О применении некоторых эвристик при исследовании задачи вершинной минимизации недетерминированных конечных автоматов методом ветвей и границ. Часть 2

М. Э. Абрамян, Б. Ф. Мельников

***Аннотация*–Работа продолжает изучение эвристик, которые могут быть применены к решению задачи вершинной минимизации недетерминированных конечных автоматов методом ветвей и границ, а точнее – к реализации наиболее трудоемкого этапа решения, связанного с нахождением минимального покрытия вспомогательной логической матрицы специальными подмножествами ее элементов со значением 1 (true). Описана новый вид эвристики («эвристика большого шага») и рассмотрены различные комбинации данной эвристики и двух видов ранее описанных эвристик. Также описана вспомогательная эвристика, основанная на применении дополнительной информации об анализируемой матрице и позволяющая более точно оценить эффективность различных комбинаций основных эвристик.**

Наряду с новыми эвристиками описывается также один из подходов к анализу результатов численных экспериментов, основанный на построении эталонного набора наилучших псевдооптимальных решений и исследовании распределений полученных решений для различных вариантов алгоритма относительно этого эталонного набора.

***Ключевые слова*–недетерминированные конечные автоматы, минимизация, метод ветвей и границ, эвристический алгоритм, программная реализация.**

VIII. Введение в часть 2

Настоящая часть 2 – продолжение части 1, т. е. [1]. Мы продолжаем нумерацию пунктов и таблиц, однако «сбрасываем» нумерацию библиографических ссылок.

Работа продолжает серию работ [2–4], посвященных исследованию различных вариантов алгоритма, связанного с основным этапом решения задачи вершинной минимизации недетерминированных конечных автоматов (НКА). Рассматриваемый подход к решению задачи вершинной минимизации основан на результатах работ [5–7], его основным этапом является решение вспомогательной задачи, которую можно описать в матричных терминах. Сама задача, описание базового алгоритма ее решения на основе метода ветвей и границ, а также детали программной реализации алгоритма приведены в [2–4]; в настоящей работе основное внимание уделяется дополнительным эвристикам, позволяющим повысить эффективность базового алгоритма.

Напомним основные факты, связанные с матричной задачей и алгоритмом ее решения.

Рассматривается матрица с логическими значениями (true и false или 1 и 0). Будем называть *гридом* набор строк и столбцов данной матрицы, на пересечении которых содержатся только единицы. Грид называется *полным*, если его нельзя расширить путем добавления новой строки или столбца. Если совокупность единичных элементов, соответствующих некоторому набору полных гридов, включает все единичные элементы исходной матрицы, то будем называть такой набор гридов *покрытием* матрицы, а число полных гридов, входящих в этот набор, – *размером* покрытия. Задача заключается в нахождении покрытия исходной матрицы, имеющего минимальный размер.

Базовый алгоритм решения данной задачи состоит в формировании и последующем анализе набора подзадач, каждая из которых представляет собой некоторый набор полных гридов для исходной матрицы (который, возможно, еще не является покрытием матрицы). Новые подзадачи создаются на основе расщепления существующей подзадачи на две новые. Это действие представляет собой основной шаг базового алгоритма и реализовано в виде метода MainStep.

В части 1 работы были описаны и проанализированы две эвристики, которые можно включить в базовый алгоритм. Первая эвристика, обозначаемая CutOff, связана с отсечением подзадач, вторая эвристика, обозначаемая MakeGr(N), связана с генерацией начального набора полных гридов, при этом параметр N определяет число шагов, выполняемых на этапе начальной генерации (для каждого единичного элемента исходной матрицы делается попытка сгенерировать N полных гридов, содержащих этот элемент; гриды, совпадающие с ранее сгенерированными, игнорируются). Численные эксперименты показали, что эвристика CutOff, взятая изолированно, не улучшает базовый алгоритм, тогда как эвристика MakeGr(N) позволяет получать, в среднем, лучшие псевдооптимальные решения.

Для тестирования различных вариантов алгоритма использовались два метода: StepRun(steps) и

Статья получена 7 сентября 2020.

Михаил Эдуардович Абрамян, Южный федеральный университет (email: m-abramyan@yandex.ru).

Борис Феликсович Мельников, Университет МГУ – ППИ в Шэньчжэне (email: bf-melnikov@yandex.ru).

TimeRun(seconds). Метод StepRun выполняет указанное в параметре steps количество вызовов метода MainStep. Продолжительность метода TimeRun (и тем самым количество вызовов метода MainStep) определяется параметром seconds следующим образом: работа метода TimeRun прекращается, если за указанное число секунд не будет найдено ни одного нового псевдооптимального решения. Варианты алгоритма тестировались на двух наборах по 100 матриц, сгенерированных случайным образом: первый набор содержал матрицы размера 15 на 25, полученные путем добавления к начальной нулевой матрице 20 различных случайных гридов, а второй содержал матрицы размера 30 на 40, построенные на основе 35 различных случайных гридов.

IX. Третья модификация базового алгоритма: дополнительная рандомизация

При использовании начального набора полных гридов, т. е. эвристики MakeGr(N), естественно ожидать, и расчеты, приведенные в части 1, это подтверждают, что результаты будут улучшаться с увеличением начального набора (т. е. с увеличением параметра N). Однако при этом время каждого шага алгоритма будет увеличиваться (в частности, за фиксированное время будут обработано меньшее число подзадач), а действия на каждом шаге будут полностью детерминированы и определяться исходным составом полных гридов.

