательный выбор разделяющего элемента происходит путём применения т. н. функций риска. А именно, сначала на шаге МВГ вместо того, чтобы сразу «найти лучший нуль» мы «ранжируем нули» – выбирая среди них *несколько* «лучших нулей». Уже потом среди них с помощью *каких-то других* эвристик мы выбираем единственный – который и станет разделяющим элементом на этом шаге МВГ. При этом здесь возможно применение:
  - классических алгоритмов многокритериальной оптимизации – [31], [32] и др.;
  - функций риска – [29], [30];

[22] Отметим, что при любом описании эвристических алгоритмов в качестве возможного продолжения такой работы *неявно* всегда предполагается желательность получения строгого доказательства уменьшения времени работы алгоритма (либо улучшения значений какой-либо другой важной целевой функции) при применении этих эвристик. Однако, как правило, подобные описания улучшений удаётся получить *только для статистических показателей* – о которых ниже.

– специальных эвристик, определяющих «похожесть» получаемой матрицы на некоторую рассматривавшуюся ранее, – [27] и др.

БЛАГОДАРНОСТИ

Работа первого автора была частично поддержана грантом научной программы китайских университетов “Higher Education Stability Support Program” (раздел “Shenzhen 2022 – Science, Technology and Innovation Commission of Shenzhen Municipality”) – 深圳市 2022 年高等院校稳定支持计划资助项目.

Список литературы

[1] Громкович Ю.: Теоретическая информатика. Введение в теорию автоматов, теорию вычислимости, теорию сложности, теорию алгоритмов, рандомизацию, теорию связи и криптографию. – СПб., БХВ-Петербург. – 2010. – 336 с.

[2] Hromkovič J.: Algorithmics for Hard Problems. Introduction to Combinatorial Optimization, Randomization, Approximation, and Heuristics. – Berlin, Springer. – 2003. – 538 p.

[3] Макаркин С.Б., Мельников Б.Ф.: Геометрические методы решения псевдогеометрической версии задачи коммивояжера // Стохастическая оптимизация в информатике. 2013. Т. 6, № 2. С. 54–72.

[4] Макаркин С.Б., Мельников Б.Ф., Тренина М.А.: Применение проблемно-ориентированных метрик в геометрических алгоритмах решения псевдогеометрической версии задачи коммивояжера // Стохастическая оптимизация в информатике. 2014. Т. 10, № 1. С. 63–71.

[5] Макаркин С.Б., Мельников Б.Ф., Тренина М.А.: Подход к решению псевдогеометрической версии задачи коммивояжера // Известия высших учебных заведений. Поволжский регион. Физико-математические науки. 2015. № 2 (34). С. 135–147.

[6] Мельников Б.Ф., Давыдова Е.В.: Математическое моделирование повышения уровня безопасности в случае отказов авиационной и космической техники // International Journal of Open Information Technologies. 2023. Vol. 6, No. 5. P. 1–6.

[7] Гэри М., Джонсон Д.: Вычислительные машины и труднорешаемые задачи. – СПб., БХВ-Петербург. – 2010. – 336 с.

[8] Мельников Б.Ф., Терентьева Ю.Ю., Чайковский Д.А.: О применении эвристических алгоритмов решения псевдогеометрической версии задачи коммивояжёра для проектирования сетей связи // Информатизация и связь. 2023. № 4. С. 7–16.

[9] Мельников Б.Ф., Мельникова Е.А.: О классической версии метода ветвей и границ // Компьютерные инструменты в образовании. 2021. № 1. С. 21–44.

[10] Dorigo M., Gambardella L.M.: Ant colony system: A cooperative learning approach to the traveling salesman problem // IEEE Transactions on Evolutionary Computation. 1997. Vol. 1, No. 1. P. 53–66.

[11] Gutin G., Punnen A. (editors): The Traveling Salesman problem. – Boston, Kluwer Academic Publishers. – 2008. – 856 p.

