УДК 004.8.023, 004.83

Б. Ф. Мельников, Е. А. Мельникова, А. Н. Радионов

ПОДХОД К ПРОГРАММИРОВАНИЮ НЕДЕТЕРМИНИРОВАННЫХ ИГР (Часть II: СПЕЦИАЛЬНЫЕ ЭВРИСТИКИ И ПРИМЕРЫ)

Аннотация.

Актуальность и цели. Создание интеллектуальных компьютерных игр является одним из основных направлений искусственного интеллекта. Кроме того, компьютерные игры предоставляют мощный арсенал разнообразных средств, используемых для обучения. Классическим метод для программирования детерминированных игр для двух лиц с полной информацией – это минимаксный алгоритм. При программировании недетерминированных игр неприменимы стандартные методы, развитые для детерминированных игр. Цель работы: разработать алгоритмы для недетеминированных игр, основанные на обработке модифицированного дерева поиска игры.

Материалы и методы. Разработаны эвристики для упорядочивания вершин в недетерминированном дереве перебора, которые сокращают время обработки узлов дерева и, следовательно, позволяют с большой вероятностью получать оценку исследуемой игровой позиции, близкую к оптимальной. Также рассмотрена возможность одновременного применения недетерминированного дерева перебора и нейронных сетей. В статье приведены примеры работы предложенных алгоритмов для построения конкретных оценок вершин (игровых позиций) различных уровней в недетерминированном дереве перебора. Для практического применения описываемых эвристик в игровых программах необходима оценочная функция позиций. В статье описаны способы ее построения и самообучения. В примерах работы алгоритмов значения оценок выбираются таким образом, чтобы примеры, несмотря на их малый объем, были бы интересными.

Результаты. Разработанные авторами алгоритмы реализованы в компьютерных игровых программах, они также находят свое применение не только непосредственно в недетерминированных играх, но и в других задачах дискретной оптимизации.

Выводы. Применение разработанных авторами эвристик позволяет повысить эффективность алгоритмов для программирования недетерминированных игр: уменьшить время работы и объем используемой памяти, улучшить качество игры.

Ключевые слова: алгоритмизация, недетерминированные игры, модифицированное дерево поиска.

B. F. Mel'nikov, E. A. Mel'nikova, A. N. Radionov

APPROACH TO NONDETERMINISTIC GAMES PROGRAMMING (Part II: SPECIAL HEURISTICS AND EXAMPLES)

Abstract.

Background. Creation of intellectual computer games is one of the main directions of artificial intelligence. Besides, computer games provide a wide range of various means used in education. The classical method for programming nondeterministic games for 2 users with full information is a minimax algorithm. In programming of nondeterministic games it is impossible to apply standard methods devel-

oped for deterministic games. The article is aimed at developing algorithms for nondeterministic games based on processing of a modified search tree of a game.

Materials and methods. The authors developed heuristics to order the top points in a nondeterministic search tree, that reduce the time of tree nodes processing and, consequently, allow to obtain an estimate of the investigated game position with greater probability, close to optimal. The authors also considered a possibility of simultaneous application of the nondeterministic search tree and the neural networks. The article adduces the performance examples of the suggested algorithms for constructing concrete estimates of top points (game positions) of various levels in the nondeterministic search tree. For practical application of the onsidered heuristics in game programs it is necessary to use the position estimator. The article describes the methods of formation and self-learning thereof. In algorithm performance examples the estimates' values are chosen in such a way that the examples, regardless of their small size, would be interesting.

Results. The algorithms, developed by the authors, are realized in computer game programs; they also find application not just in nondeterministic games, but in other problems of discrete optimization as well.

Conclusions. Application of the heuristics, developed by the authors, allows to increase effectiveness of the algorithms for nondeterministic games programming – reduce the running time and capacity of used memory, improve game quality.

Key words: algorithmics, nondeterministic games, search tree.

Статья является второй частью работы, опубликованной в 2013 г., № 4 (28). Нумерация разделов и рисунков в части II продолжается. Разделы 1–3 были приведены в части I.