С другой стороны, при выполнении алгоритма с иным инициализирующим значением датчика случайных чисел мы можем получить другой начальный набор гридов, на основе которого вполне может быть найдено псевдооптимальное решение меньшего размера. При этом можно использовать начальные наборы меньшего размера, что ускорит обработку подзадач.

Указанная идея, связанная с дополнительной рандомизацией алгоритма, была положена в основу его следующей модификации: в алгоритм добавлен метод BigStep, реализующий новый уровень – «большой шаг», на котором выполняются «основные шаги» (метод MainStep). Основная особенность большого шага состоит в том, что в его начале выполняется полная очистка набора подзадач, а начальный набор полных гридов строится заново при другом инициализирующем значении датчика случайных чисел. При этом, разумеется, существует опасность «потерять» лучшее псевдооптимальное решение, которое может быть достигнуто на прежнем наборе гридов, однако, с другой стороны, появляется шанс улучшить решение за счет совершенно новой конфигурации гридов.

Метод BigStep состоит из трех этапов. На первом из них выполняются инициализирующие действия: очищается набор subtasks, создается датчик случайных чисел с новым инициализирующим значением (которое больше предыдущего на 1) и создается новый начальный набор гридов grids и начальная (пустая) подзадача.

На втором этапе строится первое псевдооптимальное решение на основе нового набора полных гридов; для этого в цикле вызывается метод MainStep. Чтобы учесть информацию о ранее найденном псевдооптимальном решении, на данном этапе выполняется не более OptSize + N1 вызовов метода MainStep, где OptSize – размер наилучшего псевдооптимального решения, полученного на предыдущих больших шагах, а N1 – некоторое неотрицательное значение. Если за указанное число шагов первое решение не получено, то построение решения прерывается, и начинается новый большой шаг (таким образом, на этом этапе также может произойти отсечение подзадач).

Если же на втором этапе некоторое решение построено (его размер может превышать текущее псевдооптимальное решение не более чем на N1), то делается попытка его улучшить, выполнив дополнительное количество N2 вызовов метода MainStep. В этом состоит третий этап большого шага.

После завершения указанного числа шагов текущий большой шаг завершается, и выполняется очередной вызов метода BigStep со следующим инициализирующим значением датчика случайных чисел и теми же параметрами N1 и N2.

Метод StepRun(stepCount) для данной модификации по-прежнему завершает работу, как только число вызовов метода MainStep достигнет значения stepCount (независимо от того, на каком этапе очередного большого шага это произошло), в методе TimeRun(seconds) проверка, связанная с завершением алгоритма, выполняется после завершения каждого большого шага.

Описанную эвристику «большого шага» будем обозначать BigStep(N1,N2), где N1 и N2 – описанные выше параметры данной эвристики. Можно ожидать, что данная эвристика окажется эффективной в комбинации с эвристиками CutOff и MakeGr(N) при небольших значениях N.

Матрицы размера 15 × 25 и 30 × 40 были обработаны методом TimeRun(10) с использованием следующих комбинаций эвристик (16 вариантов):

- CutOff: наличие или отсутствие,
- MakeGr(N): параметр N равен 1 или 2,
- BigStep(N1, N2): параметр N1 равен 0 или 1, параметр N2 равен 100 или 1000.

Кроме того, для матриц размера 30 × 40 были дополнительно рассмотрены 8 вариантов со следующими комбинациями:

- CutOff: наличие или отсутствие,
- MakeGr(N): параметр N равен 5 или 10,
- BigStep(N1, 1000): параметр N1 равен 0 или 1.

В табл. 11 приведено шесть лучших результатов работы алгоритма с применением эвристики BigStep для матриц размера 15 × 25 и 30 × 40 (в порядке улучшения средних псевдооптимальных решений, т. е. в порядке убывания средних размеров найденных покрытий). Для сравнения приведен также лучший результат работы алгоритма с применением эвристик, рассмотренных в части 1 (этот результат выделен курсивом).

Для каждого элемента данных в таблицах указываются три характеристики: среднее значение для обработанных 100 матриц и (в скобках) минимальное и максимальное значение.

Таким образом, применение эвристики BigStep действительно улучшает полученные результаты, однако только при условии надлежащего подбора параметров. В частности, всегда целесообразно применять отсечение

CutOff и использовать большое значение параметра N2 для эвристики BigStep (в таблице нет ни одного варианта с N2 = 100). Число начальных полных гридов (определяемое параметром N эвристики MakeGr) не должно быть большим, однако для матриц большого размера оно не должно быть и слишком малым: для матриц 15 × 25 лучшие результаты достигаются при N = 1, а для матриц 30 × 40 – при N = 5. Что касается параметра N1 эвристики BigStep, то для матриц 15 × 25 лучшие результаты достигаются при N1 = 1, хотя этот параметр влияет на результат в меньшей степени, чем другие, а для матриц 30 × 40 более предпочтительным является значение N1 = 0 (в таблице нет ни одного варианта с N1 = 1 для матриц данного размера).