[12] Rego C., Gamboa D., Glover F., Osterman C.: Traveling salesman problem heuristics: Leading methods, implementations and latest advances // European Journal of Operational Research. 2011. Vol. 211, No. 3. P. 427–441.

[13] Cook W.: In pursuit of the traveling salesman: Mathematics at the limits of computation. – Princeton, Princeton University Press. – 2012. – 248 p.

[14] Лагутин М.Б.: Наглядная математическая статистика. – М., Бином. Лаборатория знаний. – 2012. – 472 с.

[15] Мельников Б.Ф., Сайфуллина Е.Ф., Терентьева Ю.Ю., Чурикова Н.П.: Применение алгоритмов генерации случайных графов для исследования надёжности сетей связи // Информатизация и связь. 2018. № 1. С. 71–80.

[16] Булынин А.Г., Мельников Б.Ф., Мещанин В.Ю., Терентьева Ю.Ю.: Оптимизационные задачи, возникающие при проектировании сетей связи высокой размерности, и некоторые эвристические методы их решения // Информатизация и связь. 2020. № 1. С. 34–40.

[17] Мельников Б.Ф., Терентьева Ю.Ю.: Построение оптимального остовного дерева как инструмент для обеспечения устойчивости сети связи // Известия высших учебных заведений. Поволжский регион. Технические науки. 2021. № 1 (57). С. 36–45.

[18] Бирюков А.А.: Информационная безопасность: защита и нападение. – М., ДМК Пресс. – 2013. – 472 с.

[19] Попков Г.В.: К вопросу оценки устойчивости функционирования элементов сети связи // Программные продукты и системы. 2018. № 2. С. 316–320.

[20] Попков Г.В., Овчинникова Е.А.: К вопросу применения сценарного подхода в формировании модели угроз информационной безопасности мультисервисных сетей связи // Телекоммуникации. 2022. № 9. С. 21–27.

[21] Рассел С., Норвиг П.: Искусственный интеллект. Современный подход. – М., Вильямс. – 2015. –1408 с.

[22] Баумгертнер С.В., Мельников Б.Ф.: Мультиэвристический подход к проблеме звездно-высотной минимизации недетерминированных конечных автоматов // Вестник Воронежского государственного университета. Серия: Системный анализ и информационные технологии. 2010. № 1. С. 5–7.

[23] Мельников Б.Ф., Эйрих С.Н.: Подход к комбинированию незавершенного метода ветвей и границ и алгоритма имитационной нормализации // Вестник Воронежского государственного университета. Серия: Системный анализ и информационные технологии. 2010. № 1. С. 35–38.

[24] Сигал И.Х., Иванова А.П.: Введение в прикладное дискретное программирование: модели и вычислительные алгоритмы. – М., Физматлит. – 2003. – 240 с.

[25] Гудман С., Хидетниеми С.: Введение в разработку и анализ алгоритмов. – М., Мир. – 1981. – 368 с.

[26] Melnikov B.: Discrete optimization problems – some new heuristic approaches // In: Proceedings – Eighth International Conference on High-Performance Computing in Asia-Pacific Region, Beijing. 2005. P. 73–80.

[27] Мельников Б.Ф.: Мультиэвристический подход к задачам дискретной оптимизации // Кибернетика и системный анализ. 2006. № 3. С. 32–40.

[28] Melnikov B., Terentyeva Y.: An approach for obtaining estimation of stability of large communication network taking into account its dependent paths // Cybernetics and Physics. 2022. Vol. 11, No. 3. P. 145–150.

[29] Мельников Б.Ф., Радионов А.Н.: О выборе стратегии в недетерминированных антагонистических играх // Программирование (РАН). 1998. № 5. С. 55–62.

[30] Melnikov B.F.: Heuristics in programming of nondeterministic games // Programming and Computer Software. 2001. Vol. 27, No. 5. P. 277–288.

[31] Березовский Б.А., Гнедин А.В.: Задача наилучшего выбора. – М., Наука. – 1984. – 198 с.