4. Эвристики для реализации дерева перебора

Как уже было сказано в части I настоящей статьи, набор вершин дерева перебора может быть организован *не* как упорядоченный список или массив, а как некоторая другая структура данных. Фактически каждый раз в процессе перебора (построения дерева) мы выбираем только одну детерминированную либо недетерминированную вершину – в зависимости от текущего состояния алгоритма. Именно поэтому каждый раз, когда мы включаем указатели на новые вершины игрового дерева в структуру данных и меняем некоторые оценки в дереве игры из-за увеличения глубины перебора, нам нет необходимости сортировать вершины в такой структуре данных. Все что нам при этом нужно – это получение *одной* вершины, а именно вершины с минимальным значением описанной ранее функции предпочтения.

Благодаря последнему факту самым удачным вариантом подобной структуры данных является так называемое правильно заполненное дерево (т.е. простая модификация дерева, использующаяся в алгоритме сортировки массивов HeapSort [1] и ему подобных). Элементы такого дерева – вершины *уже построенной части* дерева игры. Отметим еще такое важное обстоятельство: детерминированная вершина удаляется из этой структуры данных (дерева указателей на вершины) после того, как она обработана, поэтому если в процессе работы алгоритма удалены еще не все детерминированные вершины, то это означает, что возможна их дальнейшая обработка. Этот факт отражает возможность увеличения уровня оценки, т.е. нижнего индекса (у символов E в наших обозначениях). Например, первое из таких увеличений

приводит к вычислению *статической* оценки для данной вершины (т.е. E_0), второе – к вычислению статических оценок недетерминированных вершин следующего уровня и т.д.

Следующую эвристику опишем очень кратко: ее подробное описание (даже вне связи с обычными задачами дискретной оптимизации) является слишком длинным, и авторы надеются, что это станет темой дальнейших программ и соответствующих публикаций. Итак, работая с деревом поиска, мы получаем множество его вершин как множество подзадач, возникающих в процессе решения задачи дискретной оптимизации сокращенным методом ветвей и границ [2, 3]. Некоторые вершины могут весьма несущественно отличаться своими оценками (в общем случае, некоторыми специальными значениями, которые зависят от решаемой подзадачи; обычно главной из таких характеристик является размерность задачи). Как и в сокращенном методе ветвей и границ, в процессе работы алгоритма нам может быть необходима обработка двух или более таких вершин. Поэтому возникает вопрос о целесообразности использования алгоритма, который был описан в этом разделе и будет проиллюстрирован в разделе 6. Этот вопрос может быть разрешен делением множества вершин (разбиением набора ситуаций, подзадач) на некоторые классы – аналогично тому, как это делается для обычных задач дискретной оптимизации ([3–5] и др.).

Последняя из эвристик, описанных в этом разделе, весьма проста. Насколько чаще (реже) мы должны обработать недетерминированные вершины – по сравнению с детерминированными? Ответ на этот вопрос мог быть найден, например, с помощью практического программирования и использования генетических алгоритмов, но здесь существует и простое решение, основанное на рассмотрении нескольких возможных вариантов. Авторы в процессе практического программирования пришли к выводу, что для некоторых игр (короткие нарды и т.д.) для обоих видов вершин необходим примерно одинаковый объем работы. То есть алгоритмом, близким к оптимальному, является обработка одной детерминированной вершины после обработки одной недетерминированной и т.д. – хотя нет никакой гарантии, что подобная ситуация сохраняется во всех недетерминированных играх.

5. Применение нейросети для статической оценки

Одновременно с построением оценочной функции с помощью эксперта (см. [6] и др.) авторами были рассмотрены некоторые вопросы, касающиеся алгоритмов автоматизированного построения таких функций. В качестве аппроксимации статической оценки позиции использовалась трехслойная нейросеть с обратным распространением ошибки. Рассматривался ряд эвристик, предназначенных для ускорения обучения такой нейросети – впоследствии используемой в качестве оценочной функции.