Таблица 11. Размер наилучшего псевдооптимального решения и время работы алгоритма для различных режимов с применением эвристики BigStep

	TimeRun(10)	Время (в секундах)
15 × 25, MakeGr(100)	*14,76 (12-17)*	*11,72 (10,12-20,09)*
15 × 25, MakeGr(2)+BigStep(0,1000)+CutOff	14,80 (12-17)	11,17 (10,01-20,22)
15 × 25, MakeGr(2)+BigStep(1,1000)+CutOff	14,69 (12-17)	11,57 (10,05-19,95)
15 × 25, MakeGr(1)+BigStep(0,1000)	14,68 (12-17)	11,46 (10,03-19,55)
15 × 25, MakeGr(1)+BigStep(1,1000)	14,66 (12-17)	12,42 (10,03-19,98)
15 × 25, MakeGr(1)+BigStep(0,1000)+CutOff	14,60 (12-17)	11,50 (10,03-20,67)
15 × 25, MakeGr(1)+BigStep(1,1000)+CutOff	14,53 (12-17)	11,85 (10,03-20,69)
30 × 40, MakeGr(10)	*29,76 (24-35)*	*14,14 (10,26-20,14)*
30 × 40, MakeGr(10)+BigStep(0,1000)	29,57 (24-35)	17,60 (10,41-31,71)
30 × 40, MakeGr(10)+BigStep(0,1000)+CutOff	29,55 (24-35)	16,88 (10,59-30,50)
30 × 40, MakeGr(2)+BigStep(0,1000)	29,50 (24-35)	18,34 (10,16-32,07)
30 × 40, MakeGr(2)+BigStep(0,1000)+CutOff	29,50 (24-35)	18,33 (10,16-29,17)
30 × 40, MakeGr(5)+BigStep(0,1000)	29,46 (24-35)	18,68 (10,47-40,95)
30 × 40, MakeGr(5)+BigStep(0,1000)+CutOff	29,45 (24-35)	18,20 (10,44-38,44)

X. Вспомогательная эвристика, связанная с учетом дополнительной информации об исходной матрице

В данном пункте рассматривается вспомогательная эвристика, которая может использоваться для оценки эффективности ранее описанных эвристик. Данная эвристика основана на идее применения дополнительной информации о сгенерированных матрицах. Разумеется, эта информация в общем случае является недоступной и поэтому не может применяться в универсальном алгоритме. Однако с ее помощью можно попытаться получить лучшие решения и тем самым оценить качество решений, найденных с помощью различных вариантов универсального алгоритма.

Можно ожидать, что решение, близкое к оптимальному, будет получено, если рассматривать не произвольно сгенерированные полные гриды, а такие полные гриды, которые получены на основе (вообще говоря, неполных) гридов, использованных при генерации матриц. Для генерации подобных полных гридов достаточно модифицировать метод MakeGridRnd генерации полных гридов, передавая ему не единственный единичный элемент матрицы, который должен входить в созданный полный грид, а один из исходных неполных гридов, и применяя к нему алгоритм «пополнения», кратко описанный в п. III.

Этот модифицированный метод MakeGridRnd в дальнейшем используется для генерации исходного набора полных гридов способом, аналогичным описанному в п. VI. Отличие состоит лишь в том, что в новом варианте модифицированный метод MakeGridRnd вызывается указанное число раз не для каждого единичного элемента исходной матрицы, а для каждого «начального» неполного грида, использованного при ее построении. Как и в исходном варианте, чтобы исключить дублирование созданных гридов они сохраняются в виде упорядоченного множества. Таким образом, описанная эвристика, как и ранее рассмотренная эвристика MakeGr, обеспечивает формирование начального набора полных гридов. Данная эвристика будет обозначаться в виде InitGr(N), где N указывает число вызовов метода MakeGridRnd для каждого начального грида исходной матрицы.

Эвристика InitGr, как и эвристика MakeGr, использовалась в комбинации с другими ранее описанными эвристиками. В табл. 12 приведены три лучших результата работы алгоритма с применением эвристики InitGr для матриц размера 15 × 25 и 30 × 40. Для сравнения приведен также лучший результат работы алгоритма из табл. 11 (это результат выделен курсивом).

Таблица 12. Размер наилучшего псевдооптимального решения для различных режимов с применением эвристики InitGr

	TimeRun(10)
15 × 25, MakeGr(1)+BigStep(1,1000)+CutOff	*14,53 (12-17)*
15 × 25, InitGr(10)	14,46 (12-17)
15 × 25, InitGr(1)+BigStep(1,1000)	14,46 (11-17)
15 × 25, InitGr(100)	14,44 (11-17)
30 × 40, MakeGr(5)+BigStep(0,1000)+CutOff	*29,45 (24-35)*
30 × 40, InitGr(5)+BigStep(0,1000)	27,94 (24-33)
30 × 40, InitGr(1)+BigStep(0,1000)	27,86 (24-33)
30 × 40, InitGr(2)+BigStep(0,1000)	27,80 (24-33)

Применение эвристики InitGr действительно улучшает в среднем ранее полученные псевдооптимальные решения, хотя для матриц размера 15 × 25 улучшение является не столь существенным. Интересно отметить, что для данной эвристики, как и для базового алгоритма, применение эвристики отсечения CutOff не приводит к улучшению результатов. Лучшие результаты для матриц размера 15 × 25 были достигнуты либо без применения эвристики BigStep и для больших значений параметра N эвристики InitGr, либо с применением эвристики BigStep и параметре N = 1. Для матриц размера 30 × 40 наилучшие значения достигаются только при совместном применении эвристик InitGr и BigStep с параметром N = 0.

