УДК 004.023, 004.8

АЛГОРИТМИЗАЦИЯ ТРУДНОРЕШАЕМЫХ ЗАДАЧ. ЧАСТЬ II. БОЛЕЕ СЛОЖНЫЕ ЭВРИСТИКИ

Громкович Юрай,
профессор, доктор физико-математических наук,
Eidgenössische Technische Hochschule,
Цюрих, Швейцария
juraj.hromkovic@inf.ethz.ch

Мельников Борис Феликсович,
профессор, доктор физико-математических наук,
Самарский государственный университет,
Самара, Россия
bormel@rambler.ru

Аннотация. Во второй части статьи мы продолжаем излагать свой взгляд на т.н. труднорешаемые вычислительные задачи и возможные подходы к их алгоритмизации – на уровне, «несколько превышающем научно-популярный», однако «несколько меньшем, чем научный».

При этом мы делаем упор на описании методов разработки алгоритмов для них и не сводим рассмотрение трудных задач, а также нашу интерпретацию трудности, к т.н. NP-трудности. В центре нашего внимания находятся проблемы, для которых (пока) не доказаны ни NP-трудность, ни полиномиальная разрешимость – т.е. неизвестно, существуют ли алгоритмы, решающие эту задачу за т.н. полиномиальное время (время, ограниченное некоторым полиномом относительно размера входных данных). Однако, кроме того, мы считаем трудными и такие задачи, для которых полиномиальная разрешимость доказана – однако (пока) неизвестны полиномиальные алгоритмы малых степеней.

Среди рассматриваемых нами примеров задач – известные интеллектуальные игры-головоломки, а также задача коммивояжера и проблемы минимизации недетерминированных конечных автоматов и дизъюнктивных нормальных форм.

Среди алгоритмов мы в первую очередь рассматриваем эвристические – которые обычно не гарантируют получение оптимального решения, однако с приемлемо большой вероятностью дают решение, близкое к нему. Важной их разновидностью являются т.н. anytime-алгоритмы – алгоритмы реального времени, которые в каждый определенный момент работы имеют лучшее (на данный момент) решение; при этом пользователь в режиме реального времени может просматривать эти псевдоопти-

мальные решения, а последовательность таких решений в пределе обычно дает оптимальное.

Ключевые слова: труднорешаемые задачи; эвристические алгоритмы; искусственный интеллект; задачи дискретной оптимизации.

ALGORITHMICS FOR HARD PROBLEMS. PART II. SOME DIFFICULT HEURISRICS

Hromkovič Juraj,
professor, doctor of physical and mathematical sciences,
Eidgenössische Technische Hochschule,
Zürich, Switzerland
juraj.hromkovic@inf.ethz.ch

Melnikov Boris,
professor, doctor of physical and mathematical sciences,
Samara State University,
Samara, Russia
bormel@rambler.ru

Abstract. In the second part of this paper, we continue to present our views on the hard problems and possible approaches for their algorithmics. The level of presentation is “slightly larger than the popular-science”, but “slightly less than scientific”.

We call attention to the fact that we do not restrict our interpretation of hardness to so called NP-hardness in this book. Problems like primality testing, that are not known to be NP-hard (but that are also not known to be polynomial-time solvable), are in the center of our interest, too. It focuses on a systematic presentation of the fundamental concepts and algorithm design techniques.

Among the examples of the tasks before us, there are the famous intellectual puzzle games, as well as the travelling salesman problem and problems of minimization of nondeterministic finite automata and disjunctive normal forms.

Among the algorithms, we first consider the heuristic, which usually does not guarantee an optimal solution, but with an acceptably high probability yield a solution that is close to it. An important type is the so-called anytime-algorithms, i.e., real-time algorithms that have the best (for now) solution at any given moment; wherein the user in real-time may look these pseudo-optimal

solutions, and usually, the sequence of these solutions gives in the limit the optimal one.

Key words: hard problems; artificial intelligence; heuristic algorithms; discrete optimization problems.