Обучение осуществлялось с помощью так называемого TD-λ алгоритма [7]. На входы нейросети подавалась в кодированном виде игровая позиция; кроме того, несколько дополнительных входов принимали значения эвристик, вычисляемых отдельно. Среди таких эвристик (в случае бэкгеммона) были: оценки блокады фишек противника, подвижность фишек, значения статических оценок позиции, вычисленных с помощью внешней экспертной оценочной функции, и др. Позиция для входов нейросети представлялась вектором из нулей и единиц. Каждая ячейка игровой позиции кодировалась

2p элементами. Количество фишек первого игрока в некоторой ячейке доски соответствует количеству единиц в первых p элементах, второго игрока – количеству единиц в следующих p элементах; при этом если количество фишек в ячейке превышало p, то это значение «обреза́лось» до p, что также аналогично TD-λ алгоритму. Однако очень важно отметить, что аналогии с TD-λ алгоритмом на этом кончаются: получаемую оценку мы используем в качестве статической, т.е. как *вспомогательную* для окончательной, динамической оценки.

Использование значений $p > 4$ не обнаружило существенного влияния на скорость обучения нейросети. Экспертные оценки кодировались следующим образом: диапазон наиболее вероятных значений экспертной оценки разбивался на ряд интервалов, и на соответствующий вход нейросети подавалось значение 1 либо 0 – в зависимости от того, попадает ли оценка в соответствующий интервал.

Как и ожидалось, использование дополнительной эвристической информации позволило существенно ускорить процесс обучения сети на первых этапах – в том смысле, что качество игры с использованием «эвристических входов» выше, чем без их использования. «Степень полезности» той или иной эвристики оценивалась по вкладу абсолютных величин весов сети для соответствующих входных нейронов в общую сумму весов входных нейронов. Как правило, «полезные» эвристики дают относительно большое значение такой суммы.

Таким образом, сочетание нейросетевой функции оценки и эвристик для вычисления оценок детерминированных и недетерминированных позиций, описанных выше, позволило добиться более высоких темпов обучения нейросети. Так, если сеть, не использующая никакой экспертной информации о позиции, обнаруживала приемлемое качество игры после примерно 100 000 итераций обучения (т.е. сыгранных для самообучения партий), то аналогичная сеть с «экспертными входами» – после примерно 5 000 итераций.

В дальнейшем обученная сеть рассматривается как вариант оценочной функции совместно с комплексом других эвристик для вычисления оценки позиции подобно тому, как это описано выше. Отсюда вытекает дополнительное требование ко времени ее вычисления – оно должно быть малым (т.е. здесь полностью проявляется аналогия с задачами дискретной оптимизации, решаемыми с помощью так называемых anytime-алгоритмов, см. [2] и др.). Некоторые эвристики, необходимые для реализации этого требования, уже были описаны выше. Итак, сочетание скорости (а следовательно, и ограничение на количество нейронов в сети) с требованием адекватности получаемых оценок объясняет желательность использования «экспертных входов» и в процессе самообучении нейросети.

6. Пример работы алгоритма обработки вершин

Этот раздел содержит иллюстрацию работы с функциями предпочтения, задающими упорядочивание вершин в дереве недетерминированной игры. На рассматриваемом ниже примере авторы стремились показать основные преимущества нашего подхода к порядку обработки вершин такого дерева – по сравнению с подходами, близкими к полному перебору.

Фактически работа проводится с правильно заполненными деревьями, построенными из указателей на детерминированные (или недетерминирован-

ные) вершины дерева поиска, но вместо этих правильно заполненных деревьев мы собираемся рассмотреть пример, в котором используются массивы указателей на такие вершины и сортировка этих массивов. В примере подразумевается некоторая абстрактная недетерминированная игра с тремя вариантами исходов случайных событий.

В отличие от ситуации, изображенной на рис. 1, мы здесь в качестве корня рассматриваем детерминированную (малую) вершину. Поэтому в нашем примере дерево строится для поиска хода с известными вариантами исходов случайных событий – в то время как на рис. 1 дерево было построено для другой цели (не для поиска хода, а для вычисления оценки позиции).