XI. Распределение псевдооптимальных решений для алгоритма с эвристикой InitGr

В предыдущих пунктах настоящей работы в качестве основной характеристики, позволяющей судить об эффективности различных вариантов алгоритма, использовалось усредненное значение размера наилучшего (т. е. наименьшего) из найденных покрытий исходной матрицы; при этом усреднение проводилось по результатам для 100 случайных матриц выбранного размера. Однако естественно предположить, что для отдельной матрицы наилучшее решение может быть достигнуто и в случае такого варианта алгоритма, который не показал лучший результат «в среднем». Поэтому возникает вопрос о распределении полученных псевдооптимальных решений для различных вариантов алгоритма относительно некоторой «эталонной» выборки. Если бы были известны истинные оптимальные решения, тогда в качестве эталонной выборки следовало бы использовать именно их. В нашей ситуации эталонную выборку можно сформировать, включив в нее для *каждой* матрицы *наилучшее* псевдооптимальное решение из *всех* рассмотренных вариантов алгоритма.

Проиллюстрируем данный подход на примере анализа распределения решений для алгоритма с эвристикой InitGr, описанной в п. X. Вначале для набора матриц каждого размера рассмотрим по три варианта алгоритма, для которых были получены наилучшие псевдооптимальные решения в среднем (эти варианты приведены в табл. 12).

В табл. 13 приведены распределения, связанные с тремя наилучшими вариантами алгоритма для набора из 100 матриц размера 15 × 25.

Приведенная информация означает, что если для каждой матрицы выбрать наилучшее псевдооптимальное решение из трех решений, соответствующих указанным вариантам алгоритма, то в этот «комбинированный» набор наилучших псевдооптимальных решений попадут примерно 80% решений, найденных каждым алгоритмом, а остальные 20% будут отличаться от «самого лучшего» псевдооптимального решения (найденного каким-то *другим* алгоритмом из этих трех) не более чем на 1 (если не учитывать 1% случаев отличия на 2, зафиксированный во втором из приведенных вариантов алгоритма). В качестве дополнительной информации для каждого варианта алгоритма приводится среднее значение для псевдооптимальных решений (эти средние значения указаны также в табл. 12). Кроме того, в первой строке таблицы приводится среднее значение для «комбинированного» набора наилучших псевдооптимальных решений, равное 14.26.

Распределение для набора из 100 матриц размера 30 × 40 приведено в табл. 14. Здесь 85% наилучших псевдооптимальных решений, вошедших в комбинированный набор, дает первый из указанных вариантов алгоритма, а «вклад» остальных вариантов меньше.

Добавим к изучаемому набору дополнительные варианты алгоритма с эвристикой InitGr (которые показали в среднем худшие результаты). Теперь для матриц размера 15 × 25 распределение примет вид, приведенный в табл. 15.

Анализ данного распределения показывает, что добавленные варианты содержат незначительное количество лучших псевдооптимальных решений; первые три варианта по-прежнему обеспечивают примерно 80% подобных решений. Среднее значение для комбинированного набора наилучших псевдооптимальных решений (14,23) мало отличается от ранее полученного (14,26). Как было отмечено ранее, для эвристики InitGr применение дополнительного отсечения (CutOff не приводит к улучшению алгоритма.

Для матриц размера 30 × 40 распределение будет иметь больше отличий от ранее приведенного (см. табл. 16). Здесь первый вариант обеспечивает лишь 69%

лучших псевдооптимальных решений (по сравнению с 85% для ранее приведенного распределения). Это означает, что многие последующие варианты, давая в среднем худший результат, позволили найти лучшие псевдооптимальные решения для *отдельных* матриц. Например, более детальный анализ результатов показывает, что для матрицы номер 6 алгоритм InitGr(10) нашел псевдооптимальное решение 25, в то время как три первых (т. е. «лучших» в среднем алгоритма) вернули худшее решение 26. Как следствие этого обстоятельства, среднее значение для комбинированного набора наилучших псевдооптимальных решений (27,45) оказалось существенно меньше, чем ранее полученное (27,65).

Таблица 13. Распределение псевдооптимальных решений
(три лучших варианта алгоритма, содержащих эвристику InitGr; матрицы размера 15 × 25)

0	+1	+2	15 x 25, TimeRun(10)	Ср. значение
100	*0*	*0*	*Наилучшее псевдооптимальное решение*	*14,26*
82	18	0	InitGr(100)	14,44
81	18	1	InitGr(1)+BigStep(1,1000)	14,46
80	20	0	InitGr(10)	14,46

Таблица 14. Распределение псевдооптимальных решений
(три лучших варианта алгоритма, содержащих эвристику InitGr; матрицы размера 30 × 40)

0	+1	+2	30 x 40, TimeRun(10)	Ср. значение
100	*0*	*0*	*Наилучшее псевдооптимальное решение*	*27,65*
85	15	0	InitGr(2)+BigStep(0,1000)	27,80
80	19	1	InitGr(1)+BigStep(0,1000)	27,86
72	27	1	InitGr(5)+BigStep(0,1000)	27,94

