Некоторые эвристические алгоритмы в задаче вершинной минимизации недетерминированных конечных автоматов[1]

Мельников Б. Ф., д. ф.-м. н.
Мельникова Е. А., к. ф.-м. н.[2]
Самарский государственный университет
bormel@rambler.ru, elenamel@rambler.ru

В статье рассматриваются два подхода к построению эвристических алгоритмов для задачи вершинной минимизации недетерминированных конечных автоматов. Первый подход предполагает итерационное уменьшение числа вершин путем их объединения. Во втором подходе требуется предварительное построение универсального автомата. Полученные результаты вычислительных экспериментов могут свидетельствовать о возможной применимости обоих рассмотренных нами подходов.

Ключевые слова: недетерминированный конечный автомат, универсальный автомат, вершинная минимизация, эвристический алгоритм, итерационный алгоритм.

1. Введение

В статье рассматривается задача вершинной минимизации недетерминированных конечных автоматов (НКА), поставленная в 1960е годы; сто́ит отметить, что упоминание о ней имеется в [1]. В 1993 г. в [2] было доказано, что она является NP-трудной – и поэтому для решения этой проблемы важной задачей является описание *эвристических* алгоритмов, т. е. алгоритмов, дающее при приемлемом времени работы решение, близкое к оптимальному. Среди них важное место занимают т. н. anytime-алгоритмы ([3, 4] и др.); они обычно основаны на итерационной технике и работают в режиме реального времени, причем в любой момент времени они – по запросу либо пользователя, либо другой программы – дают лучшее (на данном шаге) решение, а последовательность таких (т. н.

[1]Работа частично поддержана грантом РФФИ № 13-01-97003 (р_поволжье_а).

псевдооптимальных) решений в пределе обычно (в случае хорошо разработанных алгоритмов) дает решение оптимальное.

Мы не будем подробно останавливаться на возможных "классических" предметных областях практического применения задачи вершинной минимизации НКА – упомянем лишь, что многие из этих областей описаны в серии статей ван Зейл; среди этих статей мы сошлемся только на [5] (по-видимому, наиболее интересную), в ней приведены некоторые другие ссылки. Однако к классическим предметным областям мы добавим еще и некоторые другие, связанные с предыдущими работами авторов настоящей статьи.

- В [6] описывается построение НКА, связанных с задачами определения выполнения специального отношения эквивалентности на множестве конечных языков – см. также [7, 8]; здесь предварительная минимизация автомата даст возможность получить более эффективные алгоритмы определения этой эквивалентности. Полученные алгоритмы используются в алгоритмах определения эквивалентности в некоторых подклассах класса контекстно-свободных языков, где эта проблема (в отличие от всего класса КС-языков) разрешима: см. уже упомянутую статью [7], а также [9–11]. При этом сто́т отметить, что описываемые в этих работах подклассы класса КС-языков *не* совпадают с классом детерминированных КС-языков (решение проблемы эквивалентности в котором описано в [12]).

- Связь НКА с сетями Петри (а именно – взаимное сведéние моделей, построенных в одном формализме, к моделям, построенным в другом), а также применение последних для моделирования некоторых процессов микроэкономики приведена в [13]; отметим еще, что такая модель нашла применение в [14].

- Кроме того, минимальный (либо псевдооптимальный) недетерминированный конечный автомат может найти применение при описании контекстно-свободных (причем *не* регулярных) языков – см. [15].

Итак, в каждой из предметных областей (как "классических", так и "наших") для применения различных эвристических алгоритмов, связанных с этой конкретной предметной областью, желательно

иметь автомат, предварительно полученный из некоторого (эквивалентного) заданного с помощью процедуры минимизации.