Более сложные эвристики

В [1] (в части I данной статьи) мы сформулировали (весьма сложную) задачу коммивояжера[1] и привели очень простой (т.н. жадный) подход к созданию алгоритмов для ее решения; а ранее, в [4], мы начали рассматривать существенно более сложную эвристику (т.н. метод ветвей и границ, МВГ) – но на примере значительно более простой задачи.

Здесь мы рассмотрим МВГ подробнее – именно на примере задачи коммивояжера (ЗКВ). Первые варианты описания этого алгоритма появились около 50 лет назад. Он основан на т.н. бэктрекинге – т.е. его можно рассматривать как специальную технологию полного перебора в пространстве всех допустимых решений. При этом главная проблема заключается в том, что мощность всего множества допустимых решений для входов размерности n обычно очень велика – например, 2^n, n! или даже n^n.

Итак, что же нам делать в тех случаях, когда простейший алгоритм «останавливается раньше времени»? Как мы уже отмечали в [1, 4], необходимо разделить рассматриваемую задачу на две подзадачи; в примере из [4] мы выбирали некоторую клетку (т.н. *разделяющий элемент*) – и в одной подзадаче считали, что эта клетка закрашена, а в другой – что нет. А в случае ЗКВ разделяющим элементом является некоторая пара городов (клетка матрицы,

[1] Особенностью этой задачи является то, что при относительной простоте ее постановки нахождение оптимального решения (оптимального маршрута) является весьма сложной проблемой и относится – причем как в обобщенной ее постановке, так и для большинства ее вариаций – к классу NP-полных. Более того, согласно классификации, приведенной в [2] и других источниках, задача коммивояжера является примером оптимизационной проблемы, входящей в самый сложный класс NPO(V): он содержит все оптимизационные задачи, для которых (при некотором дополнительном «естественном» предположении, возможный вариант – P≠NP) временна́я сложность всех возможных полиномиальных алгоритмов не может быть ограничена никакой полилогарифмической функцией. (Другим примером подобной задачи является проблема максимальной клики, см. [2, 3].)

ребро соответствующего графа) – и в одной подзадаче мы считаем, что этот элемент обязательно *будет* использован[2] (это – т.н. *правая* подзадача), а в другой – что обязательно *не будет* использован (левая подзадача).

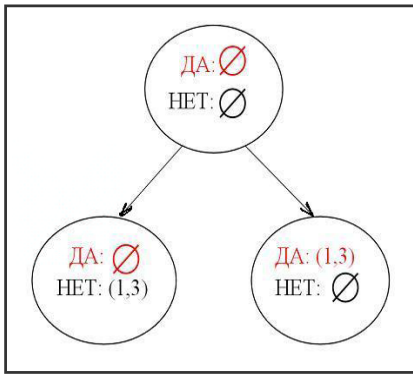

Рис. 7

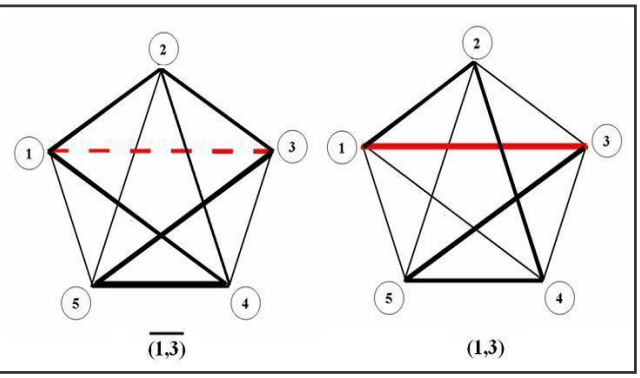

Рис. 8

Продолжим рассмотрение примера для ЗКВ с 5 городами. В качестве разделяющего элемента выберем пару *(1,3)*[3]. Задачу делим на 2 подзадачи: в правую включим все туры, проходящие через ребро графа *(1,3)*, а в левую – все туры, не проходящие через это ребро. Все это (деление пространства состояний на 2 части, соответствующее упомянутому делению задачи на 2 подзадачи) показано на рис. 7, 8. При этом на рис. 7 в каждом овале-*подзадаче* красным цветом дано множество дуг, *обязательно входящих* в тур, а черным – *обязательно не входящих.* А на рис. 8 показаны соответствующие графы для левой и правой подзадач: обязательно входящее ребро показано красной линией, обязательно не входящее – красной пунктирной, а возможные туры – жирной черной.