Таблица 15. Распределение псевдооптимальных решений
(10 вариантов алгоритма, содержащих эвристику InitGr; матрицы размера 15 × 25)

0	+1	+2	15 x 25, TimeRun(10)	Ср. значение
100	*0*	*0*	*Наилучшее псевдооптимальное решение*	*14,23*
79	21	0	InitGr(100)	14,44
78	21	1	InitGr(1)+BigStep(1,1000)	14,46
77	23	0	InitGr(10)	14,46
62	35	3	InitGr(10)+BigStep(1,1000)	14,64
61	35	4	InitGr(1)+BigStep(0,1000)	14,66
54	38	8	InitGr(1)+BigStep(0,1000)+CutOff	14,77
52	42	6	InitGr(1)+BigStep(1,1000)+CutOff	14,77
50	42	8	InitGr(2)+BigStep(0,1000)+CutOff	14,81
49	46	5	InitGr(100)+CutOff	14,79
47	45	8	InitGr(2)+BigStep(1,1000)+CutOff	14,84

Таблица 16. Распределение псевдооптимальных решений
(10 вариантов алгоритма, содержащих эвристику InitGr; матрицы размера 30 × 40)

0	+1	+2	+3	+4	30 x 40, TimeRun(10)	Ср. значение
100	*0*	*0*	*0*	*0*	*Наилучшее псевдооптимальное решение*	*27,45*
69	28	2	1	0	InitGr(2)+BigStep(0,1000)	27,80
65	31	3	0	1	InitGr(1)+BigStep(0,1000)	27,86
58	36	5	1	0	InitGr(5)+BigStep(0,1000)	27,94
45	47	7	0	1	InitGr(2)+BigStep(0,1000)+CutOff	28,10
44	44	11	0	1	InitGr(1)+BigStep(0,1000)+CutOff	28,15
44	40	12	4	0	InitGr(100)	28,21
41	41	16	2	0	InitGr(10)	28,24
37	53	9	0	1	InitGr(5)+BigStep(0,1000)+CutOff	28,20
32	49	18	0	1	InitGr(1)+BigStep(1,1000)+CutOff	28,34
30	55	12	2	1	InitGr(10)+BigStep(0,1000)+CutOff	28,34
18	44	31	6	1	InitGr(100)+CutOff	28,73
16	51	26	6	1	InitGr(10)+CutOff	28,70

XII. Распределение псевдооптимальных решений для основных эвристик исходного алгоритма

Проведем аналогичный анализ для ранее рассмотренных вариантов алгоритма, в которых не использовалась дополнительная информация, связанная с исходными матрицами (т. е. не использовалась эвристика InitGr):

- базовый алгоритм и его вариант с отсечением, т. е. эвристикой CutOff (эти варианты были рассмотрены в п. IV–V);
- варианты с генерацией начального набора полных гридов, т. е. эвристикой MakeGr (см. п. VI) и некоторые из этих вариантов с отсечением;
- варианты с дополнительной рандомизацией – эвристикой BigStep (см. п. IX, табл. 11).

Для базового алгоритма, не использующего дополнительные эвристики, в таблицах используется обозначение BasicBBM – от Branch and Bound Method.

Для матриц размера 15 × 25 распределение приведено в табл. 17. Наилучший из вариантов алгоритма дает 88% всех лучших псевдооптимальных решений. Впрочем, даже наихудший из рассмотренных вариантов (базовый алгоритм с отсечением) дает достаточно хорошие псевдооптимальные решения, отличающиеся от лучших решений в основном на 1–3 единицы.

Для матриц размера 30 × 40 приведем часть таблицы распределения, содержащую только варианты с эвристикой MakeGr, которые дают наилучшие псевдооптимальные решения (табл. 18). В этой таблице не указаны два варианта алгоритма (BasicBBM и CutOff), поскольку они дают наихудшие результаты и не влияют на распределение предыдущих вариантов. Заметим, что средние значения наилучших псевдооптимальных решений для вариантов BasicBBM и CutOff равны соответственно 34,86 и 41,34.

Таблица 17. Распределение псевдооптимальных решений
(14 вариантов алгоритма, содержащих эвристики CutOff, MakeGr и BigStep; матрицы размера 15 × 25)

0	+1	+2	+3	+4	+5	15 x 25, TimeRun(10)	Ср. значение
100	*0*	*0*	*0*	*0*	*0*	*Наилучшее псевдооптимальное решение*	*14,41*
88	12	0	0	0	0	MakeGr(1)+BigStep(1,1000)+CutOff	14,53
82	18	0	0	0	0	MakeGr(100)+CutOff	14,59
81	19	0	0	0	0	MakeGr(1)+BigStep(0,1000)+CutOff	14,60
76	23	1	0	0	0	MakeGr(1)+BigStep(1,1000)	14,66
73	27	0	0	0	0	MakeGr(1)+BigStep(0,1000)	14,68
72	28	0	0	0	0	MakeGr(2)+BigStep(1,1000)+CutOff	14,69
66	33	1	0	0	0	MakeGr(100)	14,76
65	34	1	0	0	0	MakeGr(10)	14,77
61	39	0	0	0	0	MakeGr(2)+BigStep(0,1000)+CutOff	14,80
58	38	4	0	0	0	MakeGr(2)	14,87
45	46	9	0	0	0	MakeGr(1)	15,05
42	47	11	0	0	0	BasicBBM	15,10
39	41	19	1	0	0	MakeGr(1)+CutOff	15,23
23	32	27	12	5	1	CutOff	15,88