[32] Подиновский В.В., Ногин В.Д.: Парето-оптимальные решения многокритериальных задач. – М., Физматлит. – 2007. – 256 с.

[33] Мельников Б.Ф.: Об объектно-ориентированной реализации метода ветвей и границ для задачи коммивояжёра. Часть I // Современные информационные технологии и ИТ-образование. 2022. Том 18, № 2. С. 287–299.

[34] Мельников Б.Ф.: Об объектно-ориентированной реализации метода ветвей и границ для задачи коммивояжёра. Часть II // Современные информационные технологии и ИТ-образование. 2022. Том 18, № 3. С. 644–654.

[35] Мельников Б.Ф., Терентьева Ю.Ю.: О графовой модели для задач рефлектометрии и некоторых алгоритмах их решения. Часть III. Подход к генерации тестовых данных и результаты вычислительных экспериментов // Известия высших учебных заведений. Поволжский регион. Физико-математические науки. 2022. № 4 (64). С. 31–41.

Борис Феликсович МЕЛЬНИКОВ,
профессор Университета МГУ – ППИ в Шэньчжэне
(http://szmsubit.ru/),
email$_1$: bormel@mail.ru,
email$_2$: bormel@smbu.edu.cn,
mathnet.ru: personid=27967,
elibrary.ru: authorid=15715,
scopus.com: authorId=55954040300,
ORCID: orcidID=0000-0002-6765-6800.

Юлия Юрьевна ТЕРЕНТЬЕВА,
начальник управления анализа и методологии
совершенствования информационных систем
Центра информационных технологий и систем
органов исполнительной власти им. А. В. Старовойтова
(https://citis.ru/),
email: terjul@mail.ru,
ORCID: orcidID=0000-0002-5328-1841.

On the application a number of predictors in the branch and bound method (on the example of the random variant of the traveling salesman problem)

Boris Melnikov, Yulia Terentyeva

***Abstract*—For the so-called random variant of the traveling salesman problem, there are currently no fast algorithms, including heuristic ones, that give the optimal solution (or close to it); at least for the dimension of the problem exceeding 300. This fundamentally distinguishes the random variant from the so-called geometric variant of the same problem, for which the so-called onion husk algorithms have been known for at least 25 years, giving a close to optimal solution for any dimension. At the same time, the random variant of the traveling salesman problem, as well as the more general pseudo-geometric variant, often represents an acceptable model for applied problems of different classes; this statement can hardly be attributed to the geometric variant, therefore the algorithms for solving the random variant are still important.**

Also, the relevance of the problem variant considered in the paper follows from its possible applicability in the formulation of options for calculating the so-called viability of the communication network, an integral indicator that is an average indicator of the viability estimates of all possible communication directions determined by a pair of vertices of the communication network graph. Usually, communication network developers strive to ensure that all directions have a backup route; it is assumed that we should check this, and such a check can be made using the algorithm for finding the optimal Hamiltonian cycle. So, it is necessary to be able to solve a random variant of the traveling salesman problem, and, first of all, to describe the so-called anytime algorithms for its solution.

For a random variant of the problem (and for both exact and anytime algorithms), the classical approach to its solution is relevant, associated with the use of the branch and bound method. We apply the usual description of this method, but with numerous changes and additions described in our previous publications, as well as in this paper. Apparently, to build successful specific variants of algorithms implementing this method, the most important is the auxiliary algorithm for selecting the separating element (and the last statement applies not only to the traveling salesman problem, but also to any problem solved using the branch and bound method). The auxiliary procedure for selecting a separating element is called a predictor.

In this paper, the authors propose a use case for selecting the separating element of an integral assessment using two simplest predictors; at the same time, such an option is easily generalized to the case of a larger number of predictors. The paper briefly describes the results obtained, confirming the improvement of the algorithm of the branch and bound method when using the two predictors under consideration.

***Keywords*—heuristic algorithms, branch and bound method, traveling salesman problem, variant of the problem, the random variant, predictors.**

References

[1] *Introduction to Automata, Computability, Complexity, Algorithmics, Randomization, Communication, and Cryptography*. – Berlin, Springer. – 2004. – 318 p.