Очень кратко рассмотрим более сложный пример, с 7 городами; исходная матрица приведена на рис. 9. Цель рассмотрения примера – показать продолжение работы МВГ, т.е. его второй и последующие шаги; мы рассмотрим 5 первых возможных шагов.

---

[2] Т.е. что мы обязательно поедем между этими двумя городами «прямой дорогой».

[3] Такой выбор близок к применяемым в реальных алгоритмах. Для выбора разделяющего элемента *в качестве вспомогательной* применяется жадная эвристика: ведь в исходной матрице выбранный элемент минимален. Однако стоит отметить, что описываемая ситуация несколько сложнее – и даже в простых алгоритмах обычно применяются более сложные жадные эвристики.

	1	2	3	4	5	6	7
1	∞	4	13	7	9	9	14
2		∞	8	3	3	13	12
3			∞	13	9	7	11
4				∞	12	10	4
5					∞	7	5
6						∞	12
7							∞

Рис. 9

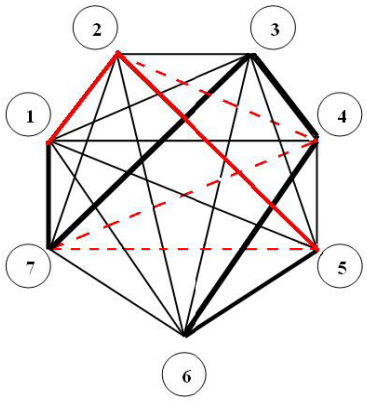

Рис. 10

На рис. 9 отмечены 5 минимальных элементов матрицы, последовательно выбираемых для ветвления МВГ – т.е. в качестве разделяющих[4]. Назовем исходную задачу α; в ней для ветвления выберем элемент *(2,5)*. Следующей будем решать ее правую подзадачу – обозначим ее β; пусть в ней разделяющим элементом является *(2,4)*. Теперь рассмотрим ее левую подзадачу – γ; для нее разделяющим элементом будет *(4,7)*. Левая подзадача последней – δ; разделяющий элемент – *(1,2)*. А ее правая подзадача – ε; в ней разделяющий элемент – *(5,7)*. Полученную в результате подзадачу описывает рис. 10, а всю указанную последовательность выбора подзадач – рис. 11; при этом можно сказать, что рис. 10 – это «усложненный рис. 8», а рис. 11 (т.н. *дерево подзадач*) – «усложненный рис. 7». На рис. 11 мы дополнительно отметили размерности рассматриваемых подзадач – и заметим, что обычно правая подзадача легче для решения, чем соответствующая ей левая[5].

[4] Отметим еще раз, что даже жадные эвристики в реальных ситуациях более сложны – однако мы сознательно упрощаем ситуацию. Кроме того, мы в связи с ограничениями объема статьи не описываем сами *алгоритмы выбора подзадачи* для ветвления – хотя это одна из самых интересных тем. Будем считать, что подзадачи (по каким-то причинам) выбираются именно в том порядке, который рассматривается в нашем примере.

[5] Однако не всегда – что мы и предполагали в рассматриваемом нами примере. А именно – в тех трех случаях, когда выбирали для дальнейшего решения левую подзадачу с еще не рассмотренной правой.