Однако еще более интересна уже упомянутая нами статья [5] тем, что в ней сформулировано (причем, по-видимому, впервые) существование *двух возможных подходов* к решению рассматриваемой нами проблемы – минимизации НКА; конечно же, в обоих подходах рассматриваются *эвристические* алгоритмы. Первый из этих подходов называется авторами статьи [5] подходом Камеды-Вайнера; однако, по-видимому, правильнее было бы назвать его *подходом Полака* (что мы и будем делать далее), несмотря на то, что статья [16] вышла значительно позже статьи [17]; ниже мы также будем называть этот подход *собственно минимизацией*. Второй подход в [5] называется редукцией (reduction) – описание его и является основным предметом той статьи; по-видимому, его можно назвать *подходом ван Зейл*. Более того, авторами статьи [5] подход Полака объявляется бесперспективным. [3]

Очень важно отметить, что существуют самые разные алгоритмы эквивалентного преобразования НКА, уменьшающие число его вершин; важное место среди них занимают итерационные алгоритмы. Например, несколько подобных алгоритмов (уменьшающих число вершин рассматриваемого автомата при его эквивалентном преобразовании) описаны в работе одного из авторов настоящей статьи – см. [21]; подробный пример рассматривается в следующем разделе. А среди последних работ в этом направлении упомянем [22].

Авторы настоящей статьи, в отличие от группы ван Зейл, считают, что *оба* подхода имеют свои возможные области применения – и объяснение этого совершенно стандартное (для эвристических алгоритмов): несмотря на то, что подход Полака работает значительно дольше, для него, в отличие от подхода ван Зейл (и подобных алгоритмов, в первую очередь – итерационных, основанных на уменьшении числа состояний), возможны реализации в виде

[3] Явно это не говорится – но, по-видимому, только так можно воспринимать раздел 2 этой статьи. Сто́ит, однако, сказать, что широко известный и часто применяемый в разных практических приложениях приложениях универсальный автомат Конвея ([18] и мн.др.) в [5] назван *so-called "universal automaton"*.

Отметим еще, что при описании подхода Полака в [5] цитируется работа одного из авторов данной статьи ([19]; она цитируется также в качестве примера, относящегося к *бесперспективному* направлению в минимизации НКА). Еще упомянем [20].

anytime-алгоритмов. В заключении данной статьи мы еще вернемся к этому вопросу.

Итак, в данной статье мы приводим результаты, связанные с разработкой и реализацией anytime-алгоритмов на основе подхода Полака, и их сравнением с нашими итерационными алгоритмами уменьшения числа состояний.

2. Пример работы алгоритма последовательного объединения состояний

В этом разделе мы рассматриваем пример работы *наших* алгоритмов последовательного объединения состояний недетерминированного автомата. При этом еще раз отметим следующее:

- они выполнены на основе теорем, приведенных одним из авторов данной статьи в [21];
- при их описании можно найти общие моменты с алгоритмами, впоследствии приведенными в [22];
- значительно меньше имеется общего с алгоритмами, описанными в [5].

Однако *все три группы* этих алгоритмов можно объединить тем, что они *не* используют универсальный автомат (и подобные ему конструкции) – т. е. не относятся к тому подходу, который мы называем подходом Полака.

Таблица 1.

		a	b
$\rightarrow$	$\{1\}$	$\{2,3\}$	$\{4\}$
	$\{2\}$	$\emptyset$	$\{5,6,7\}$
$\leftarrow$	$\{3\}$	$\emptyset$	$\emptyset$
	$\{4\}$	$\{6,8\}$	$\{9\}$
	$\{5\}$	$\{2,3\}$	$\{9\}$
$\rightarrow$	$\{6\}$	$\emptyset$	$\{4,6,8\}$
$\leftarrow$	$\{7\}$	$\emptyset$	$\emptyset$
$\leftarrow$	$\{8\}$	$\emptyset$	$\emptyset$
	$\{9\}$	$\{4,9\}$	$\emptyset$

Дальнейший материал этого раздела связан с алгоритмом последовательного объединения состояний, созданного на основе [21];

мы пользуемся некоторыми обозначениями, определенными в этой статье. В качестве примера рассмотрим недетерминированный автомат, заданный таблицей 1.