Таблица 18. Распределение псевдооптимальных решений
(14 вариантов алгоритма, содержащих эвристики CutOff, MakeGr и BigStep; матрицы размера 30 × 40),
наихудшие варианты BasicBBM и CutOff в таблице не указаны

0	+1	+2	+3	+4	+5	+6	+7	+8	30 x 40, TimeRun(10)	Ср. значение
100	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*Наилучшее псевдооптимальное решение*	*28,85*
56	26	15	3	0	0	0	0	0	MakeGr(2)+BigStep(0,1000)+CutOff	29,50
56	26	15	3	0	0	0	0	0	MakeGr(2)+BigStep(0,1000)	29,50
52	37	10	1	0	0	0	0	0	MakeGr(5)+BigStep(0,1000)+CutOff	29,45
51	38	10	1	0	0	0	0	0	MakeGr(5)+BigStep(0,1000)	29,46
47	38	13	2	0	0	0	0	0	MakeGr(10)+BigStep(0,1000)+CutOff	29,55
45	40	13	2	0	0	0	0	0	MakeGr(10)+BigStep(0,1000)	29,57
34	43	21	2	0	0	0	0	0	MakeGr(10)	29,76
32	42	18	5	2	1	0	0	0	MakeGr(100)+CutOff	29,91
30	42	19	5	3	1	0	0	0	MakeGr(100)	29,97
12	16	35	22	8	6	1	0	0	MakeGr(2)	31,05
2	9	12	27	20	16	8	5	1	MakeGr(1)	32,50
1	11	13	29	20	12	8	4	2	MakeGr(1)+CutOff	32,41

Два первых варианта алгоритма, указанные в табл. 18, дают 56% наилучших псевдооптимальных решений. Следует отметить, что большинство наилучших псевдооптимальных решений дают варианты с эвристикой BigStep.

Как уже было отмечено ранее, варианты с отсечением мало отличаются от аналогичных вариантов без отсечений. Интересно отметить, что вариант алгоритма с наилучшим средним значением решений оказался не на первом, а лишь на третьем месте, поскольку он позволил найти наилучшие решения лишь в 52% случаев. Зато для этого варианта (по сравнению с двумя первыми) имеется существенно большее количество решений, отличающихся от наилучшего на 1. При отсутствии эвристики BigStep распределение начинает ухудшаться (что, собственно, можно было определить и по среднему значению псевдооптимальных решений).

XIII. Анализ основных эвристик с учетом результатов для эвристики InitGr

Эффективность основных эвристик можно более точно оценить, если учесть результаты, полученные для вариантов алгоритма с эвристикой InitGr. Как отмечалось в п. X, эвристика InitGr позволяет получить лучшие результаты за счет использования дополнительной информации о рассматриваемых матрицах, поэтому эти результаты можно рассматривать в качестве альтернативы «истинных» оптимальных решений. Причем в данном случае в качестве «эталонного» решения целесообразно использовать не результаты, которые получены каким-то одним вариантом алгоритма с эвристикой InitGr, а *комбинированный* набор результатов, в который входят наилучшие значения для каждой матрицы, выбранные из *всех* рассмотренных вариантов алгоритма с эвристикой InitGr (подобные наборы были построены в п. XI – см. табл. 15 и 16).

Итак, в данном пункте мы рассмотрим распределения решений, полученных с помощью основных вариантов универсального алгоритма (см. п. XII), к которым будет добавлен один особый набор решений – комбинированный набор наилучших решений, полученных с применением эвристики InitGr (для этого набора будем использовать обозначение InitGrCombo). Тем самым мы сможем ответить на следующий вопрос: насколько станут хуже распределения для вариантов универсального алгоритма, если их сравнивать не только между собой, но и с набором решений, являющихся в некотором смысле «самыми лучшими» из найденных, т. е. наиболее близких к оптимальным?

Для матриц размера 15 × 25 распределение приведено в табл. 19. Хотя 94% наилучших решений достигается для вариантов, использующих эвристику InitGr, имеется 6% матриц, для которых основные варианты алгоритма позволили получить лучшее решение. Интересно отметить, что для одной из таких матриц – номер 86 – наилучшее решение, равное 14, было найдено только базовым алгоритмом, расположенным в данном списке третьим *с конца*; все прочие алгоритмы из данного списка находили для этой матрицы решения 15 или 16.

В то же время распределение основных вариантов алгоритма ухудшилось по сравнением с тем, для которого не учитывались решения, найденные с помощью эвристики InitGr (см. п. XII). В частности, для лучшего из основных вариантов MakeGr(1)+BigStep(1,1000)+CutOff распределение «88-12» из табл. 17 превратилось в «65-34-1», т. е. теперь только для 65% матриц решения, полученные данным вариантом, совпадают с наилучшими. Впрочем, для 34% матриц решения для этого варианта отличаются от наилучших всего на 1 единицу и лишь для одного процента матриц результат оказался хуже наилучшего на 2 единицы.