Будем использовать обозначения из предыдущего раздела. Для некоторой обрабатываемой вершины через u обозначим ее уровень (глубину) в дереве поиска; при этом уровни нумеруются тем же способом, как и в разд. 3, т.е. начиная с 0, причем отдельно для детерминированных и недетерминированных вершин. Пусть f – функции предпочтения, которые заданы следующим образом:

- $f = u + |E|$ – для детерминированных вершин,
- $f = u + 1{,}5 \cdot E + (k + 1)$ – для белых недетерминированных вершин,
- $f = u - 1{,}5 \cdot E + (k + 1)$ – для черных недетерминированных вершин.

При этом все обозначения взяты из предыдущего раздела. Отметим, что эти варианты функций предпочтения:

– в точности согласуются с требованиями, сформулированными в разд. 3;

– «близки» к фактическим функциям, используемым в игровых программах, функциям, *получающимся в результате самообучения* с помощью генетических алгоритмов.

Начало работы алгоритма показано на рис. 2. Запись $E_{-1} = 0$ отражает «общий» факт, что данная позиция по умолчанию классифицируется как «нейтральная» ситуация.

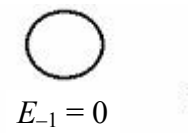

Рис. 2. Корень дерева недетерминированной игры

Следующий шаг является единственно возможным (так как пока имеется только одна вершина); его выполнение заключается в построении черных недетерминированных вершин – непосредственных потомков данной вершины. Предположим, что всего этих потомков 5; на рис. 3 они обозначены $A…E$ – здесь и ниже мы будем одним и тем же способом помечать ходы (или варианты случайных событий) и соответствующие им позиции – это не должно вызвать недоразумений. Пусть значения оценок вершины E_{-1} равны приведенным на рис. 3 под недетерминированной вершиной; тогда мы, используя эти оценки, находим значения функции предпочтения и также пишем полученные значения под соответствующими вершинами. Таким образом, мы получаем следующую последовательность элементов в дереве указателей на недетерминированные вершины: B, C, A, D, E, – поэтому берем для последующей обработки вершину B.

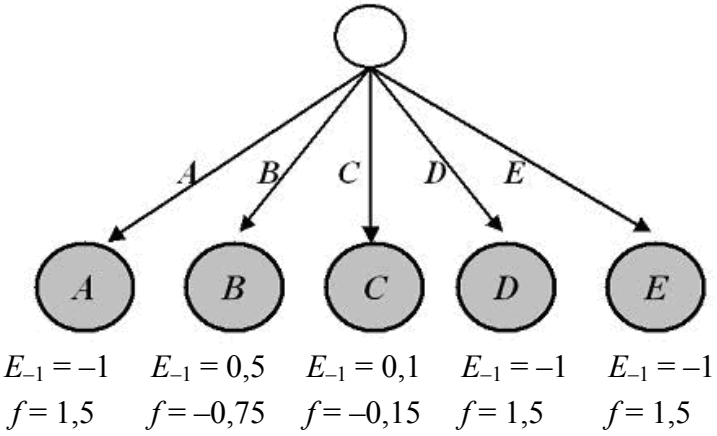

Рис. 3. Построение черных недетерминированных вершин

Как уже было сказано ранее, для каждой недетерминированной вершины (в том числе для B), мы имеем три варианта случайных событий: a, b, c. Пометки E_{-1} и f под соответствующей вершиной на рис. 4 мы не комментируем: они не отличаются от детерминированного случая.