Путем обычных построений можно показать, что этот автомат задает тот же самый язык, что и автоматы, рассматривавшиеся в примерах в [21, Sect. 3]. Для дальнейшего изложения повторим оба канонических автомата, рассматривавшиеся в упомянутых примерах:

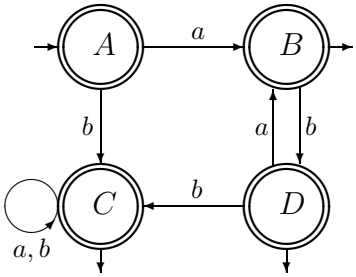

Рис. 1

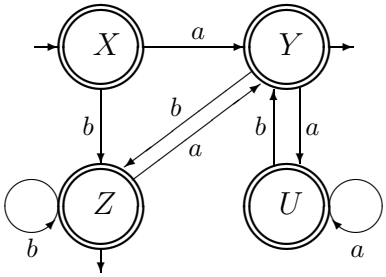

Рис. 2

после чего дополним таблицу 1, добавив значения функций φ^{in} и φ^{out} для состояний рассматриваемого автомата. (Строгие определения этих функций, для использования которых нужны обозначения вершин двух приведенных нами канонических автоматов, также см. в [21]).

Таблица 2.

		a	b	φ^{in}	φ^{out}
$\rightarrow$	$\{1\}$	$\{2,3\}$	$\{4\}$	$\{A\}$	$\{Y\}$
	$\{2\}$	$\emptyset$	$\{5,6,7\}$	$\{B\}$	$\{Z\}$
$\leftarrow$	$\{3\}$	$\emptyset$	$\emptyset$	$\{B\}$	$\{X\}$
	$\{4\}$	$\{6,8\}$	$\{9\}$	$\{C\}$	$\{Y\}$
	$\{5\}$	$\{2,3\}$	$\{9\}$	$\{D\}$	$\{Y\}$
$\rightarrow$	$\{6\}$	$\emptyset$	$\{4,6,8\}$	$\{A,C,D\}$	$\{Z\}$
$\leftarrow$	$\{7\}$	$\emptyset$	$\emptyset$	$\{D\}$	$\{X\}$
$\leftarrow$	$\{8\}$	$\emptyset$	$\emptyset$	$\{C\}$	$\{X\}$
	$\{9\}$	$\{4,9\}$	$\emptyset$	$\{C\}$	$\{U\}$

Согласно [21, Th. 4.1], можно объединить состояния 4, 8 и 9 (что на самом деле делается в два этапа – однако в нашем примере, как можно показать путем детерминизации и последующего вычисления значений функций φ^{in} и φ^{out}, оба объединения возможны).

Отметим, что состояние 6, имеющее “сложное” значение функции φ^{in}, мы пока не обрабатываем: это является аналогом преобразований, осуществляемых при конкретных программных реализациях рассматриваемых нами алгоритмов последовательного объединения состояний. Мы получаем следующий эквивалентный автомат (таб. 3):

Таблица 3.

		a	b	φ^{in}	φ^{out}
$\rightarrow$	$\{1\}$	$\{2,3\}$	$\{10\}$	$\{A\}$	$\{Y\}$
	$\{2\}$	$\emptyset$	$\{5,6,7\}$	$\{B\}$	$\{Z\}$
$\leftarrow$	$\{3\}$	$\emptyset$	$\emptyset$	$\{B\}$	$\{X\}$
	$\{5\}$	$\{2,3\}$	$\{10\}$	$\{D\}$	$\{Y\}$
$\rightarrow$	$\{6\}$	$\emptyset$	$\{6,10\}$	$\{A,C,D\}$	$\{Z\}$
$\leftarrow$	$\{7\}$	$\emptyset$	$\emptyset$	$\{D\}$	$\{X\}$
$\leftarrow$	$\{10\}$	$\{6,10\}$	$\emptyset$	$\{C\}$	$\{X,Y,U\}$

Далее после попарного объединения состояний 3 и 7, а также 1 и 5 (как и в предыдущем случае, это делается в два этапа, а возможность таких объединений следует из [21, Th. 4.2]), мы получаем автомат, определяемый таблицей 4.