Таблица 19. Распределение псевдооптимальных решений (14 вариантов алгоритма, содержащих эвристики CutOff, MakeGr и BigStep, а также набор InitGrCombo; матрицы размера 15 × 25)

0	+1	+2	+3	+4	+6	15 x 25, TimeRun(10)	Ср. значение
100	*0*	*0*	*0*	*0*	*0*	*Наилучшее псевдооптимальное решение*	*14,17*
94	6	0	0	0	0	InitGrCombo	14,23
65	34	1	0	0	0	MakeGr(1)+BigStep(1,1000)+CutOff	14,53
60	38	2	0	0	0	MakeGr(100)+CutOff	14,59
59	39	2	0	0	0	MakeGr(1)+BigStep(0,1000)+CutOff	14,60
58	35	7	0	0	0	MakeGr(1)+BigStep(1,1000)	14,66
52	45	3	0	0	0	MakeGr(1)+BigStep(0,1000)	14,68
50	48	2	0	0	0	MakeGr(2)+BigStep(1,1000)+CutOff	14,69
47	47	6	0	0	0	MakeGr(100)	14,76
46	48	6	0	0	0	MakeGr(10)	14,77
44	43	12	1	0	0	MakeGr(2)	14,87
41	55	4	0	0	0	MakeGr(2)+BigStep(0,1000)+CutOff	14,80
31	51	17	1	0	0	MakeGr(1)	15,05
28	53	17	2	0	0	BasicBBM	15,10
23	52	21	4	0	0	MakeGr(1)+CutOff	15,23
14	35	27	16	7	1	CutOff	15,88

Таблица 20. Распределение псевдооптимальных решений (6 вариантов алгоритма, содержащих эвристики CutOff, MakeGr и BigStep, а также набор InitGrCombo; матрицы размера 30 × 40)

0	+1	+2	+3	+4	+5	+6	30 x 40, TimeRun(10)	Ср. значение
100	*0*	*0*	*0*	*0*	*0*	*0*	*Наилучшее псевдооптимальное решение*	*27,42*
97	3	0	0	0	0	0	InitGrCombo	27,45
13	20	29	20	15	3	0	MakeGr(10)+BigStep(0,1000)+CutOff	29,55
12	20	30	20	15	3	0	MakeGr(10)+BigStep(0,1000)	29,57
11	29	26	18	9	5	2	MakeGr(2)+BigStep(0,1000)+CutOff	29,50
11	25	32	18	11	2	1	MakeGr(5)+BigStep(0,1000)+CutOff	29,45
11	25	31	19	11	2	1	MakeGr(5)+BigStep(0,1000)	29,46
10	29	28	18	8	5	2	MakeGr(2)+BigStep(0,1000)	29,50

Для матриц размера 30 × 40 ограничимся только теми основными вариантами алгоритма, которые используют эвристику BigStep и показали наилучшие результаты (табл. 20).

В данном случае 97%, т. е. подавляющее большинство наилучших решений достигается для вариантов, использующих эвристику InitGr, а максимальная доля наилучших решений, которые достигаются для основных вариантов алгоритма, снизилась с 56% (см. табл. 18) до 13%. Тем не менее, подобный результат для матриц большого размера может считаться вполне приемлемым, поскольку для 65% процентов матриц полученные решения отличаются от наилучших не более чем на 3 единицы. Следует также принять во внимание, что указанные результаты были получены в среднем за 16–18 секунд работы алгоритма (см. табл. 11).

XIV. Заключение

В работе приведены результаты численных исследований различных вариантов алгоритма нахождения минимального покрытия матрицы полными гридами (который является самым трудоемким этапом вершинной минимизации недетерминированных конечных автоматов). Полученные результаты свидетельствуют о том, что наиболее эффективным является вариант, в котором комбинируются различные эвристики. Также описан подход, позволяющий более точно оценить сравнительную эффективность различных вариантов алгоритма.

Для продолжения описанной здесь тематики мы предполагаем подключить вспомогательные алгоритмы выбора разделяющего элемента (из нескольких возможных), схожих с алгоритмами, приведенными в [8, 9]. Кроме того, мы планируем реализовать и исследовать параллельные версии рассмотренных алгоритмов (см. [10]).

Библиография

[1] Абрамян М.Э., Мельников Б.Ф. О применении некоторых эвристик при исследовании задачи вершинной минимизации недетерминированных конечных автоматов методом ветвей и границ. Часть 1 // International Journal of Open Information Technologies. 2020. Vol. 8. No. 9. P. 1–7.

[2] Абрамян М.Э., Мельников Б.Ф. Исследование задачи вершинной минимизации недетерминированных конечных автоматов с помощью метода ветвей и границ // Cloud of Science. 2020. Т. 7, № 2. С. 297–319.

[3] Абрамян М.Э. Об одном подходе к реализации метода ветвей и границ для оптимизационных задач // Информационные подходы в моделировании и управлении: подходы, методы, решения. Сборник научных статей II Всероссийской научной конференции с международным участием. Часть 1. Тольятти, 2019. С. 56–64.

[4] Abramyan M.E., Melnikov B.F. An approach to algorithmizing the problem of vertex minimization of nondeterministic automata. Part I. Problem statement and the brief description of the basis methods // IOP Conference Series: Materials Science and Engineering. **862** (2020) 052055. P. 1–6.

[5] Melnikov B. Once more on the edge-minimization of nondeterministic finite automata and the connected problems // Fundamenta Informaticae. 2010. Vol. 104. No. 3. P. 267–283.