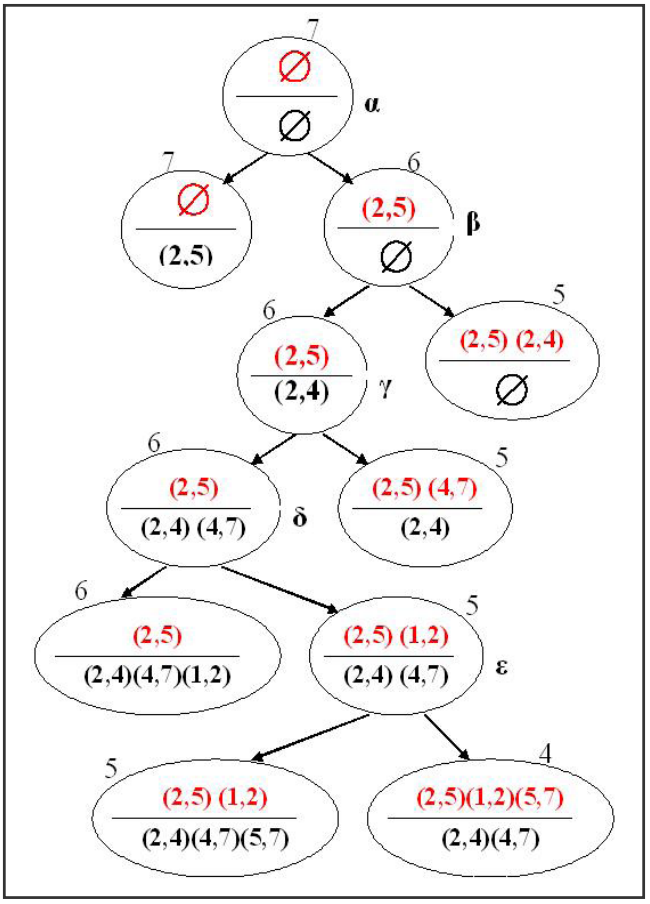

Рис. 11

Итак, можно сказать, сильно упрощая ситуацию[6], что в описанном делении рассматриваемой задачи на две подзадачи и состоит МВГ. Но лучше (хотя тоже приближенно) идею метода ветвей и границ можно сформулировать как убыстрение бэктрекинга – путем *игнорирования поиска в отдельных частях пространства допустимых решений*. Некоторые части этого пространства можно проигнорировать в том случае, когда можно распознать, что они заведомо не содержат оптимальных решений; при этом полный перебор *не* проигнорировал бы такие части. Само это распознавание обычно выполняется путем специальных оценок качества решения – т.е. быстрыми *вспомогательными* алгоритмами, вычисляющими для рассматриваемых подзадач т.н. границы.

Одна из основных целей хорошей реализации МВГ состоит в удачном выборе разделяющего элемента. Простейшая вспомогательная эвристика для такого выбора заключается в следующем.

[6] “This is my blood you drink. This is my body you eat” (Andrew Lloyd Webber and Tim Rice).

Как мы уже отмечали, обычно правая подзадача легче для решения, чем соответствующая ей левая[7], – поэтому удачная реализация заключается в таком выборе разделяющего элемента, чтобы правая подзадача получалась как можно более простой.

Возможный мультиэвристический подход

Итак, метод ветвей и границ сильно облегчает поиск в пространстве состояний (допустимых решений) и, следовательно, обычно существенно уменьшает время поиска оптимального решения. Лучше всего это видно в случае размерностей, *немного* превосходящих те, для которых возможно применение точных (иногда – простых переборных) алгоритмов. Однако если мы рассматриваем задачи еще больших размерностей, то даже МВГ «в чистом виде» не гарантирует получения оптимального решения за приемлемое время.

Выход из этой ситуации однозначно сформулировать невозможно. Точнее, этот выход можно описать самым общим образом: нужно, последовательно улучшая алгоритм для решения конкретной проблемы, «добавлять» к методу ветвей и границ все новые и новые эвристики – до тех пор, пока не получится алгоритм (и соответствующая компьютерная программа), решающий эту проблему за приемлемое время. В этом разделе мы опишем несколько возможных эвристик (точнее – несколько подходов к описанию таких эвристик), *используемых при разработке алгоритмов для конкретных трудных проблем*; но очень важно отметить, что значительно больший эффект эти эвристики дают в случае их *совместного* применения, что позволяет говорить не просто о нескольких эвристиках, а именно о *мультиэвристическом подходе* к решению оптимизационных задач.