Таблица 4.

		a	b	φ^{in}	φ^{out}
	$\{2\}$	$\emptyset$	$\{6,11,12\}$	$\{B\}$	$\{Z\}$
$\rightarrow$	$\{6\}$	$\emptyset$	$\{6,10\}$	$\{A,C,D\}$	$\boldsymbol{\{Y,Z\}}$
$\leftarrow$	$\{11\}$	$\emptyset$	$\emptyset$	$\{B,D\}$	$\{X\}$
$\leftarrow$	$\{10\}$	$\{6,10\}$	$\emptyset$	$\{C\}$	$\{X,Y,U\}$
$\rightarrow$	$\{12\}$	$\{2,11\}$	$\{10\}$	$\{A,D\}$	$\{Y\}$

Отметим изменение значения $\varphi^{out}(6)$ по сравнению с автоматом из таблицы 3, отмеченное в таблице 4 жирным шрифтом.

Как несложно видеть, дальнейшие объединения состояний (согласно алгоритмам, приведенным в [21]) невозможны. Однако полученный автомат содержит 5 состояний – в то время как в [21] для этого языка был приведен следующий автомат, содержащий 3 состояния:

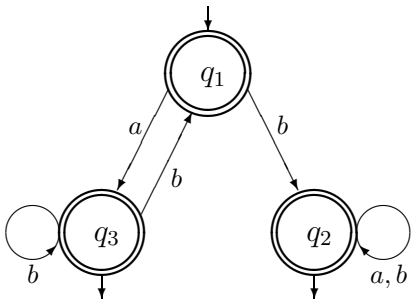

Рис. 3

Таким образом, мы можем сказать, что применение алгоритма последовательного объединения состояний может привести к попаданию в локальный оптимум (мы используем терминологию из [26, Ch. 7]). По мнению авторов настоящей статьи, при решении задачи вершинной минимизации НКА каким-либо алгоритмом локального поиска необходимо подробное исследование эвристик, увеличивающих математическое ожидание числа шагов, предшествующих подобному попаданию в локальный оптимум, и, следовательно, завершению работы алгоритма. Простейшие подобные эвристики, которые очень упрощенно могут быть названы "эвристиками близости состояний автомата", будут кратко рассмотрены далее в настоящей статье.

3. Еще раз о кластеризации ситуаций

Наша реализация подхода Полака основана на расширении метода ветвей и границ (МВГ) – т. н. мультиэвристическом подходе к задачам дискретной оптимизации ([3] и мн.др.). В этом разделе мы приводим описание алгоритмов, являющихся развитием алгоритмов, приведенных в [23, 24].

При работе алгоритма, решающего задачу минимизации недетерминированного конечного автомата таким подходом, для исходных автоматов, имеющих по 16 состояний [4], список подзадач нередко содержит десятки тысяч элементов. При этом для возможности решения подзадач из текущего списка "по аналогии" (а именно –

[4] Случайная генерация *недетерминированных* автоматов осуществлялась с помощью алгоритмов, аналогичных описанным в [25].

для применения *уже выбранного* разрешающего элемента, т. е. блока; см. [19]) необходимо иметь конкретную метрику на множестве подзадач. В наших алгоритмах мы обычно применяем один из следующих двух вариантов.

- Метрика на множестве подзадач определяется по фактическому расстоянию между двумя подзадачами, зависящему от количества несовпадающих элементов в двух матрицах, описывающих рассматриваемые подзадачи. В предыдущих публикациях мы называли такую метрику "матричной".