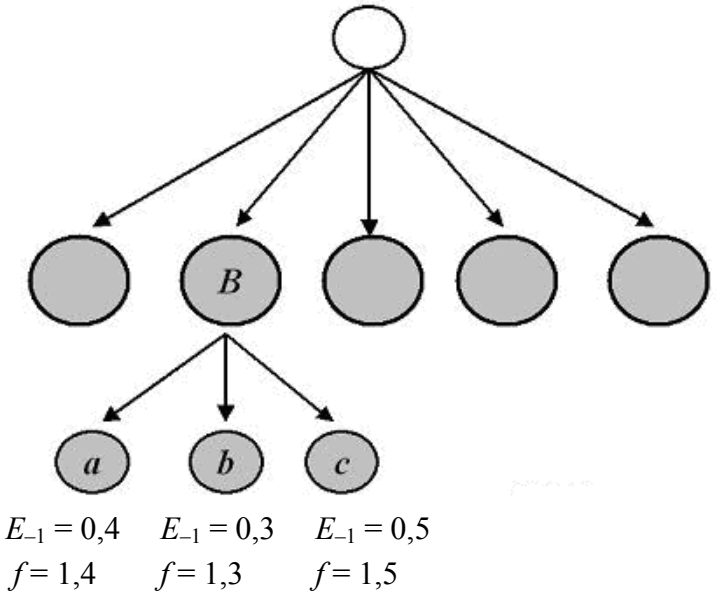

Рис. 4. Построение черных детерминированных вершин

Далее обратим внимание на следующие два обстоятельства. Во-первых, в этом примере фактическая оценка вершины B становится для белых хуже, чем это было до обработки этой вершины (можно заметить, что, скорее всего, ее окончательная оценка также менее 0,5). Во-вторых, мы *уже можем* вычислить динамическую оценку позиции – в точности на основе [8]; вычислим ее с точностью до 0,001.

Шаг первый. Предшествующая оценка позиции – $S = 0{,}4$ (среднее арифметическое). Таким образом, значения функции риска таковы:

1 для $x = 0$; 0,531 для $x = 0,5$; 0,385 для $x = 1$.

Отсюда мы (также см. [8]) получаем такую функцию риска:

$$0,646x^2 - 1,26x + 1.$$

Поэтому взвешенное среднее трех стартовых оценок (с учетом полученной функции риска) получается равным 0,397. Отметим, что это значение является весьма точным, поскольку последующие шаги вычисления оценки позиции дают значения, отличающиеся от нее только в пятом знаке. Таким образом, значение оценки вершины *B* (а следовательно, и корня дерева) равно 0,397. Последний из описываемых нами этапов построения дерева игры показан на рис. 5.

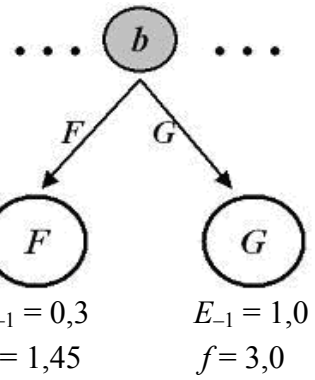

Рис. 5. Построение белых недетерминированных вершин

Предположим, что из вершины *b* (рис. 5) возможны два варианта ходов (*F* и *G*), выбранных согласно вышеупомянутому условию в зависимости от минимального значения модуля оценки позиции. Значения соответствующих предшествующих оценок и функции предпочтения даны под соответствующей вершиной. Однако мы должны учитывать, что значение функции предпочтения для вершины *B* изменилось, так как изменились ее оценка и уровень. Напомним, что эта вершина согласно материалу разд. 3 не исключается из дальнейшей обработки. Итак, согласно приведенным выше формулам значение функции предпочтения в вершине *B* становится равным 0,405 (вместо предыдущего значения – 0,75). Таким образом, последовательность существующих недетерминированных вершин становится следующей: *C*, *B*, *F*, *A*, *D*, *E*, *G*.

Поэтому для дальнейшей обработки (которую мы не будем рассматривать) мы выбираем вершину *C*. Очень важно отметить, что в этом примере мы:

– увеличиваем глубину перебора для вершины *B* (т.е. вычисляем ее статическую оценку) *перед* обработкой некоторой *более глубокой* вершины (*F* или *G*);

– обрабатываем одну из более глубоких вершин (а именно *F*) *перед* рассмотрением некоторой вершины *предыдущего уровня* (*A*, *D* или *E*).