[2] Hromkovič J.: Algorithmics for Hard Problems. Introduction to Combinatorial Optimization, Randomization, Approximation, and Heuristics. – Berlin, Springer. – 2003. – 538 p.

[3] Makarkin S.B., Melnikov B.F.: Geometric methods for solving the pseudogeometric version of the traveling salesman problem // Stochastic Optimization in Computer Science. 2013. Vol. 6, No. 2. P. 54–72. (In Russian.)

[4] Makarkin S.B., Melnikov B.F., Trenina M.A.: Application of problem-oriented metrics in geometric algorithms for solving the pseudogeometric version of the traveling salesman problem // Stochastic Optimization in Computer Science. 2014. Vol. 10, No. 1. P. 63–71. (In Russian.)

[5] Makarkin S.B., Melnikov B.F., Trenina M.A.: An approach to solving the pseudogeometric version of the traveling salesman problem // News of higher educational institutions. Volga region. Physical and mathematical sciences. 2015. No. 2 (34). P. 135–147. (In Russian.)

[6] Melnikov B.F., Давыдова E.B.: Mathematical modeling of increasing the level of safety in case of failures of aviation and space technology // International Journal of Open Information Technologies. 2023. Vol. 6, No. 5. P. 1–6. (In Russian.)

[7] Garey M., Johnson D. *Computers and Intractability. A Guide to the Theory of NP-Completeness*. – San Francisco, Freeman and Company. – 1979. – 338 p.

[8] Melnikov B.F., Terentyeva Y.Y., Chaykovsky D.A.: On the application of heuristic algorithms for solving the pseudogeometric version of the traveling salesman problem for the design of communication networks // Informatization and Communication. 2023. No. 4. P. 7–16. (In Russian.)

[9] Melnikov B.F., Melnikova E.A.: On the classical version of the branch and bound method // Computer Tools in Education. 2021. No. 1. P. 21–44. (In Russian.)

[10] Dorigo M., Gambardella L.M.: Ant colony system: A cooperative learning approach to the traveling salesman problem // IEEE Transactions on Evolutionary Computation. 1997. Vol. 1, No. 1. P. 53–66.

[11] Gutin G., Punnen A. (editors): The Traveling Salesman problem. – Boston, Kluwer Academic Publishers. – 2008. – 856 p.

[12] Rego C., Gamboa D., Glover F., Osterman C.: Traveling salesman problem heuristics: Leading methods, implementations and latest advances // European Journal of Operational Research. 2011. Vol. 211, No. 3. P. 427–441.

[13] Cook W.: In pursuit of the traveling salesman: Mathematics at the limits of computation. – Princeton, Princeton University Press. – 2012. – 248 p.

[14] Lagutin M.B.: Visual mathematical statistics. – Moscow, Binom. Laboratory of Knowledge. – 2012. – 472 p. (In Russian.)

[15] Melnikov B.F., Sayfullina E.F., Terentyeva Y.Y., Churikova N.P.: Application of random graph generation algorithms to study the reliability of communication networks // Informatization and Communication. 2018. No. 1. P. 71–80. (In Russian.)

[16] Bulynin A.G., Melnikov B.F., Meshchanin V.Y., Terentyeva Y.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. (In Russian.)

[17] Melnikov B.F., Terentyeva Y.Y.: Building an optimal spanning tree as a tool to ensure the stability of the communication network // News of higher educational institutions. Volga region. Technical sciences. 2021. No. 1 (57). P. 36–45. (In Russian.)