- Метрика на множестве подзадач определяется на основе описания подзадач, которое формируется в процессе работы метода ветвей и границ и состоит из двух списков, а именно – списка выбранных разделяющих элементов и списка запрещенных для будущего выбора (т. н. табуированных) разделяющих элементов.

Для описания конкретных матричных метрик необходимо ввести следующие обозначения. Пусть X и Y – некоторые непустые конечные множества, при этом n – число элементов их пересечения, а N – число элементов их объединения. Тогда будем писать $\Omega(X,Y) = 1 - \frac{n}{N}$.

Пусть теперь X_1 и X_2 — множества номеров строк, соответствующих 1-му и 2-му блокам, а Y_1 и Y_2 — аналогичные множества для столбцов. Тогда в качестве метрики *для матриц* мы применяем значение $\Omega(X_1,X_2) + \Omega(Y_1,Y_2)$.[5] На ее основе мы определяем

[5] В случае совпадающих блоков (т. е. совпадающих пар множеств X_1 и X_2, а также Y_1 и Y_2) это значение равно 0; выполнение остальных необходимых свойств метрики также можно показать.

А в случае больших размерностей – однако при *фиксированном* числе состояний как для канонического автомата, определяющего исходный язык (т.е., в обозначениях [21], автомата $\widetilde{L}$), так и для канонического автомата, определяющего зеркальный язык (автомата $\widetilde{L^R}$), – мы применяем более сложную метрику, а именно:

$$\alpha \cdot \Omega(X_1,X_2) + (1-\alpha) \cdot \Omega(Y_1,Y_2).$$

При этом α – некоторый коэффициент, настраиваемый заранее (например, с помощью генетических алгоритмов) и зависящий от размерностей двух канонических автоматов.

Подробно схема алгоритма для больших размерностей (в котором мы *не* строим все множество блоков заранее) приведена в [23].

метрику *для произвольных множеств блоков*

$$\mathcal{X} = \{ X_1, X_2, \ldots, X_m \} \quad \text{и} \quad \mathcal{Y} = \{ Y_1, Y_2, \ldots, Y_n \}$$

следующим образом:

$$\rho(\mathcal{X}, \mathcal{Y}) = \frac{\sum_{i,j} \Omega(X_i, Y_j)}{m \cdot n}$$

(i и j принимают все возможные значения).

А на основе последней метрики мы определяем метрику *для подзадач* следующим образом. Пусть $\mathtt{YES}_1$ и $\mathtt{NO}_1$ соответственно – множества выбранных и табуированных разделяющих элементов (блоков) для 1-й подзадачи; $\mathtt{YES}_2$ и $\mathtt{NO}_2$ – соответствующие множества для 2-й подзадачи. Введем вспомогательные расстояния

$$A = \rho(\mathtt{YES}_1, \mathtt{YES}_2), \quad B_1 = \rho(\mathtt{YES}_1, \mathtt{NO}_2), \quad B_2 = \rho(\mathtt{NO}_1, \mathtt{YES}_2),$$

и в качестве метрики для рассматриваемых подзадач будем использовать значение

$$(1 + 2\beta) \cdot A - \beta \cdot (B_1 + B_2);$$

при этом, как и ранее, коэффициент β настраивается с помощью какого-либо эвристического алгоритма, и зависит от размерности, а также от алгоритма генерации входных данных.

Очень важно отметить следующее. В [23, 24] мы описывали наши алгоритмы, лишь *частично* решающие задачу минимизации на основе подхода Полака. Фактически мы решали задачу выбора минимального множества блоков согласно [19, 20], а после получения каждого псевдооптимального решения проверяли, формирует ли полученное решение эквивалентный НКА. То есть – даже при применении комплекса эвристик для поиска схожих подзадач в текущем дереве МВГ – отыскание подзадачи, одной из близких к рассматриваемой, проводилось только на основе описания множества блоков – а не на основе описания всего конечного автомата. В следующем разделе мы опишем начало работ по формированию эвристик для поиска близких подзадач, которые (эвристики) основаны на описании не только множества блоков рассматриваемого автомата, но и его конкретных переходов.