По мнению авторов, два последних факта и формулируют основные преимущества нашего подхода к порядку обработки вершин недетерминированного дерева перебора – по сравнению с подходами, близкими к полному перебору. Этот подход дает конкретные практические результаты.

Заключение

Итак, мы рассмотрели некоторые общие эвристические алгоритмы упорядочивания вершин недетерминированного дерева перебора. Кроме того, мы рассмотрели некоторые примеры работы таких алгоритмов для некоторых конкретных значений оценок позиций, соответствующих вершинам различных уровней этого дерева. Однако для реализации этих эвристик в игровых программах необходимы функции оценки позиций, а также методы построения и самообучения таких функций [8]. Всюду мы выбрали значения оценочных функций произвольно (точнее, так, чтобы получить содержательный пример), но, конечно, в конкретных реализациях игровых программ функции вычисления статических и динамических оценок должны быть получены на основе сказанного в разд. 3–6.

Полученные результаты работы представляют собой различные способы использования «игровых эвристик» в так называемых anytime-алгоритмах. При этом стоит отметить следующие два обстоятельства.

Во-первых, в игровых программах применяются различные вспомогательные алгоритмы, используемые в «обычных» задачах дискретной оптимизации ([2–5] и др.). Например, специальные варианты незавершенного (truncated) метода ветвей и границ. При этом для подбора коэффициентов различных эвристик одновременно с функциями риска применяются генетические алгоритмы. И наоборот – упрощенное самообучение теми же генетическими методами применяется для старта незавершенного метода ветвей и границ.

Во-вторых, в другую сторону: методы и подходы, первоначально использовавшиеся при создании игровых программ, впоследствии нами применялись в «обычных» задачах дискретной оптимизации. Например, специальные версии самообучающихся генетических алгоритмов, а также так называемое турнирное самообучение [3].

Для обоих этих случаев мы имеем в виду в первую очередь следующие задачи дискретной оптимизации:

– классическую задачу коммивояжера (в первую очередь для так называемого псевдометрического случая);

– задачи минимизации недетерминированных конечных автоматов по различным критериям (вершинную, дуговую и звездно-высотную);

– задачу минимизация дизъюнктивных нормальных форм.

Демонстрационные версии соответствующих программ авторы готовы выслать по электронной почте.

В качестве возможного направления дальнейших работ отметим еще одну область, не затронутую в данной статье, но имеющую близкое отношение к эвристикам, описанным в разд. 3, а именно «недетерминированный» эвристический аналог альфа-бета алгоритма. В отличие от классического варианта, этот алгоритм не дает гарантий, что найденное им решение будет оптимальным. Тем не менее на практике оно часто оказывается оптимальным или близким к оптимальному, при этом время работы алгоритма существенно уменьшается. Авторы предполагают привести соответствующее описание недетерминированного аналога альфа-бета алгоритма в одной из будущих статей.

Список литературы

1. **Кормен, Т.** Алгоритмы – построение и анализ / Т. Кормен, Ч. Лейзерсон, Р. Ривест, К. Штайн. – М. : Вильямс, 2005. – 1296 с.

2. **Мельников, Б.** Мультиэвристический подход к задачам дискретной оптимизации / Б. Мельников // Кибернетика и системный анализ. НАН Украины. – 2006. – № 3. – С. 32–42.
3. **Melnikov, B.** Some special heuristics for discrete optimization problems / B. Melnikov, A. Radionov, V. Gumayunov // 8th International Conference on Enterprise Information Systems. – Paphos, Cyprus, 2006. – P. 360–364.
4. **Melnikov, B.** Discrete optimization problems – some new heuristic approaches / B. Melnikov // 8th International Conference on High Performance Computing and Grid in Asia Pacific Region, IEEE Computer Society Press Ed. – Beijing, China, 2005. – P. 73–80.
5. **Баумгертнер, С.** Мультиэвристический подход к проблеме звездно-высотной минимизации недетерминированных конечных автоматов / С. Баумгертнер, Б. Мельников // Вестник Воронежского государственного университета. Сер. Сист. анализ и инф. технологии. – 2010. – № 1. – С. 5–7.
6. **Адельсон-Вельский, Г.** Программирование игр / Г. Адельсон-Вельский, В. Арлазаров, М. Донской. – М. : Наука, 1978. – 255 с.
7. **Tesauro, G.** Temporal difference learning and TD-Gammon / G. Tesauro // Commun. ACM. – 1995. – Vol. 38, № 3. – P. 58–68.
8. **Мельников, Б.** Эвристики в программировании недетерминированных игр / Б. Мельников // Программирование. Известия РАН. – 2001. – № 5. – С. 63–80.