Поскольку в нашем случае – при построении anytime-алгоритмов[8] для задач больших размерностей – обычно нет возможности доводить МВГ до оптимального решения, то применяется т.н. незавершенный МВГ. Он получается с помощью такой несложной эвристики. Каждый раз при получении очередной правой задачи (назовем ее задачей T) мы строим *последовательность* правых за-

[7] Это очень хорошо видно именно на примере ЗКВ – в ней размерность правой подзадачи *всегда* на 1 меньше размерности левой.

[8] См. [1], а гораздо более подробно – в [5], где описаны многие из рассматриваемых в этом разделе эвристик.

дач (ППЗ) – т.е. саму задачу *T*, правую задачу задачи *T*, правую задачу правой задачи задачи *T* и т.д.[9]. Естественно, каждый раз формируются (и включаются в список задач для потенциального решения в последующем) и соответствующие левые подзадачи[10]. Этот процесс завершается, например, при получении задачи (небольшой размерности), для которой имеется возможность быстрого решения. Стоимость полученного решения сравнивается со стоимостью текущего оптимального, после чего текущее оптимальное решение может быть заменено на вновь полученное.

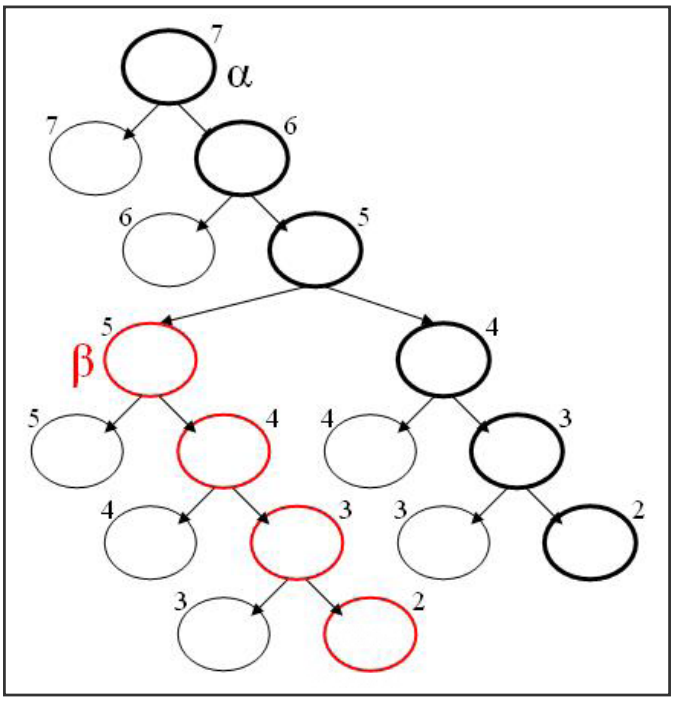

Рис. 12

На эту тему приведем очень схематичный пример дерева подзадач (рис. 12). Как и ранее, овалы соответствуют подзадачам, а рядом с ними указана размерность; считаем, что она всегда остается неизменной для левой подзадачи и уменьшается на 1 для правой. Первый раз ППЗ строится, очевидно, для исходной задачи (α); эта последовательность выделена жирными овалами. Далее, пусть мы по каким-то критериям (т.е. согласно каким-либо жадным эвристи-

[9] «Каждую ночь он спускался в гараж и рыл подземный ход, чтоб добраться до моря» (Б. Гребенщиков).

[10] Удачная организация работы с этим списком подзадач фактически является еще одной эвристикой. Однако, в связи с ограничением объема данной статьи, мы почти не затрагиваем этот вопрос.

кам) выбрали для следующего ветвления подзадачу β (см. рисунок); построенная на ее основе новая ППЗ выделена на рис. 12 красным цветом. В рассматриваемом списке подзадач *в каждый момент времени* содержатся листья такого дерева (само дерево хранить не нужно). Кроме того, как мы уже отмечали, необходимо хранить лучшее найденное текущее решение (или несколько лучших, имеющихся в данный момент времени).