4. Эвристика близости состояний

Итак, в обоих рассматривавшихся нами случаях (в наших реализациях алгоритмов локального поиска и подхода Полака) необходима дополнительная эвристика, отражающая "близость" двух рассматриваемых состояний НКА.[6] Нами предлагается следующая несложная эвристика, при описании которой мы применяем введенные выше обозначения.

Для некоторого недетерминированного конечного автомата

$$(\,Q, \Sigma, \delta, S, F\,)\,,$$

каждого его состояния $q \in Q$ и каждой буквы $a \in \Sigma$ (для простоты рассматриваем автоматы без ε-переходов) рассмотрим множества состояний

$$\mathtt{IN}^a(q) = \{\, r \in Q \mid \delta(r, a) \ni q \,\} \quad \text{и} \quad \mathtt{OUT}^a(q) = \delta(q, a)\,.$$

Мы применяем следующую эвристику "близости состояний"[7]:

$$\rho(q', q'') = \sum_{a \in \Sigma} \Big(\Omega\big(\mathtt{IN}^a(q'), \mathtt{IN}^a(q'')\big) + \Omega\big(\mathtt{OUT}^a(q'), \mathtt{OUT}^a(q'')\big) \Big).$$

(При этом мы, дополнительно к предыдущему, определяем Ω и для пустых множеств, полагая в случае *обоих* пустых множеств $\Omega(\emptyset, \emptyset) = 0$.)

Для примера рассмотрим применение этой эвристики к трем парам состояний автомата, заданного таблицей 3. Обозначения столбцов в приведенной далее таблицы 5 понятны, по-видимому, без дополнительных комментариев. В клетках сначала приведено множество для состояния q', затем – множество для состояния q'', последним – значение функции Ω для этих множеств.

---

[6] Данная эвристика также необходима при создании *гибридного* алгоритма, аналогичного описанному в [27] гибридному алгоритму для задачи коммивояжера. Этот алгоритм строится на основе МВГ и *совершенно иного* алгоритма локального поиска, чем тот алгоритм локального поиска, который был описан выше. Авторы предполагают опубликовать описание этого алгоритма и полученные с его помощью результаты вычислений в следующей публикации.

[7] Приведенная нами запись, вообще говоря, не определяет метрику. Еще отметим, что применение в ней символа Σ два раза в разных смыслах вряд ли создает неудобства.

Таблица 5.

$\{q', q''\}$	$\mathtt{IN}^a$	$\mathtt{IN}^b$	$\mathtt{OUT}^a$	$\mathtt{OUT}^b$
$\{1,5\}$	$\emptyset\ \emptyset\ 0$	$\emptyset\ \{2\}\ 1$	$\{2,3\}\ \{2,3\}\ 0$	$\{10\}\ \{10\}\ 0$
$\{3,7\}$	$\{1,5\}\ \emptyset\ 1$	$\emptyset\ \{2\}\ 1$	$\emptyset\ \emptyset\ 0$	$\emptyset\ \emptyset\ 0$
$\{2,6\}$	$\{1,5\}\ \{10\}\ 1$	$\emptyset\ \{2,6\}\ 1$	$\emptyset\ \emptyset\ 0$	$\{5,6,7\}\ \{6,10\}\ 0.75$

В примере, рассмотренном выше, мы именно на основе полученных значений определяли возможный порядок объединения пар состояний.