References

1. Kormen T., Leyzerson Ch., Rivest R., Shtayn K. *Algoritmy – postroenie i analiz* [Algorithms – formation and analysis]. Moscow: Vil'yams, 2005, 1296 p.
2. Mel'nikov B. *Kibernetika i sistemnyy analiz. NAN Ukrainy* [Cybernetics and system analysis. National Academy of Sciences of Ukraine]. 2006, no. 3, pp. 32–42.
3. Melnikov B., Radionov A., Gumayunov V. *8th International Conference on Enterprise Information Systems*. Paphos, Cyprus, 2006, pp. 360–364.
4. Melnikov B. *8th International Conference on High Performance Computing and Grid in Asia Pacific Region, IEEE Computer Society Press Ed*. Beijing, China, 2005, pp. 73–80.
5. Baumgertner S., Mel'nikov B. *Vestnik Voronezhskogo gosudarstvennogo universiteta. Ser. Sist. analiz i inf. tekhnologii* [Bulletin of Voronezh State University. Series: System analysis and information technologies]. 2010, no. 1, pp. 5–7.
6. Adel'son-Vel'skiy G., Arlazarov V., Donskoy M. *Programmirovanie igr* [Game programming]. Moscow: Nauka, 1978, 255 p.
7. Tesauro G. *Commun. ACM*. 1995, vol. 38, no. 3, pp. 58–68.
8. Mel'nikov B. *Programmirovanie. Izvestiya RAN* [Programming. Proceedings of the Russian Academy of Sciences]. 2001, no. 5, pp. 63–80.

Мельников Борис Феликсович
доктор физико-математических наук, профессор, кафедра прикладной математики и информатики, Тольяттинский филиал Самарского государственного университета (Россия, г. Тольятти, ул. Юбилейная, 31Г)

E-mail: B.Melnikov@tltsu.ru

Mel'nikov Boris Feliksovich
Doctor of physical and mathematical sciences, professor, sub-department of applied mathematics and informatics, Togliatti branch of Samara State University (31g Yubileynaya street, Togliatti, Russia)

Мельникова Елена Анатольевна
кандидат физико-математических наук, доцент, кафедра прикладной математики и информатики, Тольяттинский филиал Самарского государственного университета (Россия, г. Тольятти, ул. Юбилейная, 31Г)

E-mail: E.Melnikova@tltsu.ru

Mel'nikova Elena Anatol'evna
Candidate of physical and mathematical sciences, associate professor, sub-department of applied mathematics and informatics, Togliatti branch of Samara State University (31g Yubileyna-ya street, Togliatti, Russia)

Радионов Алексей Николаевич
кандидат технических наук, доцент, кафедра прикладной математики и информатики, Тольяттинский государственный университет (Россия, г. Тольятти, ул. Белорусская, 14)

E-mail: alx.radionov@gmail.com

Radionov Aleksey Nikolaevich
Candidate of engineering sciences, associate professor, sub-department of applied mathematics and informatics, Togliatti State University (14 Belorusskaya street, Togliatti, Russia)

УДК 004.8.023, 004.83
Мельников, Б. Ф.
Подход к программированию недетерминированных игр. Часть II: Специальные эвристики и примеры / Б. Ф. Мельников, Е. А. Мельникова, А. Н. Радионов // Известия высших учебных заведений. Поволжский регион. Физико-математические науки. – 2014. – № 2 (30). – С. 49–58.