Жадных эвристик может быть несколько ([6, 7, 8]), и они часто предпочитают разные разделяющие элементы[11]. Другими словами – как воспользоваться тем фактом, что в разных конкретных ситуациях относительно лучше работают разные эвристики? Требуется принять решение о выборе конкретного элемента для ветвления. Итак, информацию, данную разными эвристиками (т.н. «предикторами»[12]), надо каким-то образом «усреднять» для получения «единственно правильного», окончательного решения. Для этого в принципе можно пользоваться одним из алгоритмов т.н. многокритериальной оптимизации[13]. В подходе, применяемом одним из авторов данной статьи, для окончательного выбора разделяющего элемента используется аппарат динамически генерируемых функций риска – практически полностью совпадающий с описанным для недетерминированных игр в [6]. Это – специальная модификация «метода голосования», в которой каждая эвристика выдает свой порядок рассматриваемых вариантов выбора разделяющих элементов[14]. После этого «к результатам голосования»

[11] “You know his movements – we know the law” (Andrew Lloyd Webber and Tim Rice).

[12] Отметим, что они часто разрабатываются разными исследователями.

[13] См. [9], а более доступно (и с описанием конкретных алгоритмов выбора) – в [10]... Но сто́ит вспомнить поговорку про стрельбу из пушек по воробьям.

[14] Можно применить следующую аналогию с некоторыми видами спорта, в которых судьи выставляют оценки спортсменам. При этой аналогии рассматриваемый нами метод не имеет ничего общего с большинством из таких видов спорта (прыжки с трамплина, гимнастика и пр.), где оценки судей суммируются. Наш метод похож на еще недавно применявшийся в фигурном катании – где конкретные оценки, с точки зрения математика, не играли никакой роли, а имело значение лишь место, занимаемое каждым спортсменом с точки зрения каждого судьи. Однако в нашем случае (для описываемой аналогии) «судьи имеют разные веса» – причем эти веса зависят не столько от их «опыта» (хотя этот фактор тоже имеет значение), сколько от того, является ли спортсмен лидером соревнований или, наоборот, аутсайдером. Для лидера мы больше прислушиваемся к судьям, которые его оценили плохо («выясняем, почему именно это произошло»), а для аутсайдера, наоборот, «пытаемся найти его небольшие плюсы» у судей, которые все же хоть как-то смогли его оценить.

применяются специальные коэффициенты нормировки (настраиваемые, например, с помощью генетических алгоритмов) и те же самые функции риска[15].

А следующая эвристика применяется в случае необходимости решения задач «очень» больших размерностей[16]. При таких размерностях очень много времени тратится на предварительную генерацию множества потенциальных разделяющих элементов; а на основе приведенных выше примеров (задач минимизации недетерминированных конечных автоматов и дизъюнктивных нормальных форм – см. [1, 5, 8]) понятно, что вспомогательные алгоритмы для такой генерации весьма сложны[17]. Одним из возможных выходов является одновременная (попеременная) работа двух вариантов МВГ – «малого» (предназначенного для генерации множества разделяющих элементов) и «большого» (решающего основную задачу и работающего с уже имеющимся множеством таких элементов); см. [11] и др. И важно отметить, что данный вспомогательный алгоритм близок к теме распределенных (параллельных) вычислений.

В последней части (части III) данной статьи мы опишем некоторые сопутствующие проблемы, возникающие при реализации нашего подхода к алгоритмизации труднорешаемых задач, возможные пути их преодоления, а также – в заключении – некоторые задачи и алгоритмы, не вошедшие в эту статью; вероятно, мы вернемся к ним в наших будущих публикациях.