5. Результаты вычислительных экспериментов

Приведем краткое описание проведенных вычислительных экспериментов. В алфавите мы рассматривали только 2 буквы (в отличие от вычислительных экспериментов ван Зейл). Генерация случайного НКА осуществлялась аналогично описанному в [25], при этом мы генерировали автоматы, содержащие по 16 состояний (аналогично ван Зейл). При дальнейшем получении эквивалентного канонического автомата (либо канонического автомата для зеркального языка), содержащего более 100 состояний (при нашем варианте случайной генерации НКА это бывало примерно в 50% случаев), дальнейшие вычисления с этим автоматом не проводились.

Время счета итерационным алгоритмом объединения состояний не ограничивалось. Время счета незавершенным методом ветвей и границ определялось таким образом, чтобы последние 50% времени работы не было бы улучшения текущего псевдооптимального решения (но при этом на применяемом нами компьютере с тактовой частотой 2 ГГц – оно составляло бы не менее 1 секунды и не более 300 секунд). Исходный автомат алгоритму незавершенного МВГ был известен (альтернативный вариант алгоритма – когда известен только канонический автомат для рассматриваемого языка – нами не рассматривался). При этом мы *не* определяли относительное время работы двух алгоритмов (поскольку мы, в отличие от группы ван Зейл, считаем эти алгоритмы принципиально разными).

Нами были проведены 100 вычислительных экспериментов (в каждом из которых оба полученных канонических автомата содержали менее 100 состояний). При этом:

- минимальное число вершин полученного эквивалентного псевдооптимального недетерминированного автомата для обоих алгоритмов составило 13, максимальное – 16;
- подходом Полака удалось уменьшить число состояний (т. е. получить результат, меньший 16) в 48 случаях;
- не было ни одного случая, когда бы итерационный алгоритм дал лучший результат;
- в 23 случаях лучший результат дал подход Полака.

Авторы считают, что эти результаты могут свидетельствовать о возможной применимости обоих рассмотренных нами подходов.

6. Заключение

Кратко опишем возможные направления дальнейшей работы по рассмотренной теме.

- Реализовать несколько вариантов алгоритмов подхода Полака (не только на основе незавершенного метода ветвей и границ), а также все упомянутые в данной статье итерационные алгоритмы, после чего проводить более детальное сравнение их работы.
- Продолжить эксперименты с эвристикой близости состояний – различные варианты которой применить во всех данных алгоритмах.
- Улучшить результаты [25] – для случайной генерации НКА, относящихся к конкретным предметным областям.
- Применять гибридные алгоритмы (которые фактически не были использованы в известных нам статьях, связанных с вершинной минимизацией автоматов). Для этого в первую очередь предполагается описать результаты, полученные при адаптации на случай НКА гибридного алгоритма, содержащего алгоритм локального поиска и незавершенный МВГ.

Список литературы

[1] *Ахо А., Ульман Дж.* Теория синтаксического анализа, перевода и компиляции. Т. 1. – Пер. с англ. – М.: Мир. 1978. 613 с.

[2] *Jiang T., Ravikumar B.* Minimal NFA problems are hard // SIAM J. Comput. 1993. Vol. 22. No 6. P. 1117–1141.

[3] *Melnikov B.* Multiheuristic approach to discrete optimization problems // Cybernetics and Systems Analysis. 2006. Vol. 42. No 3. P. 335–341.

[4] *Melnikov B., Radionov A., Gumayunov V.* Some special heuristics for discrete optimization problems // Proc. of 8th International Conference on Enterprise Information Systems, ICEIS-2006. P. 360–364.

[5] *Geldenhuys J., van der Merwe B., van Zijl L.* Reducing nondeterministic finite automata with SAT solvers // Springer. Finite-State Methods and Natural Language Processing. Lecture Notes in Computer Science. 2010. Vol. 6062. P. 81–92.

[6] *Алексеева А., Мельников Б.* Итерации конечных и бесконечных языков и недетерминированные конечные автоматы // Вектор науки Тольяттинского государственного университета. 2011. № 3. С. 30–33.