[6] Мельников Б.Ф. Регулярные языки и недетерминированные конечные автоматы : монография. – М. : РГСУ, 2018.

[7] Melnikov B. The complete finite automaton // International Journal of Open Information Technologies. 2017. Vol. 5. No. 10. P. 9–17.

[8] Мельников Б.Ф., Мельникова Е.А. Кластеризация ситуаций в алгоритмах реального времени для задач дискретной оптимизации // Системы управления и информационные технологии. 2007. Т. 28, № 2. С. 16–20.

[9] Мельников Б., Романов Н. Ещё раз об эвристиках для задачи коммивояжёра // Теоретические проблемы информатики и ее приложений. 2001. Т. 4. С. 81.

[10] Melnikov B., Tsyganov A. The state minimization problem for nondeterministic finite automata: the parallel implementation of the truncated branch and bound method // Proceedings – International Symposium on Parallel Architectures, Algorithms and Programming. 2012. P. 194-201.

Михаил Эдуардович Абрамян,
доцент Южного федерального университета, Ростов-на-Дону (`https://sfedu.ru/`),
email: `m-abramyan@yandex.ru`,
mathnet.ru: `personid=20226`,
elibrary.ru: `authorid=17911`,
scopus.com: `authorId=8291167000`,
ORCID: `orcidID=0000-0002-2802-6144`.

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

On the application of some heuristics in the study of the state minimization problem for nondeterministic finite automata by the branch and bound method. Part 2

M. E. Abramyan, B. F. Melnikov

***Abstract*–We continue to study heuristics that can be applied to solve the problem of minimizing the states of nondeterministic finite automata by the branch and bound method, or rather, to the implementation of the most difficult stage of the solution associated with finding the minimum cover of an auxiliary logical matrix by special subsets of its elements with the value 1 (true). A new type of heuristic is described (the "big step" heuristic) and various combinations of this heuristic and the two types of previously described heuristics are considered. An auxiliary heuristic based on the use of additional information about the analyzed matrix is also described. This auxiliary heuristic allows to more accurately assess the effectiveness of various combinations of basic heuristics.**

Along with new heuristics, one of the approaches to the analysis of the results of numerical experiments is described. It is based on constructing a reference set of the best quasi-optimal solutions and studying the distributions of the obtained solutions for different variants of the algorithm with respect to this set.

***Keywords*–nondeterministic finite automata, minimization, branch and bound method, heuristic algorithm, implementation.**

REFERENCES

[1] Abramyan M.E., Melnikov B.F. On the application of some heuristics in the study of the state minimization problem for nondeterministic finite automata by the branch and bound method. Part 1 // International Journal of Open Information Technologies. 2020. Vol. 8. No. 9. P. 1–7 (in Russian).

[2] Abramyan M.E., Melnikov B.F. Investigation of the problem of state minimization of nondeterministic finite automata using the branch and bound method // Cloud of Science. 2020. Vol. 7, No. 2. P. 297–319 (in Russian).

[3] Abramyan M.E. On one approach to the implementation of the branch and bound method for optimization problems // Information approaches in modeling and control: approaches, methods, solutions. Collection of scientific papers of the II Russian scientific conference with international participation. Part 1. Togliatti, 2019. P. 56–64 (in Russian).

[4] Abramyan M.E., Melnikov B.F. An approach to algorithmizing the problem of vertex minimization of nondeterministic automata. Part I. Problem statement and the brief description of the basis methods // IOP Conference Series: Materials Science and Engineering. **862** (2020) 052055. P. 1–6.

[5] Melnikov B. Once more on the edge-minimization of nondeterministic finite automata and the connected problems // Fundamenta Informaticae. 2010. Vol. 104. No. 3. P. 267–283.

[6] Melnikov B.F. Regular languages and nondeterministic finite automata: a monograph. – M. : RGSU Publ., 2018 (in Russian).

[7] Melnikov B. The complete finite automaton // International Journal of Open Information Technologies. 2017. Vol. 5. No. 10. P. 9–17.

[8] Melnikov B.F., Melnikova E.A. Clustering situations in real-time algorithms for discrete optimization problems // Control systems and information technologies. 2007. Vol. 28, No. 2. P. 16–20 (in Russian).

[9] Melnikov B., Romanov N. Once again on heuristics for the traveling salesman problem // Theoretical problems of informatics and its applications. 2001. Vol. 4. P. 81 (in Russian).

[10] Melnikov B., Tsyganov A. The state minimization problem for nondeterministic finite automata: the parallel implementation of the truncated branch and bound method // Proceedings – International Symposium on Parallel Architectures, Algorithms and Programming. 2012. P. 194–201.

Mikhail Eduardovich Abramyan,
associate professor of the Southern Federal University, Rostov-on-Don (https://sfedu.ru/),
email: m-abramyan@yandex.ru,
mathnet.ru: personid=20226,
elibrary.ru: authorid=17911,
scopus.com: authorId=8291167000,
ORCID: orcidID=0000-0002-2802-6144.

Boris Feliksovich Melnikov,
professor of the Shenzhen MSU – BIT University (http://szmsubit.ru/),
email: bf-melnikov@yandex.ru,
mathnet.ru: personid=27967,
elibrary.ru: authorid=15715,
scopus.com: authorId=55954040300,
ORCID: orcidID=0000-0002-6765-6800.