Литература:

1. *Громкович Ю., Мельников Б.* Алгоритмизация труднорешаемых задач. Часть I. Простые примеры и простые эвристики // Философские проблемы информационных технологий и киберпространства. – 2013. – № 2. – С. 17-30. (*Hromkovič J., Melnikov B.* Algorithmics for hard problems. Part I. Some simple examples and some simple heurisrics // Filosofskiye problemy informatsionnykh tekhnologiy i kiberprostranstva. – 2013. – No. 2. – P. 17-30.)
2. *Hromkovič J.* Algorithmics for Hard Problems. Introduction to Combinatorial Optimization, Randomization, Approximation, and Heuristics. – Springer, 2004.

[15] «Я – верховный судья, и присяжные – я!» (Л. Кэролл, перевод А. Оленича-Гнененко).

[16] Скажем, таких, которые примерно в 5-10 раз превышают размерности задач, решаемых переборными алгоритмами, и примерно в 2-3 раза – размерности задач, решаемых «обычными эвристическими» алгоритмами.

[17] «А к волнам, любезный кот, не привяжешь пароход» (Ю. Мориц).

3. *Громкович Ю.* Теоретическая информатика. Введение в теорию автоматов, теорию вычислимости, теорию сложности, теорию алгоритмов, рандомизацию, теорию связи и криптографию. – СПб.: БХВ-Петербург, 2010. (*Hromkovič J.* Introduction to Automata, Computability, Complexity, Algorithmics, Randomization, Communication, and Cryptography. – Springer, 2004.)
4. *Мельников Б.* Научим машину надежде? // Философские проблемы информационных технологий и киберпространства. – 2013. – № 1. – С. 51-64. (*Melnikov B.* Let us teach computer to hope, don't we? // Filosofskiye problemy informatsionnykh tekhnologiy i kiberprostranstva. – 2013. – No. 1. – P. 51-64.)
5. *Melnikov B.* Multiheuristic approach to discrete optimization problems // Cybernetics and Systems Analysis. – 2006. – Vol. 42, No. 3. – P. 335-341.
6. *Melnikov B.* Heuristics in programming of nondeterministic games // Programming and Computer Software. – 2001. – Vol. 27, No. 5. – P. 277-288.
7. *Мельников Б., Романов Н.* Еще раз об эвристиках для задачи коммивояжера // Теоретические проблемы информатики и ее приложений. – 2001. – Т. 4. – С. 81-94. (*Melnikov B., Romanov N.* Yeschyo raz ob evristikah dlya zadaci kommivoyazhyora // Teoreticheskiye problemy informatiki i yeyo prilozheniy. – 2001. – Vol. 4. – P. 81-94.)
8. *Баумгертнер С., Мельников Б.* Мультиэвристический подход к проблеме звездно-высотной минимизации недетерминированных конечных автоматов // Вестник Воронежского государственного университета. Серия: Системный анализ и информационные технологии. – 2010. – № 1. – С. 5-97. (*Baumgärtner S., Melnikov B.* Mul'tievristicheskiy podhod k probleme zvyozdno-vysotnoy minimizacii nedeterminirovannyh konechnyh avtomatov // Vestnik Voronezhskogo gosudarstvennogo universiteta. Seriya: Sistemnyy analiz i informacionnye tehnologii. – 2010. – No. 1. – P. 5-7.)
9. *Подиновский В., Ногин В.* Парето-оптимальные решения многокритериальных задач. – М.: Наука, 1982. (*Podinovskiy V., Nogin V.* Pareto-optimal'nye resheniya mnogokriterial'nyh zadach. – M.: Nauka, 1982.)
10. *Березовский Б., Барышников Ю., Борзенко В., Кемпнер Л.* Многокритериальная оптимизация. Математические аспекты. – М.: Наука, 1989. (*Berezovskiy B., Baryshnikov Yu., Borzenko V., Kempner L.* Mnogokriterial'naya optimizaciya. Matematicheskie aspekty – M.: Nauka, 1989.)
11. *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. (*Melnikov B., Melnikova E.* Klasterizaciya situaciy v algoritmah real'nogo vremeni v nekotoryh zadachah diskretnoy optimizacii // Izvestiya vysshih uchebnyh zavedeniy. Povolzhskiy region. Estestvennye nauki. – 2007. – No. 2. – P. 3-11.)