[7] *Melnikov B.* The equality condition for infinite catenations of two sets of finite words // International Journal of Foundations of Computer Science. 1993. Vol. 4. No 3. P. 267–273.

[8] *Мельников Б.* Алгоритм проверки равенства бесконечных итераций конечных языков // Вестник Московского университета. Серия 15: Вычислительная математика и кибернетика. 1996. № 4. С. 49–53.

[9] *Дубасова О., Мельников Б.* Об одном расширении класса контекстно-свободных языков // Программирование (РАН). 1995. № 6. С. 46–58.

[10] *Melnikov B., Kashlakova E.* Some grammatical structures of programming languages as simple bracketed languages //

Informatica (Lithuanian Acad. Sci. Ed.). 2000. Vol. 11. No 4. P. 441–454.

[11] *Мельников Б.* Описание специальных подмоноидов глобального надмоноида свободного моноида // Известия высших учебных заведений. Математика. 2004. № 3. С. 46–56.

[12] *Sénizergues G.* L(A)=L(B)? decidability results from complete formal systems // Theor. Comput. Sci. 2001. Vol. 251. No 1–2. P. 1–166.

[13] *Зубова Т., Мельников Б.* Использование сетей петри для моделирования процесса принятия управленческих решений // Вектор науки Тольяттинского государственного университета. 2011. № 3. С. 33–37.

[14] *Ханова А.* Концептуальная структура системы управления предприятием на основе интегрированных моделей (на примере грузового порта) // Научно-технические ведомости Санкт-Петербургского государственного политехнического университета. Информатика. Телекоммуникации. Управление. 2012. Т. 3. № 150. С. 99–105.

[15] *Вылиток А., Зубова М., Мельников Б.* Об одном расширении класса конечных автоматов для задания контекстно-свободных языков // Вестник Московского университета. Серия 15: Вычислительная математика и кибернетика. 2013. № 1. С. 39–45.

[16] *Polák L.* Minimizations of NFA using the universal automaton // International Journal of Foundations of Computer Science. 2005. Vol. 16. No 5. P. 999–1010.

[17] *Kameda T., Weiner P.* On the state minimization of nondeterministic finite automata // IEEE Transactions on Computers. 1970. Vol. C-19. P. 617–627.

[18] *Lombardy S., Sakarovitch J.* The universal automaton // Logic and Automata. 2008. P. 457–504.

[19] *Melnikov B.* A new algorithm of the state-minimization for the nondeterministic finite automata // The Korean Journal of Computational and Applied Mathematics. 1999. Vol. 6. No 2. P. 277–290.

[20] *Melnikov B.* Once more about the state-minimization of the nondeterministic finite automata // The Korean Journal of Computational and Applied Mathematics. 2000. Vol. 7. No 3. P. 655–662.

[21] *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.

[22] *Yo-Sub Han.* State elimination heuristics for short regular expressions // Fundamenta Informaticae. 2013. Vol. 128. P. 445–462.

[23] *Мельников Б., Мельникова Е.* Кластеризация ситуаций в алгоритмах реального времени в некоторых задачах дискретной оптимизации // Известия высших учебных заведений. Поволжский регион. Естественные науки. 2007. № 2. С. 3–11.

[24] *Мельникова Е.* Применение алгоритмов кластеризации подзадач для вершинной минимизации недетерминированных конечных автоматов // Вектор науки Тольяттинского государственного университета. 2012. № 4. С. 86–89.

[25] *Мельников Б., Пивнева С., Рогова О.* Репрезентативность случайно сгенерированных недетерминированных конечных автоматов с точки зрения соответствующих базисных автоматов // Стохастическая оптимизация в информатике. 2010. Т. 6. № 1. С. 74–82.

[26] *Hromkovič J.* Algorithmics for Hard Problems. Introduction to Combinatorial Optimization, Randomization, Approximation, and Heuristics. Springer. 2003. 538 p.

[27] *Мельников Б., Эйрих С.* 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.