[18] Biryukov A.A.: Information security: defense and attack. – Moscow,

DMK Press. – 2013. – 472 p. (In Russian.)
[19] Popkov G.V.: On the issue of assessing the stability of the functioning of the elements of the communication network // Software Products and Systems. (In Russian.) 2018. No. 2. P. 316–320.
[20] Popkov G.V., Ovchinnikova E.A.: On the issue of using a scenario approach in the formation of a model of threats to information security of multiservice communication networks // Telecommunications. 2022. No. 9. P. 21–27. (In Russian.)
[21] Russell S., Norvig P. Artificial Intelligence: A Modern Approach. Englewood Cliffs: Pearson Publ., 2009. 1152 p.
[22] Baumgärtner S.V., Melnikov B.F.: A multiheuristic approach to the problem of the star-height minimization of nondeterministic finite automata // Bulletin of the Voronezh State University. Series: System Analysis and Information Technology. 2010. No. 1. P. 5–7. (In Russian.)
[23] Melnikov B.F., Eirich S.N.: An approach to combining the incomplete branch and bound method and the simulated annealing algorithm // Bulletin of the Voronezh State University. Series: System Analysis and Information Technology. 2010. No. 1. P. 35–38. (In Russian.)
[24] Sigal I.H., Ivanova A.P.: Introduction to applied discrete programming: models and computational algorithms. – Moscow, Fizmatlit. – 2003. – 240 p. (In Russian.)
[25] Goodman S. E., Hedetniemi S. T. Introduction to the design and analysis of algorithms. N.Y.: McGraw-Hill Publ., 1977. 371 p.
[26] Melnikov B.: Discrete optimization problems – some new heuristic approaches // In: Proceedings – Eighth International Conference on High-Performance Computing in Asia-Pacific Region, Beijing. 2005. P. 73–80.
[27] Melnikov B.F.: Multiheuristic approach to discrete optimization problems // Cybernetics and System Analysis. 2006. No. 3. P. 32–40. (In Russian.)
[28] Melnikov B., Terentyeva Y.: An approach for obtaining estimation of stability of large communication network taking into account its dependent paths // Cybernetics and Physics. 2022. Vol. 11, No. 3. P. 145–150.
[29] Melnikov B.F., Radionov A.N.: On the choice of strategy in non-deterministic antagonistic games // Programming (Russian Academy of Sciences Ed.). 1998. No. 5. P. 55–62. (In Russian.)
[30] Melnikov B.F.: Heuristics in programming of nondeterministic games // Programming and Computer Software. 2001. Vol. 27, No. 5. P. 277–288.
[31] Berezovsky B.A., Gnedin A.V.: The task of the best choice. – Moscow, Nauka. – 1984. – 198 p. (In Russian.)
[32] Podinovsky V.V., Nogin V.D.: Pareto-optimal solutions to multi-criteria problems. – Moscow, Fizmatlit. – 2007. – 256 p. (In Russian.)
[33] Melnikov B.F.: On the object-oriented implementation of the branch and bound method for the traveling salesman problem. Part I // Modern Information Technologies and IT Education. 2022. Vol. 18, No. 2. P. 287–299. (In Russian.)
[34] Melnikov B.F.: On the object-oriented implementation of the branch and bound method for the traveling salesman problem. Part II // Modern Information Technologies and IT Education. 2022. Vol. 18, No. 3. P. 644–654. (In Russian.)
[35] Melnikov B.F., Terentyeva Y.Y.: On a graph model for reflectometry problems and some algorithms for their solution. Part III. Approach to test data generation and results of computational experiments // News of higher educational institutions. Volga region. Physical and mathematical sciences. 2022. No. 4 (64). P. 31–41. (In Russian.)

Boris MELNIKOV,
Professor of Shenzhen MSU – BIT University, China
(http://szmsubit.ru/),
email$_1$: bormel@smbu.edu.cn,
email$_2$: bf-melnikov@yandex.ru,
mathnet.ru: personid=27967,
elibrary.ru: authorid=15715,
scopus.com: authorId=55954040300,
ORCID: orcidID=0000-0002-6765-6800.

Yulia TERENTYEVA,
Head of the Department of Analysis and Methodology
of Information Systems Improvement
of The Center for Information Technologies and Systems
of Executive Authorities named after A. V. Starovoytov
(https://citis.ru/),
email: terjul@mail.ru,
ORCID: orcidID=0000-0002-5328-1841.